Mario Hirz · Wilhelm Dietrich
Anton Gfrerrer · Johann Lang

Integrated Computer-Aided Design in Automotive Development

Development Processes, Geometric Fundamentals, Methods of CAD, Knowledge-Based Engineering Data Management

Springer

Integrated Computer-Aided Design in Automotive Development

Mario Hirz · Wilhelm Dietrich
Anton Gfrerrer · Johann Lang

Integrated Computer-Aided Design in Automotive Development

Development Processes, Geometric Fundamentals, Methods of CAD, Knowledge-Based Engineering Data Management

Springer

Mario Hirz
Institute of Automotive Engineering
Graz University of Technology
Graz
Austria

Anton Gfrerrer
Johann Lang
Institute of Geometry
Graz University of Technology
Graz
Austria

Wilhelm Dietrich
MAGNA STEYR Engineering
AG & Co KG
Graz
Austria

ISBN 978-3-642-11939-2 ISBN 978-3-642-11940-8 (eBook)
DOI 10.1007/978-3-642-11940-8
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012954064

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Introduction

Automotive development requires flexible and powerful tools. In the current, highly competitive market, the need to continually reduce development time and costs is driving the ongoing creation of strategies that can provide intelligent and functional links between the many parties involved in vehicle development, including project engineers, ergonomic specialists, safety and crash departments, designers, and many more. While a combination of virtual design and simulation methods with physical development and testing procedures represents the current state-of-the-art, the trend is moving towards integrated virtual development processes. Such processes focus on the product itself while also taking into account a wide variety of potential production and supplier interrelationships, as well as lifetime-relevant factors pertaining to customer use, support, service, and disposal.

In automotive development, computer-aided design (CAD) is used to perform the geometrical product definition, which provides the basis for three-dimensional virtual product models. The models are built by combining main assemblies, sub-assemblies, and individual components, which brings the virtual models close to the configurations of physical products. In this process, design tasks are carried out using parametric-associative techniques, which require the implementation of design-process-related guidelines and project-specific default procedural steps. The realization of parametric-associative model structures and interlinked geometry elements in turn improves the geometry representation by adding elements related to design check features, information relevant to digital mock-ups, or calculations and logical functionalities. The separation of geometry elements and geometry-defining parameters enables the integration of complex computation procedures, the creation of interfaces with design-external processes, and the direct embedding of macro-based automated routines into the design software. In this way, formerly separated working fields (e.g. calculation and simulation) are integrated into or connected to CAD models.

To enhance engineering capabilities in modern product development, intelligent solutions for the collection, storage, and distribution of product and process-oriented data and knowledge must be implemented. Powerful management concepts are necessary to manage the complex information flow, processes, and documents during the development or modification of products. Engineering data management (EDM), which organizes the data flow throughout the development processes and prevents data redundancy, represents an important component in the generation of complex product structures in the context of multi-firm and global collaborations.

The book offers a comprehensive overview of integrated CAD, with a focus on development processes in the automotive industry. This focus does not limit the application of the methods, strategies, and tools described here to a specific industry, but rather provides well-defined boundary conditions within which the topic can be effectively discussed. Nevertheless, the basic findings of this book can certainly be transferred to other industries in the area of mechanical or mechatronics product development.

One primary aim of the book is to introduce and discuss the entire process chain of product design, including the basic methods of geometry creation, the application of CAD, the integration of design and engineering, and finally the management of information related to both product and process. This comprehensive overview of the methods and tools of virtual product development will provide the reader valuable insight into the complex web of interactions and connections that characterize product development.

The following paragraphs provide brief summaries of the nine chapters included here:

Chapter 1, *Automotive Development Processes*, includes a retrospect of achievements in the automotive industry and highlights the very different factors that have influenced the development of cars over the past 120 years. The wide range of requirements for current and future cars is then elaborated, in order to clarify the current challenges facing automotive development. In addition, the stages in automobile development are explained through a detailed analysis of the different project phases, including a discussion on the integration of virtual product creation throughout the entire development chain.

Chapter 2, *Overview of Virtual Product Development*, first provides a summary of the various stages in the life cycle of mechanical products. The main terms, definitions, and methods of computer-aided product development are then introduced. This includes the historical development of CAD, simulation, and data management. In addition, some selected, representative development workflows in automotive engineering are presented and discussed, and the chapter then closes with a brief introduction to the concepts of collaborative product development.

Chapter 3, *Geometric Fundamentals*, introduces the reader to the mathematical and geometrical concepts which form the basis of a CAD system. It starts from scratch and leads the reader through the fields of curves, surfaces, freeform techniques, interpolation, approximation, and a range of other geometrical topics. In effect, this section might also be considered a manual for standard CAD concepts. However, rather than simply listing the methods and algorithms, this chapter actually explains the ideas behind these elements. A proper understanding of these ideas and properties can help engineers perform their jobs more effectively.

Chapter 4, *Modeling Techniques in CAD*, includes a detailed introduction of design methods within the CAD environment. Structures and strategies of wire-frame, surface, and solid modeling are presented and discussed in terms of their application in collaborative product development processes. Beyond the application of primary CAD functionalities, this chapter uses specific examples from the automotive industry to present a variety of methods for the efficient creation of mechanical components and assemblies.

Chapter 5, *Knowledge-Based Design*, covers the use of template models, integrated calculation and simulation procedures, and automated routines to support product design. Knowledge-based design enables the collection, storage, and reuse of expert knowledge, as well as the subsequent integration of know-how into development processes. Using examples from component and assembly development, the chapter elaborates on the potential of enhanced parametric-associative design and knowledge-based engineering used in combination with simultaneously linked calculation procedures.

Chapter 6, *Engineering Data Management* (EDM), describes the fundamental principles of this approach, which involves the interdepartmental and interdisciplinary integration of data and workflows in automotive product development. Both complete EDM use cases and the basic functional modules of CAD and computer-aided engineering (CAE) are described in the context of process-oriented product life cycle management approaches. Finally, the chapter also presents the system-oriented view by describing EDM system architecture with integrated computer-aided applications and data management systems.

Chapter 7, *Knowledge Management in Product Development*, describes product knowledge as a basis for investigation, as well as the development of such knowledge across the product life cycle. The chapter introduces and discusses the fundamentals of knowledge, knowledge management, and knowledge transfer, as well as the principle related basic models and approaches. Thus, the chapter offers a summary of current scientific findings in the area of knowledge management that serves as background for further analysis.

Chapter 8, *Knowledge-Based Engineering Data Management*, describes an approach for using process-oriented knowledge management to identify and organize knowledge-intensive activities in relation to data management activities. Modern design processes involve a variety of tasks (e.g. geometry creation, simulation) and product-specific characteristics (e.g. functional layout, materials, process-relevant data (e.g. for production), product structure, configuration), all of which can be managed with knowledge-based methods. Comprehensive knowledge exchange in product development requires effective data management strategies, which can be applied within knowledge-based EDM.

Chapter 9, *Advanced Applications of CAD/EDM in the Automotive Industry*, offers a selection of concrete use cases in automotive development. One use case includes an application of knowledge-based EDM, which highlights the importance of the interaction of knowledge processes and data management throughout virtual product development. This use case also describes the integrated application of CAD, simulation, and management throughout the daily operations of development processes. Another use case describes the integration of CAD data management in automotive engineering, which is an essential topic in the area of EDM. Finally, a use case describing an approach of a parametric-associative concept model for initial vehicle development highlights the various working fields involved in automotive concept phases and introduces a model for geometrical and functional integration.

Contents

1 Automotive Development Processes 1
1.1 Manifold Requirements in the Past and in the Future 3
1.2 The Process of Automotive Development 11
1.2.1 Project Periods. 13
1.2.2 Phases of Automotive Development. 14
1.3 Application of CAD in Automotive Development 21
References 23

2 Overview of Virtual Product Development 25
2.1 Development of Mechanical Products. 25
2.2 Virtual Product Development 29
2.2.1 Product Models 32
2.2.2 CAD-CAE Workflows in Automotive Engineering . . . 34
2.2.3 Management of Product Data 43
2.2.4 CAD-CAE Data Exchange 45
2.2.5 Concepts of Collaborative Product Development 47
References 49

3 Geometric Fundamentals 51
3.1 The 3-Space, Transformations and Motions. 52
3.1.1 Planar Reflections 55
3.1.2 Translations and Rotations 55
3.1.3 Orientation 57
3.1.4 Helical Displacements 59
3.1.5 Euclidean Motions 60
3.1.6 Some Fundamentals of Line Geometry 62
3.2 Polynomials. 65

3.3 Curves . . . 70
3.3.1 Parametric Representation of a Curve . . . 70
3.3.2 Planar Curves . . . 72
3.3.3 Derivatives and Tangents . . . 73
3.3.4 Arc Length Parameter . . . 76
3.3.5 Curvature and Torsion . . . 77
3.3.6 Osculating Circle and Osculating Plane . . . 78
3.3.7 The Frenet Frame . . . 79
3.3.8 Planar Algebraic Curves . . . 81
3.3.9 Rational Curves . . . 82
3.3.10 Second Order Curves . . . 83
3.4 Freeform Curves . . . 85
3.4.1 Bézier Curves . . . 86
3.4.2 B-Spline Curves . . . 99
3.4.3 Rational Freeform Curves, NURBS . . . 109
3.5 Univariate Interpolation . . . 116
3.5.1 Lagrange Interpolation . . . 119
3.5.2 Interpolation by Cubic Segments . . . 122
3.5.3 Parameterization . . . 134
3.6 Univariate Approximation . . . 136
3.6.1 Improving the Quality of Approximation . . . 140
3.6.2 Approximation with Cubic B-Splines . . . 143
3.7 Surfaces . . . 144
3.7.1 Parametric Representation of a Surface . . . 144
3.7.2 Surface Curves . . . 146
3.7.3 Derivatives and Tangent Planes . . . 147
3.7.4 Curvature Theory of Surfaces . . . 152
3.7.5 Surfaces Represented by Equations . . . 157
3.7.6 Algebraic Surfaces . . . 158
3.7.7 Rational Surfaces . . . 159
3.7.8 Quadrics . . . 160
3.7.9 Ruled Surfaces . . . 165
3.7.10 Developable Surfaces . . . 167
3.7.11 Surfaces of Revolution . . . 170
3.7.12 Helical Surfaces . . . 173
3.7.13 Moving a Curve or a Surface in Itself . . . 174
3.7.14 Intersection of Surfaces . . . 176
3.8 Tensor Product Surfaces . . . 180
3.8.1 Bézier Surfaces . . . 182
3.8.2 B-Spline Surfaces . . . 192
3.8.3 Rational Tensor Product Surfaces, NURBS Surfaces . . . 198

3.9 Bivariate Interpolation 204
3.9.1 Coons Patches 204
3.9.2 Interpolation of a Rectangular Point Set 206
3.9.3 Bivariate Lagrange Interpolation 208
3.9.4 Bivariate Hermite Interpolation 210
3.9.5 Bivariate Cubic B-Spline Interpolation 214
3.10 Bivariate Approximation 216
3.10.1 A Plane Fitting a Set of Scattered Points 216
3.10.2 A Tensor Product Surface Fitting Scattered Data Points 220
3.11 Triangular Bézier Patches 225
3.12 Tensor Product Volumes 232
3.13 Example: Side Window Kinematics 235
3.13.1 The Appropriate Screw Motion to a Given Surface . . . 236
3.13.2 Constructing an Ideal Side Window Surface 237
References 238

4 Modeling Techniques in CAD 241
4.1 Structures of 3D CAD Models 249
4.1.1 Surface-Based Model Structure 250
4.1.2 Solid-Based Model Structure 252
4.1.3 The Role of CAD Models in Product Development . . . 254
4.2 Wireframe and Surface Design 256
4.2.1 Reference Elements 256
4.2.2 Wireframe Design 257
4.2.3 Surface Design 260
4.2.4 Operations in Wireframe and Surface Design 265
4.2.5 Modeling in Wireframe and Surface Design 270
4.2.6 Surface Analysis Functions 274
4.3 Solid Design 276
4.3.1 Modeling of Basis Solids 277
4.3.2 Boolean Operations 279
4.3.3 Editing and Detailing Functionalities 281
4.3.4 Feature-Based Geometry Modeling 282
4.4 Combination of Wireframe, Surface, and Solid-Based Functions 285
4.5 Assembly Design 289
4.5.1 Organization of Product Structures 290
4.5.2 Methods of Component Positioning 294
4.5.3 Geometry-Based Interlinks in Assembly Design 301
4.6 Derivation of 2D Drawings 305
References 308

5 Knowledge-Based Design 309
5.1 Parameterization as a Basis for Knowledge-Based Design. 312
5.1.1 External Parameter Control 315
5.1.2 Implementation of Non-CAD Data. 316
5.2 Knowledge Integration Using Template Models. 317
5.2.1 Template-Library-Based Design. 320
5.2.2 Implementation of Mathematical and Logical Relations 323
5.2.3 Integrated Virtual Product Development Using Centralized Master Models 326
5.3 Example: Integrated Design in Automotive Bumper System Development 328
References 330

6 Engineering Data Management. 331
6.1 The Concept of Engineering Data Management (EDM) 332
6.1.1 The Y-CIM Model 332
6.1.2 PLM as a Foundation of EDM 334
6.1.3 Definition of Engineering Data Management (EDM). 336
6.2 EDM in Virtual Product Development 338
6.2.1 Process Orientation in Product Development 338
6.2.2 EDM as Integrated Management Approach 339
6.2.3 The Product Development Process 341
6.2.4 EDM Support in Virtual Product Development 341
6.2.5 EDM Process Integration 343
6.3 EDM Database 344
6.3.1 The Role of Development Data 344
6.3.2 EDM Documents 346
6.3.3 CAD Data in EDM 347
6.3.4 Digital Mock-Up (DMU) 347
6.3.5 The Virtual Product 348
6.3.6 Data Security. 349
6.4 Engineering Data Management System (EDMS) 350
6.4.1 Product Data Management System (PDMS) 350
6.4.2 Application-Related Functions of EDMS 354
6.4.3 EDMS Architecture 356
6.4.4 EDMS Interfaces 357
6.5 Computer-Supported Engineering in the Context of EDM. 358
6.5.1 How CAx Changes Product Development. 359
6.5.2 CAD Integration 360
6.5.3 CAD Implementation 361
6.5.4 Virtual Computer-Generated 3D Product Design Models. 362

6.6 Integrated EDM Applications in Product Development 364
6.6.1 Functional Dimensioning and Optimization in Early Design Phase 364
6.6.2 Consistency of Simulation Data in Optimized Design Processes 365
6.6.3 Interdisciplinary Consistency of Simulation Data 366
6.6.4 Integration of Design and Simulation 367
6.6.5 CAD/CAE Data Management 368
References 369

7 Knowledge Management in Product Development 371
7.1 Product Knowledge. 371
7.1.1 Development of Product Knowledge 371
7.1.2 Life Cycle of Product Knowledge 373
7.1.3 Defining Product Knowledge. 373
7.1.4 Product Knowledge Products. 374
7.1.5 Product Knowledge Management. 374
7.2 Fundamentals of Knowledge Management 375
7.2.1 Knowledge and Knowledge Management 375
7.2.2 Basic Elements of the Knowledge Base 376
7.2.3 Knowledge Management in Industrial Management . . . 379
7.2.4 Basic Model of Knowledge Management 380
7.2.5 System Orientation in Knowledge Management. 382
7.3 Knowledge Transfer in Product Development 383
7.3.1 Definition of Knowledge Transfer 383
7.3.2 Transfer and Transformation Processes in the Knowledge System 384
7.3.3 Direct Versus Indirect Knowledge Transfer. 385
7.3.4 Direct Knowledge Transfer 386
7.3.5 Indirect Knowledge Transfer 386
7.3.6 The Definition of Knowledge Logistics 388
7.4 Process Orientation in Knowledge Management 389
7.4.1 Knowledge-Oriented Process Management 390
7.4.2 Process-Oriented Knowledge Management 390
7.4.3 The Knowledge Process in Interaction with the Added-Value Processes 391
References 392

8 Knowledge-Based Engineering Data Management 393
8.1 Basic Models and Approaches of Knowledge-Oriented EDM 394
8.1.1 System-Oriented Reference Frame of Knowledge-Oriented EDM 394

8.1.2 The Knowledge Process as Connection Between Business Process and Support Process of EDM 394
8.1.3 Integrated Approach to Added-Value Processes 395
8.1.4 Model for the Integration of Knowledge Processes and Data Management 396
8.1.5 From the Knowledge Transfer Model to the Knowledge-Oriented Engineering Data Management 398
8.1.6 Model for the Reconstruction of the Knowledge Base and Database 399
8.2 Requirements for the IT Support of Process-Oriented Knowledge Management in EDM 400
8.2.1 Modeling Approach for the Technical Subsystem 401
8.2.2 The Database of Knowledge-Oriented EDM 403
8.2.3 EDM Workflow Support of Knowledge-Intensive Processes 403
8.2.4 Management, Transfer and Steering of Knowledge-Oriented EDM 404
8.3 Knowledgeware in Product Development 404
8.3.1 The Parametric-Associative Approach 405
8.3.2 The Fundamentals of Parametric-Associative Design 406
8.3.3 Knowledge Management and Product Configuration 407
References 408

9 Advanced Applications of CAD/EDM in the Automotive Industry 409
9.1 Applications for Knowledge-Based EDM 409
9.1.1 Relevant Knowledge Operations in EDM 410
9.1.2 Factors that Influence Knowledge Transfer Via Data Transfer at the Operational Level 410
9.1.3 Data Management Barriers in Indirect Knowledge Transfer 412
9.1.4 Reference Process for the Knowledge-Oriented Development of EDM Use Cases 413
9.2 Integrated CAD Data Management in Automotive Engineering 414
9.2.1 Challenges Related to the Topic 415
9.2.2 Concept of Integrated CAD Data Management 415
9.2.3 CAD Scheduling 416
9.2.4 A Concept of Geometry Reference 418
9.2.5 CAD Data Quality, Progress and Maturity 419

9.2.6 Generic EDM Workflow for CAD Data Management 421
9.2.7 Data Monitoring 422
9.3 A Parametric-Associative Concept Model for Initial Vehicle Development. . . . 422
9.3.1 Requirements for Automotive Concept Phases. . . . 423
9.3.2 Integrated Approach to Virtual Concept Development 428
9.3.3 Data Pool Structure and Parameterization Strategy. . . . 434
9.3.4 Geometry Creation in Conceptual Vehicle Development 437
9.3.5 Processes and Applications in the Project Flow 446
9.3.6 Product Knowledge in Integrated Virtual Concept Development 450
9.3.7 Integration of the Virtual Concept Vehicle into the Knowledge-Based EDM 452
9.4 Analysis and Design Process of the Operating EDM 454
9.4.1 Integrated Consideration of Design Measures 455
References 455

Curriculum Vitae of the Authors 457

Index 459

Acronyms

2D	Two dimensional
2WD	Two-wheel drive
3D	Three dimensional
4WD	Four-wheel drive
AGC	Automatic geometry check
AR	Augmented reality
AURORA	Automobiltechnisches, anwenderorientiertes Entwurfssystem zur Optimierung der rechnergestützten Auslegung
BE	Business engineering
BMW	Bayerische Motoren Werke
BOM	Bill of material
BR	Business reengineering
BREP	Boundary representation
B-spline	Basis spline
CA	Computer aided
CAA	Component application architecture
CAD	Computer-aided design
CAE	Computer-aided engineering
CAGD	Computer-aided geometric design
CAM	Computer-aided manufacturing
CAO	Computer-aided office automation
CAP	Computer-aided planning
CAPP	Computer-aided process planning
CAQ	Computer-aided quality
CAS	Computer-aided styling
CASE	Computer-aided software engineering
CAT	Computer-aided testing
CATIA V5	Computer-aided three-dimensional interactive application, version 5
CAVA	CATIA V5 automotive extensions vehicle architecture
CAx	Computer-aided technologies

CFD	Computational fluid dynamics
CIM	Computer integrated manufacturing
CIP	Continuous improvement process
CNC	Computer numerical control
COP	Carry over parts
C-spline	Cubic spline
CVT	Continuously variable transmission
DIN	Deutsches Institut für Normung
DM	Document management
DMS	Document management system
DMU	Digital mock-up
DOT	Department of Transportation
DPT	Digital prototype
DSP	Data synchronization point
DXF	Drawing exchange format
EDB	Engineering database
EDM	Engineering data management
EDMS	Engineering data management system
EEVC	European Enhanced Vehicle Safety Committee
ERP	Enterprise resource planning
FCM	Fast Concept Modeler
FE	Finite element
FEM	Finite element method
FMEA	Failure mode and effect analysis
GCIE	Global Car Manufacturer Information Group
GIF	Graphics interchange format
GUI	Graphical user interface
H-Point	Hip reference point
HTML	Hyper text markup language
IGES	Initial graphics exchange specification
IIHS	Insurance Institute for Highway Safety
IS	Information system
ISO	International Organization for Standardization
IT	Information technology
JT	Jupiter tesselation
KBE	Knowledge-based engineering
MBS	Multi body system
MC	Measure control
MPV	Multi purpose vehicle
NC	Numerical control
NEDC	New European driving cycle
NURBS	Non uniform rational B-splines
NVH	Noise, vibration and harshness
OEM	Original equipment manufacturer, e.g. automobile manufacturer
PCA	Principal component analysis

PDF	Portable document format
PDM	Product data management
PDMS	Product data management system
PDP	Product development process
PIM	Product information management
PLM	Product life cycle management
PMS	Project management systems
PPC	Production planning and control (German: PPS)
PS	Post script
QFD	Quality function deployment
R&D	Research & development
RC	Robot control
RPT	Rapid prototyping
SAE	Society of Automotive Engineers
SCM	Supply chain management
SDM	Security distributing and marketing
SDM	Simulation data management
SgRP	Seat relation point (=SRP)
SOP	Start of production
STEP	Standard for the exchange of product data
STL	Structural triangle language
SUV	Sport utility vehicle
TDM	Team data management
TDMS	Team data management system
TIFF	Tagged image file format
TIM	Technical information management
TIS	Technical information system
TPD	Technical product documentation
US	United States (USA)
VB.Net	Visual basic .Net
VBA	Visual basic for applications
VDA	Verband der Automobilindustrie
VDI	Verein Deutscher Ingenieure
VMU	Virtual mock-up
VPP	Virtual process planning
VPT	Virtual prototype
VR	Virtual reality
VRML	Virtual reality modeling language
VW	Volkswagen
WF	Workflow
WRL	Web rule language
xDM	Technologies of data management
XML	Extensible markup language

Chapter 1
Automotive Development Processes

Product development in the automotive industry is driven by a highly complex series of market requirements that stem from a wide range of product variants and functionalities. Stagnating sales volumes in traditional markets and increased competition are leading to both growing product diversification and reduced time-to-market processes. Furthermore, the industrial globalization of development, manufacturing and distribution has resulted in new business models, which must consider several market-specific factors. Changes in boundary conditions, resulting from both legislation and customer orientation, and a growing realization of the finite nature of crude oil supplies are spurring a reorientation of individual mobility and a continuous introduction of new vehicle concepts. In order to develop lighter, more efficient cars driven by alternative propulsion technologies, common vehicle concepts will need to be revised and completely new vehicle architecture and styling solutions will need to be introduced. At the same time, increasing customer demands in terms of safety, comfort and fashion trends are driving the creation and implementation of new technologies. This technical evolution necessitates a reduction of investment risks throughout the entire product life cycle, which makes it essential to consider factors related to production, market conditions, and disposal throughout the entire development cycle.

Development processes in the automotive industry place high demands on the performance and flexibility of the development strategies and tools applied. Besides the standard prerequisites (e.g. design, simulation and production engineering), a number of enhanced requirements will present new challenges to the architects of future development cycles. Within the boundaries of a permanent cost and time reduction in engineering-based development, new strategies must support a smart connection between the working fields of project engineers, component designers, ergonomic specialists, safety and crash departments, designers and all other involved parties. These days, automotive development is driven by the interaction of virtual design and simulation methods in combination with physical development and testing procedures (Fig. 1.1). The trend is definitely going towards integrated virtual development processes which focus on the product function itself, but also take into account both the production and supplier situations, as well as lifetime-relevant factors that pertain

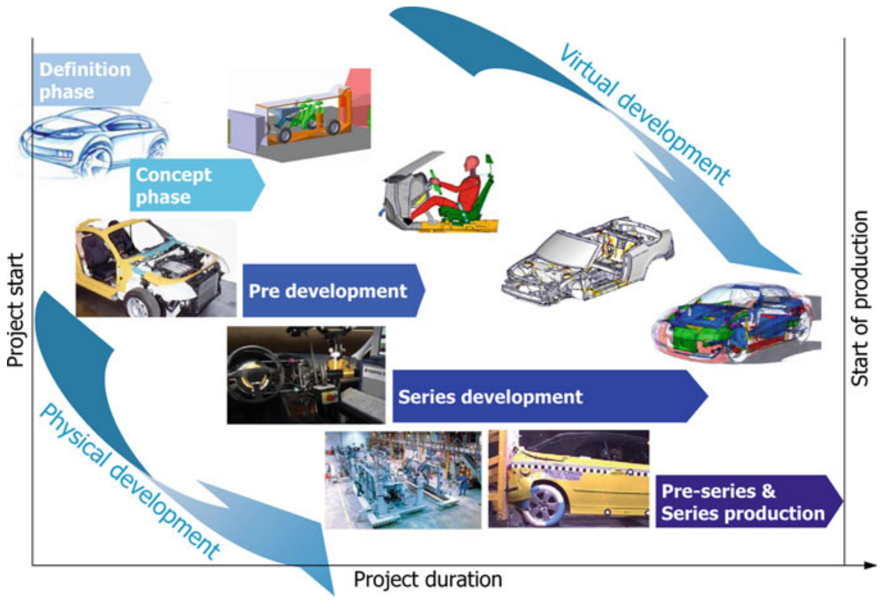


Fig. 1.1 Sample phases of an automotive full-vehicle development process [1]

to customer use, support, service and disposal. The increasing application of virtual methods is taking over several tasks formerly handled by physical development, where the focus is now shifting to data acquisition and verification procedures. This has led to a significant reduction in hardware-based test and prototype development in the last decades.

The manifold variants and characteristics of automotive products present a challenge for the entire development process chain. Unlike the limited model ranges that were typical in the mid twentieth century, the current range of personal automobiles is segmented into a wide variety of different vehicle categories, classes and configurations. New car types (e.g. MPV-multipurpose vehicles, SUV-sport utility vehicles), new body configurations (e.g. hatchback, roadster, coupe-convertible) and the development of new engine and transmission systems (e.g. supercharged gasoline and diesel engines, hybrid and electric drive, automatic transmissions, double clutch transmissions) have resulted in a significant increase in product variants and data. Besides the variety of car types, propulsion systems and chassis configurations, wide-ranging options for supplementary equipment have led to nearly endless product variations, which is good for the customers, but presents a challenge for data management in product development and throughout the product life cycle. At the same time, the development time for new automotive models has decreased from about seven years in the 1960s to about two years today (Fig. 1.2). This considerable reduction in development time has been significantly supported by virtual design, simulation and testing methods.

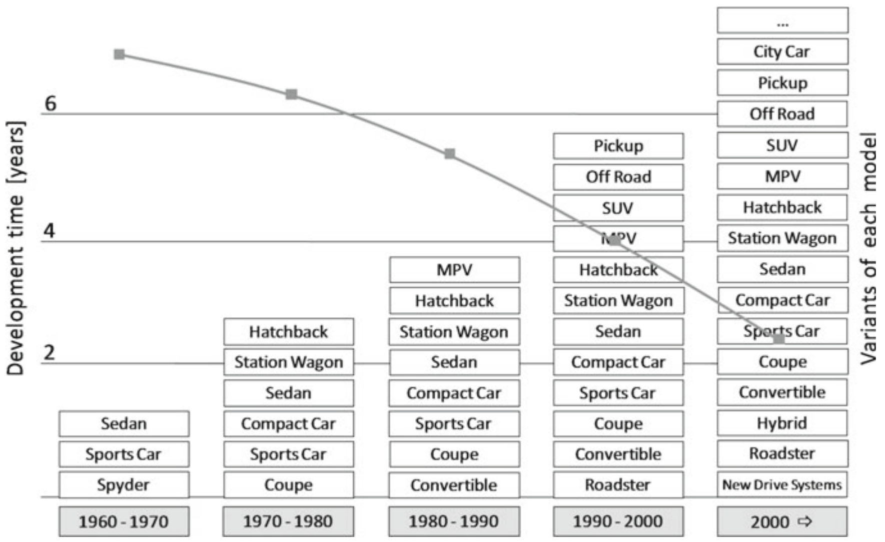


Fig. 1.2 Increase of model variants and decrease of development time in the last decades [2]

1.1 Manifold Requirements in the Past and in the Future

In addition to a brief overview of selected main topics in automotive development over the last 120 years, this chapter includes a discussion of the wide variety of development requirements for current and future cars.

More than a hundred years ago, at the beginning of automotive development, the inventors of new motorized vehicles built up their creations in simple workshops supported by a small group of specialists. Development processes were strictly problem oriented, and improvement cycles were mainly based on hardware testing procedures. Layout, design and optimization followed the functional aspects of the product itself. Figure 1.3 shows Gottlieb Daimler's first *Motorkutsche* from 1886. This vehicle was based on a coach from the Stuttgart Wagon manufacturer Wilhelm Wimpff and Sohn, which was adapted with a one-cylinder, four-stroke engine developed by Daimler. This early four-wheel automobile had a weight of 290 kg and a maximum driving speed of 18 km/h [3]. The development of the engine was completely separate from the development of the vehicle itself. Thus, people called early automobiles *horseless carriages*.

The early years were characterized by high creativity and many vehicle variants. In principle, each car was built up as an individual item. In that time, automotive development was completely hardware based and hierarchically structured. The heads of development were key people who possessed the main knowledge of construction, production and testing procedures. In many cases, the inventor, the chief of development and the business leader were one and the same person. The famous names from these early days of automotive development are well known,

Fig. 1.3 Gottlieb Daimlers first *Motorkutsche* (1886) [4]

such as Nicolaus August Otto (1832–1891, Germany), Siegfried Marcus (1831–1898, Germany, Austria), Gottlieb Daimler (1843–1900, Germany), Henry Ford (1863–1947, USA), Ferdinand Porsche (1875–1951, Austria, Germany) and many others. At the beginning of the twentieth century, the production of the first series vehicles started. Besides product-oriented technical requirements, the development of series vehicles also had to take production-related aspects into account. In those days, production engineering and operation planning were driven by the low number of vehicles produced, meaning that the development process only partially considered the organization and necessities of manufacturing.

Henry Ford made the first steps in mass production development. He adapted assembly line manufacturing knowledge from simple products to the requirements of relatively complex products, such as those produced in the automotive industry. In this way, it was possible to produce large numbers of cars through only partially automated but strictly organized stages with a surprisingly low cost per unit. The introduction of assembly line production led to a modification of the principals of automotive development processes. Besides product-oriented technical requirements, the influences of manufacturing and assembly became more and more important. The first steps of production-line-oriented car development were strictly driven by the approaches of conformity. Initially, the Ford Model T was available in just one configuration, and new configurations were based on the same platform, assembled with a high degree of carryover parts.

Figure 1.4 shows a component assembly and a final assembly line of the Ford Model T, which was introduced in 1908. It had a robust vehicle setup and a four-cylinder engine. In the year it was introduced, the Model T was sold as a low-cost model for $825, a price which dropped in each subsequent year. Although the automobile was initially produced in a conventional way, under Henry Ford's directive, the manufacturing process was subsequently improved. Seeking increased

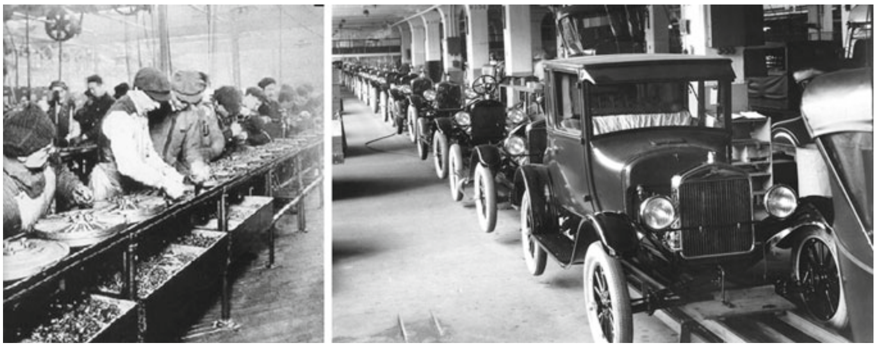


Fig. 1.4 Production of the Ford Model T in Detroit, Michigan [5, 6]

efficiency and lower costs, Ford introduced moving assembly belts into his plants in 1913, which resulted in an enormous increase in productivity. One key to success was the application of production-focused engineering in automotive development. The production of the Model T was continued until 1927, with about 15 million units sold within 19 years. The price of a basic model dropped to $290 in 1927 (which is equivalent to about $6,200 in the year 2012, based on the consumer price index), which made cars affordable for the middle class.

The increasing consideration of production-related aspects in the 1930s eventually had a significant influence on the products themselves. Growing production quantities led to the development of cost-optimized components, the use of building block units (non-variable part strategy) and the assembly of optimized structures. These measures helped lower costs, which supported the distribution of motorized vehicles into different social stratums in both Europe and North America.

At this time, the first stylistic aspects were also incorporated in the development of new cars. Stylists worked with drawings on paper and developed modeling procedures with clay. These so-called clay models can be used in different scales and levels of detail and are still an important part of the styling process today. Besides standard necessities, such as vehicle packaging and passenger space, the first influences on automotive styling came in the period between the World Wars from the aeronautic industry. As an example, Fig. 1.5 shows the famous Bugatti Type 57 SC Atlantic from 1937. Only four vehicles of this sports car were built, and it is currently one of the most valuable antique cars.

In the late 1930s and during the Second World War, the requirements of military armament influenced automotive manufacturers. Concerning automotive development methods, a breakthrough came in 1940 when Riekert and Schunk introduced a scientific approach for the calculation of the driving characteristics of a car, which were described by a series of equations and relations based on a simplified automotive model, the *single track model* [8]. The derivation of a simplified abstraction from a real vehicle to enable the general calculations of characteristic dynamic driving values represented an initial basic approach for the development of calculation and simula-

Fig. 1.5 Bugatti Type 57 SC Atlantic (1937) [7]

tion algorithms in a wide field of applications. Of course, the calculation itself was performed by hand; computer-supported simulation started more than two decades later. Today, the findings of Riekert and Schunk are still often used in different fields of vehicle dynamic simulation.

In the ensuing years, as social prosperity increased, the automotive industry was able to develop new products and to offer product variants. Mobility became more and more important, and car manufacturers offered various models in different price classes. The growing production quantity and the growing number of variants presented challenges for development departments. New development strategies and procedures arose, which led to the implementation of new departments and corporate structures. While communication had been easier in the time of compact development departments (and development targets), as task load and manpower requirements increased, it became necessary to develop new organizational structures for the organization of human resources and processes. In the early years, the development of a car was based on the definition of main vehicle modules, such as the engine module, transmission, suspension, chassis and body. Since individuals (or small groups) were assigned to each clearly defined module, information flow was relatively simple and consisted mainly of direct communication between the responsible parties. In addition, the simple setup of a car resulted in relatively short development times; in many cases, a car was developed in just a few months.

However, as the number of the development fields and people involved increased, new organizational procedures were generated for new car development. Figure 1.6 shows the Opel Record, which was introduced in 1957 and became a successful mid-class model in Europe. At that time, customers had begun to demand more comfort and performance. The development targets were therefore performance, durability and technical functionalities. In addition, styling and fashion became more important in the European automotive industry in the years following reconstruction. Thus,

Fig. 1.6 Opel Rekord P1 from 1957 [9]

styling aspects, comfort and additional customer benefits began to influence the development of personal cars.

In the second half of the 20th century, customer demands expanded to include increased safety and comfort functions, electronic features, brand identity and more. Due to growing product complexity, the automotive development became team oriented. The structured division of labor generated new jobs and areas of responsibility. In the 1960s, development processes were more strictly defined and organized. Driven by increasing responsibilities and resources, development departments were subdivided based on different tasks and requirements. The separation of automotive development into research, pre-development and series development followed. This made it possible to conduct research and develop new, future-oriented technologies without the boundary conditions of mass production and costs.

The growth in individual mobility also led to increasing numbers of traffic accidents, which forced manufacturers to take safety-relevant aspects into account. In 1968, the American Transportation Agency (DOT) started a research program for the development of vehicle safety technology. The European Enhanced Vehicle Safety Committee (EEVC) was founded in 1970, as European car manufacturers also began to intensify their commitment to crash tests and the development of passenger safety technology (Fig. 1.7). These new targets influenced the research topics in vehicle development. The vehicle body structure was analyzed in terms of stiffness and load paths in the context of different accident scenarios, and calculation and crash test procedures were added to the complete vehicle development processes. In later years, the introduction of computation technologies brought new possibilities for vehicle structure calculation. Computer-supported applications, first in simulation and later in design, led to the implementation of new organizational structures and procedures in development processes. The concern with safety continued into the 1980 and 1990s, which saw the introduction of both active and passive safety-relevant functionalities first into luxury cars, and later into all vehicle classes.

In addition to improving vehicle safety, two factors also contributed to a demand to improve vehicle efficiency. The first was a concern for the cost of operating a

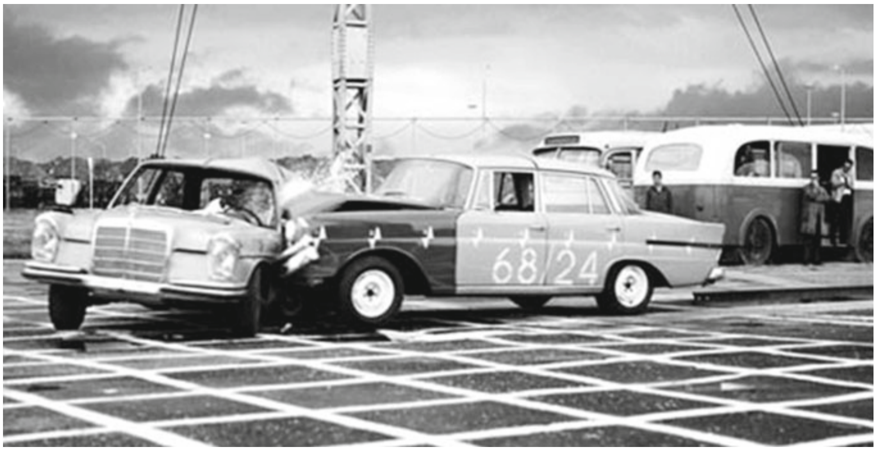


Fig. 1.7 Mercedes Benz crash test 1959 [4]

vehicle, which was brought into start relief by the oil crises of the 1970s, as prices rose and availability declined. The second factor was a growing awareness of the environmental impact of the increasing number of automobiles that arose in the mid-twentieth century. This began in California, where geographic and climate factors led to heavy smog situations with serious air pollution, particularly in densely populated urban areas. Therefore, in the early 1960s, the government of Los Angeles introduced a limitation on carbon monoxide and hydrocarbon exhaust emissions for cars. In Europe, the first exhaust emission regulation was introduced in 1970 and has been continuously upgraded since then. In the 1980 and 1990s, strict exhaust gas legislation led to the implementation of exhaust catalyst systems and electronic motor management systems for gasoline engines. With the development of supercharged diesel engines with direct injection systems, diesel vehicles were introduced into many fields of application.

Along with safety and efficiency requirements, customer demands for better comfort also increased in the second half of the twentieth century. From ride comfort (e.g. enhanced suspension systems) and convenience (e.g. electric windows) to climate control and even on-board entertainment, developers and engineers were compelled to integrate ever more technologies and features into new vehicle designs. However, after a period of focus on improved comfort and performance, the end of the twentieth century saw a distinct shift in focus towards environmental friendliness and fuel efficiency. Low fuel consumption engines and alternative drivetrain concepts, such as natural gas engines and hybrid systems, emerged, as customers began to understand the direct dependency of their mobility on the availability of crude oil. For example, Volkswagen announced the series production of a car with a fuel consumption of one liter per hundred kilometers in 2013 (Fig. 1.8, right). The car bears a notable resemblance to a 1986 Volkswagen concept vehicle, which had already emphasized the requirements of a low-fuel-consumption vehicle, such as an efficient engine,

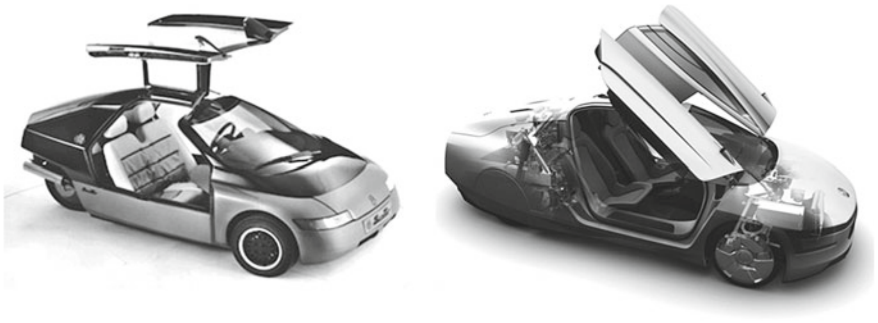


Fig. 1.8 Volkswagen concept cars: Scooter from 1986 and XL1 from 2013 [10, 11]

low weight and low driving resistance (Fig. 1.8, left). However, 20 years later, the time for this vehicle had definitely arrived. In fact, the emphasis on environmental friendliness has gone beyond simple fuel efficiency to include the use of recyclable products and sustainable materials, which is becoming increasingly widespread.

With all these different, constantly evolving demands, contemporary market researchers are constantly finding new market niches and variants to fulfill the customer requirements in terms of individualism and customizing. For example, Fig. 1.9 shows the Nissan Concept Car Pivo2, a prototype vehicle equipped with four electric hub engines, an example of a concept that is designed to meet the requirements of inner city traffic, such as an exhaust-emission-free drive unit, high flexibility for parking space, and the option of electronic supporting equipment. While such concepts represent real value for the consumers, it is important to recognize that all of these sometimes conflicting demands and the product variation that they have caused have presented significant challenges for vehicle engineers and developers.

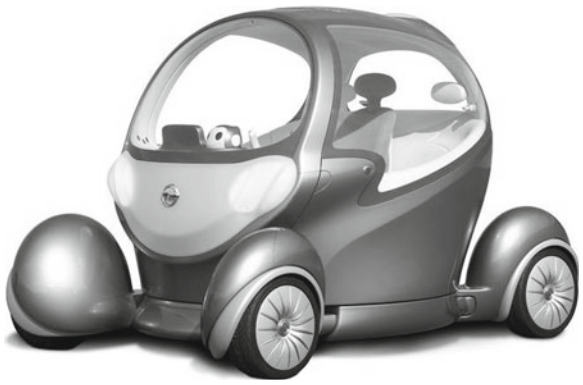


Fig. 1.9 Nissan Concept Car Pivo2 (2007) [12]

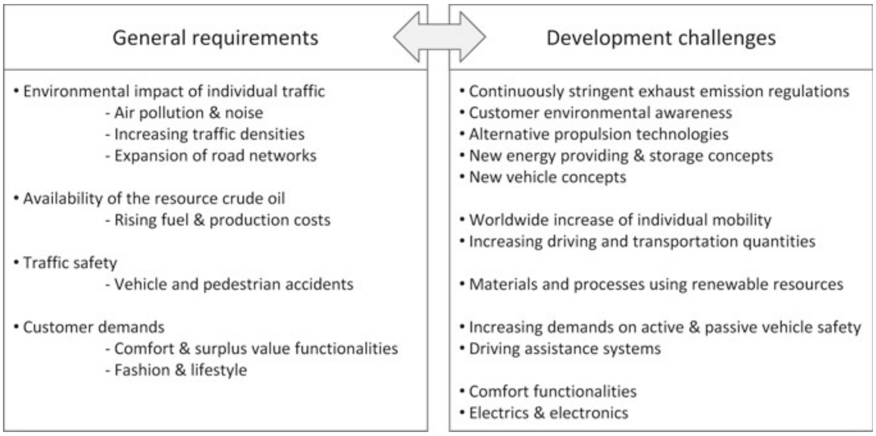
To meet these challenges, fundamental changes had to be applied to both the vehicles themselves and the processes by which they are developed. For example, legislative limits on emissions and the growing impact of fuel costs led to the implementation of new engine concepts and exhaust gas aftertreatment systems. As the complexity of the tasks required for engine development increased, it became necessary to break the development process into separate, component-specific subprocesses, such as engine, transmission and others. One important side effect of this growing complexity was the resulting impetus to cooperate with specific key suppliers for particular components or systems. This led to the definition of new, shared development procedures, which integrated suppliers into the development processes and thereby brought in their knowledge of their respective business fields. One further result of this cooperation in development was the creation of know-how networks and collaborations between different car manufacturers in specific research fields, which play an important role in contemporary vehicle development.

Beyond the increased complexity of the vehicles themselves, the increased complexity of the automobile market has also had a significant influence on modern development strategies and methods. In particular, the globalization of automobile companies has had a significant impact on development processes [13]. On the one hand, there is pressure to meet the specific demands of different customer groups and legislative bodies around the world. On the other hand, there is a counter pressure to standardize as many production processes as possible, in order to achieve synergy between the production facilities in different countries around the globe. Thus, the organization and structure of global development has led to the implementation of global-network-based product data management systems and globalized development processes, which are supported by virtual product development.

In the coming years, automotive development will face significant challenges. New propulsion systems, energy storage media, tank systems and completely new vehicle concepts must be developed to provide continued individual mobility in the coming decades. New vehicle concepts call for new development methods and new engineering tools. Thus, the importance of virtual development in automotive engineering will increase in the next years, and the generation of flexible design and simulation methods will be an important key to success. Nevertheless, the main development targets remain the same as in previous decades - reduction of the costs and development time and increase in the quality and efficiency of development processes.

Figure 1.10 provides an overview of boundary conditions for the development of new car models in the near future. Four main areas influence the general requirements: the environmental impact of individual traffic, the influence of crude oil availability, safety aspects, and the increase in customer demands in terms of comfort and lifestyle.

Focusing on automotive engineering, these boundary conditions have a significant influence on the development targets and thus on the methods applied. The creation of new vehicle concepts that fulfill increasing demands requires careful planning and refinement of existing knowledge, as well as the generation of new ideas and techniques. The early phases of development, in particular, play an important role in conceiving, developing and evaluating new ideas and technologies.



General requirements

- Environmental impact of individual traffic
  - Air pollution & noise
  - Increasing traffic densities
  - Expansion of road networks
- Availability of the resource crude oil
  - Rising fuel & production costs
- Traffic safety
  - Vehicle and pedestrian accidents
- Customer demands
  - Comfort & surplus value functionalities
  - Fashion & lifestyle

Development challenges

- Continuously stringent exhaust emission regulations
- Customer environmental awareness
- Alternative propulsion technologies
- New energy providing & storage concepts
- New vehicle concepts
- Worldwide increase of individual mobility
- Increasing driving and transportation quantities
- Materials and processes using renewable resources
- Increasing demands on active & passive vehicle safety
- Driving assistance systems
- Comfort functionalities
- Electrics & electronics

Fig. 1.10 Boundary conditions for the development of new cars

1.2 The Process of Automotive Development

Through the middle 1980s, automotive development processes were characterized by a high degree of hardware and prototype-based optimization cycles. Development projects were divided into individual tasks, which featured relatively little data sharing. Individual processes were performed in essential serial sequences, and data transfer was organized in rigid structures based on the completion of individual tasks. Computer-aided calculation methods were applied in specific areas only. For example, the application of the first computer-supported two dimensional drawing systems in the late 1970s represented a fundamental change in the way vehicles were designed.

At that time, full-vehicle development processes had a duration of about 6 years and included three prototype phases [14]. However, the integration of virtual, computer-aided methods into automotive development led to significantly revised processes. In the middle of the 1980s, computer-aided design and simulation methods began taking over engineering tasks that had formerly been done via hardware-based development. The pre-calculation of vehicle structures, durability and acoustic behavior in the layout phases supported product optimization without hardware tests, and growing design and calculation possibilities increased the influence of virtual computer-based development steps. Focusing on the design process, the introduction of parametric-associative methods pushed virtual modeling intensively.

In the closing years of the twentieth century, integrated CAD/CAE processes enabled network-based development in automotive engineering. Virtual prototypes were generated, which replaced at least one physical prototype generation. The application of virtual engineering in full-vehicle development fostered worldwide collaboration by bringing together partners and markets from different countries and regions. Today, a full-vehicle development project takes less than 3 years, and the trend is

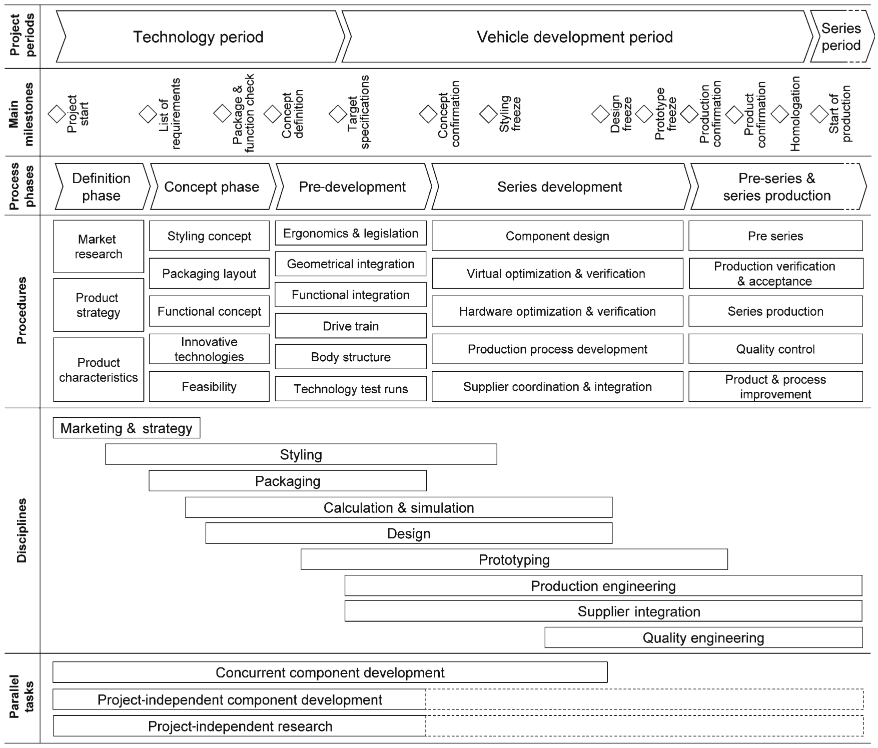


Fig. 1.11 Stages in a typical state-of-the-art full-vehicle development process

moving towards an additional decrease. Of course, since the product quality must be maintained, numerous virtual and physical test and acceptance procedures are applied throughout the entire development process.

Although the exact sequence of automotive development varies from one manufacturer to the next, in principal, the main fields of engineering can be found regardless of the specific company. Figure 1.11 shows a typical state-of-the-art automotive development process. The diagram displays the cumulative result of a detailed development process study that examined several car manufacturers and globally operating suppliers. The process depicted can be understood as a synthesized procedure which includes the main sections and milestones of a modern car development cycle. The nomenclature sometimes differs, but the major modules can be found in most development processes at car manufacturers around the world.

Therefore, the processes displayed serve here as a basis for the description and discussion of modern development cycles in the automotive industry, but do not reflect an actual development procedure from a specific company. The scheme describes the development sections of a new car model, without consideration of variants or model releases. In addition, concurrent component development and parallel project-independent research and development are not examined in detail. To support a clear illustration, the diagram is divided into different types of description. A general structure is provided by the division into three main project periods, the technology period, the vehicle development period and the series period, and an overview of the most important gates during each section is given by the definition of main milestones along the process progression. The process phases and the corresponding procedures describe the workflow, while a selection of comprehensive working disciplines points to the application of department-oriented tasks, which reflect the complex and extensively inter-linked procedures.

1.2.1 Project Periods

An automotive development process can be divided into three main periods. The technology period includes the product characteristics formulation, the vehicle concept and some initial pre-development steps. The technology period leads to the generation of a detailed target definition. This includes the definition of the vehicle layout and the drivetrain configuration from a series of production-related viewpoints. At the end of the technology period, variations pertaining to the drivetrain, transmission, special technical features and car body are considered and described, and innovations are defined and approved. Market-related research (from former models and/or competitors) is considered. The vehicle development period starts with a detailed planning of the subsequent vehicle-oriented development steps. At the beginning of this period, manufacturing-related tasks are considered, and suppliers are integrated into the development. Parallel to the design process, production engineering is performed and the influences on the detailed product design are considered. All production-related requirements have to be implemented before the concept confirmation. Technology test runs in the first section of the vehicle development period have to evaluate and optimize the interactions between the different modules in a car.

After finishing the pre-development phase (milestone: *Concept confirmation*), the series development phase takes the vehicle concept and develops the product until market-ability. During this period, the amount of production-related engineering tasks and working packages continues to increase. In addition, the influences of distribution and marketing are considered, while the product development itself decreases. There are a large number of linked processes in series development, which require an efficient project management. At the end of this period, the production, product confirmation and the homologation are completed, and the vehicle development period ends with the milestone *Start of production*.

As the consideration of manufacturing-related tasks increases, quality engineering is incorporated in the development process. Parallel to the aforementioned periods, additional responsibilities are taken into account. Project-independent research includes new technologies, strategic brand innovations and other fundamental research work. Project-independent component development is performed in the area of engine and transmission development for different vehicle types, platform development and/or development of components for the entire group. Project-related concurrent component development is performed to ensure the on-time functionality of delivered modules. This includes such tasks as parallel development, test and optimization of engines, and transmissions, which help to manage the working effort and time in the full-vehicle development process. In this way, different groups of departments and specialists work on new engines, transmissions, electronic devices and others. After completing the concurrent engineering processes, the tested and approved modules are implemented into the full-vehicle development process just in time.

1.2.2 Phases of Automotive Development

Besides the separation into three main periods, the product-generation process of a car can also be divided into process phases. Figure 1.11 shows the sequence of five main phases: the definition phase, the concept phase, the pre-development phase, the series development phase and finally the pre-series and series production phase.

The Definition Phase

The definition phase encompasses an initial characteristics estimation and evaluation of the car model which is to be developed. This phase is supported by far-reaching market research and trend prognoses to predict the requirements in target markets during the years of the projected product life cycle. In addition, the definition phase of a new car has to consider the overall product strategy of the manufacturer and boundary conditions related to the economic and financial situations. The product characteristics are set down in a *List of requirements*, which serves as a start-up schedule for the ensuing concept phase. Based on the definition phase, this list includes a description of the car classification (mini, compact, sedan, SUV, van, sports car, etc.), the main dimensions, the targeted driving behavior and other factors. In this phase, initial styling proposals are also submitted and discussed in the context of future styling trends and competing products.

The Concept Phase

The *Concept definition* milestone, one target of the concept phase, includes a detailed description of the vehicle concept itself. At this milestone, a rough approximation in terms of styling, packaging and functionalities is delivered, which provides an overview of the principal vehicle setup. Frontloading techniques support the working modules in the concept phase to enable the integration of knowledge from former project and research work.

One main working field in the concept phase is the execution of the styling process within given technical boundaries. In the initial phase, the generation of styling sketches and car body shape concepts is supported by initial packaging studies and ergonomic drawings. In later steps, the car styling has to take into account technical and legislative requirements in detail, such as space for the drivetrain and components, aerodynamics, passenger, pedestrian and crash safety. The task of the stylists is to find optimal solutions that meet the requirements of technical, economic, market and fashion trends. Benchmark studies assist in the evaluation of market trends and customer demands. In addition, typical brand characteristics have to be implemented and enhanced. Therefore, numerous styling proposals are produced to serve as a basis for discussions within the development team and in directive cycles.

The styling process is an important step in the concept phase, and even in the entire development process, because the car styling has a significant impact on customer perceptions and thus on the purchase decision. In modern vehicle development, computer-aided styling software (CAS) often supports the creation of exterior and interior styling surfaces. The styling software packages use several complex algorithms for the definition of high-quality surface models, which provide the basis for automotive styling development. The geometrical data are used for styling-related evaluation and representation within a virtual reality (VR) environment and serve as input information for subsequent engineering processes (e.g. vehicle packaging, ergonomic development, component design, module design). In most cases, styling data are transferred into the CAD-environment via neutral data formats.

In addition to styling, the packaging layout is another important task in the concept phase. It is carried out using digital mock-up (DMU) processes, which are based on a virtual assembling of the components and modules of a car. The geometrical description of these components is delivered by CAD and CAS. The working field of packaging layout handles the requirements of car module placement, such as drivetrain, tank, climate control, suspension, and others. The primary issue of the packaging investigations is the consideration of passenger and luggage space. In modern development processes, the packaging is built up around the passengers. In this context, the seating position plays an important role. Whether the passenger is sitting in a sports car, a sedan or an SUV makes a significant difference.

Besides the car class, brand-specific seating positions and space characteristics must be considered. Once the seating position, passenger's space requirements (e.g. head, elbow and knee clearance) and the luggage space have been designed, the vehicle packaging concept can be built up. This includes the placement of main modules, the review of compliance with legislative requirements and the definition of technical-based outline surfaces. All technical requests have to be double-checked with the styling surfaces of the car body to find an optimal solution that satisfies technical requirements without interfering with the car styling.

Alongside with packaging-relevant investigations, initial functional studies are performed. These studies examine the technical interaction of components and modules and also verify ergonomic functionalities. In this phase, technical modules are prepared and reviewed for their suitability for implementation into the vehicle concept. Technical modules include engine and drivetrain components, suspension and other supplied units. In addition, the functional concept includes early functional layout studies of components that must be newly developed, for example an initial estimation of the basic door kinematics and the window lifter mechanism (see also Sect. 3.13, p. 235). The package and functional check milestone includes a brief description of the vehicle architecture in terms of styling and technical requirements.

Of course, one main task in the concept phase is the consideration of innovative technologies, such as the implementation of new propulsion concepts (e.g. hybrid, electrical drive), new transmission systems (e.g. CVT, double clutch transmission), new vehicle concepts (e.g. convertible-coupe, micro city vehicles, one liter car) and others. Whether these new technologies influence the entire vehicle architecture (e.g. completely new vehicle types) or simply represent a part of the car (e.g. new transmission type) makes a significant difference. In both cases, the new technology has to be checked against the list of requirements for the new product, and its further development and applicability to the new car being developed must be verified. This can include concurrent, innovation-related virtual and hardware-based development cycles.

Finally, the concept phase concludes with a feasibility investigation of the vehicle concept. This section includes a description of different styling proposals that have been reconciled with the vehicle packaging. In addition, the functional concept is defined, and new technologies are approved for their principle usability in the new car model.

The Pre-Development Phase

Based on the concept phase, the pre-development phase includes a detailed definition of the vehicle layout taking into account all ergonomic and legislative boundary conditions. The packaging studies are enhanced by complete assembly and placement investigations, as well as functional optimizations. In addition to the definition of technical-related characteristics, comprehensive product decisions are made. This includes the integration of the new model into platform strategies, building block systems and the definition of model derivates. In many cases, new cars are based on predecessor models and carry over numerous technical features and components. In this case, the pre-development phase includes an assessment and evaluation of existing modules regarding their suitability for the new product.

The geometrical integration in the pre-development phase includes a detailed definition of the packaging, as well as of other geometrically based boundaries, whereby the knowledge from earlier models plays an important role. Brand-specific characteristics, such as the seating position, the drivetrain configuration, and suspension components, are implemented into the vehicle layout, which leads to a full-vehicle concept. This full-vehicle concept is represented as a detailed 3D CAD Model. In several steps, different specifications are adjusted in consultation with various development departments and decision makers. Styling is developed and adjusted in parallel, whereby the adjustments are performed in each department, focusing on both the styling and engineering.

In a parallel process, the functional integration verifies the technical functionalities and the interaction of all components and units. The ergonomic influences on the functionalities are investigated through ergonomic studies and simulation, for example entrance and seat position optimization, accessibility of switching devices, etc. The linkage of styling and engineering is performed in several steps, with a repeated data transfer between the departments. This procedure can sometimes be difficult, particularly when the wishes of stylists and engineers conflict. The functional integration also includes other areas, such as chassis and drivetrain, although some of these modules and components are normally developed separately because of their use in different car models. The drivetrain module can be seen as a delivered unit, but it must be considered in the vehicle setup from various viewpoints. For example, the engine packaging, an appropriate connection to the cooling system, the integration of the tank system, the development of the exhaust system (including exhaust gas aftertreatment), and the application of sensors and control units have to be adapted and approved for the new car model.

One important task of the functional integration is the full-vehicle layout related to driving performance and fuel consumption. The calculations conducted at this early stage are often based on former car models or by substituting simulation procedures because the necessary data are not yet fully available for the new model in this development phase. In many cases, there is an insufficient amount of data available for the full-vehicle simulation of future concepts. In these cases, a smart integration of existing data combined with pre-editing methods can improve the efficiency and accuracy in this important development phase in order to obtain the data necessary for simultaneous engineering.

Parallel to the general vehicle architecture development, the car body structure is defined and optimized. In the early phase, a rough estimation of supporting elements is defined and adjusted with styling and packaging. This initial structure is calculated in view of major load cases, such as different crash scenarios, stiffness and maximum stress. In many cases, the initial body structure is based on experience from former projects and/or benchmark studies. As pre-development proceeds, both the body structure design and its simulation begin to incorporate influencing factors to a higher degree. Detailed stress and durability calculations, full-vehicle crash simulations, weld spot pre-dimensioning and other production-related targets are considered and implemented.

At the end of this phase, a completely calculated design proposal for the series development is generated. In the case of a new body concept (e.g. new lightweight materials or design approaches), additional aspects have to be investigated and taken into account. In the last section of pre-development, a final styling concept is chosen and verified. This styling concept fulfills all engineering-based requirements, such as packaging, function, legislation, ergonomics, and aerodynamics. In most cases, a 1:1 hardware-styling prototype is built up to confirm the decision. Once the vehicle styling has been fixed, no subsequent modifications are allowed, due to the significant impact on the subsequent engineering steps. Parallel to the virtual development, different hardware test runs are conducted for both components and prototype vehicles to verify the functionalities.

In the course of investigating and assessing different technologies, the *Target specifications* milestone can be defined. This specification list includes a detailed description of the product characteristics and deliverables. The target specification list, which is based on the list of requirements, includes some adjustments and detailing resulting from the progress of the concept phase and the pre-development phase. The pre-development phase is completed with the *Concept confirmation* milestone. At this milestone, all product characteristics related to the target specifications have been fulfilled and verified. Modifications to the vehicle concept after the concept freeze have a significant impact on the further process timeline and the project costs. Due to the substantial influence that the initial development phases have on the product development cycle success and even on the product itself, it is necessary to apply highly flexible and efficient strategies. It is essential to use flexible tools, efficient methods and experienced engineers to identify the best product characteristics and technical solutions. In these early phases, the main facts of the entire product life cycle are defined.

Series Development

The series development phase includes a realization of the concept that takes into account process and production-related viewpoints. All of the engineering procedures in this phase are influenced by manufacturing-related boundary conditions. Thus, both the virtual and physical developments are performed in close cooperation with production engineering and with the supporting suppliers tightly integrated. In the series design process, the geometry of the components is described with a high accuracy within a CAD-environment. In this phase, the level of detail is significantly higher than during the concept phase or the pre-development phase. This precise generation of virtual models that take manufacturing processes into account requires special design methods and strategies. At present, the CAD models for series development feature a lower level of parameterization and thus less flexibility than the models created during the conceptual project phases.

The generation of new design methods, which enable a direct model transfer from highly flexible project phases into the rigid series development environment, offers substantial potential for increasing the efficiency of the entire development process. Besides component design, the virtual optimization and verification of parts and product configurations represent important steps in the series development. Digital mock-up procedures check and improve the interactions between modules in terms of functional aspects, assembly and tolerancing. Far-reaching simulation procedures ensure a failure-free performance and interaction. In the case of the vehicle body, structural investigations, virtual crash calculations and durability simulation are performed in detail under consideration of component design, materials and the applied connecting technologies. Especially in the case of modern lightweight body design, virtual preservation of failure-free operation during the product lifetime represents a significant challenge for all parties involved.

During series development, several concurrently developed components and modules are implemented into the full-vehicle architecture. Thereby, engine and drivetrain modules, brakes, suspension and electronic systems are included into the vehicle setup. One significant advantage of concurrent component and module development is the parallel generation and verification of enclosed units. These modules are optimized and tested in terms of their theoretical failure-free operation. During the integration process, they have to be verified in terms of suitability and failure-free operation within the complex vehicle system.

One example of the integration of concurrent developed modules is the engine unit. The development of internal combustion engines is a complex challenge and has to take into consideration not only engineering requirements, but also customer and legislative demands. Thus, a reduction of fuel consumption and exhaust emissions, increasing performance requirements, the implementation of alternative drivetrain concepts (e.g. hybrid power trains) and a high cost pressure influence the development of new engine technologies. The engine development process itself is divided into several sequences: design, mechanical and thermodynamic simulation, test bench and on-road application, and production-related development. Since this chapter focuses on the full-vehicle development process, it will not go into detail about engine or component development.

One important stage in the series development phase includes the mass calculation and mass management task. This task is addressed via virtual optimization and verification, as well as in hardware-related steps. The reduction of weight is an important target for the development of new models because of the direct influence of the vehicle's mass on driving dynamics, fuel consumption and exhaust gas emissions. Parallel to the virtual development, hardware optimization and verification take place. All the virtual development results are verified and fine tuned in the course of prototype test procedures. Hardware optimization and verification includes test cycles of components and modules, as well as work on full-vehicle prototypes on test benches and under different road conditions. Examples of hardware tests include tests of new propulsion concepts, engine and transmission tests, final aerodynamic and air flow (cooling) studies on hardware models, functional test benches for closures, suspension and brake tests, electronic test procedures, software tests, haptic and ergonomic studies, tests of sealing concepts, and many more.

Besides product-related hardware tests, manufacturing-related hardware optimization loops are performed in the course of the production process development. In addition, virtual manufacturing and plant engineering includes the simulation of production and assembly procedures to support the optimization of in-house processes, as well as integration with the supplier. One result of the series development is the complete generation of 3D CAD models and 2D workshop drawings. These product data are linked with different processes of data organization, supply and manufacturing. Parallel to component design, tools are designed and calculated in-house at the manufacturer, as well as in cooperation with suppliers. Detailed specifications of product quantity and quality, project schedule, logistics and manufacturing processes are defined.

When considering the main milestones of the series development phase, the dependency of product development and production engineering becomes evident. The exterior and interior styling process ends at the *Styling freeze* milestone. This target includes the complete definition of the vehicle styling under consideration of technical, production-related and, of course, aesthetic viewpoints. The *Design freeze* milestone marks the end of the design process, including all supporting calculation and simulation procedures. At this milestone, the product-influencing factors of production engineering have to be implemented. At the *Prototype freeze* milestone, the vehicle setup is confirmed across an extensive range. This includes mechanical and electrical confirmation, as well as hardware and software verification. The last milestone in the series development is *Production confirmation*. At this stage, the product is ready for production. Comprehensive workload in the area of production planning and control is frozen and approved, which includes the administration and organization of manufacturing-relevant data and procedures. The subsequent steps have to finalize the production process itself.

Pre-Series and Series Production

The production of the new car model starts with the pre-series. In this phase, although the series tools are used, the process is not performed on the ultimately planned time schedule. During the pre-series, the tooling and assembly procedures are tested and evaluated. This is performed with a reduced production rate to enable a detailed check of each individual step. Computer-controlled procedures are reviewed, and human resources are trained. Both the product itself and the production process are tested in terms of the quality guidelines. The *Product confirmation* milestone marks the acceptance of the product, including all aspects of development, production and process quality.

Due to the different market requirements, the homologation of the vehicle is a long-term process. Therefore, the homologation is carried out with pre-series vehicles or series-near prototypes. Of course, these vehicles have to fulfill the homologation-related specifications of series vehicles. The *Homologation* milestone denotes the end of the legal approval procedure. Product development ends with the *Start of production* (SOP) milestone. From this stage on, the manufacturing process is performed in the predefined time steps, and the targeted production volume is achieved. The delivery of supplied components and modules, the arrangement of the assembly line and the quality control function in the specified order. In most cases, suppliers have started their production a bit earlier, to ensure a failure-free start of the assembly process. During series production itself, continuous quality checks guarantee the predefined product features. Product and processes are checked continuously, and improvement steps are implemented into the production procedure. During the life cycle of a car model, further development and model upgrading include the implementation of new technologies (e.g. new engines and transmissions, new safety or comfort equipment) or slight styling modifications. These modifications are developed in parallel to the series production and implemented into the manufacturing process.

Once on the market, the product is maintained by a different kind of administration. Besides spare part delivery, selling centers are involved in the quality and improvement process. Feedback from the market provides important information for the current series, as well as for future development. Therefore, customer feedback data are collected and evaluated at markets around the world and collected for a detailed evaluation.

1.3 Application of CAD in Automotive Development

CAD is one of the central disciplines in modern automotive development. The efficient computer-aided creation of geometrical models provides the basis for a broad field of concerned engineering processes. The comprehensive CAD-based representation not only includes geometry data, but also provides extensive information about the product structure (e.g. bill of material - BOM), geometrical and functional interactions of components and modules, as well as much production-related information.

Section 1.2 introduced the sequences of a state-of-the-art automotive development process, including all its sub domains and areas of operation. A considerable number of working fields in this process are related to virtual product development and thus to the generation, modification and use of CAD-based information. With the goal of illustrating the main development targets and influencing factors, Fig. 1.12 shows several aspects of automotive engineering processes ordered in relation to six main areas. Economic-related aspects include the product strategy, influences from the market and of course cost management. Process-related aspects go hand in hand with production engineering, supplier integration and spare part management. Finally, traditional engineering-focused working areas contain the development of components and modules, as well as the consideration and implementation of full-vehicle-related technological parameters.

Since they are responsible for the generation of product data geometry within a virtual environment, CAD-processes are involved in development processes from the beginning. Market and product-strategy-related decisions are often influenced by fashion trends and lifestyle. In this way, a computer-aided creation and visualization of initial styling models plays an important role during assessment processes for the definition of customer requirements and market trends. In subsequent steps, the initially created styling surfaces serve as a basis for far-reaching optimization cycles, including several modifications and adaptations. The application of computer-aided styling (CAS) software supports the efficient creation of styling surfaces, data backup processes and evolutionary modifications of existing models.

Since cost calculation has an important influence on competitiveness, considerable effort is invested in the prediction, computation and evaluation of a wide range of product and production development processes in terms of their potential for cost savings. The provision of current information (including geometry, functions, materials, manufacturing and assembling related data), the implementation of components and modules from other models (COP), and providing information about the product structure throughout the entire product creation cycle offer significant support in the determination of cost-related aspects. Product data management (PDM)-based

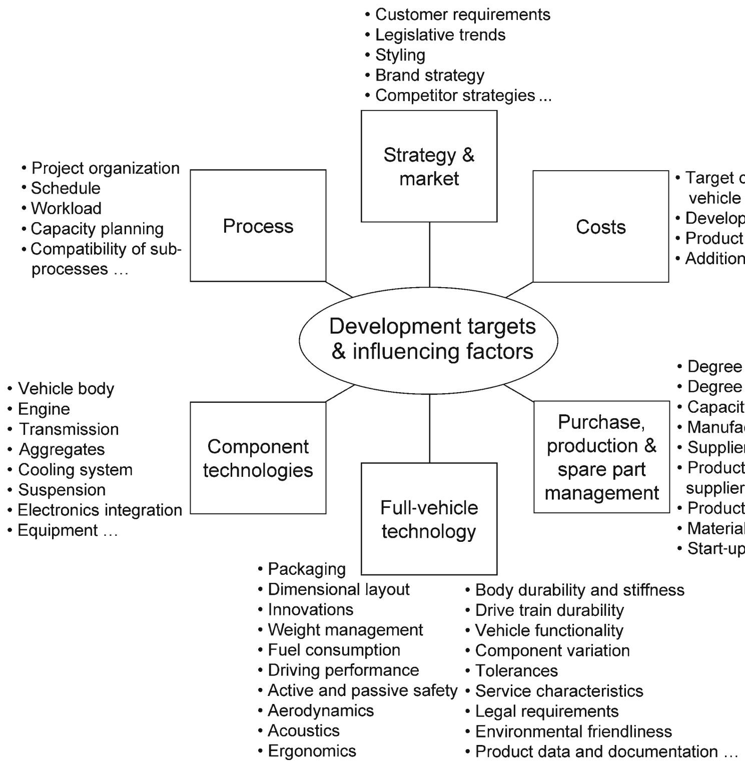


Fig. 1.12 Development targets and influencing factors of automotive full-vehicle engineering processes [15]

organization of all the involved information supports a direct access of the concerned departments and working areas to cost-calculation-relevant information.

Production planning and manufacturing engineering, including the integration of supplier and logistic processes, involves automotive DMU structures for the computation of mounting and assembly procedures, the verification of production lines, and the development and evaluation of automated manipulation and transportation. Derived from CAD data, the assembly simulation addresses the optimization of mounting procedures regarding sequences, collision detection and ergonomics investigation of operators. In addition, the management of variants supports an efficient confirmation of production processes in the case of product modifications or variations.

Full-vehicle development goes hand in hand with the development of components and modules. These processes are traditionally known as product development and include all tasks necessary for the creation and verification of all geometrical and functional product characteristics. Whereas the development of component technologies is often focused on restricted functionalities, full-vehicle development processes have to consider the complete product structure, including complex geometrical and functional interactions and requirements. Both disciplines are based on virtual product development, as CAD plays a key role in the definition of both the product structure and the product creation. Due to their central position in automotive engineering, CAD data are managed in an extensive PDM structure and provided for several parallel and subsequent calculation and simulation processes for functional layout (e.g. kinematics simulation), structural dimensioning (e.g. body crash simulation) or other optimization and verification processes (e.g. aerodynamics simulation, weight computation).

Finally, process engineering handles the development and optimization of the applied processes, beginning with a structurization of the complete vehicle development (Fig. 1.11, p. 12). The derivation of sub-processes for detailed development steps and for the integration of external engineering partners and suppliers plays an important role in the integration of conception, design, calculation and simulation. An optimized interaction of different departments, including differing development disciplines, provides the foundation for successful product development. A special focus is placed on the correct planning of different sequences and development steps, in which the transfer of data plays an important role. Process engineering has to ensure that all required information is available just in time for different processes (e.g. geometrical information regarding components developed in design processes has to be provided for subsequent stress and fatigue simulation). In modern automotive development, the planning of development sequences and workflow faces challenges from simultaneous engineering and frontloading approaches, which are based on development steps that are partially performed in parallel and on the shifting of knowledge-based, product-related decisions into early development phases. The complex interaction of CAD, CAE and knowledge-based engineering has to be planned carefully because of its importance for successful product development.

References

1. Hirz, M.: An approach of multi disciplinary collaboration in conceptual automotive development. Int. J. Collaborative Enterp. IJCEnt **2**(1), 39–56 (2011)
2. MAGNA STEYR Fahrzeugtechnik AG & Co KG: date of access: 2009–11-10. http://www.magnasteyr.com
3. Bols, U.: Die berühmtesten deutschen Autos. Podszun, Brilon (1990)
4. Mercedes-Benz Classic Archive, Daimler AG, Stuttgart, Germany
5. Courtesy of Ford Motor Company - Gross Point Public School System: date of access: 2010–05-11. http://www.gpschools.org
6. Courtesy of Ford Motor Company - Motorwayamerica: date of access: 2009–10-05. http://www.motorwayamerica.com
7. Bugatti Type 57 Atlantic, photographed by Martyn Goddard, England. http://www.martyngoddard.com
8. Riekert, P., Schunck, T.E.: Zur Fahrmechanik des gummibereiften Kraftfahrzeugs. Ing. Arch. **11**(3), 210–224 (1940)
9. Opel Classic Archiv der Adam Opel AG, Rüsselsheim, Germany
10. Volkswagen AG: date of access: 2013–03-29. http://www.volksgenag.com

11. Volkswagen Scooter 3-Wheel Microcar Press Kit, Volkswagen United States Inc., Michigan, USA 1986
12. Nissan Austria: date of access: 2009–10-05. http://www.newsroom.nissaneurope.com/at/deat/Home/Welcome.aspx
13. Jürgens, G., Hirz, M., Bader, M.: Entwicklungsmethodik. Lecture script at Graz University of Technology, Graz (2011)
14. Sorsche, J.H.: PKW-Konstruktion. Lecture script at the University of Stuttgart (1989)
15. Hirz, M.: Advanced Computer Aided Design in Conceptual Automotive Development. Habilitation Thesis at Graz University of Technology, Graz (2011)

Chapter 2
Overview of Virtual Product Development

This chapter provides an overview of product development with a focus on (but not limited to) automotive engineering. After a general definition of mechanical product development processes, the main terms, definitions and a selection of methods of virtual product development are introduced. This includes the history of CAD, CAE and PDM, a classification of fundamental methods of product modeling, and a short description of typical CAD-CAE process chains and product data management tasks in automotive engineering. The chapter closes with a brief introduction to the concepts of collaborative product development.

2.1 Development of Mechanical Products

Product engineering processes cover all operations for the development, manufacturing, use, servicing and disposal of products. Product development manages the creation of the product itself, under the consideration of different boundary conditions. In this way, product development processes include all of the operations necessary to bring a new product to market. This includes the idea generation, the concept phase, product styling and design and detail engineering, all of which are conducted in the context of market research and marketing analysis.

Figure 2.1 shows a typical product life cycle sequence. Product research encompasses both basic research work and product-specific investigations. The product planning stage is often embedded in the concept phase. In this first development phase, the main characteristics of a new product are defined and evaluated. After the concept phase, the series development includes the styling, the design and a detail engineering phase. The extent of the product testing stage depends on the product type. In the case of automotive engineering processes, the testing stage consists of far-reaching test and optimization work. Next, production-related processes are developed and implemented. After the start of production, the product manufacturing phase represents the last stage in the product creation cycle. The product distribution, use and liquidation (eventually recycling) stages take place in the market. During

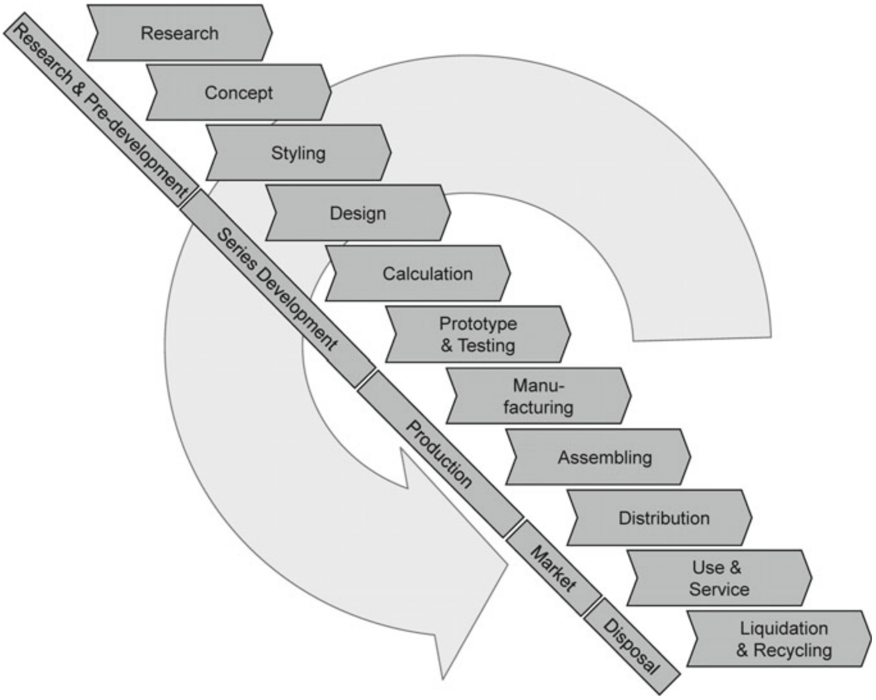


Fig. 2.1 Stages in a typical product life cycle

these phases, marketing-relevant factors, service and customer support have to be considered.

The German VDI 2221 guideline describes the stages in the development of mechanical products [1]. This standard process focuses on product development and does not include the entire life cycle. Regardless of the specific development tools and methods that are applied, the development process for mechanical products can be divided into five main stages (Fig. 2.2). In the first stage, the product requirements and specifications are defined. In the automotive industry, the description of product characteristics is supported by far-reaching market studies, research into constantly changing customer demands and an evaluation of future legislation-based boundary conditions in target markets. The cost-intensive development phase and production planning for a new car require a careful preparation of new automotive technologies or models. Typical car models have a production life time between 6 and 10 years, although some models are sold for more than 20 years. Adding a development time of about 2–4 years, a new automotive product (including new technologies) has to be competitive on the market for more than 10 years from the start of development. For this reason, it is very important to consider market tendencies and legislation-based trends in the very early phase of product development. A miscalculation of product characteristics can have negative effects on the economic success of a product, and

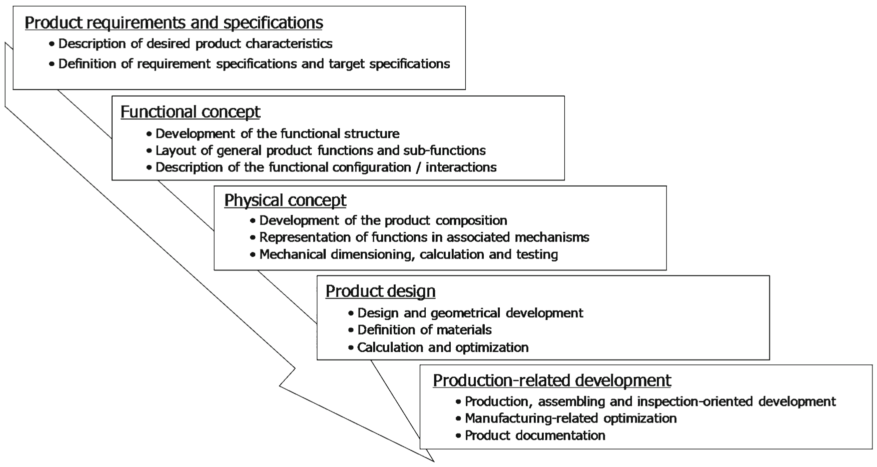

Product requirements and specifications
• Description of desired product characteristics
• Definition of requirement specifications and target specifications
Functional concept
• Development of the functional structure
• Layout of general product functions and sub-functions
• Description of the functional configuration / interactions
Physical concept
• Development of the product composition
• Representation of functions in associated mechanisms
• Mechanical dimensioning, calculation and testing
Product design
• Design and geometrical development
• Definition of materials
• Calculation and optimization
Production-related development
• Production, assembling and inspection-oriented development
• Manufacturing-related optimization
• Product documentation

Fig. 2.2 Development process for mechanical products, according to [1]

consequently, due to the immense financial investment, undesirable effects on the car manufacturer.

The description of product characteristics provides the basis for the definition of the requirement specifications and target specifications of a new model. The requirement specifications include a complete description of the new product characteristics. The project initiator is responsible for the requirement specification list. In the case of automobile development projects, this can be the management board that has ordered the development of a new car model. Requirement specifications take into account functional and non-functional specifications. They include detailed information about the requirements of product design and describe the desired behavior of a product in terms of its operation. Supplemental information, such as quality standards and manufacturing-related boundaries, complete the definitions of boundary conditions for development. The target specifications, on the other hand, define detailed approaches for the development process of the product as a function of the requirement specifications. In this way, the target specification list consists of precisely defined solutions and derived working packages for the completion of the tasks that are defined in the requirement specifications.

The second stage of the development process includes the functional concept of the new product. The new technologies implemented are defined and assessed in terms of their functional configurations and interactions. A general product layout describes the definition of functions and sub-functions. All of the interacting requirements of a new product are checked in view of the requirement specifications and other influencing boundary conditions, such as legislation-relevant tasks or production-related influences. In the case of a new car, the packaging concept plays an important role in the functional development. Besides the general layout of a car model, several technological characteristics are defined in this phase, which requires the consideration of a broad variety of influencing factors (e.g. driving performance, vehicle safety, comfort, durability, and legislative/environmental requirements). New technologies in automotive components, such as new safety equipment or environmentally friendly propulsion technologies, are implemented and verified in terms of their general functionalities within the full-vehicle system.

The third stage tackles the physical concept. This stage covers the definition of the product composition. The development of new functions is carried out in associative mechanisms. Besides simulation work, this stage includes the mechanical dimensioning and calculation of components. In automotive development processes, the physical concept defines the vehicle body structure layout in consideration of crash and stiffness requirements. In addition, basic requirements of the new car concept are addressed, such as driving performance, fuel consumption, vehicle mass, and estimated values of driving dynamics. In the case of new drivetrain concepts (e.g. electric driven vehicles), this stage includes an estimation of the performance requirements and energy density, as well as the battery layout and capacity. These factors influence the battery mass and therefore the vehicle mass, center of gravity, driving characteristics and other factors. Together, the functional concept and the physical concept form the concept phase of a new product. In automotive development processes, the concept phase covers complex procedures that take into account a wide variety of influential boundary conditions and factors.

The fourth stage handles the product design phase, which is directly dependent on the product concept phase. The geometrical development of all components has to consider the assembly of the product and the interactions of components, as specified in the concept phase. Based on knowledge from the concept phase, the product components are modeled in detail and optimized. The materials are defined, and the boundaries for the production planning are derived.

Finally, the last stage consists of the production-related development. This phase goes hand in hand with the design process because manufacturing boundaries often influence the design of components. Thus, the production, assembly and inspection-oriented development and the manufacturing-related optimization (including supplier integration) interact with geometry creation and calculation processes. In former times, these sections were performed separately, but nowadays, a close information transfer supports an effective product development. As defined by VDI 2221, the final step is the product documentation, which includes all product-relevant information, as well as manufacturing data (e.g. workshop drawings and assembling guidelines).

The development process of mechanical products stipulated by VDI 2221 does not consider the development tools and methods applied. In principle, the standard is valid for the development of all mechanical products, regardless of the manpower, machines and methods deployed. While standardized processes of product development provide a framework, in the automotive industry, and especially in conceptual development, significant additional specification and development of new models, methods and tools for conceptual design are necessary. Since the late eighties, development processes in the automotive industry have been supported by computational methods and strategies. The trend is definitely going in the direction of integrated virtual development that supports the complete generation of a new car. Taking its cue from early pioneers in aeronautical engineering disciplines, the automotive industry

Fig. 2.3 Design office circa 1900 [2]

has also played a leading role in the development of software tools and methods for improving the virtual generation of new products, new technologies and manufacturing processes. Within the boundaries of a fixed development time and cost reduction, virtual engineering methods play an important role in the optimization of technologies and products through design, simulation, calculation, organization, production, distribution and other important tasks.

Figure 2.3 shows a design office around 1900. At that time, the main development tools were a pencil, a ruler and a lot of paper. Although today's development offices are characterized by the wide-ranging application of computational tools, human beings are naturally still the most important factor in the creation of new products.

2.2 Virtual Product Development

Virtual product development includes all IT-supported, virtual product-model-based processes for the generation of a new product. Virtual product models are used to perform optimization and testing procedures in a virtual environment with the goal of saving development time and costs, while simultaneously increasing the product quality. Depending on the categories of development applied, there are different types of virtual models. They can include market-relevant or business-case-relevant data (economic models), workflow-oriented information (process models), technical characteristics and descriptions (functional models), and geometry information (design models). Depending on the different requirements and characteristics of diverse disciplines, both the virtual models and the results generated can differ significantly. In general, the disciplines of virtual product development can be classified into main groups [3].

CAD	...	Computer-aided design
CAS	...	Computer-aided styling
CAE	...	Computer-aided engineering
DMU	...	Digital mock-up
CAM	...	Computer-aided manufacturing
CAQ	...	Computer-aided quality assurance
CAT	...	Computer-aided testing

Depending on the type of product to be developed, several additional disciplines are applied within the main groups mentioned above, such as computer-aided software engineering (CASE), which is used in the development of IT-applications or mechatronic products with implemented IT-supported functionalities.

Efficient virtual product development is based on an effective interaction and integration of the various systems applied to enable a close cooperation with all participating departments and development partners. Virtual product development in the automotive industry uses a wide range of product models, which are connected by global data management systems. Data management is organized in different structures, depending on the requirements of the specific stages in the product development and life cycle. Different terms are used to describe data-management-related processes and functionalities (see Sect. 2.2.3 for an overview of product data management in automotive engineering).

Figure 2.4 shows the historical development of CAD, CAE and process-related management, with a focus on their application in automotive development.

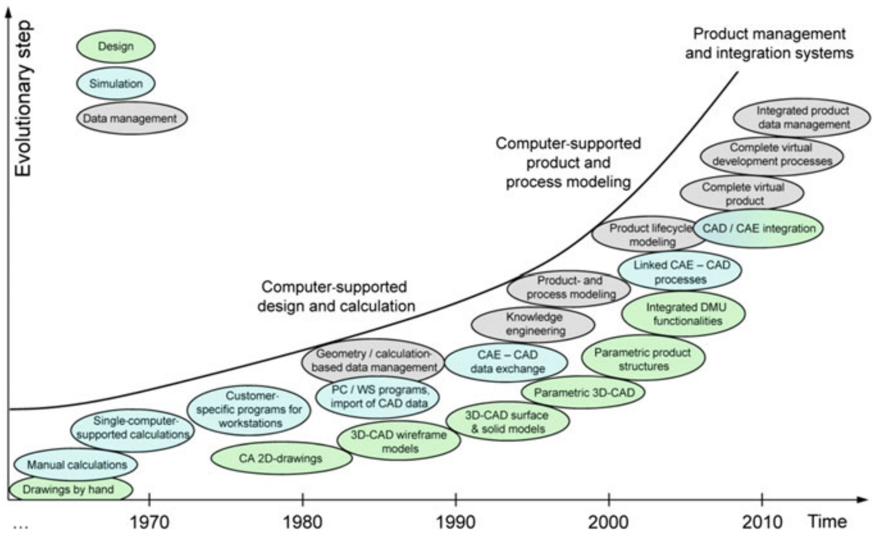

Evolutionary step
Design
Simulation
Data management
Product management and integration systems
Computer-supported product and process modeling
Computer-supported design and calculation
Integrated product data management
Complete virtual development processes
Complete virtual product
Product lifecycle modeling
CAD / CAE integration
Linked CAE – CAD processes
Product- and process modeling
Integrated DMU functionalities
Knowledge engineering
Parametric product structures
Geometry / calculation-based data management
CAE – CAD data exchange
Parametric 3D-CAD
PC / WS programs, import of CAD data
Customer-specific programs for workstations
3D-CAD surface & solid models
Single-computer-supported calculations
3D-CAD wireframe models
CA 2D-drawings
Manual calculations
Drawings by hand
...
1970
1980
1990
2000
2010
Time

Fig. 2.4 Historical development of CAD, CAE and data management, based on [4–6]

The initial computation-supported functions emerged in the late 1960s. These applications enabled mathematical operations and calculations. The first commercial computer-aided drawing programs were used in the late 70s. These programs used simple functionalities for the generation of two dimensional drawings of technical products. Initial applications defined sketches using lines and circles, while later software included additional functions, such as predefined geometrical figures or the definition of axes, pattern or dimensions. In this way, it was possible to generate 2D product description and manufacturing-related workshop drawings. In the 80s, software suppliers began offering commercial calculation programs for personal computers and work stations. At that time, integrated simulation methods for broad applications were developed. Besides automotive manufacturers, automotive component suppliers also drove the development of product-specific simulation methods for detailed investigations of their products. Geometry-based and simulation-based product data were used in different fields, and this required the implementation of initial geometry-based and physically-based data management strategies.

The transition from 2D drawings to 3D models started in the early 80s, but commercially successful 3D CAD programs were first introduced about 5 years later. 3D CAD design changed the applied design methods significantly. Most importantly, the introduction of 3D surface and solid models resulted in the evolution of design methods from static, two-dimensional drawings in several views and sections to dynamic, three-dimensional virtual geometric product models. Besides a detailed and near-real-life representation of product geometry, these models included a variety of additional information and characteristics. With the help of 3D design, it was possible to integrate production-related knowledge or assembly-related information into the model. Die casting processes and forging procedures found their requirements displayed in 3D geometry models. For example, the design of die-cast parts included all the requirements of moulds and casting process. In this way, 3D geometry models were provided with draft angles and fillets based on the selected draft directions. Design engineers obtained knowledge from cast-model manufacturers that helped them take their requirements into account during the design process. This method enabled a direct derivation of the moulds from virtual 3D CAD models. Similar procedures were applied for the definition of forged components or for the programming of numerically controlled (NC) production machines.

Direct data exchange between design and simulation started in the 90s through the use of standardized neutral data exchange formats. In this way, it was possible to use geometry data defined in design software packages for the definition of product geometries in simulation processes. Imported geometry was used to generate meshes for finite element simulations, the representation of dimensions and inertia characteristics for multibody simulation, or for other types of calculations. At that time, initial modeling strategies supported the implementation of CAD data, material characteristics, load, restraints and other boundary conditions into an integrated simulation process. 3D CAD spurred the development of product-knowledge-related engineering methods. Information from successful projects was stored in CAD models and saved in templates or simplified databases to offer guidelines and basic data for subsequent projects. These methods enabled a direct transfer of virtual product-model-based knowledge and experiences from earlier projects into new tasks.

The parameterization of geometry data represented a significant evolutionary step in 3D CAD processes. Parametric-associative 3D CAD software separated the administration of geometry and its controlling parameters. A logical and precisely defined linkage of parameters and geometry in the geometry-creation process produced fully parameterized geometry models. The parameter-based control of geometrical model characteristics opened up a wide field of application for problem-specific design applications. Parameterized CAD programs offered additional functionalities, such as data interfaces, the integration of catalogue and knowledgeware functions, and the possibility of macro-based procedures. All of these functionalities characterize state-of-the-art CAD packages, which have come into use in automotive development.

The parameterization of geometry models in turn spurred a strong development of data management systems. Initially, product and process modeling, and subsequently product life cycle modeling, were essential for the organization of exploding data volumes. Powerful product data management systems (PDMS) supported the increasing integration of design and simulation processes. The trend in the software industry is definitely going in the direction of integrated packages, which combine parametric design software and simulation software in the same environment. Although this strategy reduces data interface losses, it has been criticized for the resulting reduction in directly compatible program platforms. Nevertheless, future intelligent geometry data exchange formats, which will be able to handle both geometry data and additional characteristic product information, should increase the efficiency of virtual product development processes significantly. In the case of integrated software packages, or in the case of communicating stand-alone solutions, virtual development will be extended to the entire range of product generation, starting from the concept phase, continuing in the different development steps (including manufacturing and production), supporting sales and aftermarket, and ending in the organization of disposal processes.

2.2.1 Product Models

Virtual product development processes are based on product data models, which are able to represent the specific product characteristics. Different applications call for dissimilar product models. In general, the primary methods of product representation can be classified as [3]:

- Geometric modeling
- Feature modeling
- Parametric modeling
- Knowledge-based modeling
- Structure representation
- Technical product documentation

In automotive development, the design engineering of products is performed using CAD. Modern CAD systems offer a wide variety of functionalities for 3-dimensional product creation, 2-dimensional drafting and the creation of components and assemblies. Besides geometry creation, modern CAD systems enable the definition of several additional product characteristics, such as material specifications, process-relevant data (e.g. for production), and product structure. Geometry creation is one important task in vehicle development. Therefore, state-of-the-art automotive development processes are often based on the geometry data of a 3D CAD/DMU master model. During the product definition process, this geometry-based model represents the current state of development and interacts with all of the simultaneously performed processes. This interaction includes geometry data export for the supply of CAE, as well as the import of data for modifications and advancements, which are delivered from simultaneously performed (CAD-external) operations.

The application of 3D CAD provides the basis for a three-dimensional description of the product geometry. Modern CAD systems offer the possibility of parametric geometry control, which can be used for manifold applications of integrated strategies. The geometry is built up through the combination of single components (parts) into an assembling structure. A systematic structure of the assembly in sub-products and main products based on the structures in real life brings the virtual geometry model close to the configuration of a physical product. Structuring tools include bills of material, product configurations, assembling simulation, and others.

In the related literature, the process of virtual product generation is divided into two main sections (Fig. 2.5), [7]. Virtual product development includes all tasks necessary for the creation of product geometry and the implementation of product characteristics. The virtual plant includes the development of all manufacturing-related procedures and simulations. In this phase, the operations of production are simulated and optimized within a virtual environment. In addition, the production development takes into account the implementation of supplier, logistics and controlling mechanisms, as well as financial aspects. Optimized virtual product generation processes are based on integrated virtual product models, which include the entire product description.

Virtual product development itself can be divided into three main phases. The first stage, 3D CAD design, includes the geometry creation based on product-specific features. These features can cover technical functionalities or production-related details, such as draft angles or fillets. The design process also handles the definition of materials and the product structure. The second stage contains the digital mock-up of product components, including tasks related to assembly and packaging. DMUs contain both the product structure and simplified geometric models of individual components or assemblies. DMU procedures calculate clash and assembly procedures and are used to check several geometrical interactions in a product structure, such as clearance or accessibility. In the third stage, the functional DMU (also called VMU - virtual mock-up) considers the functional integration of the product, including all of the features and functionalities necessary for a failure-free operation. These so-called virtual prototypes take functional and physical characteristics into account and enable functional simulations or calculations (e.g. kinematics, masses, center of gravity).

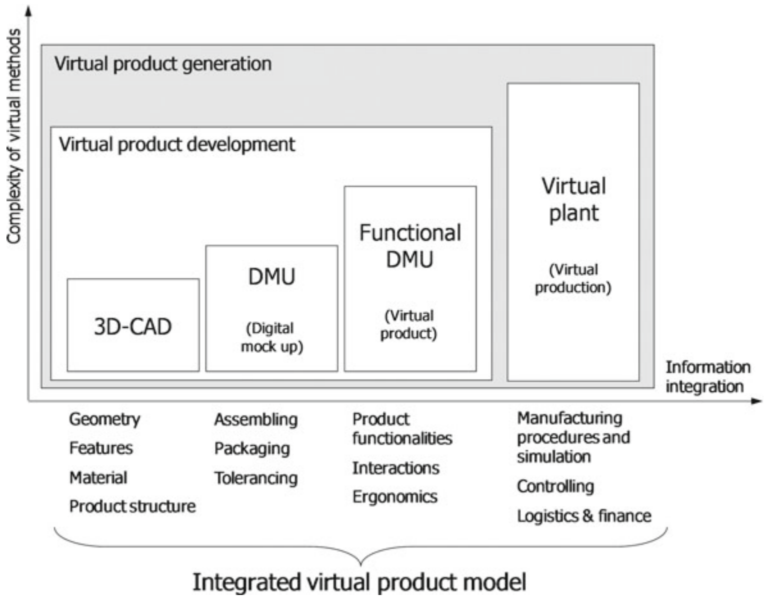


Fig. 2.5 Concepts of virtual product development, based on [7]

In automotive development, the placing of 3D CAD data at the center of virtual product generation is an efficient approach for the creation of an integrated development process. 3D CAD models of vehicle concept, styling, components and packaging serve as display units for each data status during the development project and supply all the other relevant processes with the required information. All geometry-based information is stored in the CAD model. In this way, it is possible to organize the product release updates and project progress steps via a centralized master model. The 3D CAD master model is in turn supported by a data management system, which includes the product structure and other additional information. Of course, the specific data and information required must be generated separately by the departments involved. For example, in crash simulation processes, crash-specific boundaries (e.g. forces, loads, material characteristics) are defined, in addition to the geometry-based information.

2.2.2 CAD-CAE Workflows in Automotive Engineering

CAD is used to create a geometrical product representation within a virtual environment. In automotive development, the product is composed of three-dimensional models in so-called 3D CAD programs. CAE includes a wide range of product calculation, simulation, optimization and planning processes in several disciplines

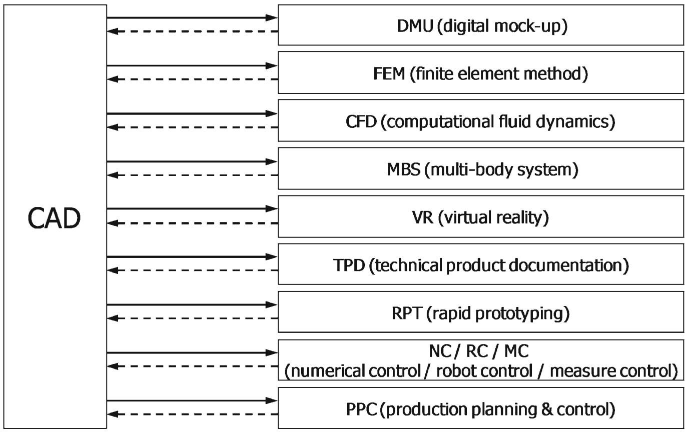

CAD
DMU (digital mock-up)
FEM (finite element method)
CFD (computational fluid dynamics)
MBS (multi-body system)
VR (virtual reality)
TPD (technical product documentation)
RPT (rapid prototyping)
NC / RC / MC (numerical control / robot control / measure control)
PPC (production planning & control)

Fig. 2.6 Examples of CAD-CAE workflows in automotive applications [8]

(e.g. mechanics, electrics, electronics, optics), which are performed parallel to the geometry creation. Of course, throughout the virtual development cycle, design always goes hand in hand with computational engineering processes. In some literature, virtual development in general is denoted as CAx (computer aided technologies). The combination of different CAD and CAE processes can be displayed as process chains.

The following sections include a short introduction of typical CAD-CAE workflows. Based on a selection of typical automotive development processes, a short description of the main targets, applied procedures and data formats provides an overview of virtual-development-related tasks (Fig. 2.6). Different CAE applications are based on a 3D CAD master model, which serves as a data source. Depending on the data format and accuracy required, the geometrical product information is converted and transferred into the CAE environment. The black arrows indicate a direct data connection from the CAD system to the corresponding CAE process. The information backflow after the evaluation and verification of simulation results always represents an important task in efficient development cycles (dotted arrows).

Digital Mock-Up (DMU)

Digital mock-ups are digital dummies, which include a simplified geometrical representation of a product. DMUs contain information about the product geometry (volume and/or surface models) and the product structure. DMU processes are used for packaging studies, clash detection, mounting and assembling simulations and other 3D CAD-based analysis steps. Converted (simplified) 3D CAD data from a virtual product model (master model) form the basis for DMU processes. The data transfer can be accomplished using native CAD data or using neutral data formats

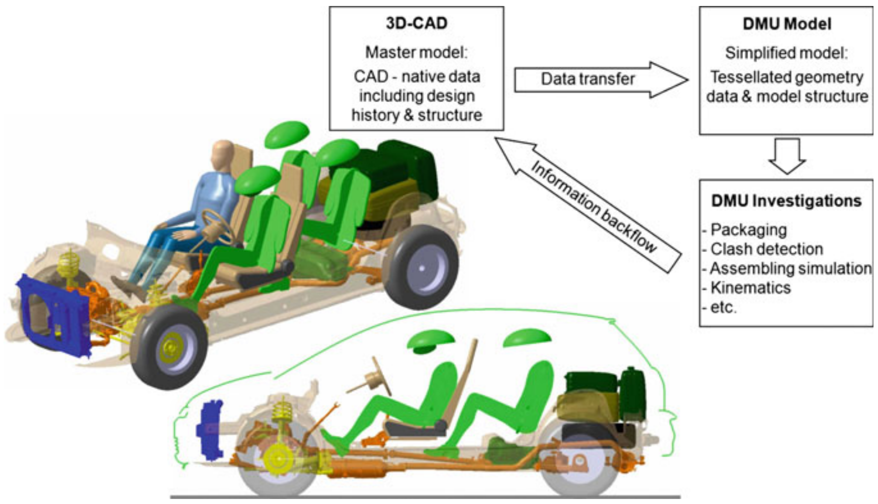


Fig. 2.7 DMU workflow based on a conceptual vehicle packaging study [8]

(e.g. STEP - standard for the exchange of product data [9], IGES - initial graphics exchange specification [10], JT - jupiter tessellation format, VDA (*Verband der Automobilindustrie*, [11]) formats). Due to a certain degree of geometry simplification, the accuracy of DMU models is lower than that of the corresponding 3D CAD master models (i.e. DMU uses tessellated geometries). Direct modification of geometry data is in general not possible within the DMU process. DMU supports simultaneous engineering approaches by handling large data structures throughout the product development, by localizing and eliminating geometrical problems in complex assemblies, and through target-oriented, assembly-based optimizations. DMU methods permit a geometrical freeze in early stages of the development cycle by integrating functional investigations into the DMU process.

In automotive development, DMU investigations cover the creation of assemblies (including components and devices), analysis and simulation (e.g. assembly procedures, movement and space investigations, collision checks, mounting and installation simulation). DMUs are often linked with simulation procedures, such as kinematical simulation processes for the optimization of movable functionalities (e.g. door-opening mechanisms, wheel suspensions, movable components in engines). Figure 2.7 depicts the DMU workflow and shows an example of a conceptual vehicle DMU, which includes initial styling information, carry-over parts from a model platform (drivetrain components, under-carriage and suspension), placeholders for wheels and luggage, and simplified human models and seat geometries. In this example, the DMU process supported the early layout of a new car model and enabled a coordination of vehicle styling proposals with the geometrical requirements of component and ergonomics configurations.

Finite Elements Method (FEM)

The finite elements method is used to calculate stress, deformation, thermal load, structural dynamics or NVH (noise, vibration and harshness). The FEM separates the continuum into finite, simple areas or volumes (finite elements), which are connected at defined nodes (mathematical discretization). FEM calculation processes are based on approximated geometries, derived from a 3D CAD model. Depending on the type of geometry and on the simulation target, different types of approximation are used. The calculation method is based on the creation of a hypothetical continuum, which is divided into simple patches or sub-bodies (*meshing*), which are in turn connected at defined nodes. The deformation of the infinitesimal mass points is approximately described as a function of the node deformation by means of a displacement approach based on an *element type*. This enables the separation of the distributed variables into space and time variables. FEM is suitable for the modeling of geometrically complicated, homogeneous structures in the fields of statics and strength and for the higher-frequency range of the motions [5, 12]. The geometry data transfer from the 3D CAD model into the FE-program is performed by a discretization process. Boundary conditions of loads (e.g. forces, moments), restraints (e.g. bearings, fixed parts), material characteristics, temperatures and other factors are defined directly in the FE-program. Figure 2.8 shows the general data flow in FEM processes using an exemplary application in engine development.

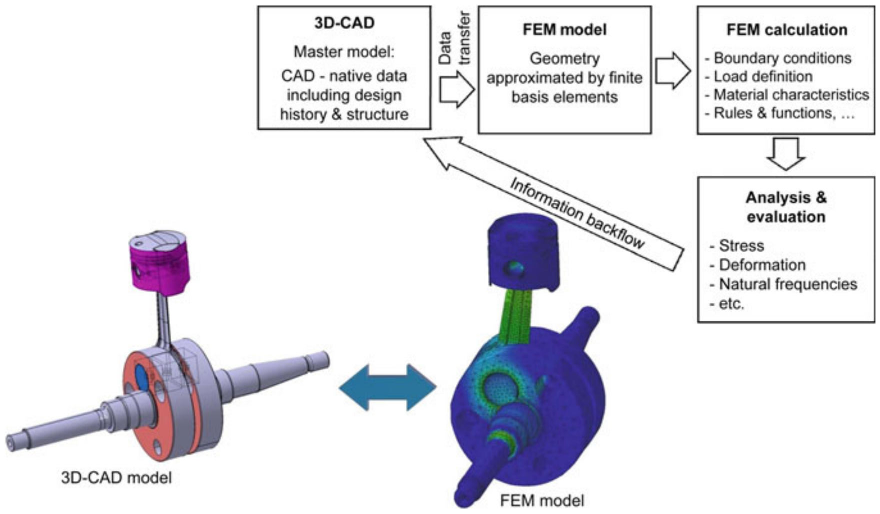


Fig. 2.8 Data flow in FEM processes and exemplary FEM simulation of a 1-cylinder engine crankshaft

Computational Fluid Dynamics (CFD)

Computational fluid dynamics simulation enables the calculation and optimization of gaseous and liquid flow processes. Similar to the FEM calculation process, the CFD model is based on an approximated geometry, the CFD mesh. The treatment of flow problems leads in general to an infinite dimensional differential equation system with space-dependent and time-dependent distributed variables (partial differential equations), which must be dicretized for the practical solution and simplified via idealizations. The idealizations selected depend on the actual interest and task and on the expected accuracy of the results. The discretization (meshing) can be performed at different levels of complexity, depending on the actual tasks and the required accuracy. One-dimensional flow calculation represents the flow characteristics along a streamline (e.g. within a tube). Three-dimensional flow calculations use spatial meshing procedures and are applied for complicated geometries, such as body flow (external aerodynamics), engine compartment flow (internal aerodynamics), and channels in combustion engines. The data transfer from the 3D CAD model into the CFD program is performed by neutral standard data formats (e.g. STEP, IGES), and the boundary conditions for the calculations are defined directly in the CFD program. Figure 2.9 shows the workflow of a CFD simulation and the results of an injection spray and air-fuel ratio simulation of a 1-cylinder motorcycle engine.

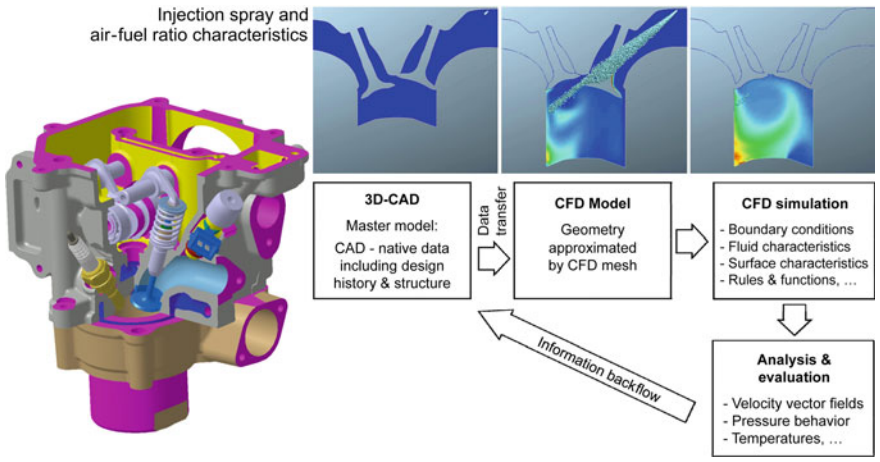


Fig. 2.9 Cylinder head assembly and CFD simulation of a motorcycle engine

Multi-Body Simulation (MBS)

Multi-body simulation is used for the kinematic and dynamic calculation and optimization of assembled (movable) parts. MBS models are based on general geomet-

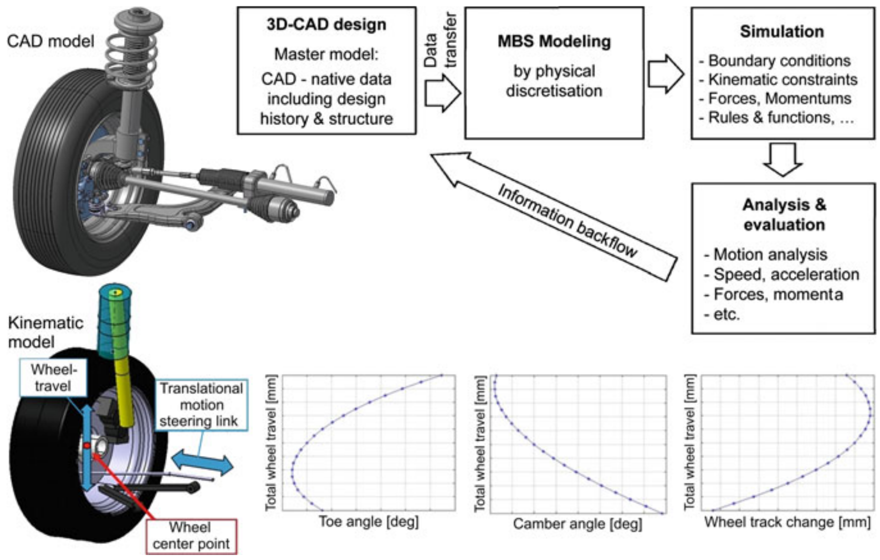


Fig. 2.10 MBS of an automotive suspension

rical definitions in CAD product structures. They consist of stiff bodies or mass points, which are connected with each other or the environment by joints (kinematic constraints) and/or by specific force laws. They are very suitable for modeling complex, inhomogeneous structures, such as full vehicles, and particularly for the low-frequency range of the motions. The boundary conditions (e.g. forces, torsional moments, masses, moments of inertia, degrees of freedom of movement) are defined directly in the MBS program. The separation of the locally distributed parameters into discrete parameters (e.g. mass, inertia, stiffness, damping) leads to a so-called physical discretization of the product model. Besides a rigid consideration of kinematic systems, elastic sub-bodies from FEM-modelings can be imported and integrated on demand (rigid-elastic MBS). Advanced MBS also enables the consideration of non-mechanical couplings (hydraulic, pneumatic and electrical state variables), as well as the modeling of active control units [5, 12]. Figure 2.10 shows the general workflow of an MBS. In addition to the CAD model, the derived MBS model and selected results of the MBS of an automotive suspension are shown.

Virtual Reality (VR)/Augmented Reality (AR)

Virtual reality and augmented reality generate virtual environments as real-time simulations for product representation and product-related investigations. They partially incorporate the user in virtual surroundings and product-related operations. AR functionalities enable the implementation of additional information and data into a VR environment (e.g. look-through functions or the accessibility of (virtual) con-

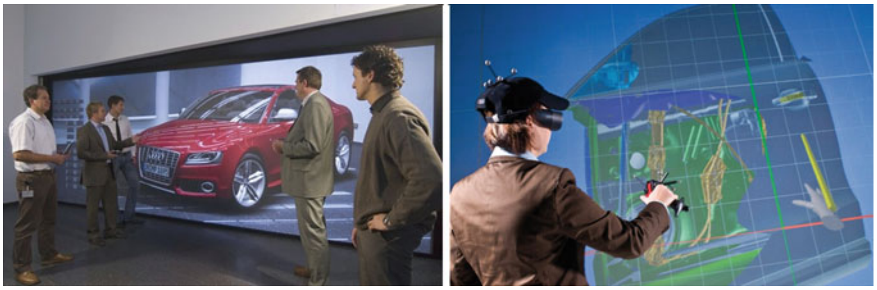


Fig. 2.11 VR equipment and illustration at a power wall [13]

trol units). The real-time interactions of geometries or functionalities support the assessment of procedures, in-use tests and product evaluation. The ability to manipulate objects gives the virtual product a near-real-life feeling. Besides development-related tasks, VR and AR technologies are used for applications in personnel training, maintenance, marketing and education. One specific data format for these types of applications (VRML - virtual reality modeling language) was designed to display 3D models and to integrate user-based interactions. VRML data, which are generated from a 3D CAD master model, include simplified geometry information without product history or structural data. Figure 2.11 illustrates examples of virtual reality application in automotive development. In the left figure, a realistic representation of exterior surfaces supports the styling evaluation process, while the figure on the right shows a look-through model of a car door module.

Technical Product Documentation (TPD)

Technical product documentation enables the derivation of technical drawings, bills of material, spare-part lists, prospects, and other items directly from the 3D CAD model. In order to enable a TPD generation, the CAD master model has to include all of the required information (e.g. product structure, geometry, tolerances, material, production related data). TPD formats include text-based formats (PDF - portable document format), 2D vector and pixel graphics (DXF - drawing exchange format, TIFF - tagged image file format, GIF - graphics interchange format), hypermedia formats (HTML - hyper text markup language, XML - extensible markup language) or 3D documentation software-based formats (3D PDF, WRL - web rule language) [14, 15]. TPD processes are always linked to standardization regulations and guidelines (e.g. ISO - International Organization for Standardization, DIN - German Institute of Standardization) and company-defined standards. Figure 2.12 shows examples of TPD, namely part lists, workshop drawings and an expanded view of a motorcycle crankshaft.

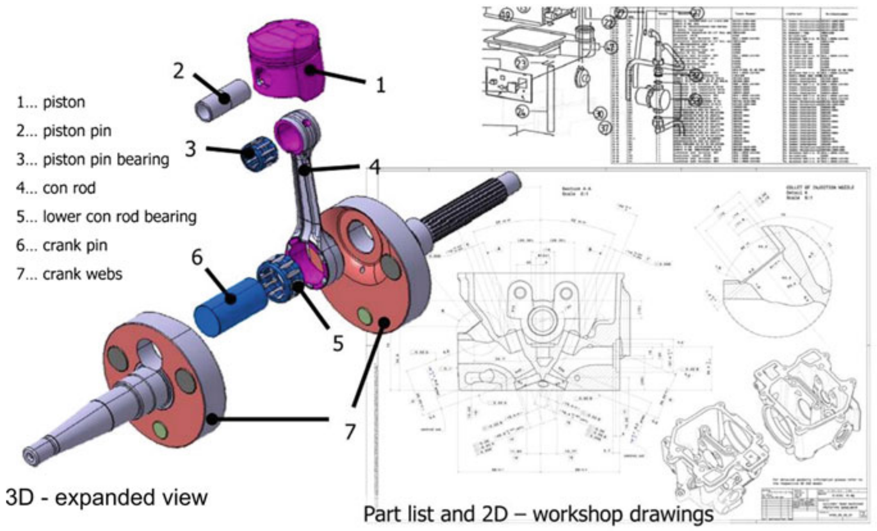


Fig. 2.12 Examples of TPD

Rapid Prototyping (RPT)

Rapid prototyping is used to generate hardware models from virtual geometry data during the product development phase. Such prototypes, which are available at an early stage, enable real-life studies, test bench optimization or customer discussions. RPT is used for concept models, design or ergonomic studies, or functional tests and optimization. Rapid prototypes are generated from tessellated geometries, which are derived from 3D CAD master models. The most common file format is STL (structural triangle language). STL geometries are calculated via triangulated surfaces with no design history or product structure information. RPT production techniques include laser sintering, stereo lithographic, 3D-printing and others. Figure 2.13 depicts the 3D CAD model, rapid prototyping parts and the physical component of an automotive cylinder head.

Numerical Control/Robot Control/Measure Control (NC/RC/MC)

Computer-aided numerical control, robot control and measure control cover manufacturing related process engineering tasks. These tasks are carried out in the course of computer-aided manufacturing (CAM), which addresses IT-supported functionalities for the control and monitoring of manufacturing resources. In NC/RC/MC processes, product geometry data are converted from a 3D CAD master model into a manufacturing-specific environment (language). Besides the geometry information, the master model includes all production-relevant data (e.g. tolerances, surface treatment, material characteristics). An NC data model contains machine-specific

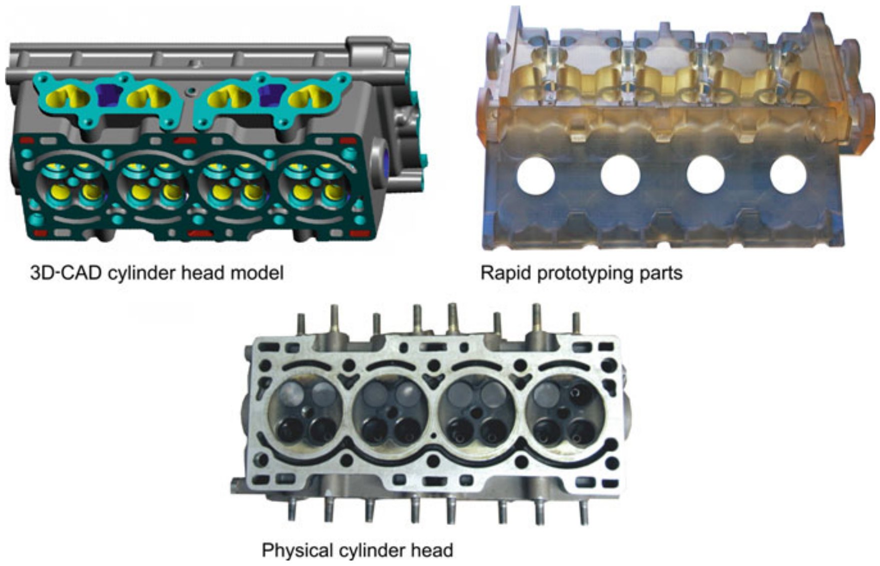


Fig. 2.13 Application of rapid prototyping in cylinder head development [16]

information about the applied tools, model fixation, cutting speed, and other factors. RC data enable the handling of assembly-relevant product information in manufacturing processes. MC data define prescribed measurement procedures for the analysis of the manufacturing process, deviation of tolerances or abrasion parameters. Figure 2.14 shows examples of numerical control machining and robot control simulation within virtual environment.

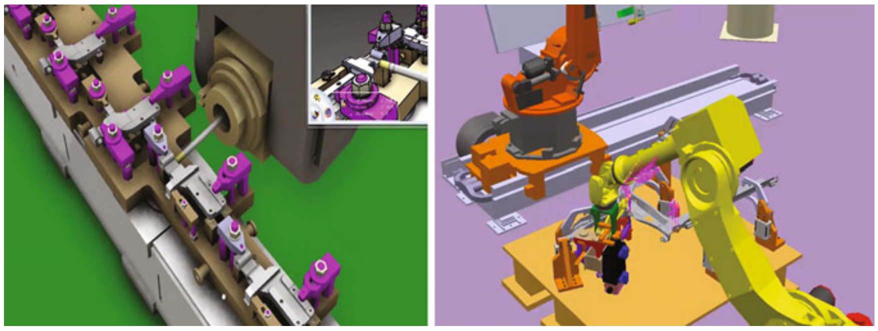


Fig. 2.14 Examples of NC and RC simulation [17]

Production Planning and Control (PPC)

Production planning and control includes the administration and organization of manufacturing-relevant data and procedures. Only released product data are transferred into the production planning process. PPC data are organized in BOM (bill of material) structures and based on 3D CAD geometry, 2D drawings and additional manufacturing-relevant information. Two important influencing factors are the working schedule and manufacturing resources. First, the basic economic and operational functions of PPC include the management of customer orders, the project calculation, the planning of requirements, the material logistics, the production capacity calculation and the order release organization. Second, PPC covers production-control-related tasks, such as production management and control, operating data logging, controlling processes (time, quantity and costs), and shipment management.

Whereas PPC covers the economic and operational tasks of manufacturing, CAD/CAM operations basically handle technical functions. The working fields of CAD/CAM can also be divided into two groups. The first group includes planning processes, such as the product concept development, design and simulation, process planning material logistics and the programming of production machines and resources. The second group includes the control of manufacturing and quality-management-related procedures. This covers the control of NC-machines, transportation control, storage management, assembly control, maintenance management and quality management.

2.2.3 *Management of Product Data*

Product life cycle management (PLM) includes all organizational tasks necessary for the identification, supply and archival storage of product-related data during the product life cycle. The management of all data flow, processes and documents during the development or modification of products across the product life cycle provides the basis for an efficient virtual product generation because complex product structures or product variations create numerous product parameters and a great amount of information. Product data management (PDM) organizes the data and information flow throughout the development process of a product, prevents data redundancy and is an important component in the generation of complex product structures in multi-firm and global collaboration [14].

Product data can be classified into different categories.

- Product-defining data related to technical requirements include all kinds of data for the product specification. In the case of automotive development, this can be driving performance, car weight, fuel consumption, dimensions, targeted vehicle configurations and other factors.
- Product-describing data related to technical product documentation are all of the information that can be found in lists (e.g. BOM).

- Geometry data include CAD model files, styling data, geometry data exchange formats, CAD-based product structures and other design-based data.
- Information concerning the development process itself includes workflow data, management of resources, data for engineering organization and others.
- Product configuration data include information about possible variants. They define the setup of the car in accordance with the customer order, including the type of engine and transmission, safety features, colors and all the possible accessories.
- Metadata describe additional product-related facts, such as production-related information or data for calculation and organization.

Of course, the management of product data includes the maintenance of different types of documents. Besides the main examples of product data, the documents can be classified using characteristic criteria, which address data quality, the age and maturity of data in the project progress, and the status. Further classifications are made based on data formats and the utilization and validity of data.

Due to the introduction of virtual methods in design, simulation and data management, modern IT-supported technologies have emerged in nearly all areas of mechanical industries in recent years. New methods and strategies have been developed to tap the potential of virtual product development processes, which has led to the implementation of new procedures, methods and tools in industry, education and research. IT-supported engineering in design, calculation and documentation characterize virtual product development, whereas specific CAx methods handle the generation of tasks, product models and process models that span multiple disciplines. In the automotive industry, virtual product development processes focus on different fields and disciplines. Depending on the specific tasks, these can address styling, vehicle packaging, component design, assembly, different types of simulation, production development and others. The goal is to integrate individual processes into incorporated virtual development processes supported by efficient data transfer and management.

The integration of virtual product and process models has enabled new approaches in design, simulation and manufacturing, which presents a challenge in the development of new IT methods for data transfer and data organization. Complex development processes require IT support and therefore the generation of PDM systems and other network-based information technologies. In addition, integrated data organization simplifies the documentation across the entire development process. An efficient documentation concept supports process and data standardization and enables network-based development by supplying product information that is available worldwide. Finally, knowledge databases, drawn from successful projects, can be created using an existing PDM network. These databases can be used for the re-integration of knowledge and product data into new project creation process chains and the re-use of experiences for variant studies and product optimization.

The new technologies lead to new possibilities and require new working methods and procedures: from static, theme-related working methods to dynamic, process-related working methods; from individual, hierarchic work to team-based work embedded in multifunctional structures (e.g. matrix structures); from document-based methods to integrated, virtual-product-model-based development. This change calls for the implementation of new networking processes and structures with a high demand for flexibility. Process-oriented working methods require an intensive time schedule with a detailed planning and management of several part-processes and sequences. To tap the full potential of virtual product development, a powerful management system of the entire process structure is needed. Earlier processes, which focused on digital models of the product, were unable to manage the data and information flow efficiently. The increased orientation on virtual products requires an increased implementation of integrated and linked development. Compared to the past, the processes and methods applied are becoming more important, which results in a change in the working procedure and working organization and presents a challenge to the people involved. In Sect. 6.4.1, *product data management* is treated in more detail.

2.2.4 CAD-CAE Data Exchange

The exchange of product data within the same software environment is mainly performed by native data formats. Native data are generated during the development of product-related information within one program. They include product data, information concerning the product development history, and methodological and organizational data. Thus, native data contain much more than purely discipline-specific information (e.g. the product geometry in the case of 3D CAD). Heterogeneous data exchange (e.g. importing product geometry information from a 3D CAD program into FE simulation software) is accomplished with the use of neutral data formats.

Neutral data exchange formats enable the transfer of product data between different software applications. In the case of product design and simulation-related processes, neutral data formats provide product geometry data and limited additional product information for the exchange between different CAD software packages or between CAD and CAE applications. Neutral data formats can be divided in geometrically accurate systems, which transform CAD-native product information into sets of mathematical descriptions of the model geometries. Examples are IGES, STEP or VDA formats. Besides a purely geometrical representation, enhanced neutral data formats are able to include additional facts, such as product structuring information or the configuration of components, modules and sub-assemblies.

The second group of neutral data formats concerns solutions which are able to convert the product geometry into models with approximated (tessellated) geometry. These formats yield geometry information that are less accurate but which also contain significantly lower data volumes. Examples of neutral data formats with tessellated geometries include STL, VRML and WRL. In addition, XML-based (extensible markup language) geometry description also works with tessellation algorithms. These approximated product data are often used in DMU-based investigations or for visualization tasks.

The JT format is a special neutral data format. This type of data exchange language provides a mathematical geometry description by applying BREP-based (boundary representation) algorithms, which rebuild the model geometry with groups of faces, edges and vertices. In addition, the JT format also provides tessellation algorithms for the creation of approximated geometries. Furthermore, product metadata can be transferred using this type of neutral data format [14, 18].

For the application of neutral data formats, specific converters are required to convert CAD-native geometry information into the corresponding format. In the case of data import, neutral data have to be converted or integrated into the corresponding software-specific format, which means that the target program must be able to read the specific type of neutral data format.

Unlike native data formats, neutral data languages cannot contain detailed design-process-related knowledge, such as the design history, product parameterization, implemented algorithms or macros, functionalities, etc. Normally, only the geometrical contour can be transmitted, although in some cases it is possible to transmit some selected additional information. This limits the application of neutral data formats to geometry-based processes and restricts advanced automation or design integration. For this reason, alternative methods and strategies have arisen in recent years. One such method is to manage data exchange by integrating PDM systems. In this method, a PDM system manages the entire product data range, including geometry data, product structure and additional data, such as materials, tolerances, and production-related information. In the case of data exchange (e.g. between automotive companies and suppliers), the required data are provided by the PDM system. This method requires the integration of all corresponding development partners into a comprehensive PDM system and a clear definition of the data models applied.

One trend is moving in the direction of integrated PDM systems, which integrate the corresponding CAD and/or CAE software packages, e.g. [17]. This strategy enables a universal data exchange, depending on the applied workflow. A centralized PDM system provides CAD-native data, as well as neutral geometry data and data for calculation, simulation or testing procedures. Depending on the type of development process, the specific required data are supplied and transferred. The ability to closely integrate CAD and PDM enables direct access by the PDM system to design-related native data (e.g. the product parameterization or specific constraints or links) which have been defined during the design process. In this way, collaborative design methods can be supported directly, without applying neutral data formats (e.g. via internet or intranet-based communication technologies), even in the case of a worldwide distribution of development partners.

The disadvantages of integrated PDM systems with directly integrated tools are relatively rigid structures and stiff data management in terms of the applicable software. The integration of external software (software from other software suppliers) into product development processes requires the application of neutral data formats. In addition, the data exchange performance depends on the data link provided, which can cause problems in the case of the large data amounts characteristic of complex products. Independent of PDM-related strategies, collaborative development methods call for the application of online data exchange and direct product data access across the entire development process chain. In addition, future data exchange formats will facilitate object-oriented methods for the supply of simulation processes and the creation of associative referencing techniques within complex virtual product structures [19].

2.2.5 Concepts of Collaborative Product Development

Industrial development processes have included methods for collaborating strategies for many years. More powerful IT systems and the capabilities of virtual engineering and data management have supported an increase of data exchange and have opened up new ways of collaborating. Modern development strategies, as they occur in automotive development, are only possible with modern networked processes. Although integrated, cross-linked strategies offer a significant potential for the improvement of development efficiency and data quality compared to traditional, sequential development methods, they require a significant re-configuration of established structures and procedures. Two important factors of success are the implementation of knowledge-based workflows in early phases of product generation and the integration of different development disciplines into a comprehensive consideration of product-related parameters. These approaches require extensive planning and preparation phases under consideration of the specific requirements in the relevant organization structures.

The integration of collaborative concepts of product development faces several challenges, including the application of complex communication and data flow structures, as well as the implementation of knowledge management strategies. Besides a resource shift into early project phases, the methods of collaborative development applied are based on the utilization of partially unreleased data and procedures. This requires efficient evaluation and assessment methods to avoid incorrect decisions in the very sensitive phases of product definition. The following short overview introduces three collaborating product development concepts, which have been applied within the current fields of research and development.

Concurrent Design

Concurrent design is a cooperative product development method that is based on the breakdown of complex design tasks into several subtasks. These subtasks are carried out by specialists more or less in parallel sequences. All engineering processes are linked, and the complete development process is supported by a data management system. In concurrent design, an efficient data and information exchange between the subtasks is an important factor. All participating parties (specialists) have to be informed about the sequences of processes, the content of each step and the data exchange procedures.

Synchronization of individual development steps can be performed with the help of DMU methods, in which the product is assembled and checked in the context of geometrical tasks. For example, the design process of a cylinder head is arranged as parallel design steps of the cast part, valve train components, other inner parts and the cylinder head cover (Fig. 4.9, p. 255). One important factor in concurrent design is the definition of design interfaces, such as flanges, adapter geometries or the implementation of skeleton models.

Simultaneous Engineering

Simultaneous engineering covers not only the design process, but the entire product development phase. Instead of working sequentially through stages, simultaneous engineering defines a parallel workflow of development tasks. For example, the tool design might be started before the detailed designs of the product components have been finished. The engineer begins the detailed design of solid models before the concept design surface models have been completed. Although simultaneous engineering does not necessarily reduce the amount of manpower required for a project, it can drastically reduce the development time and thus the time to market entry (Fig. 2.15). Especially in conceptual development, it is important that all responsible departments participate in simultaneous engineering. In automotive engineering, the concept phase has to consider several requirements, including legislative demands, crash and safety, functional aspects, packaging, production-related boundary conditions and many more. The early availability of broad information enables efficient, target-oriented product development.

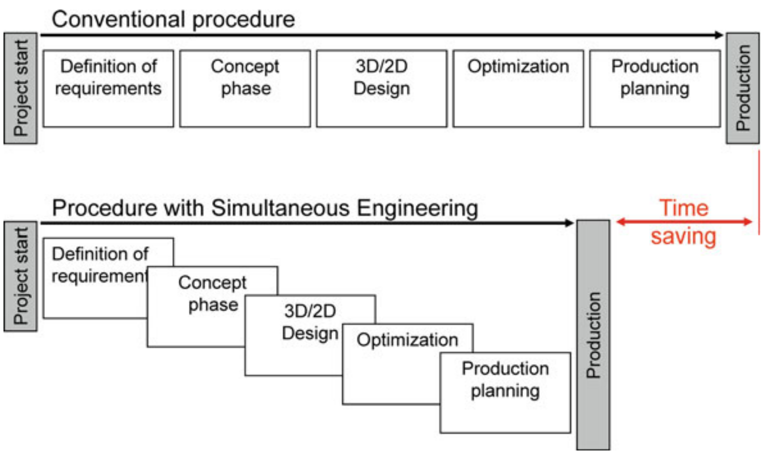


Fig. 2.15 Workflow in simultaneous engineering [12]

Frontloading

Frontloading is used in product feasibility studies, product planning and the early phase of product development. The goal is to define the product specifications as early as possible (as many characteristics and tasks as possible) in the concept phase. To achieve this, it is essential to use knowledge from former projects. Frontloading methods are based on a resources shift into the concept phase, in order to find solutions in the initial phase. This leads to an intensive application of knowledge-based engineering methods in combination with design and simulation steps. This can lead to a reduction in development time through the determination of product characteristics as accurately as possible and the elimination of functional product failure in the early phase, when the degree of freedom for product-related characteristics is high. The foundation is the availability of knowledge from former product development processes, which is supported by the provision of product-related information, the support by expert knowledge in the early phase, the use of simulation methods, and knowledge from former product life cycles (knowledge databases, lessons learned). In this way, frontloading methods are able to reduce the risk of failure in later development steps [2, 20].

References

1. Verein Deutscher Ingenieure: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. VDI-Richtlinie (1993)
2. Eigner, M.: Virtuelle Produktentwicklung I. Lecture script at the Lehrstuhl für Virtuelle Produktentwicklung, Technische Universität Kaiserslautern (2008)
3. Dubbel, H., Grote, K.H., Feldhusen, J.: Dubbel-Taschenbuch für den Maschinenbau, 22nd edn. Springer, Berlin (2007)
4. Ern, M.: Methodik zur Einsatz- und Ausbauplanung von CAE- Methoden für Entwicklungsprojekte in der Automobilindustrie. Ph.D. thesis, Universität Stuttgart, Stuttgart (2004)
5. Meywerk, M.: CAE-Methoden in der Fahrzeugtechnik. Springer, Berlin (2007)
6. Weissberg, D.E.: The Engineering Design Revolution: The People, Companies and Computer Systems That Changed Forever the Practice of Engineering. Cyon Research Corporation (2008)
7. Grabowski, H., Anderl, R., Polly, A.: Integriertes Produktmodell. Beuth, Berlin (1993)
8. Hirz, M.: Product data management in automotive engineering. Lecture script at Graz University of Technology (2008)
9. International Organization for Standardization: Industrial automation systems and integration, product data representation and exchange, (STEP-Standard for the Exchange of Product model data). ISO 10303–11 (2004)
10. U.S. Product Data Association: Initial Graphics Exchange Specification IGES 5.3 (2012)
11. Verband der Automobilindustrie: VDA-FS guideline for CAD data exchange. http://www.vda.de (1993). Accessed 20 April 2012
12. Hirz, M.: CAx in automotive and engine technology. Lecture script at Graz University of Technology (2011)
13. AUDI AG, Ingolstadt, Germany
14. Eigner, M., Stelzer, R.: Product Lifecycle Management. Springer, Berlin (2009)
15. Schäfer, D., Roller, D.: XML: Grundlagen und Anwendung des neuen Internet Standards. CAD-CAM Rep. **20**(2), 28–33 (2001)

16. Schumacher, J.: Auslegung und Konstruktion thermodynamischer Teile eines 4-Ventil/4-Zylinder CBR-Motorenkonzeptes. Diploma thesis, Graz University of Technology (2000)
17. Dassault Systems: CATIA V6. http://www.3ds.com/products/catia. Accessed 10 Nov 2009
18. International Organization for Standardization: Industrial automation systems and integration-JT file format specification for 3D visualization. ISO/DIS 14306 (2012)
19. Meyer, B.: Objektorientierte Softwareprogrammierung. Hanser, München (1990)
20. Dankwort, C.W., Ovtcharova, J., Weidlich, R.: A concept of engineering objects for collaborative virtual engineering: automotive case study. In: Fraunhofer, I.R.P. (ed.) Proceedings of the ProSTEP iViP Science Days, Dresden (2003)

Chapter 3
Geometric Fundamentals

An attribute of an object is called *geometric* if it is related to the spatial extension or the shape. Obviously, there are lots of geometric attributes in an automobile. The discipline dealing with geometric features is called *Computer-Aided Design* (CAD).

In the past few decades CAD has rapidly developed and broadly diversified. One reason for the amazing growth of CAD was the fertile soil of geometric founding on which it is based. CAD is one essential tool of research and development. Because of its enormous economic importance, CAD has been a major driving force for research in the theory of splines, approximation theory, computational geometry, geometry processing, discrete differential geometry and computer graphics. CAD deals with the process of design and design-documentation using a computer; these days, the use of a computer is so ubiquitous that the words *computer-aided* could just as well be omitted.

This chapter intends to give an introduction to CAD. Of course, it is of little use to address hardware issues or the details of specific software since such information might already be out of date by the time it is published.

CAD is used to design curves and shapes in the plane or curves, surfaces, and solids in 3-space. Many of the applications described in this chapter basically involve 3D-issues, others can be viewed in either a 2D- or a 3D-guise. A central branch of CAD deals with the design of geometric models for object shapes, which is called *Computer-Aided Geometric Design* (CAGD). Many of the topics addressed in this chapter belong to that field.

We start with fundamental attributes of 3-space (Sect. 3.1). Mathematical properties of polynomials—a core tool of modeling and graphics—are followed by general properties of curves in the plane and in 3-space (Sects. 3.2 and 3.3). Our next topic—freeform curves—can be viewed as a combination of both, polynomials and curves (Sect. 3.4). The fields of univariate interpolation and approximation are also closely related to the previous ones. They are of considerable significance from the engineer's point of view (Sects. 3.5 and 3.6). Surfaces (Sect. 3.7) are a key issue of CAD. This is why the concept of surfaces is also pivotal for the following sections in this chapter. The field of tensor product surfaces (Sect. 3.8) is a vital part of

CAD. We describe different types of tensor product surfaces as well as their common characteristics. The sections on bivariate interpolation (Sect. 3.9) and approximation (Sect. 3.10) are an introduction to further methods of surface generation. Triangular patches deserve separate attention as they require particular techniques (Sect. 3.11). Finally, a section on solid representation rounds off the chapter (Sect. 3.12); the definitions and methods in this section greatly resemble the ones for surfaces. We finally include a section where we discuss a single engineering task which is meant to show a number of geometrical and technical aspects occurring in one job (Sect. 3.13).

Although we introduce the basic tools from the bottom up, we have to assume that the reader is familiar with fundamental mathematical concepts. This chapter is particularly geared towards engineers who want to see below the varnish of the applied software and to understand how and why things work. Our intention is to deliver more than a compilation of statements and definitions. While space limitations do not allow for the proof of each single result, we endeavor to explain and contextualize the individual steps and definitions at least to some extent. Thus, this chapter on geometric fundamentals will hopefully serve as a useful reference source for key terms and geometrical aspects.

Whenever a new task or problem arises in the course of automotive development or more generally in mechanical engineering and design, there may be several ways of solving it. Since human beings can only recognize things with which they are familiar, an engineer's geometric background is crucial for his or her capacity to see the geometric aspects within a problem and to find the solutions.

3.1 The 3-Space, Transformations and Motions

For good reason we start with the 3-dimensional space $\mathbb{E}_3$ where most of our geometric considerations will be situated. However, some particular issues may essentially be 2-dimensional, i.e., they happen in a plane.

A *planar* Cartesian coordinate frame in a plane ε is defined by its origin O and two perpendicular coordinate axes x, y through O. Any point P in ε is uniquely determined by two real numbers x, y (Fig. 3.1) called the *Cartesian coordinates of* P. Alternatively, we can define a *polar coordinate system* in the plane by its origin O and a reference line x through O. In this case a point P is described by two real numbers r, u, namely its distance r from O and the angle u between the reference line x and the line OP. Cartesian and polar coordinates are linked via

$$x = r\cos u, \quad y = r\sin u. \tag{3.1}$$

A *spatial* Cartesian coordinate frame is defined by a point O (*origin*) and a triple of pairwise orthogonal, oriented lines through O—the *coordinate axes* x, y, z (Fig. 3.2). The planes spanned by two of the coordinate axes are called *coordinate planes*. So we have the xy-, the yz- and the zx-plane.

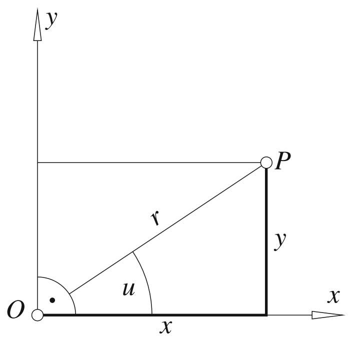


Fig. 3.1 Planar Cartesian coordinate frame

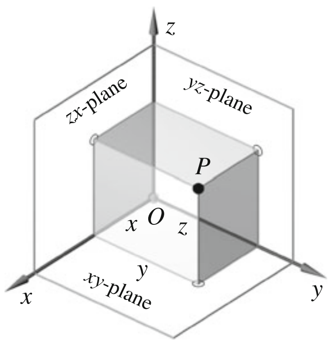


Fig. 3.2 Spatial Cartesian coordinate frame

With respect to a Cartesian coordinate system each point P in 3-space is assigned a triple x, y, z of real numbers, the *Cartesian coordinates* of P (Fig. 3.2). These numbers define the *position vector*

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

We identify the point P and its position vector $\mathbf{p}$ which is why we usually say: *the point* $\mathbf{p}$.

For some specific purposes it may be favorable to use other types of spatial coordinate systems like *cylindrical* or *spherical* coordinate systems.

A cylindrical coordinate system determines a point P by polar coordinates r, u of its top view in the xy-plane and the additional Cartesian z-coordinate: $P \ldots r, u, z$.

A spherical coordinate system, on the other hand, originates from geographic coordinates on the globe. It is determined by its origin O and two planes: the *equator plane* and the *zero meridian plane*. A point P is assigned two angles u and v and its distance r from the origin (cf. Example 3.13 and Fig. 3.77, p. 158). Cartesian and spherical coordinates are related via

$$x = r \cos u \cos v, \quad y = r \sin u \cos v, \quad z = r \sin v. \tag{3.2}$$

Now we consider mappings, also called *spatial transformations* on the set of points in 3-space. To each point $\mathbf{p}$ such a mapping assigns an image point $\mathbf{p}^*$ (see also [1], pp. 238).

Definition 3.1. Affine transformation.[1] Let $\alpha : \mathbf{p} \longrightarrow \mathbf{p}^*$ be a spatial transformation of the form

$$\mathbf{p}^* = \mathbf{A} \cdot \mathbf{p} + \mathbf{d} \tag{3.3}$$

where $\mathbf{d} := \begin{bmatrix} a_{10} \\ a_{20} \\ a_{30} \end{bmatrix}$ is a given 3-vector and $\mathbf{A} := \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{32} \end{bmatrix}$ is a given 3×3-matrix. If $\mathbf{A}$ be regular, i.e., $\det \mathbf{A} \neq 0$. then α is called an *affine transformation*. Similarly, we speak of *planar affine transformations* which are defined within a plane.

One core property of an affine transformation is that it always transforms collinear points into collinear points which means that straight lines are preserved. Due to the regularity of $\mathbf{A}$ an affine transformation is one-to-one and onto.

Definition 3.2. Isometry. An affine transformation (3.3) where additionally

$$\mathbf{A} \cdot \mathbf{A}^\top = \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

is called an *isometry*.

Matrices $\mathbf{A}$ complying with (3.4) are referred to as *orthogonal matrices*. An isometry can be geometrically characterized by the property that the distance of each pair of points is preserved: $\operatorname{dist}(\mathbf{p}, \mathbf{q}) = \operatorname{dist}(\mathbf{p}^*, \mathbf{q}^*)$.

Examples for isometries include reflections in planes, translations, rotations and helical displacements. In fact it is easy to verify that an isometry does not only preserve lengths but also angles and areas.

Isometries play an important role in the field of mechanical engineering. Designing a part with a CAD tool frequently requires the application of isometries. On the other hand, many mechanical parts such as cylinders, rotationally symmetric or helical components are manufactured by means of appropriate congruence transformations. A body of revolution, for example, can be produced on a lathe by applying a continuous rotation.

[1] For an application of affine transformations in CAD see also p. 267.

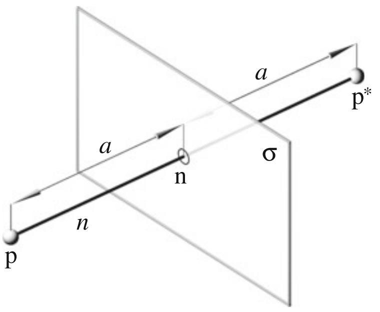


Fig. 3.3 Reflection of a point $\mathbf{p}(x, y, z)$ in a plane σ

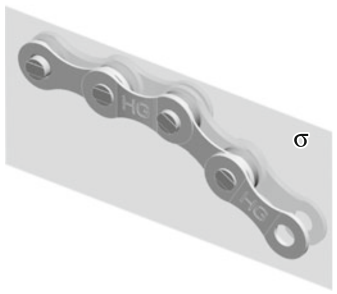


Fig. 3.4 The chain (bike drivetrain) is symmetric with respect to the plane σ

3.1.1 Planar Reflections

As a simple example of an isometry (3.3) we first consider the reflection in a plane (mirror plane, Fig. 3.3).

Definition 3.3. Reflection in a plane. Let σ be a plane. The *planar reflection* in the plane σ transforms a point $\mathbf{p}$ into a point $\mathbf{p}^*$ in the following way:

- Construct the line n through $\mathbf{p}$ perpendicular to σ. It intersects σ in a point $\mathbf{n}$.
- Find the point $\mathbf{p}^*$ on n with distance $\operatorname{dist}(\mathbf{n}, \mathbf{p}^*) = \operatorname{dist}(\mathbf{p}, \mathbf{n})$ such that $\mathbf{p}^*$ is opposite to $\mathbf{p}$ with respect to $\mathbf{n}$.

Mirroring a point in a coordinate plane is a particularly easy task in terms of computation: A point $\mathbf{p}(x, y, z)$ mirrored in the xz-plane delivers the point $\mathbf{p}^*(x, -y, z)$.

Of course, we can apply a reflection in a plane σ to all points of an object $\mathcal{O}$ arriving at a mirrored object $\mathcal{O}^*$. If $\mathcal{O} = \mathcal{O}^*$ we say that $\mathcal{O}$ is symmetric with respect to σ. Figure 3.4 shows an example.

A reflection can physically be realized by a glossy, reflective plane σ. The light rays mirrored in σ make believe that they are emitted from the mirrored object.

3.1.2 Translations and Rotations

Composing a reflection with itself transforms any point onto itself: $\mathbf{p}^* = \mathbf{p}$. Hence, this composition yields the identical transformation.

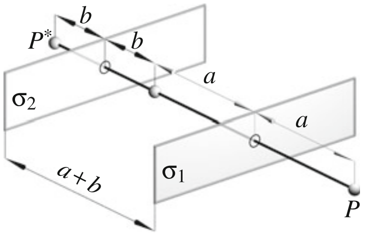


Fig. 3.5 The composition of two reflections in parallel planes σ_1 and σ_2 yields a translation

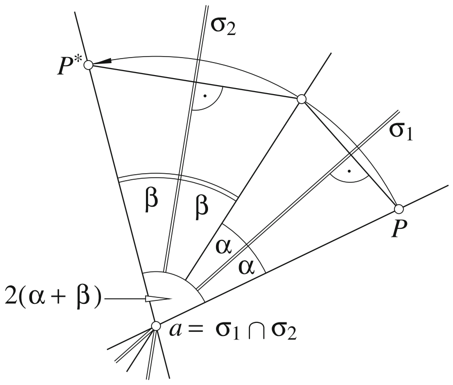


Fig. 3.6 The composition of two reflections in planes σ_1 and σ_2 intersecting in a line a yields a rotation about the axis a

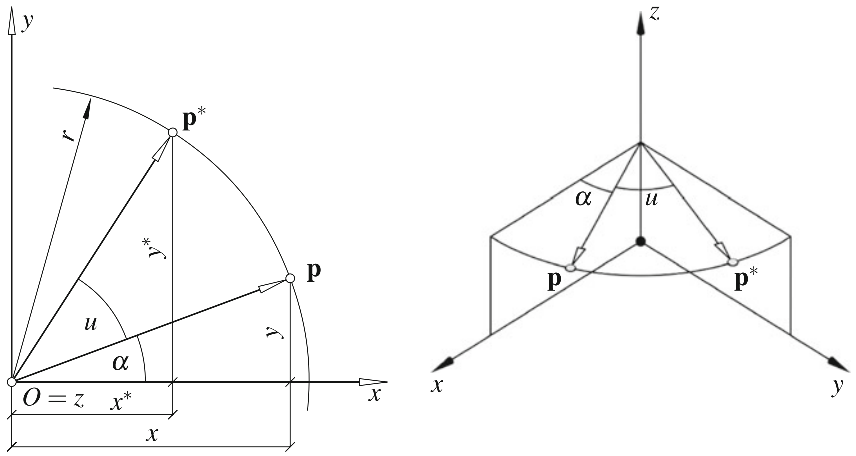


Fig. 3.7 *Left* analytical description of a rotation the point $\mathbf{p}$, rotated about the z-axis by the angle u, yields the revolved point $\mathbf{p}^*$. *Top* view in z-direction. *Right* general view

The composition of two reflections in different but parallel planes (Fig. 3.5) is a *translation* in the direction $\mathbf{v}$ orthogonal to the planes. Each point is moved in that direction by twice the distance between the two parallel planes.

Analytically the resulting point $\mathbf{p}^*$ can simply be obtained by adding the translation vector $\mathbf{v}$ to $\mathbf{p}$:

$$\mathbf{p}^* = \mathbf{p} + \mathbf{v}$$

If two planes σ_1 and σ_2 intersect in a line $a = \sigma_1 \cap \sigma_2$ the composition of the two reflections yields a rotation about the axis a. Each point is revolved by an angle u which is twice the angle between the planes σ_1 and σ_2. Figure 3.6 shows the situation. The planes σ_1 and σ_2 appear in edge view.

In the special case where σ_1 and σ_2 happen to be orthogonal we have $u = \pi \mathrel{\hat{=}} 180°$ for the rotation angle. This is a *half turn* about a, also dubbed *axial reflection* or *reflection in a line*.

In order to describe rotations analytically we put the z-axis of the coordinate system into the rotation axis a. (Fig. 3.7): Let

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\alpha \\ r\sin\alpha \\ z \end{bmatrix}, \tag{3.5}$$

where r, α, z are the *cylindrical coordinates* of $\mathbf{p}$. Revolving $\mathbf{p}$ about the z-axis by an angle u we arrive at the point $\mathbf{p}^*$ with cylindrical coordinates r, $u+\alpha$, z. Returning to Cartesian coordinates according to (3.1) we obtain

$$\mathbf{p}^* = \begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} r\cos(u+\alpha) \\ r\sin(u+\alpha) \\ z \end{bmatrix}.$$

It is easy to verify by means of the addition theorems for the sine and the cosine functions that this implies

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \cos u \cdot x - \sin u \cdot y \\ \sin u \cdot x + \cos u \cdot y \\ z \end{bmatrix}.$$

Using matrix notation we arrive at

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.6}$$

3.1.3 Orientation

In 3-space there are two types of Cartesian coordinate frames: *left-handed* and *right-handed* systems. In a left-handed system the x-, y- and z-axis are oriented just

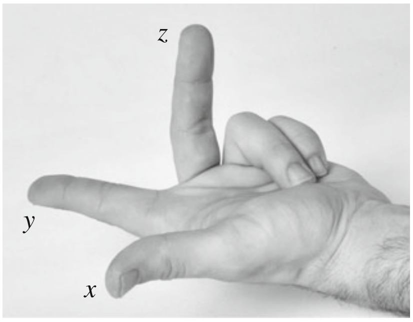


Fig. 3.8 The thumb (x-axis), the index finger (y-axis) and the middle finger (z-axis) of the *left* hand provide a *left-handed coordinate system*

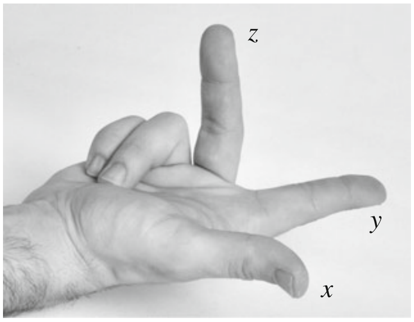


Fig. 3.9 The same declaration for the *right* hand delivers a *right-handed coordinate system. Right*-handed systems are generally being used

like a person's left hand thumb, index finger and middle finger (see Fig. 3.8). In a right-handed system the same holds for a person's right hand (Fig. 3.9).

We say that two right-handed systems (two left-handed systems) have the same *orientation* whereas a right- and a left-handed system have different orientations. In CAD-packages right-handed systems are standard. This is why we will also use right-handed systems in this book.

A reflection in a plane turns a right-handed into a left-handed system and vice versa. If an isometry can be obtained as a composition of an even (odd) number of reflections in planes it preserves (reverses) the orientation of any coordinate system. We call it an *even isometry* (*odd isometry*). As for the isometries considered above we can say:

Reflections in planes are odd isometries. Translations and rotations are even isometries.

As a further example let us compose three reflections in pairwise orthogonal planes. Without loss of generality we choose the xy-, the yz- and the xz-plane of the coordinate system as reflection planes. A point $\mathbf{p}(x, y, z)$ is eventually transformed into the point $\mathbf{p}^*(-x, -y, -z)$. This is the *point reflection* in the origin $\mathbf{o}(0, 0, 0)$. More generally, we can say that the composition of three planar reflections in pairwise orthogonal planes is a point reflection in the point $\mathbf{s}$ which the three planes have in common. A point reflection in 3-space is an odd isometry.

Note that a reflection in a line is the same as a rotation about this line by $u = \pi \mathrel{\hat{=}} 180°$ and hence an even isometry whereas a reflection about a plane or a point is odd.

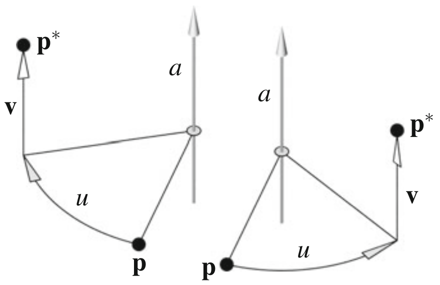


Fig. 3.10 A helical displacement is the composition of a rotation (axis a, angle u) and a translation along a (translation vector $\mathbf{v}$). *Left* left-handed screw displacement. *Right* right-handed screw displacement

3.1.4 Helical Displacements

We now consider the composition of a rotation about an axis a (rotation angle u) and a translation with a translation vector $\mathbf{v}$ parallel to a (Fig. 3.10). According to Sect. 3.1.2 this can also be viewed as a composition of four reflections in planes, the first two containing the axis a (rotation about a) and the latter two being perpendicular to a (translation along a). The emerging even isometry is called a *helical displacement* or *screw displacement* with *screw axis a*, *screw angle $u > 0$* and *screw parameter*

$$p := \pm\frac{\|\mathbf{v}\|}{u}.$$

Here we presume that u is positive. The sign of p can still be chosen. A helical displacement is a composition of four reflections in planes and hence an even isometry.

If we choose a right-handed coordinate system with the screw axis a as z-axis we obtain the analytical representation

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ p \cdot u \end{bmatrix} \tag{3.7}$$

where $\mathbf{p}(x, y, z)$ and $\mathbf{p}^*(x^*, y^*, z^*)$ denote a point and its transformed point and $\mathbf{v} = [0, 0, p \cdot u]^\top$.

A helical displacement is called *right-handed* if the rotation—viewed in the xy-plane against the $\mathbf{v}$-direction—is counterclockwise. If this rotation is clockwise the helical displacement is called *left-handed*. In (3.7) the condition $p > 0$ characterizes the right-handed helical displacements just as $p < 0$ characterizes the left-handed. Figure 3.11 shows an ordinary steel bolt with a right-handed thread.

Remark 3.1 It can be shown that any even isometry in 3-space is either a helical displacement or a rotation or a translation. This important statement is commonly known as *Chasles' Theorem* [2].

Fig. 3.11 The thread of a steel bolt as an application for screw motions

3.1.5 Euclidean Motions

A continuous series of even isometries is called a *Euclidean motion* (in short: *motion*, Fig. 3.12). The series be parameterized by a real number $u \in [u_0, u_1]$, the *motion parameter*, which can as well be interpreted as *time parameter*.

One example of such a spatial motion, for instance, is the movement of a robot's end-effector. As another example we can think of a car window glass being moved from its closed position into the door body by some window lifter mechanism. Due to the curved shape of the window pane, to find the appropriate spatial motion can be a challenge for the engineer (compare Sect. 3.13).

Analytically a Euclidean motion can be described via

$$\mathbf{p}^* = \mathbf{A}(u) \cdot \mathbf{p} + \mathbf{d}(u) \tag{3.8}$$

where u is the motion parameter and

$$\mathbf{p} := \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \mathbf{p}^* := \begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix}$$

denote a point and its image. The 3-vector $\mathbf{d} = \mathbf{d}(u)$ is the *translational part* of the motion and $\mathbf{A} = \mathbf{A}(u)$ is a proper orthogonal 3×3-matrix (*rotation matrix*), i.e.,

$$\mathbf{A} \cdot \mathbf{A}^\top = \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \ \det \mathbf{A} = 1.$$

Note that the point $\mathbf{p}^*$ depends on the parameter u; that means

$$\mathbf{p}^* = \mathbf{p}^*(u) = \begin{bmatrix} x^*(u) \\ y^*(u) \\ z^*(u) \end{bmatrix}.$$

This parametric representation describes the *trajectory* (*path*, *orbit*) of the point $\mathbf{p}$. We now consider a couple of simple examples of Euclidean motions.

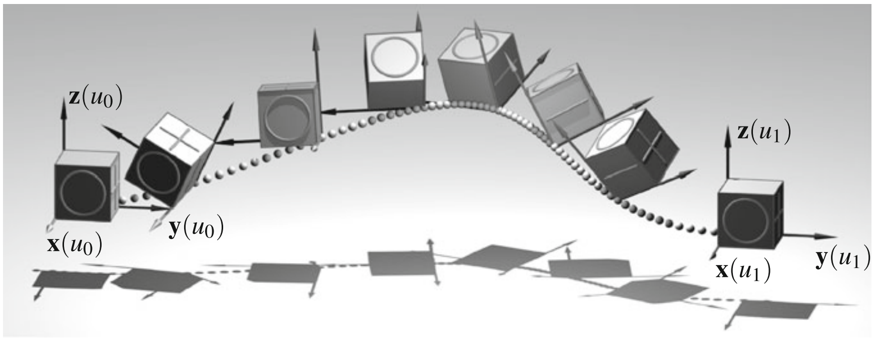


Fig. 3.12 This continuous Euclidean motion smoothly moves the given coordinate system and a cube along their orbit in 3-space

Example 3.1 A *continuous translation* along a given direction $\mathbf{v}$ can be defined by the constant vector $\mathbf{v} \neq [0, 0, 0]^\top$. Its analytic description is

$$\mathbf{p}^* = \mathbf{p} + u \cdot \mathbf{v}.$$

The path of any point $\mathbf{p}$ is obviously a straight line with direction $\mathbf{v}$. In this case the translational part of the motion is $\mathbf{d}(u) = u \cdot \mathbf{v}$ and the rotation matrix is the identity matrix: $\mathbf{A} = \mathbf{I}_3$.

Example 3.2 In a *continuous revolution* about an axis a the orbit of each point $\mathbf{p} = [x, y, z]^\top$ is a circle with axis a. For the parameterization we use the rotation angle u as motion parameter and we put the z-axis into a; then the trajectory $\mathbf{p}^*(u) = [x^*(u), y^*(u), z^*(u)]^\top$ of $\mathbf{p}$ reads as

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos u \cdot x - \sin u \cdot y \\ \sin u \cdot x + \cos u \cdot y \\ z \end{bmatrix}. \tag{3.9}$$

Example 3.3 A *screw motion* is defined by its axis a and its screw parameter p. A point $\mathbf{p} = [x, y, z]^\top$ is moved on its screw line trajectory (helix) $\mathbf{p}^*(u) = [x^*(u), y^*(u), z^*(u)]^\top$. As above we use the rotation angle u as parameter and put the z-axis into a:

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ pu \end{bmatrix} = \begin{bmatrix} \cos u \cdot x - \sin u \cdot y \\ \sin u \cdot x + \cos u \cdot y \\ z + pu \end{bmatrix} \tag{3.10}$$

The screw parameter p is the constant ratio between the translational component pu and the rotation angle u. The translational component relating to $u = 2\pi$ is commonly called the *pitch h* of the screw motion:

$$h = 2\pi p$$

One could say that the translation, the revolution and the screw motion play a pivotal role in the history of mechanical engineering. We just mention two fundamental inventions: the wheel (revolution) and the screw (screw motion).

3.1.6 Some Fundamentals of Line Geometry

Line geometry is one intriguing facet of geometry. As its fundamentals are not generally included in the basics of geometry we mention a few details in this section.

Plücker Coordinates

Let g be an oriented line in 3-space determined by a point **a** and a direction unit vector **d** (see Fig. 3.13). The straight line g can also be represented by the two vectors

$$\mathbf{g} := \mathbf{d} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \quad \bar{\mathbf{g}} := \mathbf{a} \times \mathbf{d} = \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \bar{g}_3 \end{bmatrix}, \tag{3.11}$$

called *normalized Plücker vectors of* g. The term *normalized* refers to the fact that the vector **g** is a unit vector. Combining the two vectors **g** and $\bar{\mathbf{g}}$ to a 6-vector

$$\begin{bmatrix} \mathbf{g} \\ \bar{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \bar{g}_1 \\ \bar{g}_2 \\ \bar{g}_3 \end{bmatrix}$$

we arrive at a point G in 6-space $\mathbb{R}^6$.

The six values $g_1, g_2, g_3, \bar{g}_1, \bar{g}_2, \bar{g}_2$ are called *normalized line coordinates* or *normalized Plücker coordinates* of the oriented line g. According to this procedure each oriented line g in 3-space is mapped to a point G in 6-space $\mathbb{R}^6$:

$$g \longrightarrow G$$

Further on we will denote this mapping by $\mathscr{G}$ and the image of a line g by $\mathscr{G}(g)$.

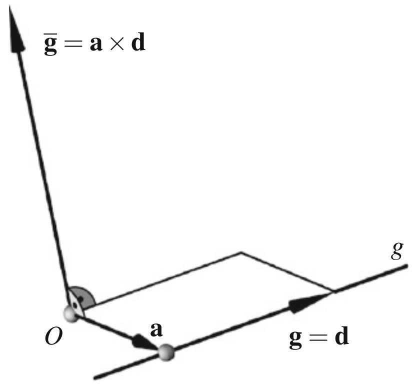

$\overline{\mathbf{g}} = \mathbf{a} \times \mathbf{d}$
g
O
$\mathbf{a}$
$\mathbf{g} = \mathbf{d}$

Fig. 3.13 The normalized Plücker vectors $\mathbf{g}$, $\overline{\mathbf{g}}$ of a *straight line* g. They are determined by the point $\mathbf{a}$ on g and the normalized direction vector $\mathbf{d}$ of g. The first Plücker vector is $\mathbf{g} = \mathbf{d}$ itself, whereas the second is the cross product $\overline{\mathbf{g}} = \mathbf{a} \times \mathbf{d}$

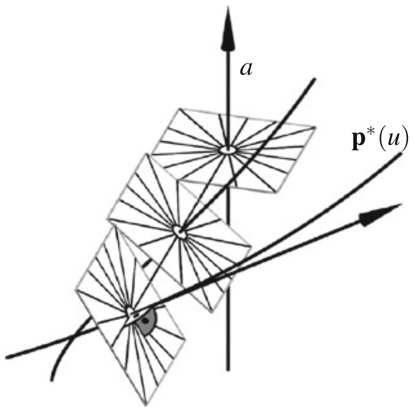


Fig. 3.14 The screw normals of a given screw motion (axis a and screw parameter p) establish a linear complex. The figure illustrates two screw *lines* and the axis a. One point is shown on each of these *lines*, together with its pencil of normals

Owing to their definition the Plücker vectors $\mathbf{g}$, $\overline{\mathbf{g}}$ are perpendicular to each other and thus satisfy the condition[2]

$$\langle \mathbf{g}, \overline{\mathbf{g}} \rangle = g_1 \cdot \overline{g}_1 + g_2 \cdot \overline{g}_2 + g_3 \cdot \overline{g}_3 = 0. \tag{3.12}$$

Moreover, as $\mathbf{g} = \mathbf{d}$ is normalized we also have

$$\|\mathbf{g}\|^2 = g_1^2 + g_2^2 + g_3^2 = 1. \tag{3.13}$$

The Eqs. (3.12), (3.13) determine two quadratic hypersurfaces in $\mathbb{R}^6$. Hence we see, that by means of $\mathscr{G}$ each oriented line g in 3-space is mapped into a point $G = \mathscr{G}(g)$ contained in the intersection of those two hypersurfaces in $\mathbb{R}^6$. Conversely, it can be shown that each point $G \in \mathbb{R}^6$ in the intersection of the two hypersurfaces (3.12), (3.13) is the $\mathscr{G}$-image of exactly one oriented line g of the 3-space.

[2] Throughout this text we denote the *dot-product* of two vectors $\mathbf{a}$ and $\mathbf{b}$ by $\langle \mathbf{a}, \mathbf{b} \rangle$.

Remark 3.2 The classical way of constructing a point model of the set of lines in 3-space goes back to Grassmann and Klein[3] and uses a projective image space of dimension 5. Details on this so-called *Plücker-mapping* can be found in ([3], pp. 133). The point model in $\mathbb{R}^6$ of the oriented lines in $\mathbb{R}^3$ which we have preferred instead can be seen as a variant of this classical Plücker-mapping. This way we could get around the notions of *projective spaces* and *homogeneous coordinates*.

Line Geometric View of a Screw Motion

In this section we use the term *continuous screw motion* (cf. Example 3.3) also for the special cases of continuous rotations (zero translational part; cf. Example 3.2) and continuous translations (zero rotational part; cf. Example 3.1).

Let M be a continuous screw motion. Then the trajectory of a point $\mathbf{p}$ is a screw line (helix) $\mathbf{p}^*(u)$, a circle (in the special case of a rotation) or a straight line (in the special case of a translation). In each of its points the curve $\mathbf{p}^*(u)$ has a one parametric set of normals g: These are the straight lines through the respective point of $\mathbf{p}^*(u)$ which lie in the plane orthogonal to the tangent of $\mathbf{p}^*(u)$ (see Fig. 3.14). The set of all such lines is called *linear line complex* or simply *linear complex*. The following geometric statements are well worth mentioning (cf. [3], pp. 163):

- All $\mathscr{G}$-image points $G = \mathscr{G}(g)$ in $\mathbb{R}^6$ of a linear complex lie in a common 5-dimensional hyperplane H through the origin O of $\mathbb{R}^6$.
- Conversely, each hyperplane H of $\mathbb{R}^6$ containing the origin of this space represents a well-defined linear complex and thus a unique screw motion (or rotation or translation) M.
- In general, any five path-normals $g_1, g_2.g_3, g_4, g_5$ of a given screw motion (or rotation or translation) determine the corresponding hyperplane $H \subset \mathbb{R}^6$ uniquely.

Let now $H \subset \mathbb{R}^6$ be a hyperplane through the origin O of $\mathbb{R}^6$ and let

$$\langle \mathbf{v}, \mathbf{g} \rangle + \langle \mathbf{w}, \bar{\mathbf{g}} \rangle = 0 \tag{3.14}$$

be its equation. Here $\mathbf{v} = [v_1, v_2, v_3]^\top$ and $\mathbf{w} = [w_1, w_2, w_3]^\top$ are constant coefficient vectors. Then the corresponding screw motion M is determined in the following way:

$$p = \frac{\langle \mathbf{w}, \mathbf{v} \rangle}{\|\mathbf{w}\|^2} \tag{3.15}$$

[3] Hermann Günther Grassmann (1809–1877) was a German mathematician and philologist. He is renowned as the founder of vector and tensor calculus. Felix Christian Klein (1849–1925) was one of the most prolific and significant German mathematicians of his time, particularly in the field of geometry.

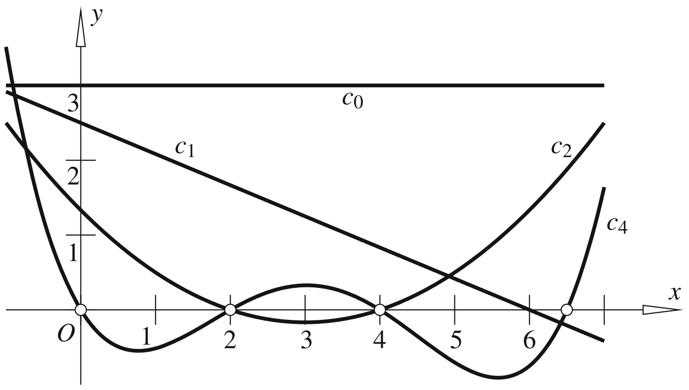


Fig. 3.15 The constant polynomial $p_0(x) = 3$ is of degree 0. It determines a constant function c_0. The polynomial $p_1(x) = -\frac{5}{12}x + \frac{5}{2}$ is linear (of degree 1). We have a linear function c_1. The polynomial $p_2(x) = \frac{1}{6}x^2 - x + \frac{4}{3}$ is of degree 2. We arrive at the quadratic function c_2. The polynomial $p_4(x) = \frac{1}{32}x^4 - \frac{25}{64}x^3 + \frac{47}{32}x^2 - \frac{13}{8}x$ is of degree 4. It yields the quartic function c_4

is the screw parameter,

$$\mathbf{d} = \mathbf{w} \tag{3.16}$$

is a direction vector of the screw axis a and

$$\mathbf{a} = \frac{\mathbf{w} \times \mathbf{v}}{\|\mathbf{w}\|^2} \tag{3.17}$$

is the position vector of the point $A \in a$ lying closest to the origin in $\mathbb{R}^3$.

Remark 3.3 The Eqs. (3.15), (3.16), (3.17) are valid only if $\mathbf{w} \neq [0, 0, 0]^\top$. The case $\mathbf{w} = [0, 0, 0]^\top$ characterizes continuous translations whereas a continuous rotation is characterized by $\langle \mathbf{w}, \mathbf{v} \rangle = 0$, $\mathbf{w} \neq [0, 0, 0]^\top$.

This classical connection between line geometry and screw motions has quite a few applications in practical problems. Section 3.13 contains one example.

3.2 Polynomials

Polynomial functions play a crucial role in CAD and, of course, in many other computational applications. One reason for this is that they can easily be evaluated. Another reason is that almost any function occurring in practical applications can be approximated by a polynomial with arbitrary degree of accuracy.

So whether or not the engineer knows it, polynomials are as ubiquitous as they are indispensable. The user will not always be aware of the polynomial background of freeform curves, surfaces or solids. But still, the knowledge of polynomials and their properties will certainly be valuable.

Definition 3.4. Polynomial in one variable. Let n be a nonnegative integer and let $a_0, \dots, a_n$ be real numbers (elements of $\mathbb{R}$) or complex numbers (elements of $\mathbb{C}$). Then the expression

$$p(x) = \sum_{i=0}^{n} a_i \cdot x^i \tag{3.18}$$

is called a univariate polynomial or, more specifically, a *polynomial in the variable x with coefficients a_i in $\mathbb{R}$ or $\mathbb{C}$.*

The set of all polynomials in the variable x is usually denoted by $\mathbb{K}[x]$ where $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$. We also call $\mathbb{K}[x]$ the *set of polynomials over the field* $\mathbb{K}$ in x.

Definition 3.5. Degree of a polynomial. If $p(x) = \sum_{i=0}^{n} a_i \cdot x^i$ is a polynomial and $a_n \neq 0$ the number n is called the *degree* of $p(x)$, in symbols:

$$\deg p(x) = n$$

Hence, the degree of a polynomial is the largest exponent occurring in (3.18). If $a_0 = \cdots = a_n = 0$ we call $p(x) = 0$ the *zero polynomial* and put $\deg p(x) = -\infty$.

A polynomial of degree 0, 1, 2, 3, 4, ... is also called a *constant*, *linear*, *quadratic*, *cubic*, *quartic*, ... polynomial, respectively. Figure 3.15 shows some examples of polynomial functions (see Definition 3.13, p. 69) which are defined by polynomials of degrees 0, 1, 2, 4.

We now consider the set $\mathbb{K}[x]$ of all polynomials over a field $\mathbb{K}$ and some of its subsets. The set $\mathbb{K}[x]$ has the structure of a vector space[4]: Two polynomials can be added and a polynomial can be multiplied with an element from $\mathbb{K}$ (multiplication with a scalar). In both cases the result is again an element of $\mathbb{K}[x]$. In this respect polynomials behave like vectors. This is why concepts from the theory of vector spaces (e.g., *linear independence*, *generating systems* and *bases*) also make sense for polynomials.

Definition 3.6. Linearly independent polynomials. A set $\{p_1(x), \dots, p_k(x)\}$ of polynomials is called *linearly independent* if none of its elements can be expressed as a linear combination of the remaining elements. In the opposite case the set of polynomials is called *linearly dependent.*

We denote the set of all polynomials over $\mathbb{K}$ with degree $\leq n$ by $\mathbb{K}_n[x]$. As can easily be verified, this subset of $\mathbb{K}[x]$ is also a vector space: $\mathbb{K}_n[x]$ is a subspace of $\mathbb{K}[x]$.

[4] For the notion of a vector space see for instance [4].

Definition 3.7. Basis of a subspace. A set $\{p_1(x), \ldots, p_k(x)\}$ of polynomials in $\mathbb{K}_n[x]$ is called a *generating system* if each element $q(x)$ of $\mathbb{K}_n[x]$ can be expressed as a linear combination of the elements $p_1(x), \ldots, p_k(x)$, i.e.,

$$q(x) = \sum_{i=0}^{k} \lambda_i \cdot p_i(x)$$

where $\lambda_i \in \mathbb{K}$. A linearly independent generating system of $\mathbb{K}_n[x]$ is called a *basis* of $\mathbb{K}_n[x]$.

We mention the so-called *monomial basis* $\{1, x, x^2, \ldots, x^n\}$ as one obvious example of a basis of $\mathbb{K}_n[x]$. Other examples of bases of $\mathbb{K}_n[x]$ are the *Bernstein basis* (Sect. 3.4.1, p. 86) and the *Lagrange basis* (Sect. 3.5, p. 116).

We add the following two important facts:

- Each basis of $\mathbb{K}_n[x]$ has exactly $n+1$ elements. In other words: The vector space $\mathbb{K}_n[x]$ of all polynomials of degree $\leq n$ is $(n+1)$-dimensional.
- If $\{p_0(x), \ldots, p_n(x)\}$ is a basis of $\mathbb{K}_n[x]$ then each polynomial $q(x) \in \mathbb{K}_n[x]$ can be expressed as a linear combination of the basis polynomials, i.e.,

$$q(x) = \sum_{i=0}^{n} \lambda_i \cdot p_i(x). \tag{3.19}$$

This is obvious since a basis is always a generating system of $\mathbb{K}_n[x]$. But due to the linear independence of a basis we additionally have that this representation is unique: If $q(x) = \sum_{i=0}^{n} \lambda_i \cdot p_i(x) = \sum_{i=0}^{n} \mu_i \cdot p_i(x)$ then we inevitably have $\mu_i = \lambda_i$ for $i = 0, \ldots, n$.

We can multiply two polynomials in the usual way. In more detail, the product $p(x) \cdot q(x)$ of a polynomial $p(x)$ of degree m and a polynomial $q(x)$ of degree n is a polynomial $r(x) = p(x) \cdot q(x)$ of degree $m+n$. In this case the polynomials $p(x), q(x)$ are called *divisors* of $r(x)$.

Note that any polynomial of degree 0—that is an element of $\mathbb{K} \setminus \{0\}$—is a divisor of any other polynomial. A divisor of degree > 0 is called a *proper divisor*.

Definition 3.8. Greatest common divisor of two polynomials. Let $p(x), q(x) \in \mathbb{K}[x]$ be nonzero polynomials. A polynomial $d(x)$ is called *greatest common divisor* of $p(x), q(x)$ if

1. $d(x)$ is a divisor of both $p(x)$ and $q(x)$ and
2. any common divisor of $p(x)$ and $q(x)$ is also a divisor of $d(x)$.

It is easy to see that the greatest common divisor of two polynomials is uniquely defined up to a constant nonzero factor. We denote the greatest common divisor of $p(x)$ and $q(x)$ by $gcd(p(x), q(x))$.

Definition 3.9. Zero of a polynomial. Let $p(x) = \sum_{i=0}^{n} a_i \cdot x^i$ in $\mathbb{K}[x]$ be a nonzero polynomial. An element $x_0 \in \mathbb{K}$ is called a *zero* or *root* of $p(x)$ if

$$p(x_0) = \sum_{i=0}^{n} a_i \cdot x_0^i = 0.$$

Many properties of polynomials heavily depend on their field of coefficients $\mathbb{K}$. The following theorem is a good example.

Theorem 3.1. The Fundamental Theorem of Algebra. *A polynomial $p(x)$ of degree $n > 0$ in $\mathbb{C}[x]$ can be written as the product of n factors of degree 1 (called linear factors), i.e.,*

$$p(x) = a \cdot (x - x_1) \cdot (x - x_2) \cdot \ldots \cdot (x - x_n) \tag{3.20}$$

where x_i are elements of $\mathbb{C}$.

Of course, the numbers $x_1, \ldots, x_n$ are roots of $p(x)$. Certainly, it may also happen that roots x_i coincide.

One conclusion from the Fundamental Theorem of Algebra can be: Every non-zero polynomial over $\mathbb{C}$ of degree $n > 0$ has at least one zero and at most n zeros.

As $\mathbb{R} \subset \mathbb{C}$, any polynomial over $\mathbb{R}$ can as well be viewed as a special case of a polynomial over $\mathbb{C}$. Thus, for any polynomial $p(x) \in \mathbb{R}[x]$ of degree n Theorem 3.1 can also be applied, however, as some of the values $x_1, \ldots x_n$ from (3.20) may well be non-real complex numbers, we only have the weaker conclusion: There are no more than n roots $x_i \in \mathbb{R}$ if $p(x) \in \mathbb{R}[x]$.

We now regard polynomials with more than one variables:

Definition 3.10. Multivariate polynomials. The expression $p(x, y) = \sum_{i,j} a_{ij} \cdot x^i \cdot y^j$ is called a *bivariate polynomial in the variables x, y* if there are only finitely many summands. Similarly a *trivariate polynomial in the variables x, y, z* is defined as $p(x, y, z) = \sum_{i,j,k} a_{ijk} \cdot x^i \cdot y^j \cdot z^k$. In the same way we can increase the number of variables and generally call any finite sum of the form

$$p(x, y, \ldots) = \sum_{i,j,\ldots} a_{ij\ldots} \cdot x^i \cdot y^j \cdot \ldots$$

a *multivariate polynomial in the variables $x, y, \ldots$*.

The concepts of the *degree of a polynomial* (see Definition 3.5 for univariate polynomials) and of the *greatest common divisor* (gcd) of polynomials can as well be extended to multivariate polynomials.

Definition 3.11. Degree of a multivariate polynomial. We consider a multivariate polynomial $p(x, y, \ldots) = \sum_{i,j,\ldots} a_{ij\ldots} \cdot x^i \cdot y^j \cdot \ldots$. The maximal exponent sum

$i + j + \dots$ among all elements $a_{ij\dots} \cdot x^i \cdot y^j \cdot \dots$ is defined as the *degree of the polynomial* $p(x, y, \dots)$.

As an example we write down the trivariate polynomial $p(x, y, z) = 1 + 2x - yz^2 + y^3 + 4x^2yz$ which is of degree 4.

The set of all polynomials in the variables u, v shall be denoted by $\mathbb{K}[u, v]$. We shortly consider the subset of all polynomials of degree $\leq n$ in 2 variables u, v. They form a subspace $\mathbb{K}_n[u, v]$ of $\mathbb{K}[u, v]$. The dimension of this space of polynomials can easily be computed as the monomial basis is obvious:

- The basis elements of degree 0 are multiples of the constant 1.
- The basis elements of degree 1 are u and v.
- The basis elements of degree 2 are u^2, uv and v^2.
- The basis elements of degree 3 are u^3, u^2v, uv^2 and v^3 and so on.

We can instantly count the number of elements up to degree n. As the set of these polynomials can easily be recognized as a linearly independent generating system in $\mathbb{K}_n[u, v]$ we eventually arrive at

Remark 3.4 **Dimension of the space of bivariate polynomials of degree** $\leq n$. The dimension of the space $\mathbb{K}_n[u, v]$ of polynomials in 2 variables u, v of degree $\leq n$ is $(n + 1) \cdot (n + 2)/2$.

A *divisor of a multivariate polynomial* $p(x, y, \dots)$ is a polynomial $d(x, y, \dots)$ such that

$$p(x, y, \dots) = d(x, y, \dots) \cdot e(x, y, \dots)$$

for some polynomial $e(x, y, \dots)$.

Definition 3.12. Greatest common divisor of two multivariate polynomials. Two non-zero multivariate polynomials $p(x, y, \dots)$ and $q(x, y, \dots)$ in $\mathbb{K}[x, y, \dots]$ be given. A polynomial $d(x, y, \dots)$ is the *greatest common divisor* of $p(x, y, \dots)$ and $q(x, y, \dots)$ if

1. $d(x, y, \dots)$ is a divisor of both, $p(x, y, \dots)$ and $q(x, y, \dots)$ and
2. any common divisor of $p(x, y, \dots)$ and $q(x, y, \dots)$ is also a divisor of $d(x, y \dots)$.

Definition 3.13. Polynomial functions and rational functions. Let $p(x) = \sum_{i=0}^{n} a_i \cdot x^i$ in $\mathbb{R}[x]$. The function

$$f : x \mapsto f(x) = p(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

is called *(univariate) polynomial function*. It maps every element $t \in \mathbb{R}$ into the element $\sum_{i=0}^{n} a_i \cdot t^i \in \mathbb{R}$.

Let $p(x)$ and $q(x)$ in $\mathbb{R}[x]$ with $\gcd(p(x), q(x)) = 1$. The function

$$f : x \mapsto f(x) = \frac{p(x)}{q(x)}$$

is called *(univariate) rational function*. Of course, a rational function is only defined where $q(x) \neq 0$.

In the same way *multivariate* polynomial functions and *multivariate* rational functions can be defined.

Polynomials will be the key tool for freeform curves (Sect. 3.4), freeform surfaces (Sect. 3.8) and tensor product volumes (Sect. 3.12). They are also vital for interpolation (Sects. 3.5 and 3.9) and approximation (Sects. 3.6 and 3.10).

3.3 Curves

A continuous 1-parameter set of points in 3-space (or in the plane) is generally called a *curve*. We can think of a curved piece of (infinitely thin) wire. In mechanical engineering curves frequently occur as curved edges of parts or surfaces.

3.3.1 Parametric Representation of a Curve

A curve c is a one-parameter set of points in the 3-dimensional space (or in the plane). Hence, it can be described by

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad t \in [t_0, t_1]. \tag{3.21}$$

where $x(t)$, $y(t)$ and $z(t)$ are functions of one parameter t. This means that to every t in the interval $[t_0, t_1] \in \mathbb{R}$ we assign a point $\mathbf{p}(t)$. These points trace the curve c (Fig. 3.16). The interval $[t_0, t_1]$ is also called *parameter domain* of $\mathbf{p}(t)$. The description (3.21) is also referred to as a *parametric representation* or parameterization of c.

Example 3.4 A straight line l can be parameterized via

$$\mathbf{p}(t) = (1 - t) \cdot \mathbf{a} + t \cdot \mathbf{b}, \quad t \in \mathbb{R}. \tag{3.22}$$

where $\mathbf{a}$, $\mathbf{b}$ are two different points on l (Fig. 3.17). The parameter values $t = 0$ and $t = 1$ in (3.22) refer to the points $\mathbf{a}$ and $\mathbf{b}$. If we substitute $u := 1 - t$ and $v := t$ (cf. Sect. 3.11, p. 225) we can rewrite (3.22) as

$$\tilde{\mathbf{p}}(u, v) = u \cdot \mathbf{a} + v \cdot \mathbf{b} \ \text{ with } u + v = 1. \tag{3.23}$$

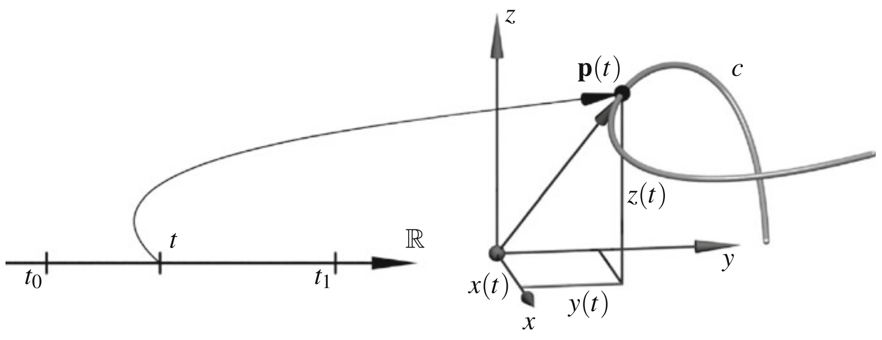


Fig. 3.16 The mapping on the interval $[t_0, t_1]$ assigns a point $\mathbf{p}(t)$ to every value $t \in [t_0, t_1]$. The points $\mathbf{p}(t)$ define a curve c

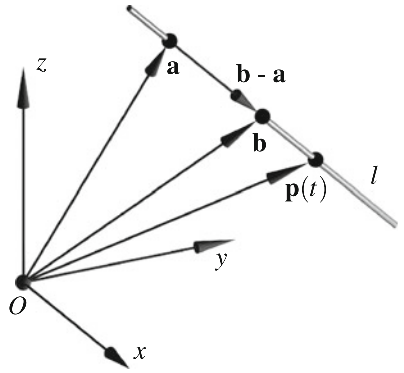


Fig. 3.17 Parametric representation $\mathbf{p}(t)$ of a *straight line* l through the points $\mathbf{a}$ and $\mathbf{b}$

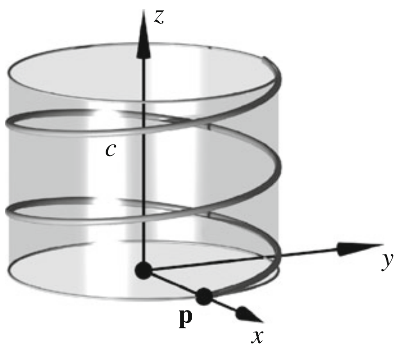


Fig. 3.18 A point $\mathbf{p}$ is subjected to a continuous screw motion. Its path c lies on the corresponding screw cylinder about the screw axis (z-axis)

Example 3.5 A *screw line* or *helix* c is the path of a point $\mathbf{p}$ under a screw motion (Fig. 3.18; see also Example 3.3, p. 61). We take the z-axis as the screw axis and consider the path c of $\mathbf{p}(r, 0, 0)$. The trajectory c has the parameterization

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos t \\ r\sin t \\ pt \end{bmatrix}. \tag{3.24}$$

Definition 3.14. Admissible parameterization of a curve. Let c be a curve with some parameterization (3.21). If $x(t)$, $y(t)$, $z(t)$ are differentiable functions and[5]

$$\dot{\mathbf{p}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ for all } t \in [t_0, t_1], \tag{3.25}$$

then the parameterization of c is called *admissible*.

Remark 3.5 To any given parameterization (3.21) of a curve c we can find infinitely many alternative representations: By applying a function $t = t(\tau)$ (*parameter transformation*) to the given parameterization (3.21) we obtain

$$\mathbf{q}(\tau) = \mathbf{p}(t(\tau)) = \begin{bmatrix} x(t(\tau)) \\ y(t(\tau)) \\ z(t(\tau)) \end{bmatrix}, \quad \tau \in [\tau_1, \tau_2] \tag{3.26}$$

which—for $t(\tau_0) = t_0$ and $t(\tau_1) = t_1$—describes the same curve c. If we additionally demand that $t = t(\tau)$ is a differentiable function with

$$\frac{dt}{d\tau} \neq 0 \text{ for all } \tau \in [\tau_1, \tau_2].$$

then $t = t(\tau)$ is called an *admissible parameter transformation*. Due to the chain rule for derivatives it is clear that if the given parameterization (3.21) of c and the parameter transformation both are admissible then this is also true for the new parameterization (3.26).

3.3.2 Planar Curves

As a special case we can consider *planar curves* where we have $z(t) \equiv 0$. Of course, in this case z can be omitted in the parametric representation altogether. A planar curve c can be described by a parametric representation

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \quad t \in [t_0, t_1]. \tag{3.27}$$

[5] The jth derivative of a vector function $\mathbf{p}(t)$ with respect to t is written as $\frac{d^j\mathbf{p}}{(dt)^j}$. If the variable is unambiguous we shortly write $\mathbf{p}^{(j)}$ or $\dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}, \ldots$.

At times it can also be represented by an equation

$$F(x, y) = 0. \tag{3.28}$$

This way of describing a planar curve has its merits (see also Sect. 3.3.8, p. 81): A point $\mathbf{p}(x, y)$ is contained in c if and only if (3.28) holds.

Example 3.6 An equation of the form

$$a \cdot x + b \cdot y + c = 0 \tag{3.29}$$

where $(a, b) \neq (0, 0)$ represents a straight line g in the xy-plane. The coefficients a, b determine a vector $\mathbf{n} = [a, b]^\top$ which is perpendicular to g. Mind that the same straight line can also be described by a parametric representation of the form (3.22).

Example 3.7 A circle in the xy-plane centered in $C(m, n)$, with radius r has the equation

$$(x - m)^2 + (y - n)^2 - r^2 = 0. \tag{3.30}$$

A parametric representation of the same circle could be

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} m + r\cos t \\ n + r\sin t \end{bmatrix}. \tag{3.31}$$

3.3.3 Derivatives and Tangents

This section and the following three are a short introduction to the differential geometry of curves. For more details we refer the reader to one of the excellent monographs on this subject (cf., for instance, [5]).

Let us now suppose that the functions we use for the representation of curves are differentiable and that the derivatives we need are still continuous.

Definition 3.15. Class of differentiability. Let k be a non-negative integer and let c be a curve with parameterization $\mathbf{p} = \mathbf{p}(t)$. If the k-th derivative

$$\mathbf{p}^{(k)}(t) \quad := \quad \frac{d\mathbf{p}}{dt}(t)$$

exists[6] and if this function is a continuous vector function we say that the parameterization $\mathbf{p} = \mathbf{p}(t)$ is *of class C^k*. C^k is also called the *class of differentiability* and c is called a C^k*-curve*.

If the k-th derivative of a vector function $\mathbf{p}(t)$ exists for *every* $k \in \mathbb{N}$ we say that $\mathbf{p}(t)$ is *of class C^∞* or *smooth*.

[6] Of course, this automatically implies that all derivatives of order $< k$ also exist.

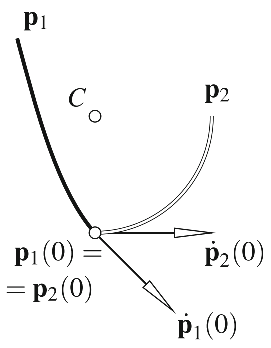


Fig. 3.19 Two curves joining C^0-**continuously**. A parabola arc $\mathbf{p}_1(t) = [t, -t + \frac{3}{2}t^2]^\top$ and a circular arc $\mathbf{p}_2(t) = [\sin t, 1 - \cos t]^\top$ joining C^0-continuously at $t = 0$. The tangents at the junction point are different the two curves join without a gap, however in a sharp kink

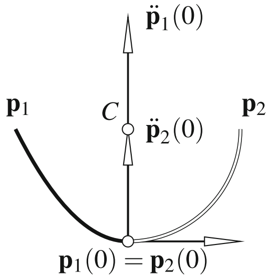


Fig. 3.20 Two curves joining C^1-**continuously**. A parabola arc $\mathbf{p}_1(t) = [t, t^2]^\top$ and a circular arc $\mathbf{p}_2(t)$ (see Fig. 3.19). The first derivative vectors of both parametric representations are identical: $\dot{\mathbf{p}}_1(0) = \dot{\mathbf{p}}_2(0)$. The tangents at the junction point are the same. The curvatures at the junction point differ

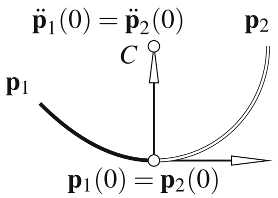

$\ddot{\mathbf{p}}_1(0) = \ddot{\mathbf{p}}_2(0)$
$\mathbf{p}_2$
C
$\mathbf{p}_1$
$\mathbf{p}_1(0) = \mathbf{p}_2(0)$

Fig. 3.21 Two curves which join C^2-**continuously**. A parabola arc $\mathbf{p}_1(t) = [t, \frac{t^2}{2}]^\top$ and again a circular arc $\mathbf{p}_2(t)$ (see Fig. 3.19). The first and the second derivative vectors of both parameterizations are identical: $\dot{\mathbf{p}}_1(0) = \dot{\mathbf{p}}_2(0)$ and $\ddot{\mathbf{p}}_1(0) = \ddot{\mathbf{p}}_2(0)$. As a consequence we even have the same curvature at the junction point

Definition 3.16. Tangent of a curve. If c is a curve with an admissible parameterization (3.21) then the straight line through $\mathbf{p}(t)$ with direction $\dot{\mathbf{p}}(t)$ is called *tangent to* c at $\mathbf{p}(t)$.

We now regard two curves with a common point. The concept of C^k*-continuity* describes to what extent the two parameterizations fit together.

Definition 3.17. **C^k-continuity of two curves.** Let c_1 and c_2 be two curves with the parameterizations $\mathbf{p}_1 = \mathbf{p}_1(t)$ and $\mathbf{p}_2 = \mathbf{p}_2(t)$, both of class C^k, $k \geq 0$. The parameterizations are said to be *C^k-continuous at t_0* if

$$\mathbf{p}_1^{(i)}(t_0) = \mathbf{p}_2^{(i)}(t_0), \quad \text{for all} \quad i = 0, \dots, k$$

Note that joining C^k-continuously is a local property of two parameterizations of two curves. It indicates that—in one common point $\mathbf{p}_1(t_0) = \mathbf{p}_2(t_0)$—they fit together up to the kth derivative vector.

In the same way we can define C^k-functions $y = f_1(x)$ and $y = f_2(x)$ joining C^k-continuously at some value $x = x_0$.

We now illustrate the geometric meaning of curves joining C^k-continuously for $k = 0, 1, 2$ (compare Figs. 3.19, 3.20, 3.21):

- If two curves join C^0-continuously in a point we can only say that c_1 and c_2 have a common point at t_0: $\mathbf{p}_1(t_0) = \mathbf{p}_2(t_0)$
- Curves joining C^1-continuously additionally have the same tangent vectors at the respective point. Note that, conversely, two curves may have a common tangent in a point though their first derivative vectors (tangent vectors) are not identical; they might only be of the same direction.
- In the case of curves joining C^2-continuously both curves, additionally, have the same 0th, 1st and 2nd derivative vector at the considered parameter value. As a consequence they also have the same point, the same tangent and the same curvature[7] at the respective point.

The last two cases show that in terms of geometry the concept of curves joining C^k-continuously could as well be replaced by a somewhat weaker definition which is called *geometric C^k-continuity*:

Definition 3.18. **$\mathbf{GC^k}$**-continuity of two curves. Let two curves c_1 and c_2 be given by parameterizations $\mathbf{q}_1 = \mathbf{q}_1(\sigma)$ and $\mathbf{q}_2 = \mathbf{q}_2(\tau)$ and let them have a common point, i.e., $\mathbf{q}_1(\sigma_0) = \mathbf{q}_2(\tau_0) =: \mathbf{a}$; then c_1 and c_2 are said to *join GC^k-continuously at the point* $\mathbf{a}$ if there exist reparameterizations

$$\mathbf{p}_1(t) := \mathbf{q}_1(\sigma(t))$$

and

$$\mathbf{p}_2(t) := \mathbf{q}_2(\tau(t))$$

of c_1 and c_2 such that the new parametric representations $\mathbf{p}_1(t)$, $\mathbf{p}_2(t)$ of c_1, c_2 are C^k-continuous at t_0 in the sense of Definition 3.17. (t_0 is the parameter value with $\mathbf{p}_1(t_0) = \mathbf{p}_2(t_0) = \mathbf{a}$.)

Mind that curves joining GC^k-continuously cannot be distinguished visually from curves joining C^k-continuously. However, if the parametric representations of two

[7] As for the concept of *curvature* see Sect. 3.3.5, p. 77

curves which merely join GC^k-continuously, are used for CNC-machining the difference to genuine C^k-parameterizations might become noticeable. As an example, one obvious effect might just be that the machining speed changes abruptly at the transition point.

3.3.4 Arc Length Parameter

Definition 3.19. Arc length parameter. Let c be a curve with an admissible parameterization (3.21); then we can introduce a new parameter $s = s(t)$ for the curve c defined by

$$\frac{ds}{dt} = \|\dot{\mathbf{p}}(t)\| \;=\; \sqrt{\dot{x}^2(t) + \dot{y}^2(t) + \dot{z}^2(t)}. \tag{3.32}$$

This parameter s called *arc length.*

The arc length parameter s is aptly named as we have:

Theorem 3.2. Arc length property. *Let c be a curve parameterized by its arc length s:*

$$\mathbf{p}(s) = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix}, \quad s \in [s_0, s_1].$$

Let moreover $\mathbf{p}_0 = \mathbf{p}(s_0)$ and $\mathbf{p}_1 = \mathbf{p}(s_1)$ be the points on c. Then $|s_1 - s_0|$ is the length of the arc on c between $\mathbf{p}_0$ und $\mathbf{p}_1$.

A proof to this theorem can be found in ([5], p. 19). In contrast to derivatives with respect to an arbitrary parameter t (indicated by dots) we mark derivatives with respect to the arc length s by primes: $\mathbf{p}'(s) = \frac{d\mathbf{p}}{ds}(s)$, $\mathbf{p}''(s) = \frac{d^2\mathbf{p}}{(ds)^2}(s), \ldots$.

Another important property of the arc length parameter s is the following:

Theorem 3.3. Derivative vector $\mathbf{p}'$*(s). Let c be a curve parameterized by its arc length s:*

$$\mathbf{p}(s) = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix}, \quad s \in [s_0, s_1]$$

Then we have

$$\|\mathbf{p}'(s)\| \;=\; \left\|\frac{d\mathbf{p}}{ds}(s)\right\| \overset{s}{\equiv} 1. \tag{3.33}$$

This theorem directly follows from (3.32) and from the chain rule of differentiation. It states that, at any point $\mathbf{p}(s)$, the tangent vector (first derivative vector) of a curve c parameterized by its arc length is a unit vector. If we interpret s as the time parameter and, consequently, the derivative $\mathbf{p}'(s)$ as the velocity we can say: The curve c is tracked with constant speed 1.

From $\|\mathbf{p}'\|^2 = \langle \mathbf{p}', \mathbf{p}' \rangle \equiv 1$ we obtain

$$\langle \mathbf{p}', \mathbf{p}'' \rangle \overset{s}{\equiv} 0. \tag{3.34}$$

This means that the first derivative vector and the second are perpendicular are orthogonal.

3.3.5 Curvature and Torsion

In this section we will develop the concepts of *curvature* and *torsion* which are two significant functions pertaining to any given space curve. The definition of these functions requires that the curve is—at least—of class C^3 according to Definition 3.15, p. 73.

Definition 3.20. Curvature and torsion of a curve. Let c be a space curve with an admissible parameterization (3.21). Thus we have $\dot{\mathbf{p}}(t) \neq [0, 0, 0]^\top$.

1. If $\mathbf{p}(t)$ is of class C^2 the *curvature* $\kappa(t)$ of c can be defined via

$$\kappa(t) = \frac{\left\|\dot{\mathbf{p}} \times \ddot{\mathbf{p}}\right\|}{\left\|\dot{\mathbf{p}}\right\|^3}. \tag{3.35}$$

2. If $\mathbf{p}(t)$ is of class C^3 the *torsion* $\tau(t)$ of c in $\mathbf{p}(t)$ is

$$\tau(t) = \frac{\det\left[\dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}\right]}{\left\|\dot{\mathbf{p}} \times \ddot{\mathbf{p}}\right\|^2}. \tag{3.36}$$

For a planar curve

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

the definition of the curvature has to be slightly modified:

$$\kappa(t) = \frac{\det(\dot{\mathbf{p}}, \ddot{\mathbf{p}})}{\|\dot{\mathbf{p}}\|^3} = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}. \tag{3.37}$$

Remark 3.6 We record a couple of properties related to curvature and torsion:

(a) The curvature and torsion of a curve are invariant with respect to parameterization and change of the coordinate system. In other words: Curvature and torsion both bear geometric meaning which will be referred to in the following section.
(b) A space curve which is contained in a plane, and thus is a planar curve is characterized by $\tau(t) \equiv 0$.
(c) As can be inferred from (3.35) or (3.37) the concept of curvature is even defined for curves of class C^2.
(d) In the planar case the curvature κ even has a sign which is geometrically meaningful.[8] If we interpret a planar curve as the track of a car being driven in the plane, at any instant κ can be understood as the current steering wheel position. The sign of κ distinguishes between left and right cornering.

In the special case where the spatial curve c is parameterized by its arc length $\mathbf{p} = \mathbf{p}(s)$ the formulae for the curvature κ and the torsion τ become a lot simpler:

$$\kappa = \left\|\mathbf{p}''\right\| \tag{3.38}$$

$$\tau = \frac{\det\left[\mathbf{p}', \mathbf{p}'', \mathbf{p}'''\right]}{\|\mathbf{p}''\|^2} \tag{3.39}$$

3.3.6 Osculating Circle and Osculating Plane

Let g be the tangent to a curve c at $\mathbf{p}(u_0)$ and let $\mathbf{p}(u_1)$ be an additional point on c, then in general there exists a unique circle l^* tangent to g at $\mathbf{p}(u_0)$ and passing through $\mathbf{p}(u_1)$. The limiting process $u_1 \longrightarrow u_0$ moves the point $\mathbf{p}(u_1)$ along the curve c into the point $\mathbf{p}(u_0)$. The circle l^* simultaneously adjusts and eventually turns into a circle l which is called *osculating circle* of the curve c at $\mathbf{p}(u_0)$ (see Fig. 3.22 and [5], p. 55). It can be shown that the radius ρ of l is the inverse of the above-mentioned curvature (3.35):

$$\rho = \frac{1}{\kappa} \tag{3.40}$$

Hence, ρ is also called *radius of curvature* of c at $\mathbf{p}(u_0)$. The plane σ containing the osculating circle, is usually called *osculating plane of c at* $\mathbf{p}(t_0)$. It is spanned by the first and the second derivative vector of $\mathbf{p}(t)$.

Points with $\kappa = 0$ are called *inflection points*. They are characterized by the condition that their first and second derivative vectors are linearly dependent. The osculating plane and the osculating circle do not exist. Instead, the tangent assumes the role of the osculating circle.

[8] For more details we refer to ([5], p. 26).

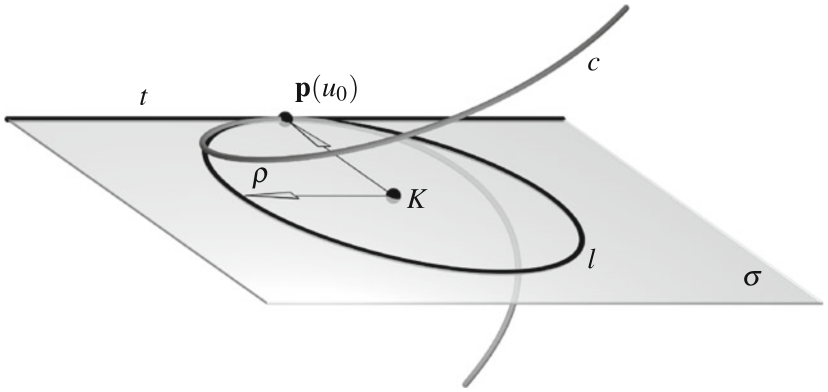


Fig. 3.22 The curve $c \ldots \mathbf{p}(u)$ is evaluated at $u = u_0$ which yields the point $\mathbf{p}(u_0)$. At this point it determines the tangent t and the osculating plane σ. The osculating *circle* l lies in σ and has the radius $\rho = \frac{1}{\kappa}$

Basically, the curvature in a point indicates to what extent the curve strives away from the tangent in that point. If the curvature is zero within a whole interval the respective part of the curve is a straight line segment.

The torsion of a curve in a point indicates to what extent the curve strives away from the osculating plane σ at that point. In particular, a planar curve c is characterized by $\tau \equiv 0$. The plane which contains c is identical with the osculating plane σ at each of its points.

Let c_1 and c_2 be two curves joining C^2-continuously at u_0 (see Definition 3.17, p. 75). Then c_1 and c_2 have the same curvature, the same osculating plane and the same osculating circle at $\mathbf{p}(u_0)$. This is obvious as the derivative vectors occurring in (3.35) are the same. If c_1 and c_2 are even C^3-continuous at $\mathbf{p}(u_0)$, the two curves additionally have the same torsion at that point (cf. (3.36)).

3.3.7 The Frenet Frame

Let $\mathbf{p} = \mathbf{p}(s)$, $s \in [s_0, s_1]$ be a curve parameterized with respect to its arc length s, then

$$\|\mathbf{p}'\| \equiv 1, \tag{3.41}$$

$$\langle \mathbf{p}', \mathbf{p}'' \rangle \equiv 0 \tag{3.42}$$

according to (3.33) and (3.34), p. 74. Moreover, with (3.38), p. 78, we have

$$\|\mathbf{p}''\| \equiv \kappa. \tag{3.43}$$

where κ is the curvature of c. Let us assume that $\kappa \neq 0$ in the considered interval $[s_0, s_1]$, i.e., there is no inflection point on the considered part of the curve c. Then

Eq. (3.42) tells us that the two unit vectors

$$\mathbf{t} := \mathbf{p}', \tag{3.44}$$

$$\mathbf{h} := \frac{\mathbf{p}''}{\kappa} \tag{3.45}$$

are perpendicular. In order to complete the two to an orthogonal vector basis we add the third vector

$$\mathbf{b} := \mathbf{t} \times \mathbf{h}. \tag{3.46}$$

Definition 3.21. Frenet frame of a curve.[9] Let c be a curve parameterized by $\mathbf{p} = \mathbf{p}(s)$, $s \in [s_0, s_1]$ with respect to its arc length s and let $\|\mathbf{p}''\| = \kappa \neq 0$ within $[s_0, s_1]$. Then the orthonormal vector basis $\{\mathbf{t}, \mathbf{h}, \mathbf{b}\}$ defined via (3.44), (3.45), (3.46) is called the *Frenet frame* of c. The vectors $\mathbf{t}, \mathbf{h}, \mathbf{b}$ are called the *tangent vector*, the *principal normal vector* and the *binormal vector* at the respective point $\mathbf{p}(s)$ of c.

Note that the tangent vector $\mathbf{t}$ and the principal normal vector $\mathbf{h}$ span the osculating plane σ of c (see Sect. 3.3.6, p. 78). Figure 3.23 shows a space curve c and its Frenet frame for six different values of s.

The Frenet frame is defined for any point of the curve where $\|\mathbf{p}''\| = \kappa \neq 0$. It makes sense to compute the derivatives of the vectors $\mathbf{t}, \mathbf{h}, \mathbf{b}$ in the Frenet basis. We arrive at the so-called *Frenet-Serret formulae*.

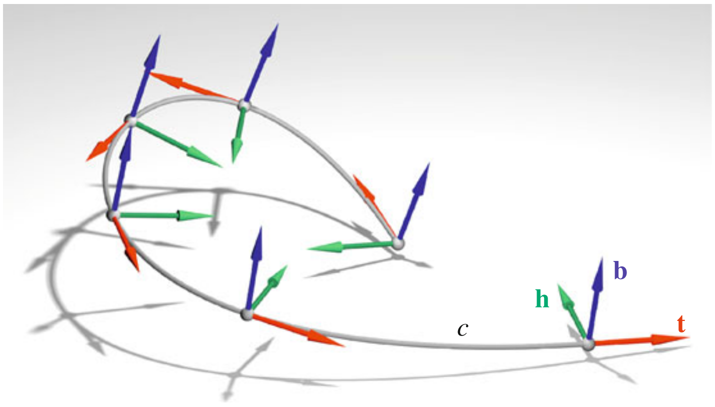


Fig. 3.23 A curve c and its Frenet frame $\{\mathbf{t}, \mathbf{h}, \mathbf{b}\}$ displayed for different values of the arc length parameter s. The tangent vector $\mathbf{t}$, principal normal vector $\mathbf{h}$ and binormal vector $\mathbf{b}$ are displayed in *red*, *green* and *blue*, respectively

[9] Jean Frédéric Frenet (1816–1900) was a French mathematician who published these formulae which were also found by Joseph Alfred Serret (1819–1885).

$$\left.\begin{array}{rlll} \mathbf{t}' = & & \kappa\cdot\mathbf{h} & \\ \mathbf{h}' = & -\kappa\cdot\mathbf{t} & & +\tau\cdot\mathbf{b} \\ \mathbf{b}' = & & -\tau\cdot\mathbf{h} & \end{array}\right\} \tag{3.47}$$

The coefficients occurring in these formulae are the curvature κ and torsion τ of the curve c (cf. Sect. 3.3.5, p. 77). We omit the proof which is based on (3.43) and the orthogonality of the vectors **t**, **h**, **b** (see, for instance, [5], p. 58).

3.3.8 Planar Algebraic Curves

In this section we focus on a particular class of planar curves which are given by their equation as in (3.28), p. 73.

Definition 3.22. Planar algebraic curve. A planar curve c is called *algebraic* if it can be described by a polynomial equation

$$F(x,y) \;=\; \sum_{finite} a_{ij}\, x^i\, y^j = 0 \tag{3.48}$$

where at least one of the coefficients a_{ij} is nonzero. The integer

$$n := \max\{i+j \mid a_{ij} \neq 0\} \tag{3.49}$$

i.e., the degree of the polynomial $\sum\limits_{finite} a_{ij}\, x^i\, y^j$, is called the *order* of the algebraic curve c.

Non-algebraic curves are those which cannot be characterized by any equation of this type (3.48). They are also referred to as *transcendent curves*.

Example 3.8 **Straight lines as** 1st **order planar algebraic curves.** Planar algebraic curves of order 1 have an equation of the form

$$ax + by + c = 0$$

with $(a,b) \neq 0$. They are straight lines.

Example 3.9 2nd **order planar algebraic curves.** Planar algebraic curves of order 2 have an equation of type

$$ax^2 + by^2 + 2cxy + 2dx + 2ey + f = 0 \tag{3.50}$$

with $(a,b,c) \neq (0,0,0)$.

The concept of the *order of a planar algebraic curve* bears some geometrical relevance as is shown in the following two theorems.

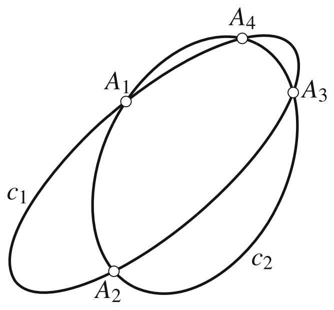


Fig. 3.24 A simple example to Bézout's Theorem: two different regular 2nd order curves (conics) c_1 und c_2 cannot have more than four intersection points

Theorem 3.4. Intersection of a planar algebraic curve with a straight line. *Let c be a planar algebraic curve of order n and let g be a straight line, $g \not\subset c$; then c und g can not have more than n points in common.*

Proof Let c be determined by (3.48) and g be given by its parameterization $\mathbf{p}(t) = (1-t)\cdot\mathbf{a} + t\cdot\mathbf{b}$ which reads as

$$\begin{bmatrix} x \\ y \end{bmatrix} = (1-t)\cdot\begin{bmatrix} x_A \\ y_A \end{bmatrix} + t\cdot\begin{bmatrix} x_B \\ y_B \end{bmatrix}. \tag{3.51}$$

Substituting (3.51) into (3.48) yields

$$\sum a_{ij}\left((1-t)\cdot x_A + t\cdot x_B\right)^i \left((1-t)\cdot y_A + t\cdot y_B\right)^j = 0.$$

The left-hand side of this equation is a polynomial of degree n in the variable t. Hence, it has at most n zeroes (Theorem 3.1: Fundamental Theorem of Algebra, p. 68). Any of the zeroes might be either a complex or a real number. Substituted into (3.51) each of the real-valued zeroes delivers a common point of c and g.

Theorem 3.5. Bézout's Theorem.[10] *Let c_1 and c_2 be planar algebraic curves with order n_1 und n_2. Let us further assume that c_1 und c_2 do not have any common component (algebraic curve which is part of both c_1 and c_2). Then c_1 und c_2 have at most $n_1\cdot n_2$ common points (see Fig. 3.24).*

The proof to this theorem is again based on the Fundamental Theorem of Algebra. It can be found in every book on basic Algebraic Geometry (see, for instance, [6], p. 59).

3.3.9 Rational Curves

Here we consider curves which can be parameterized by fractions of polynomials:

[10] Named after the French mathematician Étienne Bézout (1730–1783). It was first stated 1779 in É. Bézout's *Théorie générale des équations algébriques*.

Definition 3.23. Rational curve. A curve c is called a *spatial rational curve* if it can be represented by a rational parameterization

$$\mathbf{p}(t) = \begin{bmatrix} \frac{q_1(t)}{q_0(t)} \\ \frac{q_2(t)}{q_0(t)} \\ \frac{q_3(t)}{q_0(t)} \end{bmatrix} \tag{3.52}$$

Here $q_i(t)$ are polynomials with[11]

$$\gcd\{q_1(t), q_0(t)\} = \gcd\{q_2(t), q_0(t)\} = \gcd\{q_3(t), q_0(t)\} = 1.$$

In order to obtain a *planar rational curve* we only have to put $q_3(t) \equiv 0$ in (3.52).

If especially $q_0(t) \equiv 1$ the denominator can be omitted; we call c a *polynomial curve*.

The integer $n := \max\{\deg q_0(t), \deg q_1(t), \deg q_2(t), \deg q_3(t)\}$ is called the *degree of the rational curve c*.

It can be shown that any planar rational curve of degree n is also an algebraic curve of order $\leq n$. However, the converse is not true: A planar algebraic curve need not be rational.

Due to their simple representation rational and particularly polynomial curves are frequently used in CAD applications. Bézier curves (Sect. 3.4.1), B-Spline curves (Sect. 3.4.2) and NURBS (Sect. 3.4.3) are typical examples of curves consisting of rational or even polynomial components.

3.3.10 Second Order Curves

Second order curves include a couple of well-known and frequently occurring planar curves.

An algebraic curve of order 2 (= second order curve) is a curve with an equation of the form (3.50) with $(a, b, c) \neq (0, 0, 0)$. Selecting the coordinate system in the plane adequately this equation can be put into exactly one of the following *normal forms* ($a, b, p \in \mathbb{R}^+$):

(a) $x^2 = 0$.
This is a *straight line*. It is sometimes referred to as a *doubly counted straight line*.

(b) $a^2x^2 + b^2y^2 = 0$.
This can also be written in the form $(ax + iby) \cdot (ax - iby) = 0$ where i is the complex unit ($i^2 = -1$). This shows that we have a *pair of conjugate complex*

[11] gcd denotes the *greatest common divisor* of polynomials; see Definition 3.8, p. 67.

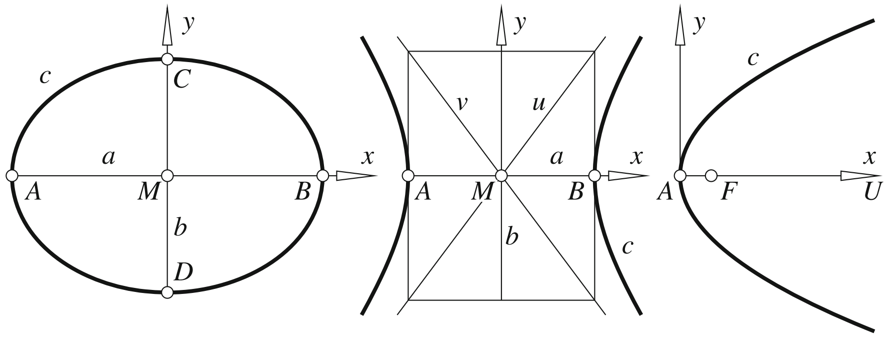


Fig. 3.25 Conic sections. *Left* ellipse, vertices A, B and C, D; the values a, b from (3.53) are marked. *Center* hyperbola, vertices A, B and asymptotes u, v; a, b from (3.54) are again marked. *Right* parabola, vertex A, focus point F, parabola axis x, point at infinity U

lines. This curve only contains one real point, namely $\mathbf{o}(0, 0)$—the point of intersection of the two complex lines.

(c) $a^2x^2 - b^2y^2 = 0$.
Because of $a^2x^2 - b^2y^2 = (ax + by) \cdot (ax - by)$ the curve c is a *pair of real straight lines*.

(d) $x^2 + a^2 = 0$.
As this is equivalent to $(x + ia) \cdot (x - ia) = 0$, we have a *pair of conjugate complex parallel lines*. The curve does not contain any real point apart from the point at infinity!

(e) $x^2 - a^2 = 0$.
Again we can factorize the left-hand side of the equation: $x^2 - a^2 = (x + a) \cdot (x - a)$. This shows that c consists of *two real parallel lines*.

(f) $\frac{x^2}{a^2} + \frac{y^2}{b^2} = -1$.
This curve does not contain any real point and is called *null conic*.

(g) $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$.
In this case c is an *ellipse* whose axes coincide with the coordinate axes and have lengths $2a$ and $2b$ (Fig. 3.25, left).

(h) $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$.
We have a *hyperbola* whose axes lie on the coordinate axes. The distance of the two apexes is $2a$ and its asymptote angle α is defined by $\tan \frac{\alpha}{2} = \frac{b}{a}$ (Fig. 3.25, center).

(i) $y^2 = 2px$.
This is a *parabola* with its apex in the origin and its symmetry axis on the x-axis. Its focus is $F(\frac{p}{2}, 0)$ (Fig. 3.25, right).

Admittedly, the types (a)–(e) are not particularly thrilling as they consist of straight lines; they are called *singular second order curves*. The other types (f)–(i) are called *regular second order curves*. The three types (g), (h) and (i) (ellipse, hyperbola and parabola) are referred to as *conics* or *conic sections*.

For the ellipse (g) we have the following standard parameterization:

$$\mathbf{p}(t) = \begin{bmatrix} a\cos t \\ b\sin t \end{bmatrix}, \quad t \in [0, 2\pi) \tag{3.53}$$

In the hyperbola case (h) we have:

$$\mathbf{p}(t) = \begin{bmatrix} \pm a\cosh t \\ b\sinh t \end{bmatrix}, \quad t \in \mathbb{R} \tag{3.54}$$

The parabola (i) can be parameterized by:

$$\mathbf{p}(t) = \begin{bmatrix} \frac{1}{2p}t^2 \\ t \end{bmatrix}, \quad t \in \mathbb{R}. \tag{3.55}$$

In Fig. 3.25 the three types of conic sections are illustrated. As for the standard parameterizations (3.53), (3.54) and (3.55) the coordinate system is specifically adapted to the curve to obtain a particularly simple form. In these normal forms the symmetry axes of the curves are the axes of the coordinate system.

Obviously, the standard parameterizations (3.53), (3.54) of the ellipse and the hyperbola are not rational. However, there also exist rational parameterizations and, as a consequence, conics are rational curves. We will harness this property in Proposition 3.9, p. 113.

A parabola is even a polynomial curve. Moreover, we have the following result:

Theorem 3.6. Parameterization of a parabola. *Let $a_0, a_1, a_2, b_0, b_1, b_2$ be constant real numbers. Then*

$$\mathbf{p}(t) = \begin{bmatrix} a_0 + a_1 t + a_2 t^2 \\ b_0 + b_1 t + b_2 t^2 \end{bmatrix}, \quad t \in \mathbb{R}$$

is the parameterization of a parabola if

$$a_1 b_2 - a_2 b_1 \neq 0 \tag{3.56}$$

This can be proved by means of suitable coordinate transformations.

3.4 Freeform Curves

Working in a CAD environment requires adequate, simple tools for the control of pretty complex geometric objects. The user ought to be given the option of easily modifying their shapes by interactive manipulation.

As for the case of one-parametric objects so-called *freeform curves* have emerged as standard. The most important types will be defined and discussed in detail in the following sections: Bézier curves, B-spline curves, Lagrange interpolants and Hermite interpolants. We mention that there exist further types of freeform curves: μ-splines, β-splines, Wilson-Fowler-splines, splines in tension, exponential splines and nonlinear splines. A closer look at these interesting curves is beyond the scope of this introduction. It can be found in [7].

We first launch the general concept of a *freeform curve*:

Definition 3.24. Freeform curve. Let $n+1$ points $\mathbf{a}_0, \dots, \mathbf{a}_n$ in 3-space (or in the plane) be given and let $F_i(t)$, $i = 0, \dots, n$ be a family of functions of a parameter t. The curve with the parametric representation

$$\mathbf{p}(t) = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i, \quad t \in [a, b] \tag{3.57}$$

is called the *freeform curve* to the given *control polygon* $\mathbf{a}_0, \dots, \mathbf{a}_n$ belonging to the function family $\{F_i(t)\}$; $[a, b]$ is the corresponding parameter interval.

Remark 3.7 The definition of freeform curves requires the choice of a family $\{F_i(t)\}$ of functions. The whole concept only makes sense if the curve is finally determined by the control polygon $\mathbf{a}_0, \dots, \mathbf{a}_n$ and does not depend on the choice of the coordinate system used for its description. It is easy to prove that this independence is guaranteed if and only if the condition

$$\sum_{i=0}^{n} F_i(t) \equiv 1 \tag{3.58}$$

holds. In that case the function family $\{F_i(t)\}$ is called a *partition of unity*.

If (3.58) is fulfilled there is a close *geometric* bond between the control points and the resulting freeform curve. This relationship greatly depends on the choice of the functions $F_i(t)$. In the following sections we will study the most common types.

3.4.1 *Bézier Curves*

Within the last few decades the Bézier curves have emerged as the forefathers of various free form spline techniques. No matter which type of freeform technique is applied in a CAD system, the basic properties of Bézier curves are key. Bézier curves are freeform curves where the underlying family $\{F_i(t)\}$ of functions is specified as the family of *Bernstein polynomials*.

Definition 3.25. Bernstein polynomials. For a given positive integer n the *Bernstein polynomials* $B_i^n(t)$ of degree n are defined as

$$B_{i,n}(t) = \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} = \frac{n!}{i! \cdot (n-i)!} \cdot t^i \cdot (1-t)^{n-i}, \quad i = 0, \ldots, n. \tag{3.59}$$

This family of polynomials plays an important role throughout this chapter and beyond.

Proposition 3.1 Properties of Bernstein polynomials. *The Bernstein polynomials $B_{i,n}(t)$ are polynomials of degree n (see Definition 3.5, p. 66). Further properties are:*

1. *The Bernstein polynomials are bounded within the interval* [0, 1]*:*

$$0 \le B_{i,n}(t) \le 1 \text{ for } t \in [0, 1].$$

2. *From the Binomial Theorem we have*

$$1 = (1 - t + t)^n = \sum_{i=0}^{n} \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i}$$

   which in turn implies

$$\sum_{i=0}^{n} B_{i,n}(t) \equiv 1. \tag{3.60}$$

   The Bernstein polynomials are a partition of unity (cf. (3.58)).
3. *The set of all Bernstein polynomials $\{B_{0,n}(t), \ldots, B_{n,n}(t)\}$ of degree n is a basis of the vector space $\mathbb{R}_n[t]$ of all polynomials of degree $\le n$ (Bernstein basis).*
4. *For $i = 1, \ldots, n-1$ we have the recursion formula*

$$B_{i,n}(t) = (1-t) \cdot B_{i,n-1}(t) + t \cdot B_{i-1,n-1}(t) \tag{3.61}$$

   and, as a supplement:

$$\begin{aligned} B_{0,n}(t) &= (1-t) \cdot B_{0,n-1}(t) \\ B_{n,n}(t) &= t \cdot B_{n-1,n-1}(t) \end{aligned} \tag{3.62}$$

5. *For every $i = 0, \ldots, n$ we have*

$$B_{n-i,n}(1-t) = B_{i,n}(t). \tag{3.63}$$

   The Bernstein polynomials are symmetric (see also Figs. 3.26 and 3.27).
6. *For the k-th derivatives $B_{i,n}^{(k)}(t)$ of the polynomial functions $B_{i,n}(t)$ at $t = 0$ and $t = 1$ we have*

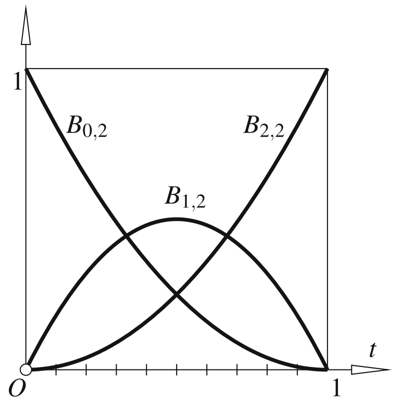


Fig. 3.26 The Bernstein basis functions of degree 2 on the unit interval [0, 1]. Mind that according to (3.60) their values add up to 1 for every t

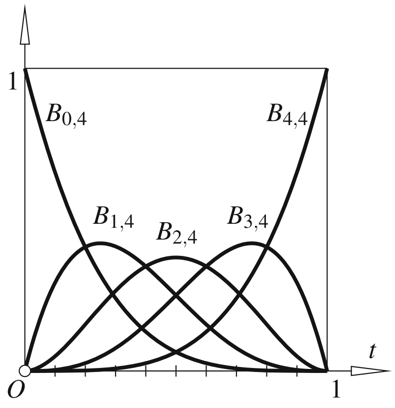


Fig. 3.27 The Bernstein basis functions of degree 4 on [0, 1]

$$B_{i,n}^{(k)}(0) = 0 \text{ for } i = k+1, \dots, n \tag{3.64}$$

$$B_{i,n}^{(k)}(1) = 0 \text{ for } i = 0, \dots, n-k-1 \tag{3.65}$$

The proofs to these claims are an easy exercise (see also [7], pp. 118 and [8], p. 38).

The Bernstein polynomials were introduced as early as in 1912 (see [9]) by Sergei Natanovich Bernstein (1880–1968). He used this family of polynomials to prove the so-called Stone-Weierstrass approximation theorem: *Every real continuous function on a closed interval can be uniformly approximated by polynomial functions*.

The Bézier curves are defined as the freeform curves (see Definition 3.24) where the family $\{F_i(t)\}$ of functions is specified as the family of Bernstein polynomials $\{B_{i,n}(t)\}$:

Definition 3.26. Bézier curve. Let $n+1$ points $\mathbf{a}_0, \dots, \mathbf{a}_n$ in 3-space or in the plane be given. The curve c defined by

$$\mathbf{p}(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot \mathbf{a}_i, \quad t \in [0, 1] \tag{3.66}$$

is called the *Bézier curve of degree n* to the given *control polygon* $\mathbf{a}_0, \ldots, \mathbf{a}_n$.

Basically, a Bézier curve of degree n is a polynomial curve of degree n. On the other hand, any polynomial curve of degree n can be represented as a Bézier curve of degree n with an appropriate control polygon. This follows from the fact that the set $\{B_{0,n}(t), \ldots, B_{n,n}(t)\}$ of Bernstein polynomials is a basis of $\mathbb{R}_n[t]$ (Proposition 3.1, 3.).

Proposition 3.2 Properties of Bézier curves. *Let a Bézier curve of degree n be defined by a control polygon* $\mathbf{a}_0, \ldots, \mathbf{a}_n$. *Then we have:*

1. $\mathbf{p}(0) = \mathbf{a}_0$, $\mathbf{p}(1) = \mathbf{a}_n$. *The Bézier curve and its control polygon have the same starting point* $\mathbf{a}_0$ *and the same end point* $\mathbf{a}_n$ *(see also Figs. 3.28 and 3.29).*
2. *Let* $k \in \{1, \ldots, n\}$ *be an integer. The k-th derivative vector* $\mathbf{p}^{(k)}(t)$ *at* $t = 0$ *only depends on the points* $\mathbf{a}_0, \ldots, \mathbf{a}_k$, $k \in \{0, \ldots, n\}$. *More detailed, we have:*

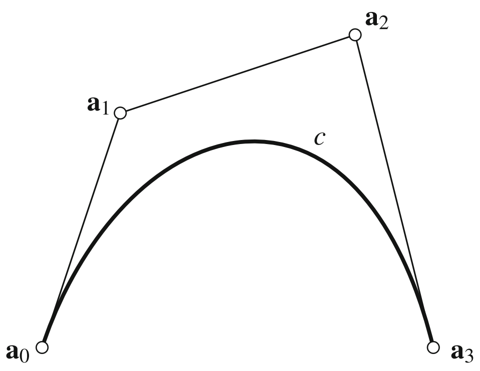


Fig. 3.28 A planar Bézier curve c of degree 3, determined by the control polygon $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$

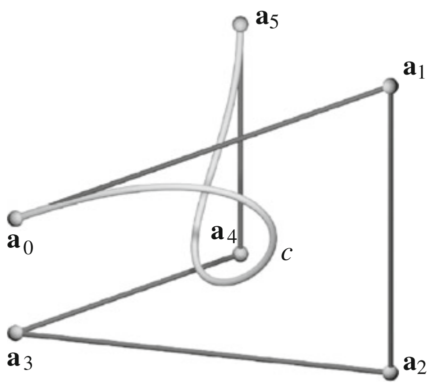


Fig. 3.29 A spatial Bézier curve c of degree 5 together with its control polygon $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5$

$$\mathbf{p}^{(k)}(0) = n \cdot (n-1) \cdot \ldots \cdot (n-k+1) \sum_{i=0}^{k} (-1)^i \cdot \binom{k}{i} \cdot \mathbf{a}_{k-i} \qquad (3.67)$$

Analogously: The k-th derivative vector $\mathbf{p}^{(k)}(t)$ *at* $t = 1$ *only depends on the points* $\mathbf{a}_n, \ldots, \mathbf{a}_{n-k}$, $k \in \{0, \ldots, n\}$*:*

$$\mathbf{p}^{(k)}(1) = n \cdot (n-1) \cdot \ldots \cdot (n-k+1) \sum_{i=0}^{k} (-1)^i \cdot \binom{k}{i} \cdot \mathbf{a}_{n-i} \qquad (3.68)$$

3. *The curve is invariantly connected with its control polygon* $\mathbf{a}_0, \ldots, \mathbf{a}_n$ *with respect to affine transformations (see also Remark 3.7, p. 86) and particularly with respect to isometries.*
4. *Exploiting the symmetric behavior (3.63) of the Bernstein polynomials we get*

$$\sum_{i=0}^{n} B_{i,n}(t) \cdot \mathbf{a}_i = \sum_{i=0}^{n} B_{n-i,n}(1-t) \cdot \mathbf{a}_i \;=\; \sum_{i=0}^{n} B_{i,n}(1-t) \cdot \mathbf{a}_{n-i}.$$

   The Bézier curve defined by the reversed control polygon is the same as the original curve c. Reversing the polygon only reverses the orientation of c. The overall curve c remains the same.
5. *The curve is completely contained in the convex hull of its control polygon. (convex-hull property, see Fig. 3.30). The property also holds for spatial Bézier curves where the convex hull of the control polygon is, in general, a polyhedron.*
6. *Let c be a planar Bézier curve to a control polygon p. We consider a straight line g and intersect it with c and p. The number of intersection points with c is less or equal to the number of intersection points with p (see Fig. 3.31). This interesting characteristic is usually called the* variation diminishing property. *It can also be formulated for spatial Bézier curves where the number of intersection points with some plane* γ *has to be regarded.*

Proposition 3.2 contains a lot of fundamental issues: Eq. (3.67) for $k = 1$ simply yields $\dot{\mathbf{p}}(0) = n \cdot (\mathbf{a}_1 - \mathbf{a}_0)$. This means that the first edge $\mathbf{a}_0\mathbf{a}_1$ of the control polygon is tangent to the Bézier curve c in the starting point $\mathbf{a}_0$. Equally, the last edge $\mathbf{a}_{n-1}\mathbf{a}_n$ is tangent to the curve c in the endpoint $\mathbf{a}_n$ (see Figs. 3.28 and 3.29).

Property 3.) follows from the fact that the Bernstein polynomials of degree n are a partition of unity. If α is an affine transformation (cf. Definition 3.1, p. 54) we have

$$\alpha(\mathbf{p}(t)) = \alpha\left(\sum_{i=0}^{n} B_{i,n}(t) \cdot \mathbf{a}_i\right) \;=\; \sum_{i=0}^{n} B_{i,n}(t) \cdot \alpha(\mathbf{a}_i). \qquad (3.69)$$

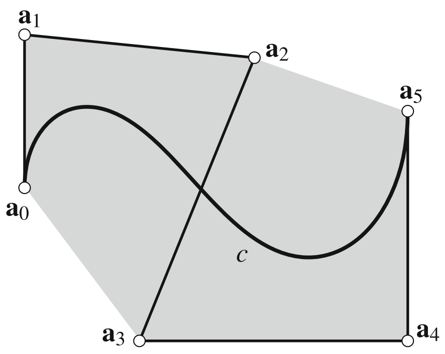


Fig. 3.30 Convex hull property: the planar Bézier curve c is contained in the convex hull of its control points $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5$ (*grey area*)

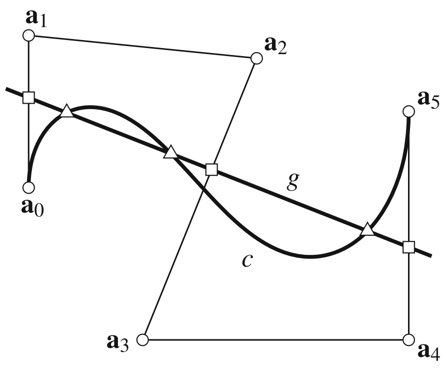


Fig. 3.31 Variation diminishing property a *straight line* g intersects the planar Bézier curve c in r points $\triangle$ and its control polygon in s points $\square$. The variation diminishing property of Bézier curves says: $r \leq s$ for every *straight line* g. Here we have $r = s = 3$

Applying α to the control polygon and then constructing the corresponding Bézier curve (right hand side of (3.69)) gets us the same thing as though we applied α to the Bézier curve $\mathbf{p}(t)$ in the first place (left hand side of (3.69)).

For the proofs see also ([7], pp. 115–128) or ([8], pp. 30–32 and 40–46).

Bézier Curves on an Arbitrary Support Interval

Obviously the Bernstein polynomials are perfectly adapted to the unit interval [0, 1] (Proposition 3.1, 1.), 6.)). Sometimes though, it is indispensable to transform them to an arbitrary interval $[\tau_0, \tau_1]$ by the linear substitution

$$t = \frac{\tau - \tau_0}{\tau_1 - \tau_0}. \tag{3.70}$$

We arrive at:

Definition 3.27. Bernstein polynomials, general form. For any positive integer n the generalized Bernstein polynomials are

$$B_{i,n}^{[\tau_0,\tau_1]}(\tau) = \frac{1}{(\tau_1 - \tau_0)^n} \binom{n}{i} \cdot (\tau - \tau_0)^i \cdot (\tau_1 - \tau)^{n-i}; \quad i = 0, \dots, n \tag{3.71}$$

By substituting (3.70) into the parametric representation (3.66) of a Bézier curve c we obtain the new parameterization of c:

$$\mathbf{q}(\tau) = \sum_{i=0}^{n} B_{i,n}^{[\tau_0,\tau_1]}(\tau) \cdot \mathbf{a}_i, \qquad \tau \in [\tau_0, \tau_1] \tag{3.72}$$

As we merely have applied a *linear* parameter transformation (3.70) the kth derivative vectors are only multiplied by a constant factor, namely $\frac{1}{(\tau_1-\tau_0)^k}$, which is a consequence of the chain rule of differentiation. In particular, the kth derivative vectors with respect to the new parameter τ at the starting point and at the end point are

$$\left.\begin{aligned} \mathbf{q}^{(k)}(\tau_0) &= \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{(\tau_1 - \tau_0)^k} \cdot \sum_{i=0}^{k} (-1)^i \cdot \binom{k}{i} \cdot \mathbf{a}_{k-i}, \\ \mathbf{q}^{(k)}(\tau_1) &= \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{(\tau_1 - \tau_0)^k} \cdot \sum_{i=0}^{k} (-1)^i \cdot \binom{k}{i} \cdot \mathbf{a}_{n-i}, \end{aligned}\right\} \tag{3.73}$$

which is a direct consequence of (3.67), (3.68).

For example, for $k = 1$ we obtain the tangent vectors at τ_0 and τ_1:

$$\left.\begin{aligned} \dot{\mathbf{q}}(\tau_0) &= \frac{n}{\tau_1 - \tau_0} \cdot (\mathbf{a}_1 - \mathbf{a}_0) \\ \dot{\mathbf{q}}(\tau_1) &= \frac{n}{\tau_1 - \tau_0} \cdot (\mathbf{a}_n - \mathbf{a}_{n-1}) \end{aligned}\right\} \tag{3.74}$$

The de Casteljau Algorithm

If we are to compute the point $\mathbf{p}(t)$ on a Bézier curve c to a given parameter value $t \in [0, 1]$ we sure can use (3.66) as proposed in Definition 3.26. However there are several reasons why users and software engineers prefer the following simple geometric algorithm instead (see Fig. 3.32 and the scheme (3.75)):

Algorithm 3.1. De Casteljau algorithm for Bézier curves. *To a given control polygon* $\mathbf{a}_0, \dots, \mathbf{a}_n$ *the point* $\mathbf{p}(t)$ *on the Bézier curve c can be constructed as follows (see also* (3.75)*):*

1. *Rename* $\mathbf{c}_{i,0} := \mathbf{a}_i$ *for all* $i = 0, \dots, n-1$. *Starting with the polygon* $\mathbf{c}_{0,0}, \dots, \mathbf{c}_{n,0}$, *to each segment* $\mathbf{c}_{i,0}\mathbf{c}_{i+1,0}$ *compute the point*[12]

$$\mathbf{c}_{i,1} := (1-t) \cdot \mathbf{c}_{i,0} + t \cdot \mathbf{c}_{i+1,0}.$$

[12] The point $\mathbf{c}_{i,1}$ is constructed by subdivision of the segment by the prescribed ratio $t : (1-t)$.

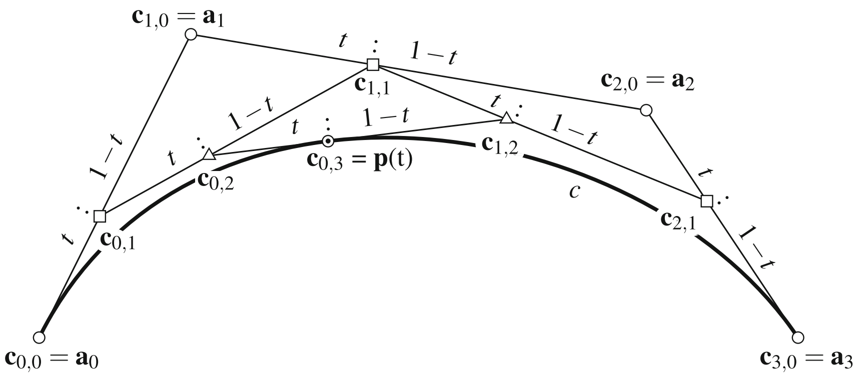

$\mathbf{c}_{1,0} = \mathbf{a}_1$
$t : 1-t$
$\mathbf{c}_{1,1}$
$\mathbf{c}_{2,0} = \mathbf{a}_2$
$1-t$
$t : 1-t$
$1-t$
$\mathbf{c}_{0,3} = \mathbf{p}(t)$
$\mathbf{c}_{1,2}$
t
$\mathbf{c}_{0,2}$
c
$\mathbf{c}_{2,1}$
t
$\mathbf{c}_{0,1}$
$1-t$
$\mathbf{c}_{0,0} = \mathbf{a}_0$
$\mathbf{c}_{3,0} = \mathbf{a}_3$

Fig. 3.32 The de Casteljau algorithm for a Bézier curve of degree $n = 3$ to some given parameter value $t \in [0, 1]$—here $t = 0.4$. The algorithm eventually delivers the point $\mathbf{c}_{0,3} = \mathbf{p}(t)$

This provides a new polygon $\mathbf{c}_{0,1}, \dots, \mathbf{c}_{n-1,1}$ *having one point less than the former.*

2. *Next apply the same routine to the new polygon* $\mathbf{c}_{0,1}, \dots, \mathbf{c}_{n-1,1}$ *and so on.*
3. *Every round of this iteration process reduces the number of points in the polygon by 1.*
4. *After n steps one arrives at one single point* $\mathbf{c}_{0,n}$.

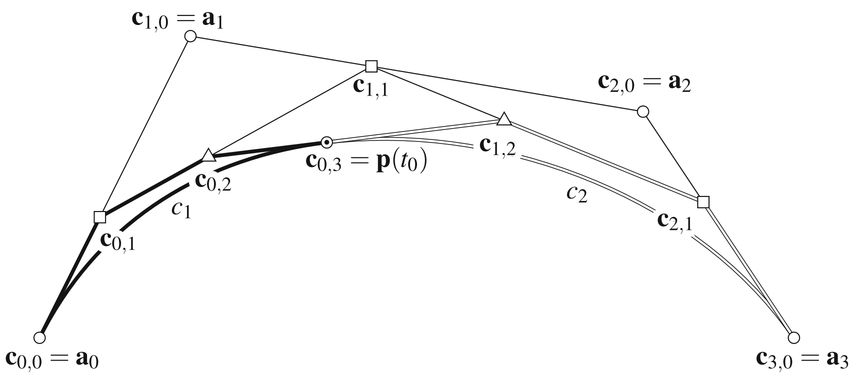
$\mathbf{c}_{0,n}$ *is just the desired point* $\mathbf{p}(t)$ *on the Bézier curve c.*

This construction can easily be proved by means of the recursion formula (3.61) (see also [8], pp. 167). The following scheme (3.75) briefly outlines the procedure.

$$\begin{array}{llllll}
\mathbf{c}_{0,0} = \mathbf{a}_0 & \mathbf{c}_{0,1} & * & * & * \; \mathbf{c}_{0,n-1} & \mathbf{c}_{0,n} = \mathbf{p}(t) \\
\mathbf{c}_{1,0} = \mathbf{a}_1 & \mathbf{c}_{1,1} & * & * & * \; \mathbf{c}_{1,n-1} & \\
\quad * & \quad * & * & * & * & \\
\quad * & \quad * & * & * & & \\
\quad * & \quad * & * & & & \\
\quad * & \mathbf{c}_{n-1,1} & & & & \\
\mathbf{c}_{n,0} = \mathbf{a}_n & & & & &
\end{array} \tag{3.75}$$

Figure 3.32 shows the de Casteljau algorithm graphically for the case $n = 3$. Basically, the de Casteljau algorithm is a series of *subdivisions* of straight line segments and hence, it consists of repeated linear combination of vectors. With this algorithm the evaluation of Bernstein polynomials can be avoided in the first place.

The de Casteljau algorithm once again shows what has already been stated in Proposition 3.1, 3.): The Bézier curve is affinely connected with its control polygon: Dividing a straight line segment by a given ratio is an affine construction. Moreover the convex hull property (Proposition 3.2),5.)) of a Bézier curve can easily be verified by the algorithm as—for $t \in [0, 1]$—each of the occurring points $\mathbf{c}_{i,j}$ lies in the interior of the line segment $[\mathbf{c}_{i,j-1}, \mathbf{c}_{i+1,j-1}]$; hence, no point of the Bézier curve can lie outside the convex hull of the control polygon.



$\mathbf{c}_{1,0} = \mathbf{a}_1$
$\mathbf{c}_{1,1}$
$\mathbf{c}_{2,0} = \mathbf{a}_2$
$\mathbf{c}_{1,2}$
$\mathbf{c}_{0,3} = \mathbf{p}(t_0)$
$\mathbf{c}_{0,2}$
c_1
c_2
$\mathbf{c}_{2,1}$
$\mathbf{c}_{0,1}$
$\mathbf{c}_{0,0} = \mathbf{a}_0$
$\mathbf{c}_{3,0} = \mathbf{a}_3$

Fig. 3.33 Splitting a Bézier curve at a point $\mathbf{p}(t_0)$ the de Casteljau algorithm for a Bézier curve c of degree $n = 3$ to the parameter value $t_0 = 0.4$. The algorithm delivers control polygons to each of the two sections into which c is split by the point $\mathbf{c}_{0,3} = \mathbf{p}(t_0)$

Continuation of a Bézier Curve

The de Casteljau scheme (3.75) applied to the parameter value $t = t_0$ does not only yield the point $\mathbf{c}_{0,n} = \mathbf{p}(t_0)$ on the Bézier curve c as described above. In fact it delivers a lot more (Fig. 3.33):

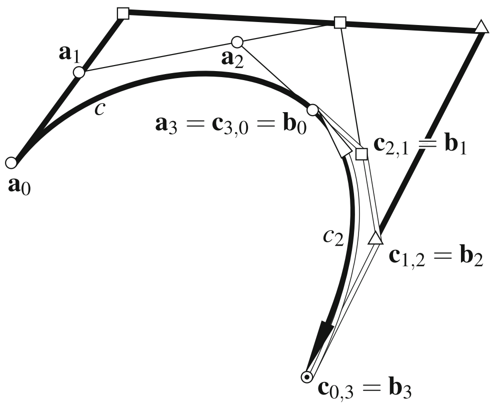
Construction 3.1 Splitting a Bézier curve into two segments. *The points* $\mathbf{c}_{0,0}$, $\mathbf{c}_{0,1}, \ldots, \mathbf{c}_{0,n-1}, \mathbf{c}_{0,n}$ *in the first row of the scheme (3.75) represent a control polygon to a Bézier curve* c_1 *which happens to be identical with the very part of* c *belonging to the parameter interval* $[0, t_0]$. *On the other hand, the points* $\mathbf{c}_{0,n}, \mathbf{c}_{1,n-1}, \ldots, \mathbf{c}_{n-1,1}, \mathbf{c}_{n,0}$ *in the diagonal of the scheme are the control points of another Bézier curve* c_2 *which incidentally is the remaining part of* c*, belonging to the parameter interval* $[t_0, 1]$.

The Bézier curve is usually regarded in the domain corresponding to the interval $[0, 1]$. As it is a polynomial curve we can as well extend the parameter area to any interval contained in $\mathbb{R}$ or even to $\mathbb{R}$ itself.

As a consequence, all the considerations above remain equally valid if we omit the restriction $t_0 \in [0, 1]$. If $t_0 > 1$ we end up in a point $\mathbf{p}(t_0)$ belonging to the *continuation of* c beyond the parameter interval $[0, 1]$ (Fig. 3.34).

In case of $t_0 > 1$ the two emerging parts c_1 and c_2 are parameterized by $[0, t_0]$ and $[t_0, 1]$. The first part c_1 ranges from the starting point $\mathbf{a}_0$ all across c to the point $\mathbf{c}_{0,n} = \mathbf{p}(t_0)$. The second part c_2 obtained from our splitting procedure starts at $\mathbf{c}_{0,n} = \mathbf{p}(t_0)$ and leads all the way back to the endpoint $\mathbf{a}_n$ (Fig. 3.34). In fact, it is part of c_1.

Up to now we have gained a continuation c_2 of the original Bézier curve c such that c_2 is defined by a separate control polygon $\mathbf{c}_{0,n}, \mathbf{c}_{1,n-1}, \ldots, \mathbf{c}_{n-1,1}, \mathbf{c}_{n,0}$. The two curves c and c_2 are joined at their junction point $\mathbf{a}_n$. They are GC^∞ (see Definition 3.18) which means that after some appropriate parameter transformation the new



$\mathbf{a}_1$
$\mathbf{a}_2$
c
$\mathbf{a}_3 = \mathbf{c}_{3,0} = \mathbf{b}_0$
$\mathbf{c}_{2,1} = \mathbf{b}_1$
$\mathbf{a}_0$
c_2
$\mathbf{c}_{1,2} = \mathbf{b}_2$
$\mathbf{c}_{0,3} = \mathbf{b}_3$

Fig. 3.34 Continuation of a Bézier curve of degree 3. The splitting algorithm applied for $t_0 = 1.65 > 1$ determines the continuation c_2 (*white*) of c (*black*) beyond its endpoint $\mathbf{a}_3$. The control polygon $\mathbf{c}_{0,3}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{3,0}$ of c_2 has to be reversed: $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$. The parameterization of c_2 is to be adjusted accordingly. Finally the two curves c and c_2 join at $\mathbf{a}_n = \mathbf{b}_0$ with C^∞-continuity

parameterizations could even be C^∞-continuous. It is not a big deal to reparameterize c_2 accordingly. To this end the following two steps are necessary:

1. We reverse the order (see Proposition 3.2, 4.) of the control points of c_2, arriving at $\mathbf{b}_0 := \mathbf{c}_{n,0}, \mathbf{b}_1 := \mathbf{c}_{n-1,1}, \ldots, \mathbf{b}_{n-1} := \mathbf{c}_{1,n-1}, \mathbf{b}_n := \mathbf{c}_{0,n}$. This control polygon again delivers the Bézier curve c_2, however starting at $\mathbf{c}_{n,0} = \mathbf{a}_n \ldots t = 0$ and ending at $\mathbf{c}_{0,n} \ldots t = 1$.
2. We now reparameterize this curve via (3.70), p. 91 putting $\tau_0 = 1$ and $\tau_1 = t_0$, i.e., $t = \frac{\tau-1}{t_0-1}$.

This new parametric representation is identical to $\mathbf{p}(t)$ within the interval $[1, t_0]$. We can summarize:

Construction 3.2 Continuation of a Bézier curve. *We apply the de Casteljau algorithm with some parameter* $t_0 > 1$. *The points* $\mathbf{c}_{0,n}, \mathbf{c}_{1,n-1}, \ldots, \mathbf{c}_{n-1,1}, \mathbf{c}_{n,0}$ *in the diagonal of the scheme (3.75) are the control points of a Bézier curve* c_2 *which represents the continuation of the segment* c *beyond the parameter value* $t = 1$. c_2 *belongs to the parameter interval* $[t_0, 1]$. *Reversing its control polygon gives it the correct orientation* $\mathbf{c}_{n,0} = \mathbf{b}_0, \mathbf{c}_{n-1,1} = \mathbf{b}_1, \ldots, \mathbf{c}_{1,n-1} = \mathbf{b}_{n-1}, \mathbf{c}_{0,n} = \mathbf{b}_n$ *and can be dealt with all on its own. In order to have a* C^∞*-continuation in terms of the parameterization we have to apply the parameter transformation* $t = \frac{\tau-1}{t_0-1}$.

The fact that we now have access to two separate control polygons to two adjacent segments of a Bézier curve will enable us to easily modify one of the segments without affecting the other.

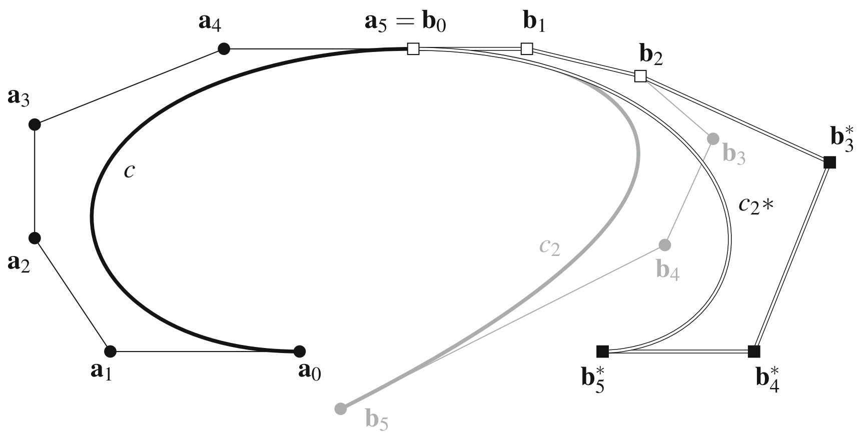


Fig. 3.35 We start with a Bézier curve c of degree $n = 5$ and apply the de Casteljau continuation algorithm (Construction 3.2) to its control polygon $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5$. We obtain a control polygon $\mathbf{a}_5 = \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5$ of the continuation segment c_2 which extends c with C^∞-continuity. The modification of c_2 amounts to replacing the points $\mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5$ by some modified (chosen) points $\mathbf{b}_3^*, \mathbf{b}_4^*, \mathbf{b}_5^*$. The modified curve c_2^* still extends c with C^2-continuity as the first three points $\mathbf{a}_5 = \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ remain unaltered

Modeling Bézier Curves

Modeling and shaping a curve is one fundamental task of any stylist. Powerful algorithms have to be complemented by creativity. The continuation of a given Bézier curve as described above can be the starting point. Depending on the desired degree of smoothness at the junction point of two segments particular points of the control polygon can be modified.

Construction 3.3 Modeling Bézier curves. *Let c be a Bézier curve with the control polygon $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_n$ (Fig. 3.35). Let $\mathbf{b}_0, \dots, \mathbf{b}_n$ be the control polygon of a continuation c_2 belonging to the parameter t_0 (Construction 3.2). We replace the control points $\mathbf{b}_{k+1}, \dots, \mathbf{b}_n$ of c_2 by some (deliberately chosen) new points $\mathbf{b}_{k+1}^*, \dots, \mathbf{b}_n^*$ and leave the first $k+1$ control points $\mathbf{b}_0, \dots, \mathbf{b}_k$ untouched.*

Then the Bézier curve c_2^* defined by the control polygon $\mathbf{b}_0, \dots, \mathbf{b}_k, \mathbf{b}_{k+1}^*, \dots, \mathbf{b}_n^*$ extends the original curve c C^k-continuously at the junction point.

The reason for C^k-continuity at the junction point lies in Proposition 3.2, 2.): The new curve c_2^* still yields the same derivative vectors up to order k as c_2 at the transition point $\mathbf{a}_n$ of c and c_2^*.

The choice of the parameter t_0 used for the continuation is essential, as it determines the first $k+1$ points of the control polygon to c_2^*. This parameter t_0 is commonly named *design parameter*.

In the same way as we have created the continuation beyond the endpoint $\mathbf{a}_n$ with some design parameter $t_0 > 1$ we could also generate a continuation in the other

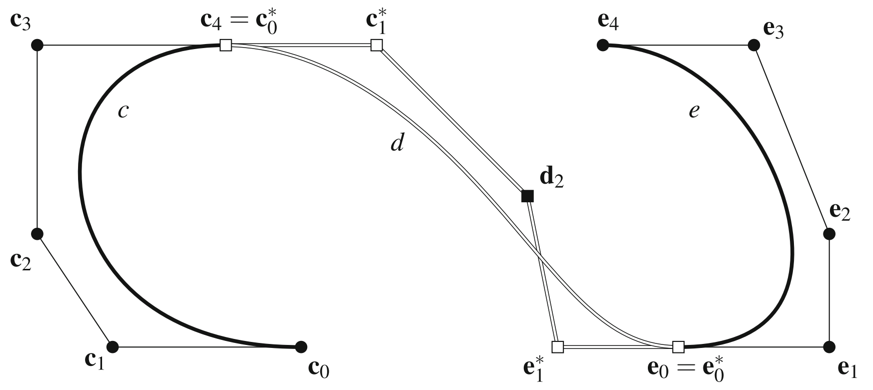


Fig. 3.36 Example for a C^1-connection to two given Bézier curves c and e (*black*), both of degree 4. The connecting curve d is again of degree 4. At either end d fits C^1-continuously. As we have $n = 4$ and $k = 1$ there is only one point $\mathbf{d}_2$ *left* to be chosen by the user

direction across the endpoint $\mathbf{a}_0$. To this avail we would have to choose a design parameter $t_0 < 0$.

One standard situation in curve design is the task of creating a smooth connection between two given Bézier curves (Fig. 3.36).

Construction 3.4 Connecting two Bézier curves. *We start with two Bézier curves c and e, both of degree n. Let k be a non-negative integer with $k \le \frac{n-1}{2}$. In order to connect c and e by a C^k-continuous transition curve the following steps have to be taken:*

- *Construct a continuation c^* of c with a design parameter $t_0 > 1$ via Construction 3.3. The resulting control polygon be $\mathbf{c}_0^*, \dots, \mathbf{c}_n^*$.*
- *Analogously, construct a continuation e^* of e with some design parameter $s_0 < 0$; $\mathbf{e}_0^*, \dots, \mathbf{e}_n^*$ be the control polygon of e^*.*

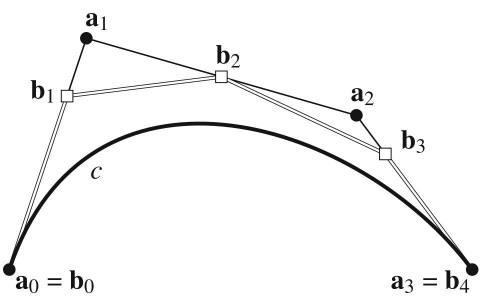


Fig. 3.37 Bézier curve c ($n = 3$). The control polygon $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ is subjected to the degree elevation algorithm (3.76). The new control polygon $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4$ determines the same Bézier curve c—now formally of degree 4. In this case the segments of the control polygon are subdivided by the ratios 3 : 1, 2 : 2, and 1 : 3

- *Generate a new series of points as follows:* $\mathbf{c}_0^*, \ldots, \mathbf{c}_k^*, \mathbf{d}_{k+1}, \ldots, \mathbf{d}_{n-k-1}, \mathbf{e}_k^*, \ldots, \mathbf{e}_0^*$. *The points* $\mathbf{d}_{k+1}, \ldots, \mathbf{d}_{n-k-1}$ *of this series can be deliberately chosen by the user.*
  The Bézier curve d *to this control polygon delivers a transition between* c *and* e *which is a* C^k*-continuation at either end.*

Obviously we have to observe the restriction $k \geq \frac{n-1}{2}$. The number of points $\mathbf{d}_{k+1}, \ldots, \mathbf{d}_{n-k-1}$ which are left for the user to be chosen is $n - 2k - 1$. E.g., for $n = 4$ and $k = 1$ (C^1-continuous connection at either end) we have $n - 2k - 1 = 1$; so, there is only one point left for manipulation.

The number $n + 1$ of control points obviously determines the polynomial degree n of a Bézier curve c. Larger values of n will deliver more freedom for modeling and/or a higher degree k of continuity at the junction point. Though it is not always preferable to use high degree Bézier curves, it sometimes may be useful to increase the degree n.

Degree Elevation

In some cases it may be beneficial to represent a Bézier curve of degree n as a higher degree Bézier curve. It is easy to see that such a thing is always possible, as every polynomial of degree n can be displayed in the Bernstein basis of degree $n + 1$. This is how *degree elevation* basically works.

Let c be a Bézier curve of degree n with the control polygon $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_n$. In order to represent the curve c as a Bézier curve of degree $n + 1$, the control polygon is to be replaced by some control polygon $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}$. The points of the new control polygon can easily be computed. Basically, we have to express the Bernstein polynomials of degree n in the Bernstein basis of degree $n + 1$: We—formally—multiply the representation $\mathbf{p}(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot \mathbf{a}_i$ by the factor $1 = (1 - t) + t$ and collect the summands in terms of the Bernstein polynomials $B_{i,n+1}(t)$. This immediately yields the recipe for the construction of the new control polygon:

$$\begin{aligned}\mathbf{b}_0 &= \mathbf{a}_0\\ \mathbf{b}_k &= \frac{k}{n+1} \cdot \mathbf{a}_{k-1} + \Big(1 - \frac{k}{n+1}\Big) \cdot \mathbf{a}_k, \quad k = 1, \ldots, n\\ \mathbf{b}_{n+1} &= \mathbf{a}_n\end{aligned} \tag{3.76}$$

The construction consists of subdivisions, where the ratios $(1 - \frac{k}{n+1}) : \frac{k}{n+1}$ are variable, depending on the index k. Figure 3.37 illustrates the procedure.

Construction 3.5 Degree elevation of a Bézier curve. *Every Bézier curve of degree* n *can be represented as a Bézier curve of degree* $n + 1$. *The corresponding points of the new control polygon can be constructed from the original control polygon via (3.76).*

To give an example, we consider the modeling of Bézier curves (Construction 3.3). In order to achieve C^k-continuity for the continuation c_2 of a Bézier curve c we have to keep the first $k + 1$ points $\mathbf{b}_0, \ldots, \mathbf{b}_k$ untouched. The other control points can be used for modeling. If there are too few remaining points, degree elevation my well be a good strategy as it could provide more leverage for the stylist.

3.4.2 B-Spline Curves

Continuing a Bézier curve and modeling it according to Sect. 3.4.1 enables us to construct a curve consisting of an arbitrary number of polynomial segments of degree n with C^k-continuity at the transition points. The idea of piecing a curve together this way leads to the concept of a *spline curve*.

The term *spline* is commonly used in the CAD context. It denotes a curved line element. In the olden days spline curves were materialized by sleek wooden beams (stringers) used in the construction of ship bodies. The stringers were forced to run through given points on the predefined cross-sections (frames). These points fractionize the stringer into several segments. Before we delve into particular types of spline curves we consider the concept more generally:

Definition 3.28. Spline functions and spline curves. Mathematically splines are defined as follows:

1. The real numbers in the series $t_0 < t_1 < \cdots < t_m$ are called *knots*. The ordered set $(t_0, t_1, \ldots, t_m)$ is called a *knot vector*.
2. Let a knot vector $(t_0, t_1, \ldots, t_m)$ be given. A function $F(t), t \in [t_0, t_m]$ consisting of polynomial segments $f_j(t), t \in [t_j, t_{j+1}]$ of degree $k-1$ is called a *spline function* if it is $(k-2)$-times differentiable at the knots. This means that consecutive polynomials $f_j(t)$, $f_{j+1}(t)$ have identical derivatives up to order $k-2$ at t_{j+1}. If the order of differentiability at these points is less than $k-2$ we call it *subspline function*.
3. The set of all spline functions of degree $k-1$ to a given knot vector $(t_0, t_1, \ldots, t_m)$ shall be denoted by $\mathscr{S}_{k-1}(t_0, t_1, \ldots, t_m)$.
4. A freeform curve $\mathbf{p}(t) = \sum_{i=0}^{m} F_i(t) \cdot \mathbf{a}_i$ is called *spline curve* (*subspline curve*) if the functions $F_i(t)$ are spline functions (subspline functions) with a common knot vector $(t_0, t_1, \ldots, t_m)$.
5. The points $\mathbf{p}(t_i)$ belonging to the knots are also called *junction points* of the spline curve.

The proof to the following remark can be found in ([8], p. 158).

Remark 3.8 The set of all spline functions $\mathscr{S}_{k-1}(t_0, t_1, \ldots, t_m)$ is a vector space of dimension

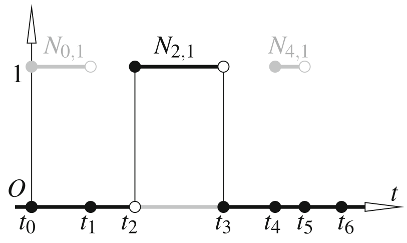


Fig. 3.38 The B-spline basis functions $N_{i,1}(t)$ for $i = 0$, $i = 2$ and $i = 4$. For the sake of clarity the ones for $i = 1$ and $i = 3$ are not displayed. As an example the basis function $N_{2,1}(t)$ is highlighted in *black*

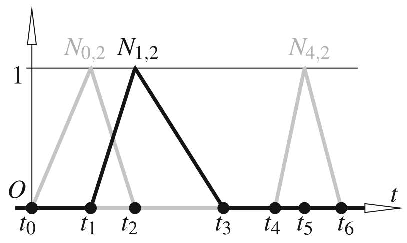


Fig. 3.39 The B-spline basis functions $N_{i,2}(t)$ for $i = 0, 1, 4$. The basis functions $N_{3,2}(t)$, $N_{4,2}(t)$ in between have been *left* out. The basis function $N_{1,2}(t)$ is highlighted in *black*

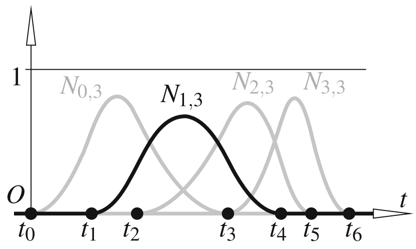


Fig. 3.40 The B-spline basis functions $N_{i,3}(t)$ for $i = 0, 1, 2, 3$. $N_{1,3}(t)$ is highlighted in *black*

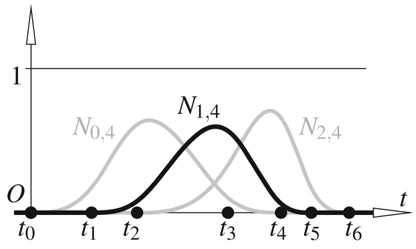


Fig. 3.41 The B-spline basis functions $N_{i,4}(t)$ for $i = 0, 1, 2$. $N_{1,4}(t)$ is highlighted in *black*

$$d = \dim \mathscr{S}_{k-1}(t_0, t_1, \ldots, t_m) = m + k - 1. \tag{3.77}$$

Remark 3.9 The definition of spline functions as given above can be generalized by allowing multiple knots (cf. Definition 3.32, p. 106). This means that the assumption $t_0 < t_1 < \cdots < t_m$ is replaced by $t_0 \le t_1 \le \cdots \le t_m$. The emerging *generalized spline functions* have slightly different properties; for instance, the differentiability at a multiple knot decreases. The dimension formula (3.77) also has to be adapted.

Multiple knots can be used to shape the spline functions in a way. We will use this concept in the case of B-spline functions.

An in-depth introduction to spline theory can be found in [10, 11] or [12]. In this paragraph we particularly discuss one prominent example of spline curves: the *B-spline curves*. Remember that the introduction of Bézier curves (Sect. 3.4.1) was preceded by the definition of the corresponding basis functions (Bernstein polynomials). In the case of B-spline curves we equally have to define a specific set of suitable functions.

Definition 3.29. B-spline basis functions. Let n and $k < n$ be positive integers and let $(t_0, t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+k})$ be a knot vector. Then we define

- in case of $k = 1$:

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } \; t_i \le t < t_{i+1} \\ 0 & \text{if } \; \text{else} \end{cases} \tag{3.78}$$

- and in case of $k > 1$:

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} \cdot N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} \cdot N_{i+1,k-1}(t) \tag{3.79}$$

where $i = 0, \ldots, n$. The functions $N_{i,k}(t)$ are called the *B-spline basis functions*.

We have:

Proposition 3.3 Properties of B-spline basis functions. *The following properties of the B-spline basis functions are easy to check (see also [7], pp. 167 and [8], p. 38):*

1. $N_{i,k}(t) > 0$ *for* $t \in (t_i, t_{i+k})$ *and* $N_{i,k}(t) = 0$ *anywhere else (see Figs. 3.38, 3.39, 3.40, 3.41).*
2. $N_{i,k}(t)$ *consists of polynomial segments of degree* $k-1$*, each defined in an interval* $[t_j, t_{j+1}]$ *between two consecutive knots.*
3. *Each B-spline basis function* $N_{i,k}(t)$ *is* $(k-2)$*-times differentiable at each knot* $t_i, t_{i+1}, \ldots, t_{i+k}$*. Everywhere else it is smooth.*
4. *Within the interval* $[t_l, t_{l+1}]$ *the only B-spline basis functions different from* 0 *are* $N_{l-k+1,k}(t), \ldots, N_{l,k}(t)$*.*
5. *The B-spline basis functions are a partition of unity:*

$$\sum_{i=0}^{n} N_{i,k}(t) \equiv 1 \tag{3.80}$$

6. *The B-spline basis functions* $N_{0,k}(t), \ldots, N_{n,k}(t)$ *establish a basis of the linear function space* $\mathscr{S}_{k-1}(t_{k-1}, \ldots, t_{n+1})$*. Their number* $n+1$ *coincides with (3.77).*

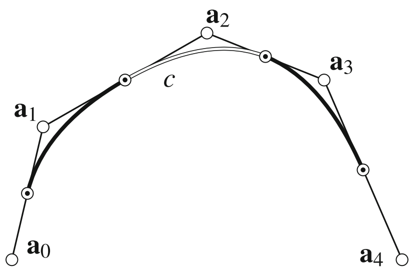


Fig. 3.42 B-spline curve: $n = 4$ and $k = 3$. If the knot vector is uniform (see Definition 3.31, p. 103) the junction points are the midpoints of the control polygon edges. The different sections of the B-spline curve are parabola arcs tangent to the control polygon

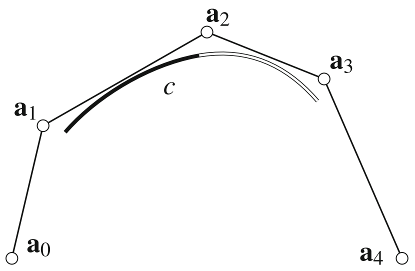


Fig. 3.43 B-spline curve: $n = k = 4$. Adjacent segments are joined C^2-continuously. According to Definition 3.30 the parameter interval $[t_{k-1}, t_{n+1}] = [t_3, t_5]$ provides only two segments with their junction point $\mathbf{p}(t_4)$

Based on the functions $N_{i,k}(t)$ we now define the *B-spline curves*.[13]

Definition 3.30. B-spline curve. We are given two positive integers k, n with $k < n + 2$ and a knot vector $(t_0 < t_1 < \cdots < t_n < t_{n+1} < \cdots < t_{n+k})$ as in Definition 3.28. We additionally have $n + 1$ points $\mathbf{a}_0, \ldots, \mathbf{a}_n$ in 3-space (or in the plane). The curve c with the parametric representation

$$\mathbf{p}(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot \mathbf{a}_i, \quad t \in [t_{k-1}, t_{n+1}] \tag{3.81}$$

is called the *B-spline curve* to the given *control polygon* $\mathbf{a}_0, \ldots, \mathbf{a}_n$ and the given knot vector $(t_0, \ldots, t_{n+k})$. The points $\mathbf{a}_i$ are called *control points* or *de Boor points*.

[13] The properties of B-splines were essentially known in the nineteenth century. One of the first mathematicians to investigate the properties of such spline functions was J.C.F. Haase (1870, see [13]). The introduction of B-spline curves dates back to the Romanian mathematician I. Schoenberg (1903–1990). The theory of approximation by B-splines and algorithms for working with them are—among others—owed to the German-American mathematician Carl de Boor (born 1937 in Stolp, today Slupsk, Poland).

Note that B-spline curves are one example of freeform curves according to Definition 3.24 where the family of functions is specified as the set of B-spline basis functions.

Due to Proposition 3.3, 3.) the differentiability class of the B-spline basis functions is C^{k-2} and thus, the same holds for the B-spline curves. According to Fig. 3.38 the B-Spline curve for $k = 1$ would not be worth mentioning: It consists of the discrete vertices of the control polygon and would not even deserve the term *curve*. For $k = 2$ we would obtain a B-spline curve which is identical to the edges of the control polygon which is C^0-continuous (without gaps) but still pointed at the vertices. For $k \geq 3$ the B-spline curve (see Figs. 3.42 and 3.43) is differentiable.

Definition 3.31. Uniform and non-uniform B-spline curves. If the knot intervals $[t_i, t_{i+1}]$ are of equal lengths we call the knot vector *uniform*. A B-spline curve to such a knot vector is called a *uniform B-spline curve*. Knot vectors with non-uniform distribution yield *non-uniform B-spline curves*.

Proposition 3.4 Properties of B-spline curves.

1. *Owing to Proposition 3.3, 4.) we have*

$$\mathbf{p}(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot \mathbf{a}_i = \sum_{i=l-k+1}^{l} N_{i,k}(t) \cdot \mathbf{a}_i \tag{3.82}$$

   for $t \in [t_l, t_{l+1}]$. All the other B-spline basis functions vanish within this interval. As a consequence: The segment of the B-spline curve relating to the interval $[t_l, t_{l+1}]$ only depends on the k control points $\mathbf{a}_{l-k+1}, \ldots, \mathbf{a}_l$.
2. *The curve is invariantly connected with its control polygon $\mathbf{a}_0, \ldots, \mathbf{a}_n$ with respect to affine transformations (see also Remark 3.7, p. 86).*
3. *A B-spline curve is smooth within each subinterval $[t_i, t_{i+1}]$. It is C^{k-2}-continuous at the junction points $\mathbf{p}(t_i)$.*

We remember that the polynomial degree of a Bézier curve is 1 less than the number of control points. As for B-spline curves we see: The polynomial degree is $k - 1$ and hence does not depend on the number $n + 1$ of control points. Increasing n arbitrarily does not affect the polynomial degree which is a great benefit.

Proposition 3.4, 1.) indicates that changing the control point $\mathbf{a}_l$ will only affect the segments of the B-spline curve relating to $[t_l, t_{l+k}]$. All other segments remain unaltered (see Fig. 3.44). This is generally referred to as the *local control property* of B-spline curves.

The Cox-de Boor Algorithm

For Bézier curves we easily obtained the point $\mathbf{p}(t)$ belonging to the parameter value t by employing the de Casteljau scheme (3.75), p. 93. For B-spline curves we are

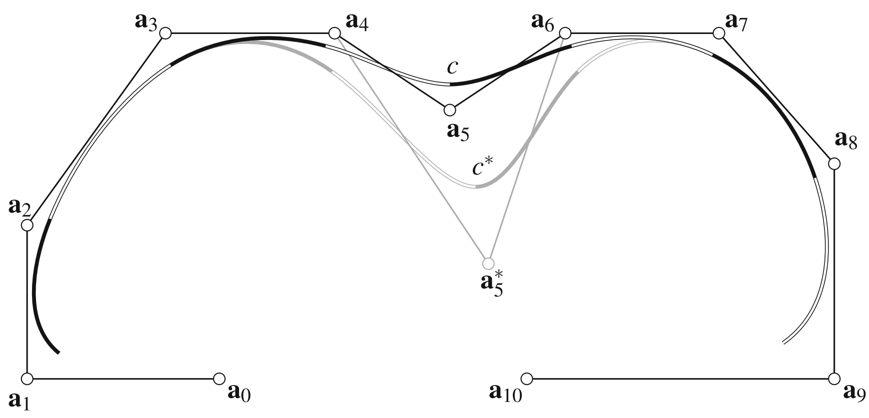


Fig. 3.44 Local control of a B-spline curve, $k = 4$. Changing the control point $\mathbf{a}_5$ to $\mathbf{a}_5^*$ changes the 4 segments of the B-spline curve which relate to the intervals $[t_5, t_6]$, $[t_6, t_7]$, $[t_7, t_8]$, $[t_8, t_9]$. The segments relating to $[t_3, t_4]$, $[t_4, t_5]$ and $[t_9, t_{10}]$, $[t_{10}, t_{11}]$ remain unchanged

lucky to have a similarly powerful method, called[14] *Cox-de Boor algorithm*. Starting with the set of control points (and the knot vector) we recursively define a series of points (see Fig. 3.45) and arrive at a truly helpful scheme (3.83):

Algorithm 3.2. Cox-de Boor algorithm. *The control polygon* $\mathbf{a}_0, \dots, \mathbf{a}_n$ *and the knot vector* $(t_0, \dots, t_{n+k})$ *of a B-spline curve c be given. In order to obtain the point* $\mathbf{p}(t)$ *on c belonging to a given parameter value* $t \in [t_{k-1}, t_{n+1}]$ *we take the following steps:*

1. *Determine* l *with* $t_l \le t < t_{l+1}$*; in case of* $t = t_{n+1}$ *put* $l = n$.
2. *Put* $\mathbf{d}_{l-k+i,0} := \mathbf{a}_{l-k+i}$ *for* $i = 1, \dots, k$.
3. *For* $j = 1, \dots, k-1$ *and* $i = 1, \dots, k-j$ *compute the points*

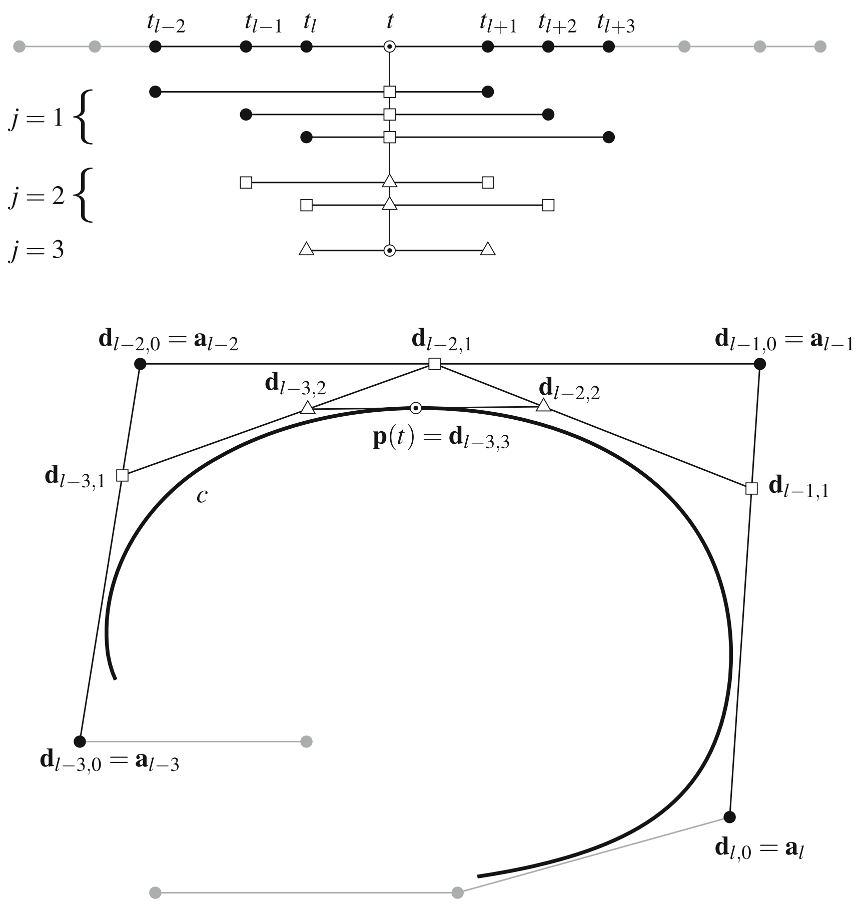
$$\mathbf{d}_{l-k+i,j} := (1 - \beta(i,j,k,l,t)) \cdot \mathbf{d}_{l-k+i,j-1} + \beta(i,j,k,l,t) \cdot \mathbf{d}_{l-k+i+1,j-1}$$

with

$$\beta(i,j,k,l,t) := \frac{t - t_{l+i-k+j}}{t_{l+i} - t_{l+i-k+j}}.$$

4. *The desired point appears at the right top end of the scheme (3.83):* $\mathbf{p}(t) = \mathbf{d}_{l-k+1,k-1}$.

[14] As for C.R. de Boor see p. 107.



t_{l-2}
t_{l-1}
t_l
t
t_{l+1}
t_{l+2}
t_{l+3}
$j = 1$
$j = 2$
$j = 3$
$\mathbf{d}_{l-2,0} = \mathbf{a}_{l-2}$
$\mathbf{d}_{l-2,1}$
$\mathbf{d}_{l-1,0} = \mathbf{a}_{l-1}$
$\mathbf{d}_{l-3,2}$
$\mathbf{d}_{l-2,2}$
$\mathbf{p}(t) = \mathbf{d}_{l-3,3}$
$\mathbf{d}_{l-3,1}$
$\mathbf{d}_{l-1,1}$
c
$\mathbf{d}_{l-3,0} = \mathbf{a}_{l-3}$
$\mathbf{d}_{l,0} = \mathbf{a}_l$

Fig. 3.45 The Cox-de Boor algorithm for a B-spline curve c with $k = 4$ to some given parameter value $t \in [t_{k-1}, t_{n+1}]$. The algorithm is based on successive subdivision as in the de Casteljau scheme for Bézier curves. However, the division ratio in the Cox-de Boor algorithm varies from step to step as illustrated (on *top*). The algorithm results in the point $\mathbf{p}(t) \in c$

$$\begin{array}{cccccccc}
\mathbf{d}_{l-k+1,0} & \mathbf{d}_{l-k+1,1} & * & * & * & \mathbf{d}_{l-k+1,k-2} & \mathbf{d}_{l-k+1,k-1} = \mathbf{p}(t) \\
\mathbf{d}_{l-k+2,0} & \mathbf{d}_{l-k+2,1} & * & * & * & \mathbf{d}_{l-k+2,k-2} & \\
* & * & * & * & * & & \\
* & * & * & * & & & \\
* & * & * & & & & \\
* & \mathbf{d}_{l-1,1} & & & & & \\
\mathbf{d}_{l,0} & & & & & &
\end{array} \tag{3.83}$$

This construction is based on the recursive Definition 3.39 of the B-Spline basis functions. For the proof see ([8], p. 154ff).

Endpoint Interpolation

As for Bézier curves we are familiar with the fact that they start at the first point of the control polygon and are even tangent to its first edge. The same thing happens at the end point. We call this property the *endpoint interpolation property*. In general B-spline curves behave differently (see Figs. 3.43 or 3.45). But they are extremely versatile. If desired we can easily tweak the knot vector such that the resulting B-spline curve also has the endpoint interpolation property. To this end we have to allow the concept of multiple knots:

Definition 3.32. Multiple knots. Let $(t_0, t_1, \dots, t_{n+k})$ be a knot vector and let l be an integer with $l \in \{0, \dots, k-1\}$. If

$$t_{i-1} < t_i = t_{i+1} = \cdots = t_{i+l} < t_{i+l+1}$$

for some $i \in \{0, \dots, n+k-l\}$ then t_i is called a *knot with multiplicity* $l+1$.

If we set $\frac{0}{0} := 0$ the recursive Definition 3.29, p. 100 of the B-spline basis functions is still valid in case of knot vectors with multiple knots in the sense of Definition 3.32.

For example, in the knot vector $(t_0, t_0, t_1, t_1, t_1, t_2)$ the knots t_0 and t_1 occur with multiplicity 2 and 3, respectively. Another typical example is the knot vector

$$(\underbrace{t_0, \dots, t_0}_{k-\text{fold}}, \underbrace{t_1, \dots, t_1}_{k-\text{fold}}), \tag{3.84}$$

which consists of only two distinct knots t_0, t_1 each of them with multiplicity k. It can easily be shown that in that case a B-spline curve $\mathbf{p}(t)$ to this knot vector is identical with the Bézier curve which has the DeBoor polygon of $\mathbf{p}(t)$ as control polygon (see also [7], p. 173). Hence, we can state:

Proposition 3.5 *Every Bézier curve can be represented as a B-spline curve.*

In this sense B-spline curves can be seen as a generalization of Bézier curves. Additionally we have:

Proposition 3.6 Endpoint interpolation property. *Let c be a B-spline curve with some control polygon $\mathbf{a}_0, \dots, \mathbf{a}_n$ and a knot vector with knots of multiplicity k at either end:*

$$t_0 = t_1 = \cdots = t_{k-1},\ t_k, \dots, t_n,\ t_{n+1} = t_{n+2} = \cdots = t_{n+k} \tag{3.85}$$

Then the B-spline curve c has the endpoint interpolation property (see Figs. 3.46 and 3.47):

- *c starts at $\mathbf{a}_0$ and ends at $\mathbf{a}_n$:*

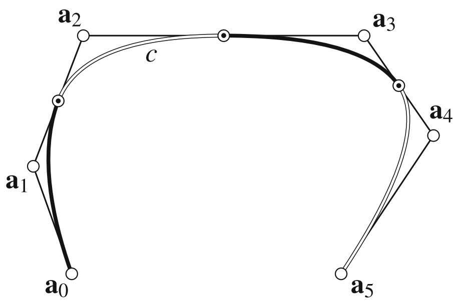


Fig. 3.46 Endpoint interpolation of a B-spline curve for $n = 5$ and $k = 3$

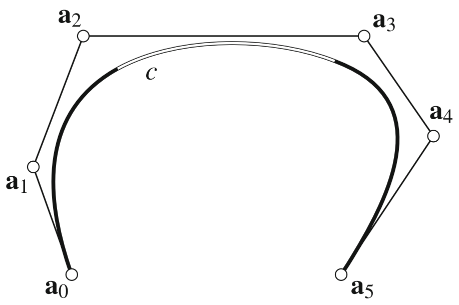


Fig. 3.47 Endpoint interpolation of a B-spline curve for $n = 5$ and $k = 4$

$$\mathbf{p}(t_{k-1}) = \mathbf{a}_0, \quad \mathbf{p}(t_{n+1}) \;=\; \mathbf{a}_n \tag{3.86}$$

- *c is tangent to the first edge* $\mathbf{a}_0\mathbf{a}_1$ *of the control polygon at the starting point and to the last edge* $\mathbf{a}_{n-1}\mathbf{a}_n$ *at its end point. More detailed, the tangent vectors in those points are*

$$\dot{\mathbf{p}}(t_{k-1}) = \frac{k-1}{t_k - t_{k-1}} \cdot (\mathbf{a}_1 - \mathbf{a}_0), \quad \dot{\mathbf{p}}(t_{n+1}) \;=\; \frac{k-1}{t_{n+1} - t_n} \cdot (\mathbf{a}_n - \mathbf{a}_{n-1}) \tag{3.87}$$

The k coinciding knots $t_0 = \cdots = t_{k-1}$ enforces the B-spline curve to behave just like a Bézier curve of degree $k - 1$ whose first two control points are $\mathbf{a}_0$ and $\mathbf{a}_1$—at least up to the first derivative. The same holds for the end point.

Closed B-Spline Curves

A curve is generally called *closed* if its end point coincides with its starting point. Such curves are also called *loops*. Other curves are called *open*.

B-splines can easily be adapted to deliver loops if desired. Not only can we manage that the end point coincides with the starting point, we even can achieve that—at this point—the emerging curve is also C^{k-2}-continuous just as at any other knot point $\mathbf{p}(t_i)$. To this end we have to adapt the knot vector and make some arrangements:

Proposition 3.7 Closed B-spline curve. *We start with a B-spline curve defined by a de Boor polygon* $\mathbf{a}_0, \ldots, \mathbf{a}_n$ *and a knot vector* $(t_0, \ldots, t_{n+k})$. *First we continue the control polygon periodically by putting* $\mathbf{a}_{n+i} = \mathbf{a}_{i-1}$ *for* $i = 1, \ldots, k-1$. *Additionally, the knot vector is continued by adding* $k-1$ *new knots in the following way:*

$$t_{n+k+i} = t_{n+k} - t_{k-1} + t_{k-1+i}, \quad i = 1, \ldots, k-1 \tag{3.88}$$

Then the B-spline curve c defined by the control polygon $\mathbf{a}_0, \ldots, \mathbf{a}_{n+k-1}$ *and the knot vector* $t_0, \ldots, t_{n+2k-1}$ *is a closed curve. The parameter interval for one complete round is* $[t_{k-1}, t_{n+k}]$. *We have* $\mathbf{p}(t_{k-1}) = \mathbf{p}(t_{n+k})$*; even the derivatives up to order* $k-2$ *coincide at* t_{k-1} *and* t_{n+k}. *The bonding point* $\mathbf{p}(t_{k-1}) = \mathbf{p}(t_{n+k})$ *behaves just like any other junction point* $\mathbf{p}(t_i)$.

Engineering applications often call for closed curves which are all but seamless. This is exactly what such B-spline applications are capable of. Adapting the order k

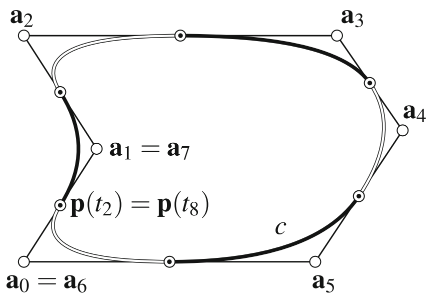

$\mathbf{a}_2$
$\mathbf{a}_3$
$\mathbf{a}_4$
$\mathbf{a}_1 = \mathbf{a}_7$
$\mathbf{p}(t_2) = \mathbf{p}(t_8)$
c
$\mathbf{a}_0 = \mathbf{a}_6$
$\mathbf{a}_5$

Fig. 3.48 Closed B-spline curve c for $n = 5$ and $k = 3$. The number of points being repeated at the end of the de Boor polygon is $k - 1 = 2$. The bonding point is $\mathbf{p}(t_2) = \mathbf{p}(t_8)$

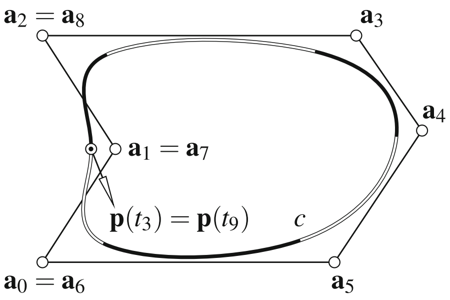

$\mathbf{a}_2 = \mathbf{a}_8$
$\mathbf{a}_3$
$\mathbf{a}_4$
$\mathbf{a}_1 = \mathbf{a}_7$
$\mathbf{p}(t_3) = \mathbf{p}(t_9)$
c
$\mathbf{a}_0 = \mathbf{a}_6$
$\mathbf{a}_5$

Fig. 3.49 Closed B-spline curve c for $n = 5$ and $k = 4$. There are $k - 1 = 3$ de Boor points being repeated at the end of the de Boor polygon. Apart from this, the de Boor polygon is the same as in the example on the *left*. The bonding point of the curve is $\mathbf{p}(t_3) = \mathbf{p}(t_9)$

will easily allow for any desired degree of smoothness all over the loop. Figures 3.48 and 3.49 show two examples with C^1- and C^2-continuity even at the junction points.

3.4.3 Rational Freeform Curves, NURBS

Having selected an appropriate family of basis functions $F_i(t),\ i = 0, \dots, n$ any control polygon determines a freeform curve (see Definition 3.24). We now describe the more general notion of *rational* freeform curves.

We consider a polygon $\mathbf{a}_0, \dots, \mathbf{a}_n$ in the $[x, y]$-plane:

$$\mathbf{a}_i = \begin{bmatrix} a_{i,1} \\ a_{i,2} \end{bmatrix}, \quad i = 0, \dots, n.$$

To define a rational freeform curve $\mathbf{r}(t)$ we choose one further value w_i for every control point $\mathbf{a}_i$. This additional real number is called *the weight of the point* $\mathbf{a}_i$. To construct $\mathbf{r}(t)$ we apply the following three steps (see Fig. 3.50):

Step 1: We assume that the plane π containing the control polygon $\mathbf{a}_0, \dots, \mathbf{a}_n$ is embedded into 3-space $\mathbb{E}_3$. We choose a spatial coordinate system Σ (origin O^*, coordinate axes x_0, x_1, x_2) in $\mathbb{E}_3$ so that the plane π of the control polygon is given by $\pi \dots x_0 = 1$. More precisely, a point $\mathbf{p}$ with coordinates x, y in π has coordinates $x_0 = 1, x_1 = x, x_2 = y$ with respect to Σ.

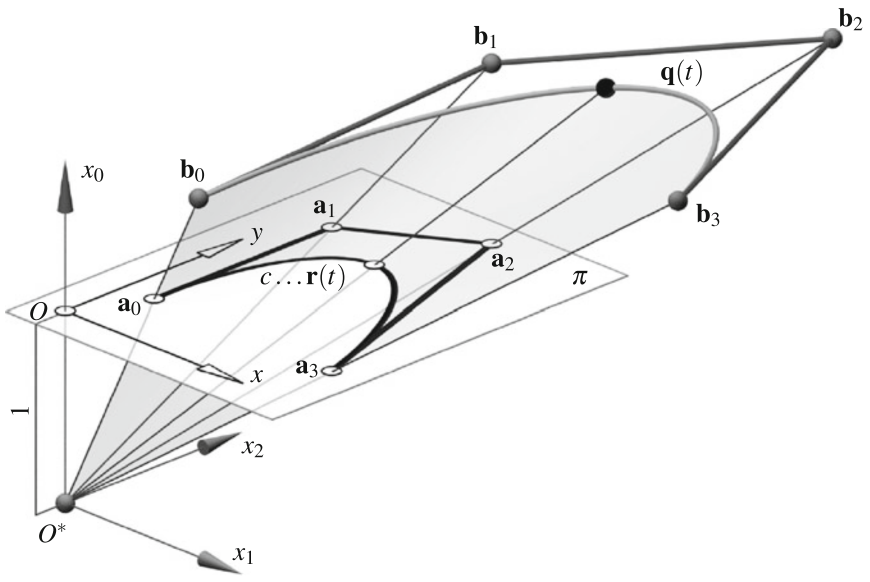


Fig. 3.50 The planar rational freeform curve $c \dots \mathbf{r}(t)$ can be interpreted as the central projection of an ordinary spatial freeform curve $\mathbf{q}(t)$

Step 2: Every point $\mathbf{a}_i$ is now viewed as a 3-vector $[1, a_{i,1}, a_{i,2}]^\top$. Its weight w_i shall be used as the scale factor of a dilation from the center O^*. The dilation gets $\mathbf{a}_i$ into a point $\mathbf{b}_i = [w_i, w_i a_{i,1}, w_i a_{i,2}]^\top$. This new point $\mathbf{b}_i$ bears the information of both, the coordinates of $\mathbf{a}_i$ and its weight w_i. The points $\mathbf{b}_0, \ldots, \mathbf{b}_n$ form a new (spatial) polygon which can be used as a control polygon of a spatial freeform curve

$$\mathbf{q}(t) = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{b}_i, \quad t \in [t_0, t_1] \tag{3.89}$$

with basis functions $F_i(t)$ as defined in (3.57), p. 86. Moreover, we require that the $F_i(t)$ are polynomial functions or at least consist of polynomial segments.

Step 3: We now consider the central projection δ from the center O^* into the plane $\pi \ldots x_0 = 1$. A point $\mathbf{q} = [x_0, x_1, x_2]^\top$ in 3-space with $x_0 \neq 0$ is mapped to its image point $\mathbf{p} = [1, \frac{x_1}{x_0}, \frac{x_2}{x_0}]^\top$. We omit the coordinate $x_0 = 1$ and put $\delta(\mathbf{q}) = [\frac{x_1}{x_0}, \frac{x_2}{x_0}]^\top =: [x, y]^\top$. The rational freeform curve $\mathbf{r}(t)$ is then defined as the central projection of the ordinary (polynomial) freeform curve $\mathbf{q}(t)$ into the plane π:

$$\mathbf{r}(t) = \delta(\mathbf{q}(t)) \;=\; \frac{\sum\limits_{i=0}^{n} F_i(t) \cdot w_i \cdot \mathbf{a}_i}{\sum\limits_{i=0}^{n} F_i(t) \cdot w_i} \tag{3.90}$$

Summarizing we can give the following

Definition 3.33. Rational freeform curve. The *rational freeform curve* $\mathbf{r}(t)$ to a given control polygon $\mathbf{a}_0, \ldots, \mathbf{a}_n$, given weights $w_0 \neq 0, \ldots, w_n \neq 0$ and polynomial basis functions $F_0(t), \ldots, F_n(t)$ is defined by the parameterization (3.90).

Remark 3.10 The fore-mentioned recipe shows how to create a *rational* freeform curve from an ordinary one.

(a) For the sake of clarity we started with a *planar* control polygon $\mathbf{a}_i = [a_{i,1}, a_{i,2}]^\top$, $i = 0, \ldots, n$ to define a *planar* rational freeform curve. We can easily generalize this concept to the case of a *spatial* rational freeform curve by starting with a *spatial* control polygon $\mathbf{a}_i = [a_{i,1}, a_{i,2}, a_{i,3}]^\top$, $i = 0, \ldots, n$. Definition 3.33 is still valid, because adding a further coordinate does not change anything significantly. We subsequently carry out a central projection. In the planar case this projection led from the 3-space into the plane, whereas in case of spatial rational freeform curves it leads from the 4-space containing $\mathbf{q}(t)$ into the 3-space containing the resulting rational curve $\mathbf{r}(t)$.
The very reason why we initially confined ourselves to planar rational freeform curves was that the interpretation as a central projection from 3-space to the plane is easier to imagine.

(b) A rational freeform curve (3.90) is well-defined for all parameters t except for the zeroes of the denominator $\sum_{i=0}^{n} F_i(t) \cdot w_i$.

(c) The *weights* w_i which can additionally be chosen provide more leverage. The range of curves which can be represented this way, is considerably wider than with ordinary freeform curves.

How do the weight factors affect the curve shape? Of course, the answer depends on the type of the freeform curve (i.e., Bézier curve, B-spline curve, etc.). But still we can generally confirm:

Proposition 3.8 Properties of rational freeform curves. *Let $c \ldots \mathbf{r}(t)$ be a rational freeform curve defined by the control polygon $\mathbf{a}_0, \ldots, \mathbf{a}_n$, the weights $w_0, \ldots, w_n$ and the polynomial basis functions $F_0(t), \ldots, F_n(t)$ according to (3.90).*

1. *The shape of c can be modified without altering the control polygon, just by changing some of the weights w_i.*
2. *Let $w_0, \ldots, w_n$ be positive. If one weight w_{i_0} is increased to $w_{i_0} + \Delta$ with $\Delta > 0$ then $c \ldots \mathbf{r}(t)$ turns into a new curve $c^* \ldots \mathbf{r}^*(t)$. Moreover, for every value t the points $\mathbf{r}(t)$, $\mathbf{r}^*(t)$ and $\mathbf{a}_{i_0}$ are collinear (see Fig. 3.51).*
   If the functions $F_i(t)$ are non-negative within their support interval[15] *we can even observe that for a positive Δ the point $\mathbf{r}(t)$ moves towards $\mathbf{a}_{i_0}$. The new point $\mathbf{r}^*(t)$ is closer to $\mathbf{a}_{i_0}$ than $\mathbf{r}(t)$. More detailed we have*
$$\mathbf{r}^*(t) - \mathbf{r}(t) = \frac{F_{i_0}(t) \cdot \Delta}{\sum_{i=0}^{n} F_i(t) \cdot w_i - F_{i_0}(t) \cdot \Delta}(\mathbf{a}_{i_0} - \mathbf{r}(t)) \tag{3.91}$$
   as can easily be verified.
   Increasing the weight of a control point really puts more emphasis on this point.
3. *Multiplying all weights with the same constant factor $\alpha \neq 0$ would not change the resulting rational freeform curve: The common factor could be cancelled in the denominator and numerator of (3.90).*
4. *If all weights w_i are equal ($w_0 = \cdots = w_n = w \neq 0$) the rational freeform curve is identical to the ordinary freeform curve $\mathbf{p}(t) = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i$. This is a consequence of the partition of unity (3.58) of the functions $F_i(t)$:*
$$\mathbf{r}(t) = \frac{\sum\limits_{i=0}^{n} F_i(t) \cdot w \cdot \mathbf{a}_i}{\sum\limits_{i=0}^{n} F_i(t) \cdot w} = \frac{w \cdot \sum\limits_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i}{w \cdot \underbrace{\sum\limits_{i=0}^{n} F_i(t)}_{=1}} = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i.$$

[15] Bézier and B-spline curves are defined by basis functions with this handy property. So rational Bézier or rational B-spline curves comply with this rule.

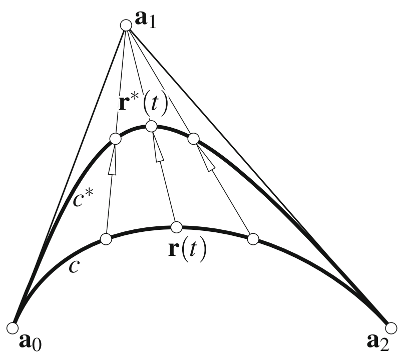


Fig. 3.51 Rational Bézier curve c of degree 2. Increasing the weight w_1 to $w_1 + \Delta$ delivers a different rational Bézier curve c^*. Increasing the weight w_1 moves every point towards the control point $\mathbf{a}_1$

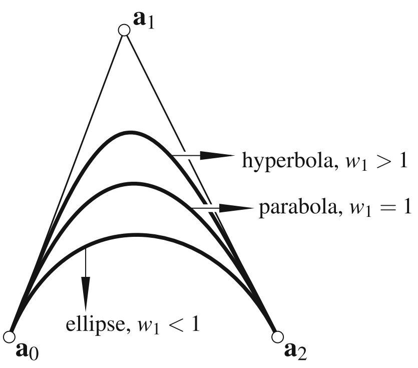


Fig. 3.52 A rational Bézier curve of order 2 with weights $w_0, w_2 = 1$ at the entpoints. Depending on the weight w_1 we c an create an ellipse, a parabola or a hyperbola c

In this paragraph we have addressed rational freeform curves in a general context. This is to say that we have not specified the defining family $F_i(t), i = 0, \ldots, n$ of polynomials. Whenever we go for a particular family of such functions we possibly can render more details of the corresponding rational freeform curve.

Rational Bézier Curves

Remember the Bernstein polynomials $B_{i,n}(t)$ of degree n see Definition 3.25, p. 87). Specifying $F_i(t) = B_{i,n}(t)$ we obtain the parameterization of a *rational Bézier curve*:

$$\mathbf{r}(t) = \frac{\sum\limits_{i=0}^{n} B_{i,n}(t) \cdot w_i \cdot \mathbf{a}_i}{\sum\limits_{i=0}^{n} B_{i,n}(t) \cdot w_i} \tag{3.92}$$

Ordinary (polynomial) Bézier curves bring about lots of limitations. Even in the case $n = 2$ these limitations become obvious. Every polynomial Bézier curve of degree $n = 2$ is some part of a parabola (see Theorem 3.6, p. 85).

It is impossible to represent the other types of regular 2nd order curves (hyperbolae, ellipses, and esp. circles) as polynomial Bézier curves. We now use rational Bézier curves to fill this gap. This will be feasible as arcs of circles, ellipses or hyperbolae can be parameterized by rational functions.

As every rational curve of degree 2 represents a planar[16] algebraic curve of order 2 (see Sect. 3.3.9, p. 82), a rational Bézier curve c of degree 2 is an arc of a parabola, a hyperbola or an ellipse (circles included) whenever the three control points $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$ are not collinear. If they are collinear c is a straight line segment.

The general representation of a rational Bézier curve c of degree 2 is

$$\begin{aligned}\mathbf{r}(t) &= \frac{\sum\limits_{i=0}^{2} B_{i,2}(t)\cdot w_i \cdot \mathbf{a}_i}{\sum\limits_{i=0}^{2} B_{i,2}(t)\cdot w_i} \\ &= \frac{(1-t)^2\cdot w_0\cdot \mathbf{a}_0 + 2\cdot t\cdot (1-t)^2\cdot w_1\cdot \mathbf{a}_1 + t^2\cdot w_2\cdot \mathbf{a}_2}{(1-t)^2\cdot w_0 + 2\cdot t\cdot (1-t)^2\cdot w_1 + t^2\cdot w_2}. \end{aligned} \tag{3.93}$$

Proposition 3.9 Conics as rational Bézier curves. *Let c be a rational Bézier curve of degree 2 with the parameterization (3.93) and let moreover $w_0 = w_2 = 1$ (cf. Fig. 3.52).*

$$\textit{Then } c \textit{ is an arc of} \left\{ \begin{array}{l} \textit{an ellipse if } 0 < w_1 < 1. \\ \textit{a parabola if } w_1 = 1. \\ \textit{a hyperbola if } w_1 > 1. \end{array} \right\}$$

Now we can easily offer a (rational) Bézier representation of any circular arc:

Proposition 3.10 Circular arcs as rational Bézier curves. *Let c be a circular arc centered in $\mathbf{m}$ with radius r and endpoints $\mathbf{a}_0, \mathbf{a}_2$. Let $\varphi < \pi$ be the central angle of c and let $\mathbf{a}_1$ be the intersection point of the tangents at $\mathbf{a}_0$ and $\mathbf{a}_2$ (see Fig. 3.53).*

If we choose the weights $w_0 = w_2 = 1$ and put $w_1 = \cos\frac{\varphi}{2}$ for a rational Bézier curve $\mathbf{r}(t)$ of degree 2 then $\mathbf{r}(t)$ is a parametric representation of the given circular arc c.

The interpretation of rational freeform curves (or particularly rational Bézier curves) as central projections of ordinary freeform curves put us in the position to assess the effect of increasing or reducing the weights. In the same ways we can interpret the effect of positive or negative weights. But all the same, we confine ourselves to positive weights.

$\mathbf{r}(t)$ be a (planar) rational Bézier curve, obtained by central projection of the (ordinary) spatial Bézier curve $\mathbf{q}(t)$. If all the weights w_i are positive the control polygon of $\mathbf{q}(t)$ is entirely contained in the upper half space $x_0 > 0$. Owing to the convex hull property (Proposition 3.2, 5., p. 89) the same thing holds for the curve $\mathbf{q}(t)$ itself.

[16] Note that any rational curve of degree 2 is necessarily planar.

Projecting the points of $\mathbf{q}(t)$ centrally from the origin O^* onto the plane $\pi : x_0 = 1$ is trouble-free; the denominator in (3.92) has no zeroes within the interval [0, 1].

If there is some weight $w_i < 0$, though, we cannot guarantee this any more. It may well happen that for some particular points on $\mathbf{q}(t)$ the denominator in (3.92) vanishes. The projection ray is parallel to the image plane $x_0 = 1$ and thus intersects in a point at infinity of the resulting rational Bézier curve $\mathbf{r}(t)$. In the computation such singular cases have to be addressed adequately.

Nevertheless, in some cases negative weights may well be useful, as the following proposition shows.

Proposition 3.11 Complementary conic arcs. *We consider a rational Bézier curve* $\mathbf{r}(t)$ *of degree* 2*. Its weights be* $w_0 = w_2 = 1$ *and* $w_1 > 0$ *(see Fig. 3.54). Due to Proposition 3.9 the arc* $\mathbf{r}(t)$ *lies on a conic section c. Replacing the weight* w_1 *by* $w_1^* = -w_1$ *we obtain another rational Bézier arc* $\mathbf{r}^*(t)$ *with the same endpoints* $\mathbf{a}_0$*,* $\mathbf{a}_2$ *(see Fig. 3.54). We have:*

$\mathbf{r}^*(t)$ *and* $\mathbf{r}(t)$ *are complementary arcs on the same conic section c.*

This can particularly be applied to circular arcs described in Proposition 3.10. Replacing $w_1 = \cos\frac{\varphi}{2}$ by $w_1 = -\cos\frac{\varphi}{2}$ amounts to the same thing as replacing φ by $2\pi - \varphi$.

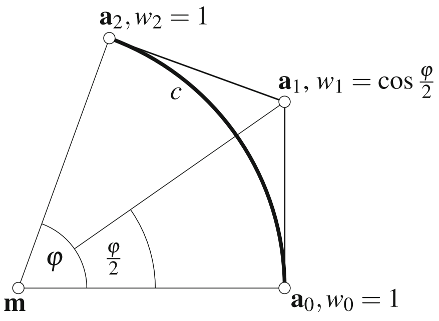


Fig. 3.53 Circular arc, represented as a rational Bézier curve c: $w_0 = w_2 = 1,\ w_1 = \cos\frac{\varphi}{2}$

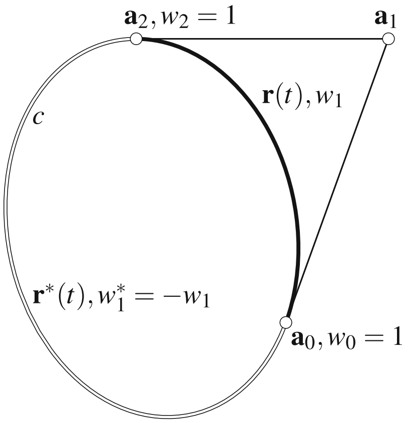


Fig. 3.54 Two complementary arcs $\mathbf{r}(t)$ and $\mathbf{r}^*(t)$ on a conic section c, represented as rational Bézier curves

Rational B-Spline Curves, NURBS

As we could already see the B-spline curves are an extremely versatile tool for the engineer. We now extend this concept to *rational* B-spline curves which even gives us a lot of additional options.

Definition 3.34. Rational B-spline curve. Let a (planar or spatial) control polygon $\mathbf{a}_0, \ldots, \mathbf{a}_n$, weight factors $w_0, \ldots, w_n$ be given. Let $N_{i,k}(t)$, $i = 0, \ldots, n$ be the B-spline basis functions belonging to the knot vector $(t_0, t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+k})$. Then we call

$$\mathbf{r}(t) = \frac{\sum\limits_{i=0}^{n} N_{i,k}(t) \cdot w_i \cdot \mathbf{a}_i}{\sum\limits_{i=0}^{n} N_{i,k}(t) \cdot w_i} \tag{3.94}$$

the corresponding rational B-spline curve.

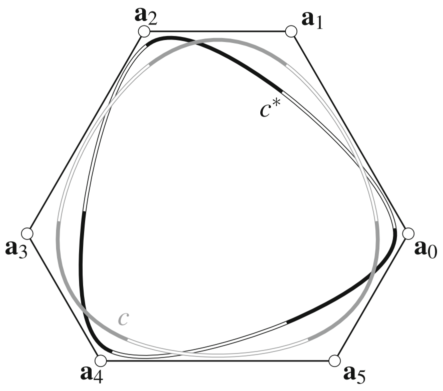


Fig. 3.55 Two closed rational B-spline curves c and c^* with $k = 4$. As for c (*grey*) all weights $w_i = 1$, so c is an ordinary B-Spline curve (see Proposition 3.8, 4., p. 111). For c^* (black) the weights w_0, w_2, w_4 have been increased to 3

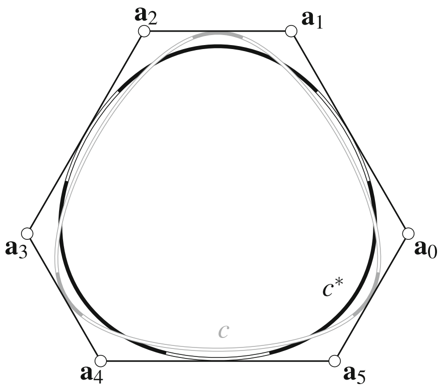


Fig. 3.56 Two B-spline curves c and c^*. Here, the weights of c (*grey*) remained untouched. The variation c^* (*black*) is due to a modification of the knot vector

The knot vector of a rational B-spline curve can be uniform or non-uniform (see Definition 3.31, p. 103).

Definition 3.35. NURBS. Rational B-spline curves with non-uniform knot vectors are generally called *NURBS* (*N*on-*U*niform *R*ational *B*-*S*pline).

NURBS offer lots of parameters to control the shape: Apart from modifying the de Boor polygon the user can subtly change the weights or the knot vector. This versatility may be the reason why NURBS have emerged as one of the standard tools in CAD and graphics. Figures 3.55 and 3.56 illustrate some of the options of modification applied to closed B-spline curves.

3.5 Univariate Interpolation

Interpolation deals with the problem of finding a function (out of a given preselected class) which adopts certain prescribed data. If the function to be found depends on *one* variable the task is called *univariate interpolation*. The following interpolation problem frequently occurs in practical applications:

Problem 3.1 Univariate interpolation of real values. Let parameter values $s_0 < \dots < s_n \in \mathbb{R}$ within a closed interval $[a, b]$ be given as well as $n + 1$ corresponding values $c_0, \dots, c_n \in \mathbb{R}$ (see also Fig. 3.57). Moreover let $\mathscr{F}$ be a linear space[17] of continuous functions on $[a, b]$. We further demand[18]: $\dim \mathscr{F} = n + 1$.

Find a function $p(t) \in \mathscr{F}$ such that

$$p(s_j) = c_j, \quad j = 0, \dots, n. \tag{3.95}$$

The problem can be addressed as follows. We choose a basis $\{F_0(t), \dots, F_n(t)\}$ in $\mathscr{F}$; then the interpolating function $p(t)$ to be found must have a unique representation of the form

$$p(t) = \sum_{i=0}^{n} F_i(t) \cdot a_i. \tag{3.96}$$

Substituting this representation into the interpolation conditions (3.95) we obtain the $n + 1$ linear equations

[17] This means that $\mathscr{F}$ has the structure of a vector space: With $F_1(t), F_2(t) \in \mathscr{F}$ and $\lambda \in \mathbb{R}$ the functions $F_1(t) + F_2(t)$ and $\lambda \cdot F_1(t)$ also belong to $\mathscr{F}$. The set $\mathscr{F}$ is *closed* with respect to sums and scalar multiples.

[18] This is equivalent to the existence of a basis $\{F_0(t), \dots, F_n(t)\}$ consisting of $n + 1$ elements.

$$\sum_{i=0}^{n} F_i(s_j) \cdot a_i = c_j, \quad j = 0, \dots, n \tag{3.97}$$

in the $n + 1$ yet unknown coefficients a_i.

We can easily rewrite (3.97) in matrix form as follows:

$$\begin{bmatrix} F_0(s_0) & \dots & F_n(s_0) \\ \vdots & & \vdots \\ F_0(s_n) & \dots & F_n(s_n) \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \tag{3.98}$$

At this point it is clear that the interpolation task has a unique solution if and only if the coefficient matrix

$$\mathbf{F} := \begin{bmatrix} F_0(s_0) & \dots & F_n(s_0) \\ \vdots & & \vdots \\ F_0(s_n) & \dots & F_n(s_n) \end{bmatrix} \tag{3.99}$$

of the linear system is invertible, i.e., $\det \mathbf{F} \neq 0$. In that case the solution is

$$\begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{F}^{-1} \cdot \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix}. \tag{3.100}$$

The following concept proves particularly helpful.

Definition 3.36. Chebyshev space. Let $\mathscr{F}$ be a linear space of continuous functions on an interval $[a, b]$ with $\dim \mathscr{F} = n + 1$.

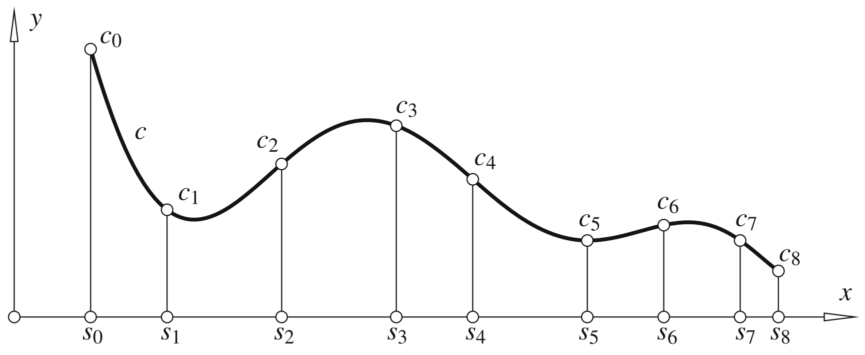


Fig. 3.57 Nine values $c_0, \dots, c_8$ are prescribed. The curve c is the solution to the Interpolation Problem 3.2 in the space $\mathbb{R}_8[t]$ of polynomials of degree 8

1. $\mathscr{F}$ is called a *Chebyshev space*[19] if for any basis $\{F_0(t), \ldots, F_n(t)\}$ of $\mathscr{F}$ and for any series $s_0 < s_1 < \cdots < s_n$ of real values in the interval $[a, b]$ the matrix $\mathbf{F}$ (Eq. (3.99)) is invertible.
2. Any basis of a Chebyshev space is called a *Chebyshev system*.

We mention a few valuable characterizations of Chebyshev spaces:

Theorem 3.7. Characterization of Chebyshev spaces. *Let the assumptions of Definition 3.36 be fulfilled. The following items are equivalent:*

1. *$\mathscr{F}$ is a Chebyshev space.*
2. *Every function $p(t)$ in $\mathscr{F}$ has no more than n zeroes in $[a, b]$.*
3. *Every interpolation task akin to Problem 3.1 has a unique solution in $\mathscr{F}$.*

Example 3.10 We consider the linear space $\mathbb{R}_n[t]$ of polynomials of degree $\leq n$ (see Sect. 3.2, pp. 65), however viewed as continuous functions on the interval $[a, b]$. This is a Chebyshev space of dimension $n + 1$. We have already got to know two different bases of this space: the monomial basis $\{1, t, t^2, \ldots, t^n\}$ and the Bernstein basis $\{B_{0,n}(t), \ldots, B_{n,n}(t)\}$ (see p. 87). Problem 3.1 has a unique solution in $\mathbb{R}_n[t]$. Any choice of a basis in $\mathbb{R}_n[t]$ delivers an approach. Each of such approaches, of course, leads to the same solution. The key of the matter is the coefficient matrix $\mathbf{F}$ (Eq. (3.99)) whose inverse delivers the solution (3.100).

This matrix $\mathbf{F}$, however, depends on the choice of the basis. The monomial basis, for one, delivers the so-called *Vandermonde-Matrix*

$$\mathbf{F} = \begin{bmatrix} 1 & s_0 & \ldots & s_0^n \\ \vdots & & & \vdots \\ 1 & s_n & \ldots & s_n^n \end{bmatrix} \tag{3.101}$$

whose determinant is

$$\det \mathbf{F} = \prod_{0 \leq i < j \leq n} (s_j - s_i) \neq 0.$$

It would be a quite natural question whether there is a basis of $\mathbb{R}_n[t]$ such that $\mathbf{F}$ is the unit matrix. This question will be answered positively in Sect. 3.5.1.

Example 3.11 The space of trigonometric functions spanned by the functions $B = \{1, \sin t, \cos t, \sin(2t), \cos(2t), \ldots, \sin(nt), \cos(nt)\}$ on the interval $[0, 2\pi]$ is a Chebyshev space of dimension $2n + 1$ and B is a basis.

The interpolation of values $c_i \in \mathbb{R}$ can easily be generalized to the interpolation of data in a k-dimensional space. The following problem deals with the case $k = 2$ and $k = 3$.

[19] Named after the Russian mathematician Pafnuty Lvovich Chebyshev (1821–1894). Sometimes these spaces are also called *Haar spaces* after the Hungarian mathematician Alfred Haar (1885–1933).

Problem 3.2 Univariate interpolation of points. Let $n+1$ points $\mathbf{c}_0, \ldots, \mathbf{c}_n$ in the plane or in 3-space and corresponding parameter values $s_0 < \cdots < s_n$ in a closed interval $[a, b]$ be given. Furthermore let $\mathscr{F}$ be a linear space of continuous functions on $[a, b]$ as in Problem 3.1. Let $\{F_0(t), \ldots, F_n(t)\}$ be a basis of $\mathscr{F}$.

Find a curve of the form

$$\mathbf{p}(t) = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i \tag{3.102}$$

interpolating the given data, i.e.,

$$\mathbf{p}(s_j) = \mathbf{c}_j, \quad j = 0, \ldots, n. \tag{3.103}$$

The solution is given by

Proposition 3.12 Solvability of the interpolation problem. *If the matrix* $\mathbf{F}$ *defined by (3.99) is invertible there is a unique solution to Problem 3.2; the coefficient vectors* $\mathbf{a}_i$ *belonging to this solution are*[20]

$$\begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \mathbf{F}^{-1} \cdot \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_0 \end{bmatrix}. \tag{3.104}$$

Remark 3.11 To compute the inverse to a large matrix $\mathbf{F}$ can sometimes be numerically challenging. Depending on the entries of the matrix large sets of input data—in special cases—may even render the method inapplicable.

3.5.1 Lagrange Interpolation

In case of the polynomial space $\mathscr{F} = \mathbb{R}_n[t]$ the most traditional approach to Problem 3.2 is owing to J.-L. Lagrange.[21]

For a given knot vector $(s_0, s_1, \ldots, s_n)$ we consider the set $\{L_{0,n}(t), \ldots, L_{n,n}(t)\}$ of polynomials in $\mathbb{R}_n[t]$ defined by

$$L_{i,n}(t) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{t - s_k}{s_i - s_k}, \quad i = 0, \ldots, n. \tag{3.105}$$

They are called *Lagrange polynomials*; they form a basis of $\mathbb{R}_n[t]$, the so-called *Lagrange basis*. Moreover, we have

[20] Of course, one has to interprete Eq. (3.104) for every coordinate of the vectors $\mathbf{a}_i$, $\mathbf{c}_i$ separately!

[21] Joseph-Louis Lagrange (1736–1813) was a French mathematician who also contributed to analytical mechanics and mathematical physics.

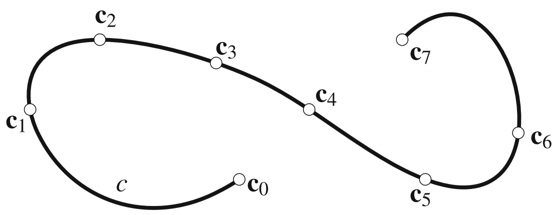


Fig. 3.58 The Lagrange solution to Interpolation Problem 3.2 with input points $\mathbf{c}_0, \ldots, \mathbf{c}_7$ and a uniform knot vector $s_0, \ldots, s_7$. The resulting interpolation curve c is C^∞ all over its domain

$$L_{i,n}(s_j) = \left\{ \begin{array}{ll} 0 \text{ if } & j \neq i \\ 1 \text{ if } & j = i \end{array} \right\}. \tag{3.106}$$

This is a stunning result as it simply implies that the matrix **F** (see (3.99)) is the unit matrix. Hence, using the Lagrange basis the straightforward solution to Problem 3.2 is

$$\mathbf{a}_i = \mathbf{c}_i, i = 0, \ldots, n.$$

with respect to the Lagrange basis the polynomial interpolation curve for the given input data can be written as

$$\mathbf{p}(t) = \sum_{i=0}^{n} L_{i,n}(t) \cdot \mathbf{c}_i. \tag{3.107}$$

The solution to this task certainly does not depend on the chosen basis. But still, the choice of the basis has an impact on the computational cost. The Lagrange basis is optimal in this sense; inverting the matrix **F** can be avoided in the first place. Though the solution curve is unaffected by the used basis, it traditionally is named *Lagrange interpolation curve*. Figure 3.58 gives an example of a Lagrange interpolation curve to a given set of input points. A uniform knot vector has been chosen for this task.

Aitken's Algorithm

Remember that Bézier curves can conveniently be evaluated by the de Casteljau algorithm. The de Boor algorithm similarly does a good job for B-spline curves. We now describe a geometric algorithm[22] called *Aitken's algorithm* to obtain the Langrange interpolation curve (see Fig. 3.59). Again, it is based on repeated subdivision of straight line segments (see also [8], pp. 67–70). The whole algorithm can be observed in Fig. 3.59.

Algorithm 3.3. Aitken's algorithm. *Let* $n+1$ *points* $\mathbf{a}_0, \ldots, \mathbf{a}_n$ *and the parameter values* $s_0 < \cdots < s_n$ *of a Lagrange interpolation curve (3.107) be given. In order*

[22] This algorithm is due to Alexander Craig Aitken (1895–1967), a mathematician from New Zealand.

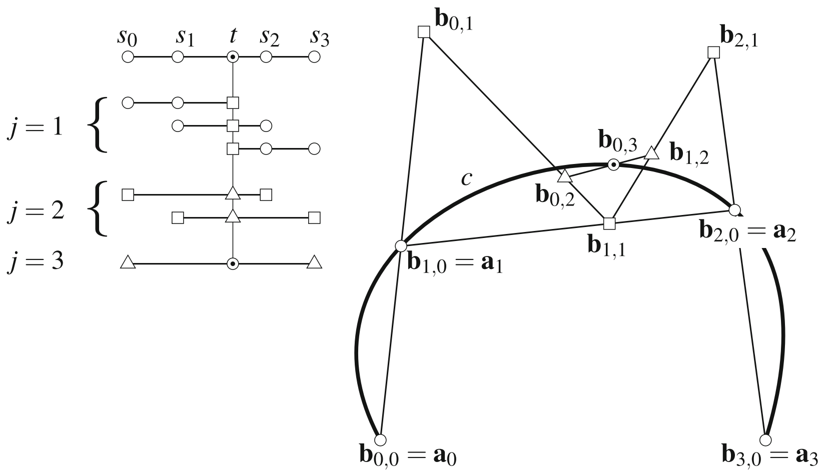


Fig. 3.59 Aitken's algorithm for n = 3. It consists of successive subdivision, starting with the given points $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ (*right*). The ratio of subdivision is to be computed from the given parameter values s_0, s_1, s_2, s_3 as indicated (*left*). The algorithm results in the point $\mathbf{b}_{0,3} = \mathbf{p}(t) \in c$

to obtain the point $\mathbf{p}(t)$ on this curve belonging to the parameter value $t \in [s_0, s_n]$ we take the following steps:

1. *Rename $\mathbf{b}_{i,0} := \mathbf{a}_i$ for $i = 0, \dots n$.*
2. *For $l = 1, \dots, n$ and $i = 0, \dots, n - l$ compute the points*

$$\mathbf{b}_{i,l}(t) := (1 - \alpha(t, i, l)) \cdot \mathbf{b}_{i,l-1}(t) + \alpha(t, i, l) \cdot \mathbf{b}_{i+1,l-1}(t)$$

   with

$$\alpha(i, l, t) = \frac{t - s_i}{s_{i+l} - s_i},$$

   successively (see scheme (3.108)).
3. *The desired point eventually shows up at the right top end of scheme (3.108): $\mathbf{p}(t) = \mathbf{b}_{0,n}$.*

$$\begin{array}{llllll}
\mathbf{b}_0 & \mathbf{b}_{0,1} & * & * & * \, \mathbf{b}_{0,n-1} & \mathbf{b}_{0,n} = \mathbf{p}(t) \\
\mathbf{b}_1 & \mathbf{b}_{1,1} & * & * & * \, \mathbf{b}_{1,n-1} & \\
* & * & * & * & * & \\
* & * & * & * & & \\
* & * & * & & & \\
* \, \mathbf{b}_{n-1,1} & & & & & \\
\mathbf{b}_n & & & & &
\end{array} \tag{3.108}$$

A graphic proof to this handy geometric algorithm can be found in ([8], pp. 67).

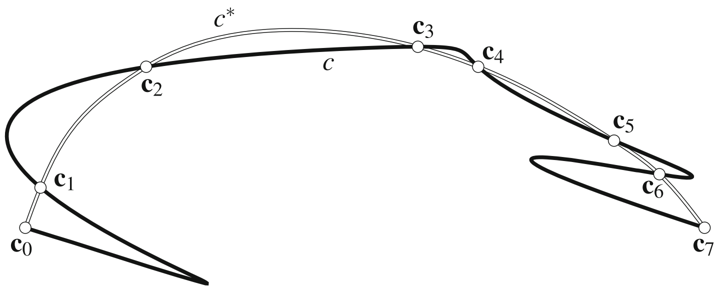


Fig. 3.60 The Lagrange interpolation curve c to a given set of points $\mathbf{c}_0, \ldots, \mathbf{c}_7$ and a uniform knot vector $s_0, \ldots, s_7$. In this case the solution curve c shows considerable oscillation. For comparison, the curve c^* (*double line*) indicates what the user might have expected

We summarize that a Lagrange interpolation curve can easily be computed or—if desired—be constructed geometrically via Aitken's algorithm. The number of input points determines the polynomial degree: $n + 1$ points imply an interpolation curve of degree n. We though have to admit that this also entails one significant drawback: High degree polynomial curves are prone to oscillation. So, if the number of input points is large we may get undesired phenomena (Fig. 3.60). Even though Lagrange interpolation delivers a smooth (C^∞) solution we have to face the fact that—in particular cases—it may not be the right tool.

The following section is some kind of trade-off: We avoid any tendency to oscillation by confining ourselves to polynomial curves of degree 3. On the other hand we have to make do with C^1-continuity at the junction points of the resulting subspline curve. However, in the subsequent paragraph we will improve the continuity to C^2, obtaining a cubic spline curve solution.

3.5.2 Interpolation by Cubic Segments

C^2-continuity means that apart from the common first derivative vector at the junction points there is also a second derivative vector that both adjacent segments have in common. In terms of geometry C^2-continuity means that the segments have the same tangent and the same curvature at their junction point. So, the visual impression of C^2-splines is indeed compelling. A spline curve with cubic segments delivers C^2-continuity even at the knots (see Definition 3.28). Later in this section we will provide such cubic splines solving the interpolation problem.

In many instances even a cubic subspline with C^1-continuity will do the job: The derivative vectors at the junction point of two consecutive segments are identical. One simple way of constructing a cubic interpolating subspline curve was suggested by A.W. Overhauser,[23] [14]:

[23] Albert W. Overhauser, born 1925 in San Diego, California is an American physicist.

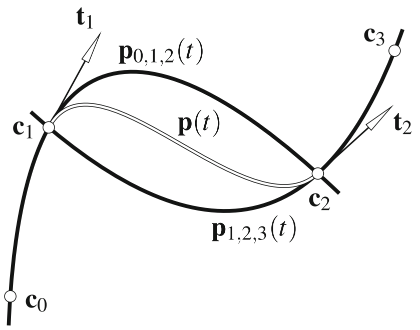


Fig. 3.61 We put $i = 1$ in (3.110). The Overhauser subspline segment is constructed as a linear blending of parabola arcs $\mathbf{p}_{0,1,2}(t)$ and $\mathbf{p}_{1,2,3}(t)$. The blending curve $\mathbf{p}(t)$ inherits the tangent vector from $\mathbf{p}_{0,1,2}(t)$ at s_1 and the tangent vector from $\mathbf{p}_{1,2,3}(t)$ at s_2. This way the resulting subspline segments join C^1-continuously at their junction points

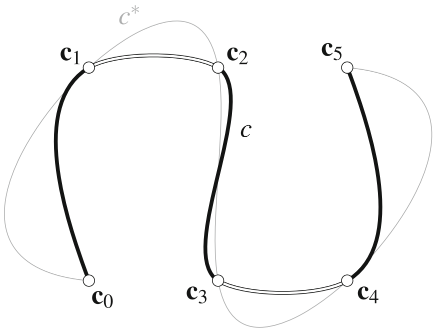


Fig. 3.62 Overhauser subspline as a solution to the Interpolation Problem 3.2. The different segments (*black* and *white*, alternately) are joined C^1-continuously. They are generated as explained in (3.110) and illustrated in Fig. 3.61. For comparison, the thin grey curve c^* is the Lagrange solution with a uniform knot vector

Interpolation by Overhauser Subsplines

We first introduce the concept of *linear blending of two curves*. Let $\mathbf{p}_1(t)$ and $\mathbf{p}_2(t)$ be the parameterizations of two curves on an interval $[s_1, s_2]$. The linear blending of $\mathbf{p}_1(t)$ and $\mathbf{p}_2(t)$ is defined as the curve

$$\mathbf{p}(t) = \frac{s_2 - t}{s_2 - s_1}\mathbf{p}_1(t) + \frac{t - s_1}{s_2 - s_1}\mathbf{p}_2(t). \tag{3.109}$$

If we apply linear blending (3.109) to the curves $\mathbf{p}_1(t)$ and $\mathbf{p}_2(t)$ and additionally assume that $\mathbf{p}_1(s_1) = \mathbf{p}_2(s_1)$ and $\mathbf{p}_1(s_2) = \mathbf{p}_2(s_2)$, the resulting blending curve $\mathbf{p}(t)$ inherits its tangent vectors at $t = s_1$ from $\mathbf{p}_1$ and at $t = s_2$ from $\mathbf{p}_2$. This is the basic idea for the following construction of Overhauser subsplines (Figs. 3.61 and 3.62).

We now introduce a simple but efficient algorithm to construct interpolation curves capable of coping with large numbers of input points without unwanted oscillation.

Definition 3.37. Overhauser subspline. Let $n + 1$ points $\mathbf{c}_0, \ldots, \mathbf{c}_n$ and corresponding parameter values $s_0, \ldots, s_n$ be given. Let $\mathbf{p}_{i,i+1,i+2}(t)$ denote the Lagrange interpolation curve to the input data $\mathbf{c}_i, \mathbf{c}_{i+1}, \mathbf{c}_{i+2}$ and s_i, s_{i+1}, s_{i+2}. Then the curve $\mathbf{p}(t)$ defined by

$$\mathbf{p}(t) = \left\{ \begin{array}{ll} \mathbf{p}_{0,1,2}(t) & \text{if } t \in [s_0, s_1] \\ \frac{s_{i+1}-t}{s_{i+1}-s_i}\mathbf{p}_{i-1,i,i+1}(t) + \frac{t-s_i}{s_{i+1}-s_i}\mathbf{p}_{i,i+1,i+2}(t) & \text{if } t \in [s_i, s_{i+1}], \\ & i = 1, \ldots, n-2 \\ \mathbf{p}_{n-2,n-1,n}(t) & \text{if } t \in [s_{n-1}, s_n] \end{array} \right\} \tag{3.110}$$

is called *Overhauser subspline*.

Proposition 3.13 Properties of Overhauser subsplines.

1. *The first segment of an Overhauser subspline curve (relating to the interval* $[s_0, s_1]$ *is a Lagrange interpolation curve of degree* 2*, i.e., a parabola segment. The same holds for the last segment.*
2. *The intermediate segments (relating to the intervals* $[s_i, s_{i+1}]$*,* $i = 1, \ldots, n-2$*) are linear blendings of parabola segments. Thus they are cubic curves (polynomial curves of degree* 3*).*
3. *At their junction point adjacent segments meet with* C^1*-continuity, i.e., they have a common first derivative vector.*

Overhauser supsplines are capable of interpolating a large number of input points without being susceptible to unwanted oscillation effects.

The only downside to this useful algorithm is that it merely delivers C^1-continuity. However, this limitation can also be overcome by resorting to non-linear blending of higher degree elements (see [15]).

Hermite Interpolation

In this paragraph we describe another truly classical solution to the interpolation problem. It employs cubic spline segments which are joined to a C^1-subspline curve.

Hermite interpolation starts with a set of points to be interpolated and—additionally—a set of derivative vectors at the given points to be adopted.

The restriction to low degree spline elements avoids oscillation in the first place. In the subsequent paragraph we will even be able to modify the approach and deliver a C^2-spline solution with cubic segments.

Problem 3.3 Hermite interpolation.[24] Let a set of points $\mathbf{c}_0, \ldots, \mathbf{c}_n$, a set of corresponding parameter values $s_0 < \cdots < s_n$ and a corresponding set $\mathbf{t}_0, \ldots, \mathbf{t}_n$ of derivative vectors be given (Fig. 3.63).

[24] Charles Hermite (1822–1901) was a French mathematician who particularly contributed to the fields of number theory and algebra.

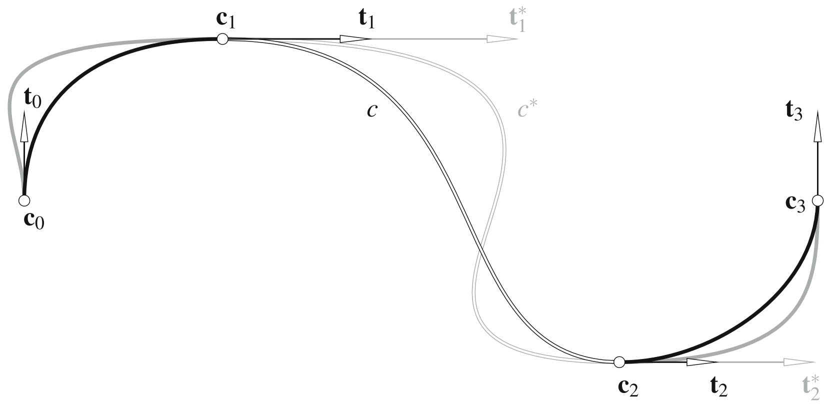


Fig. 3.63 Hermite interpolation problem: the points $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$, the corresponding derivative vectors $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ and the knot vector s_0, s_1, s_2, s_3 are given. The sections of the emerging Hermite solution curve c are marked in *black* and *white*. The same input, however with different derivative vectors $\mathbf{t}_1^* = 2 \cdot \mathbf{t}_1$ and $\mathbf{t}_2^* = 2 \cdot \mathbf{t}_2$, delivers the solution c^* (*grey*)

Find a curve $\mathbf{p}(t)$ consisting of cubic segments such that for every $j = 0, \ldots, n$:

$$\mathbf{p}(s_j) = \mathbf{c}_j \tag{3.111}$$

$$\dot{\mathbf{p}}(\mathbf{s_j}) = \mathbf{t}_j \tag{3.112}$$

Basically, this task is taken on separately for every single interval $[s_j, s_{j+1}]$: A cubic curve $\mathbf{p}_j(t)$ is uniquely determined by prescribing two points $\mathbf{c}_j, \mathbf{c}_{j+1}$ and the corresponding derivative vectors $\mathbf{t}_j, \mathbf{t}_{j+1}$ at the respective parameters s_j, s_{j+1}.

We represent the desired curve as a cubic Bézier curve on the interval $[s_j, s_{j+1}]$ (compare with (3.72), p. 92):

$$\mathbf{p}_j(t) = \sum_{i=0}^{3} B_{i,3}^{[s_j, s_{j+1}]}(t) \cdot \mathbf{a}_{j,i} \tag{3.113}$$

A Bézier curve interpolates the first and the last point of its control polygon, so from (3.111) we get:

$$\mathbf{a}_{j,0} = \mathbf{c}_j, \;\; \mathbf{a}_{j,3} \; = \; \mathbf{c}_{j+1} \tag{3.114}$$

According to (3.74), p. 92 the derivative vectors of this segment at the starting point and at the end point are:

$$\dot{\mathbf{p}}_j(s_j) = \frac{3}{\Delta_j} \cdot (\mathbf{a}_{j,1} - \mathbf{a}_{j,0}),$$
$$\dot{\mathbf{p}}_j(s_{j+1}) = \frac{3}{\Delta_j} \cdot (\mathbf{a}_{j,3} - \mathbf{a}_{j,2})$$

where $\Delta_j = s_{j+1} - s_j$.

As we are also given the prescribed derivative vectors $\mathbf{t}_j$, $\mathbf{t}_{j+1}$ at s_j, s_{j+1} according to (3.112) we obtain:

$$\mathbf{a}_{j,1} = \mathbf{c}_j + \frac{\Delta_j}{3} \cdot \mathbf{t}_j, \quad \mathbf{a}_{j,2} \;=\; \mathbf{c}_{j+1} - \frac{\Delta_j}{3} \cdot \mathbf{t}_{j+1} \tag{3.115}$$

Inserting (3.114) and (3.115) into (3.113) perfectly conveys the representation of the cubic segment to the interval $[s_j, s_{j+1}]$. In terms of the input data this formula can be rewritten as

$$\mathbf{p}_j(t) = H_{j,0}(t) \cdot \mathbf{c}_j + H_{j,1}(t) \cdot \mathbf{t}_j + H_{j,2}(t) \cdot \mathbf{t}_{j+1} + H_{j,3}(t) \cdot \mathbf{c}_{j+1} \tag{3.116}$$

where the functions

$$\left.\begin{aligned} H_{j,0}(t) &= \frac{1}{\Delta_j^3} \cdot (s_{j+1} - t)^2 \cdot (2t + s_{j+1} - 3s_j), \\ H_{j,1}(t) &= \frac{1}{\Delta_j^2} \cdot (t - s_j) \cdot (s_{j+1} - t)^2, \\ H_{j,2}(t) &= -\frac{1}{\Delta_j^2} \cdot (t - s_j)^2 \cdot (s_{j+1} - t), \\ H_{j,3}(t) &= \frac{1}{\Delta_j^3} \cdot (t - s_j)^2 \cdot (-2t + 3s_{j+1} - s_j) \end{aligned}\right\} \tag{3.117}$$

are the classical *cubic Hermite polynomials* on the support interval $[s_j, s_{j+1}]$.

Remark 3.12 The Hermite interpolation problem can also be solved if higher order derivative vectors at the junction points are prescribed. In this case more than C^1-continuity can be achieved (see [8], pp. 79–80, [7], pp. 96–98). An explicit formula for Hermite interpolants with various numbers of given higher order derivatives at the knots can be found in [16], p. 121.

In the next paragraph the option of preselecting the tangent vectors at every data point will be discarded in exchange for higher smoothness at the junction points.

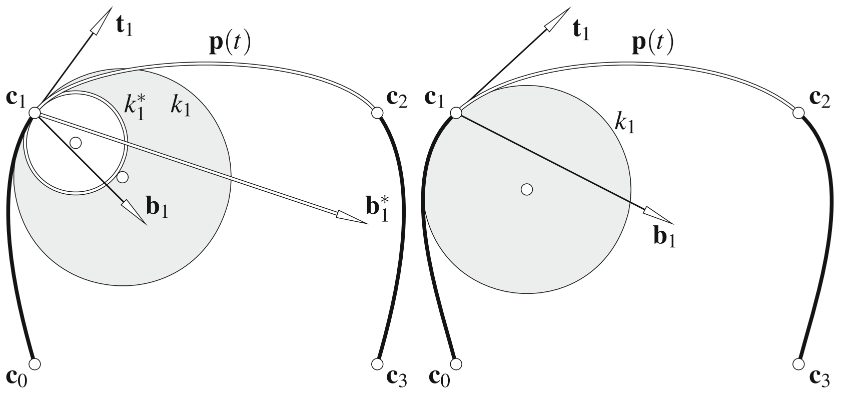


Fig. 3.64 C^1-continuity as opposed to C^2-continuity. *Left* interpolation by Overhauser subsplines delivers spline segments joined C^1-continuously. In the figure this is illustrated for the junction point $\mathbf{c}_1$: the two segments joined at $\mathbf{c}_1$ have the same tangent vector $\mathbf{t}_1$ whilst the second derivative vectors $\mathbf{b}_1$ and $\mathbf{b}_1^*$ are different. The osculating circles k_1 and k_1^* of the two segments are also different; the curvature changes abruptly at the junction point. *Right* the Hermite approach as explained in this section delivers a C^2-continuous interpolation curve. Even at the junction points (in the figure illustrated for $\mathbf{c}_1$) the first derivative vectors of the two segments coincide, and so do the second derivative vectors and the osculating circles. The curvature varies smoothly all over the interpolation curve, even at the junction points

Cubic Spline Interpolation: The Hermite Approach

Cubic segments are well capable of delivering more than C^1-continuity at the junction points. Figure 3.64 highlights the difference between C^1- and C^2-continuity.

Generally speaking, segments of degree k with $k-1$-continuity are the hallmark of proper *splines* (see Definition 3.28, p. 99). The case $k = 3$ can already deliver curvature-continuity (i.e., C^2-continuity) of consecutive spline segments in their junction points which is—in most applications—just right. We now solve the interpolation problem by splines of degree $k = 3$ (cubic splines).

Problem 3.4 Cubic spline interpolation. Let a set of points $\mathbf{c}_0, \ldots, \mathbf{c}_n$ (in 3-space or the plane) and a set of corresponding parameter values $s_0 < \cdots < s_n$ be given. Find a cubic spline curve $\mathbf{p}(t)$, $t \in [s_0, s_n]$ with the knot vector $(s_0, \ldots, s_n)$ interpolating the given data, i.e.,

$$\mathbf{p}(s_j) = \mathbf{c}_j, \quad j = 0, \ldots, n. \tag{3.118}$$

The following strategy is based on Hermite interpolation (see above):

- We record the representation of the jth segment of the desired cubic spline interpolant represented in terms of the Hermite polynomials

$$\mathbf{p}_j(t) = H_{j,0}(t) \cdot \mathbf{c}_j + H_{j,1}(t) \cdot \mathbf{t}_j + H_{j,2}(t) \cdot \mathbf{t}_{j+1} + H_{j,3}(t) \cdot \mathbf{c}_{j+1}, \quad j = 0, \dots, n-1 \tag{3.119}$$

with yet unknown derivative vectors $\mathbf{t}_j, \mathbf{t}_{j+1}$ at the knots s_j, s_{j+1}; those vectors are still to be computed.

- As we go for C^2-continuity we set up the conditions

$$\ddot{\mathbf{p}}_j(s_{j+1}) = \ddot{\mathbf{p}}_{j+1}(s_{j+1}), \quad j = 0, \dots, n-2. \tag{3.120}$$

This is a system of $n-1$ linear equations for the $n+1$ unknown vectors $\mathbf{t}_0, \dots, \mathbf{t}_n$.

- We have two more unknown vectors than equations, so we are still to add two more conditions. At this point we have several alternatives how to go ahead. We describe two (standard) options of continuing the computation:

(a) We prescribe the derivative vectors $\mathbf{t}_0$, $\mathbf{t}_n$ at either endpoint $\mathbf{c}_0$ and $\mathbf{c}_n$ in the following way:
The first three points $\mathbf{c}_0$, $\mathbf{c}_1$, $\mathbf{c}_2$ uniquely determine a Lagrange interpolation curve according to (3.107), p. 120, which is—in this case—a parabola. Its tangent vector at $\mathbf{c}_0$ can easily be computed by

$$\mathbf{t}_0 = \gamma_0 \cdot \mathbf{c}_0 + \gamma_1 \cdot \mathbf{c}_1 + \gamma_2 \cdot \mathbf{c}_2 \tag{3.121}$$

with

$$\gamma_0 := -\frac{2\Delta_0 + \Delta_1}{\Delta_0(\Delta_0 + \Delta_1)}, \quad \gamma_1 := \frac{\Delta_0 + \Delta_1}{\Delta_0 \Delta_1}, \quad \gamma_2 := -\frac{\Delta_0}{(\Delta_0 + \Delta_1)\Delta_1}.$$

We equally have

$$\mathbf{t}_n = \gamma_{n-2} \cdot \mathbf{c}_{n-2} + \gamma_{n-1} \cdot \mathbf{c}_{n-1} + \gamma_n \cdot \mathbf{c}_n \tag{3.122}$$

for the tangent vector at $\mathbf{c}_n$ with the abbreviations

$$\gamma_{n-2} := \frac{\Delta_{n-1}}{\Delta_{n-2}(\Delta_{n-2} + \Delta_{n-1})}, \quad \gamma_{n-1} := -\frac{\Delta_{n-2} + \Delta_{n-1}}{\Delta_{n-2}\Delta_{n-1}},$$
$$\gamma_n := \frac{2\Delta_{n-1} + \Delta_{n-2}}{(\Delta_{n-2} + \Delta_{n-1})\Delta_{n-1}}.$$

Adding these two conditions the final system of linear equations reads as[25]

$$\mathbf{D} \cdot \begin{bmatrix} \mathbf{t}_0 \\ \vdots \\ \mathbf{t}_n \end{bmatrix} = \tilde{\mathbf{D}} \cdot \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_n \end{bmatrix} \tag{3.123}$$

[25] Equation (3.123) has to be interpreted for each coordinate of the vectors $\mathbf{t}_i$, $\mathbf{c}_i$ separately, so it basically consists of three equations.

with the matrices $\mathbf{D}$, $\tilde{\mathbf{D}}$:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & & \dots & 0 \\ \Delta_1 & 2(\Delta_0+\Delta_1) & \Delta_0 & 0 & & & \\ 0 & \Delta_2 & 2(\Delta_1+\Delta_2) & \Delta_1 & & & \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ & & & & & & 0 \\ \vdots & & & & \Delta_{n-1} & 2(\Delta_{n-2}+\Delta_{n-1}) & \Delta_{n-2} \\ 0 & \dots & & & 0 & 0 & 1 \end{bmatrix} \tag{3.124}$$

$$\tilde{\mathbf{D}} = \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & 0 & & \dots & 0 \\ -3\frac{\Delta_1}{\Delta_0} & 3(\frac{\Delta_1}{\Delta_0} - \frac{\Delta_0}{\Delta_1}) & 3\frac{\Delta_0}{\Delta_1} & 0 & & & \\ 0 & -3\frac{\Delta_2}{\Delta_1} & 3(\frac{\Delta_2}{\Delta_1} - \frac{\Delta_1}{\Delta_2}) & 3\frac{\Delta_1}{\Delta_2} & & & \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ & & & & & & 0 \\ \vdots & & & & -3\frac{\Delta_{n-1}}{\Delta_{n-2}} & 3(\frac{\Delta_{n-1}}{\Delta_{n-2}} - \frac{\Delta_{n-2}}{\Delta_{n-1}}) & 3\frac{\Delta_{n-2}}{\Delta_{n-1}} \\ 0 & \dots & & & \gamma_{n-2} & \gamma_{n-1} & \gamma_n \end{bmatrix} \tag{3.125}$$

(b) We offer another proposal how to add two additional constraints by simply demanding

$$\ddot{\mathbf{p}}_0(s_0) = \ddot{\mathbf{p}}_{n-1}(s_n) = \mathbf{0}. \tag{3.126}$$

Owing to these conditions the resulting interpolation curve has points of inflection at its endpoints. Interpreting the curve as a flexible beam (Sect. 3.4.2, p. 99) this means that outside the support interval there are no forces exerted to the beam. At each endpoint it behaves as though it was continued as a straight line.
If we apply these conditions (3.126) instead of the aforementioned the first and the last row in matrix $\mathbf{D}$ have to be replaced by

$$\begin{gathered} 2\Delta_0, \Delta_0, 0, \dots\dots\dots, 0, \\ 0, \dots\dots, 0, \ \Delta_{n-1}, \ 2\Delta_{n-1}. \end{gathered}$$

Furthermore the first and last row of $\tilde{\mathbf{D}}$ have to be replaced by

$$\begin{gathered} -3, \ 3, \ 0, \dots\dots, \ 0, \\ 0, \dots\dots, 0, -3, \ 3. \end{gathered}$$

- The tridiagonal matrix $\mathbf{D}$ is invertible in both cases a) and b). The reason for this is that $\mathbf{D}$ is main diagonal dominant: In every row the element in the main diagonal is greater than the other entries of the same row. The solution to (3.123) is given by[26]

$$\begin{bmatrix} \mathbf{t}_0 \\ \vdots \\ \mathbf{t}_n \end{bmatrix} = \mathbf{D}^{-1} \cdot \tilde{\mathbf{D}} \cdot \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_n \end{bmatrix}. \tag{3.127}$$

- Having found all the tangent vectors $\mathbf{t}_j$, $j = 0, \ldots, n$ we can easily finish the task: The jth segment $\mathbf{p}_j(t)$ of the required cubic spline curve is determined by the given points $\mathbf{c}_j$, $\mathbf{c}_{j+1}$ and the just computed tangent vectors $\mathbf{t}_j$, $\mathbf{t}_{j+1}$. Its explicit representation is (3.119), p. 128.

Figure 3.65 shows an example of cubic Hermite spline interpolation.

Physical Interpretation of Cubic Spline Interpolation

The first time that we referred to the use of flexible beams passing through given points, was at the beginning of Sect. 3.4.2, p. 99. The Interpolation Problem 3.2, p. 119, is closely related to this task. If we force such a beam through given points it will try to assume a balanced state. The beam $\mathbf{p}(t)$ minimizes the linearized bending energy

$$E = c \cdot \int \kappa^2(s)\mathrm{d}s \quad = \quad c \cdot \int \|\mathbf{p}''\|^2\mathrm{d}s \tag{3.128}$$

where c is a constant, s denotes the arc length on $\mathbf{p}(t)$ and primes denote differentiation with respect to s (compare with (3.38), p. 78). We can state (without going into details):

Among all curves through the given points $\mathbf{c}_i$ it is the cubic spline curve which minimizes the bending energy E. This means that this curve roughly behaves like a flexible beam through these points.

Cubic Spline Interpolation: The B-Spline Approach

In this paragraph we describe another way of solving Problem 3.4, p. 127. The resulting interpolation curve is the same as it is uniquely determined by the input and thus independent of the way how to get there.

For the approach at hand we use the cubic B-spline basis functions $N_{i,4}(t)$ (Definition 3.29, p. 100). In contrast to the Hermite approach we compute the control

[26] The inverse of a tridiagonal matrix can for instance be computed by the method suggested in [17] though we do not recommend going this way. Instead we opt for the numerically stable *Gauss-Jordan method with pivoting* for solving the linear equation system (3.123) which is a standard mathematical tool.

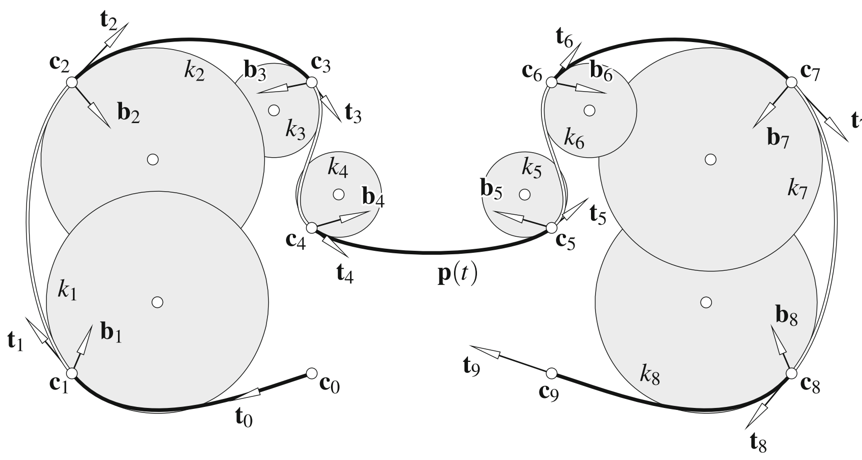


Fig. 3.65 Cubic Hermite splines deliver a C^2-continuous interpolation curve to the given data points $\mathbf{c}_0, \ldots, \mathbf{c}_9$. Note that at each junction point the spline segments (marked in *black* and *white*, alternately) have identical first and second derivative vectors (named $\mathbf{t}_i$ and $\mathbf{b}_i$ in the figure). The common osculating circle k_i of adjacent segments at the junction point $\mathbf{c}_i$ is also shown. At the endpoints we went for option (3.126): the second derivative vectors at either end ($\mathbf{b}_0$ and $\mathbf{b}_9$) vanish

points $\mathbf{a}_i$ of the resulting B-spline curve $\mathbf{p}(t)$ directly (without computing the tangent vectors at the knots s_i beforehand).

We generally demand that the knots $t_0, \ldots, t_m, \ldots, t_{m+k}$ of the B-spline curve $\mathbf{p}(t)$—which is meant to be our solution—coincide with the interpolation knots s_j. As we use a cubic B-spline curve $\mathbf{p}(t)$ we have to put $k = 4$; the support interval of $\mathbf{p}(t)$ has to be $[t_3 = s_0, t_{m+1} = s_n]$. The appropriate knot vector is

$$t_0, t_1, t_2, t_3 = s_0, \ldots, t_{n+3} = s_n, t_{n+4}, t_{n+5}, t_{n+6}. \tag{3.129}$$

We intend to compute the control points $\mathbf{a}_i$ of the cubic B-spline curve

$$\mathbf{p}(t) = \sum_{i=0}^{n+2} N_{i,4}(t) \cdot \mathbf{a}_i, \quad t \in [s_0, s_n]. \tag{3.130}$$

The free knots t_0, t_1, t_2 and $t_{n+4}, t_{n+5}, t_{n+6}$ at either end can be chosen arbitrarily. We suggest $t_0 = t_1 = t_2 = t_3 = s_0$ and $s_n = t_{n+3} = t_{n+4} = t_{n+5} = t_{n+6}$ which—by the way—does not affect the final outcome.

This choice, on the other hand, brings about one convenient side-effect (see Proposition 3.6, p. 106): For the first and the last control point we have

$$\left.\begin{aligned} \mathbf{a}_0 &= \mathbf{p}(s_0) = \mathbf{c}_0, \\ \mathbf{a}_{n+2} &= \mathbf{p}(s_n) = \mathbf{c}_n. \end{aligned}\right\} \tag{3.131}$$

The remaining conditions for the points $\mathbf{c}_j$ to be interpolated at s_j are:

$$\mathbf{p}(s_j) = \sum_{i=0}^{n+2} N_{i,4}(s_j = t_{j+3}) \cdot \mathbf{a}_i, \;=\; \mathbf{c}_j,\;\; j = 1, \ldots, n-1 \tag{3.132}$$

Due to ((Proposition 3.3, 4.), p. 100), there are only three non-zero terms in every sum of (3.132) as we have:

$$N_{j,4}(t_{j+3}) \cdot \mathbf{a}_j + N_{j+1,4}(t_{j+3}) \cdot \mathbf{a}_{j+1} + N_{j+2,4}(t_{j+3}) \cdot \mathbf{a}_{j+2}, = \mathbf{c}_j, \quad j = 1, \ldots, n-1 \tag{3.133}$$

The non-vanishing values $N_{j,4}(t_{j+3}), N_{j+1,4}(t_{j+3}), N_{j+2,4}(t_{j+3})$ can directly be expressed in terms of the knots t_i:

$$\left.\begin{aligned}
N_{j,4}(t_{j+3}) &= \frac{(t_{j+4}-t_{j+3})^2}{(t_{j+4}-t_{j+1})(t_{j+4}-t_{j+2})} \\
N_{j+1,4}(t_{j+3}) &= \frac{(t_{j+3}-t_{j+1})(t_{j+4}-t_{j+3})}{(t_{j+4}-t_{j+1})(t_{j+4}-t_{j+2})} + \frac{(t_{j+5}-t_{j+3})(t_{j+3}-t_{j+2})}{(t_{j+5}-t_{j+2})(t_{j+4}-t_{j+2})} \\
N_{j+2,4}(t_{j+3}) &= \frac{(t_{j+3}-t_{j+2})^2}{(t_{j+5}-t_{j+2})(t_{j+4}-t_{j+2})}
\end{aligned}\right\} \tag{3.134}$$

The $n+3$ de Boor points $\mathbf{a}_i$ of the required interpolation curve $\mathbf{p}(t)$ are to be computed. As of now, we have $n+1$ conditions, i.e., the two constraints (3.131) plus the $n-1$ conditions (3.133). There are, obviously, two additional constraints to be chosen.[27]

One option would certainly be to select (arbitrary) tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_n$ at $\mathbf{c}_0$ and $\mathbf{c}_n$, respectively. This is what commonly used CAD programs suggest. They request the tangent vectors at either end and they interactively show a preview of the resulting interpolation curve.

In many cases the additional free choice of two vectors is not exactly what the user expects. As an alternative we regard the Lagrange interpolation curve to the three points $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$: Its tangent vector $\mathbf{t}_0$ at $\mathbf{c}_0$ has been computed in (3.121), p. 128. Equally, we obtain the tangent vector $\mathbf{t}_n$ of the Lagrange interpolation curve to $\mathbf{c}_{n-2}, \mathbf{c}_{n-1}, \mathbf{c}_n$ in (3.122). These two vectors are a good suggestion for the required additional input as they perfectly blend in the shape of the input data.

The Lagrange interpolation curve to the first (or the last) three given data points is a parabola which—in most cases—yields a suitable tangent at the end point of the segment. Sometimes, though, the shape of the parabolic segment may cause some inappropriate kink near the endpoint. In such a case, a circle through the first (or the last) three data points may be the more preferable choice.

As we have k coinciding knots at the beginning the tangent vector $\mathbf{t}_0$ can be written in terms of the first two de Boor points $\mathbf{a}_0, \mathbf{a}_1$ (see Proposition 3.6, (3.87), p. 106):

[27] Remember the similar situation with the Hermite approach to the same task.

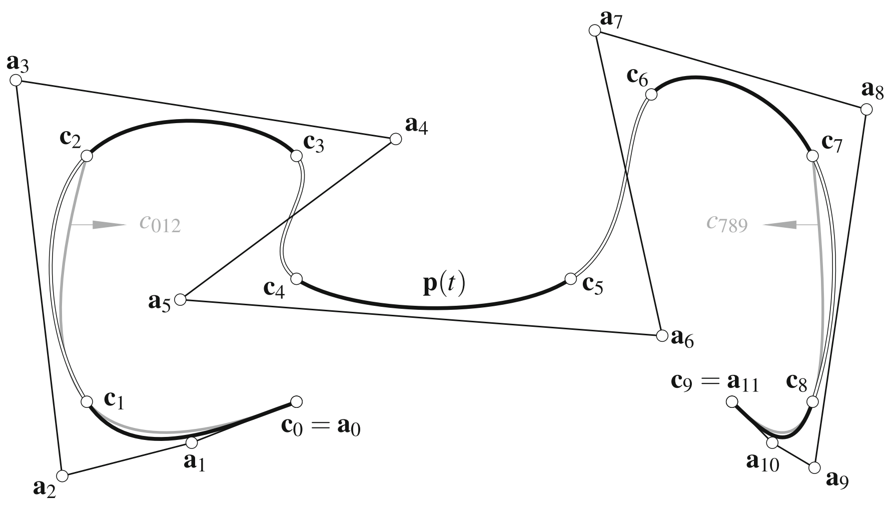


Fig. 3.66 The B-spline approach delivers a C^2-continuous interpolation curve to the given data points $\mathbf{c}_0, \ldots, \mathbf{c}_9$. In order to obtain the resulting B-spline curve we compute the de Boor points $\mathbf{a}_0, \ldots, \mathbf{a}_{11}$. As for the two free conditions at the endpoints $\mathbf{c}_0$ and $\mathbf{c}_9$ we here used (3.121) and (3.122) the parabola arcs c_{012} and c_{789} defined by the first and the last three points, respectively, are highlighted in *grey*. We demand that the interpolation curve inherits its tangent vectors at either end from these two parabolae

$$\frac{3}{s_1 - s_0} \cdot (\mathbf{a}_1 - \mathbf{a}_0) = \mathbf{t}_0 \tag{3.135}$$

In the same way we get

$$\frac{3}{s_n - s_{n-1}} \cdot (\mathbf{a}_{n+2} - \mathbf{a}_{n+1}) = \mathbf{t}_n, \tag{3.136}$$

as we also have k coinciding knots at the endpoint.

Combining the $n - 1$ conditions (3.133), the two conditions (3.131) and the last two (3.135), (3.136) we get a system of $n + 3$ linear equations

$$\mathbf{F} \cdot \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{n+1} \\ \mathbf{a}_{n+2} \end{bmatrix} = \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_n \\ \mathbf{t}_n \end{bmatrix} \tag{3.137}$$

where

$$\mathbf{F} := \begin{bmatrix} -\frac{3}{s_1-s_0} & \frac{3}{s_1-s_0} & 0 & \dots & & \dots & 0 \\ 1 & 0 & \dots & & & \dots & 0 \\ 0 & N_{1,4}(s_1) & N_{2,4}(s_1) & N_{3,4}(s_1) & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & N_{n-1,4}(s_{n-1}) & N_{n,4}(s_{n-1}) & N_{n+1,4}(s_{n-1}) & 0 \\ 0 & \dots & & & \dots & 0 & 1 \\ 0 & \dots & & \dots & 0 & -\frac{3}{s_n-s_{n-1}} & \frac{3}{s_n-s_{n-1}} \end{bmatrix}. \tag{3.138}$$

Certainly, the values of $N_{i,4}(s_j)$ occurring in this matrix can be computed right away by (3.134).

The matrix **F** is tridiagonal and invertible.[28] The de Boor points of the solution curve to the interpolation problem appear as

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{n+1} \\ \mathbf{a}_{n+2} \end{bmatrix} = \mathbf{F}^{-1} \cdot \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_n \\ \mathbf{t}_n \end{bmatrix}. \tag{3.139}$$

To know that the solution is unique and can basically be expressed via the inverse matrix $\mathbf{F}^{-1}$ is comforting. For the actual computation—particularly if the matrix **F** is very large—we recommend an alternative option (see footnote on p. 130). In Figs. 3.66 and 3.67 two examples of cubic B-spline spline interpolation are illustrated. Figure 3.66 refers to the planar case whereas Fig. 3.67 shows the solution to a spatial interpolation problem.

3.5.3 Parameterization

In the Interpolation Problem 3.2, p. 119, we started with given data points $\mathbf{c}_0, \dots, \mathbf{c}_n$ and a sequence $s_0 < \dots < s_n$ of given parameter values. In practice we may only be given the points $\mathbf{c}_j$ whereas the parameter values s_j can—theoretically—be chosen at will. We have to be careful as the resulting interpolation curve heavily depends upon this choice. For in-depth considerations on numerical methods we also refer to [18] and [16].

[28] In fact its invertibility is easy to check after expanding the matrix along the 2nd and the last but one row. The remaining matrix is main diagonal dominant (see p. 130) and hence its determinant is non-zero.

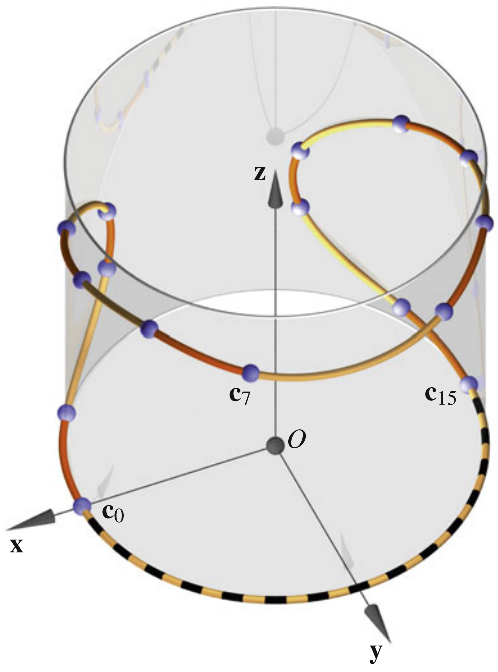


Fig. 3.67 The 16 points $\mathbf{c}_0, \ldots, \mathbf{c}_{15}$ lie on a *right* cylinder. They are interpolated by a cubic B-spline curve as described above; chordal parameterization. The tangents at either end ($\mathbf{c}_0$ and $\mathbf{c}_{15}$) are prescribed as tangents to the boundary circle (zebra striped) in order to guarantee a flush transition. Strictly speaking, the interpolation curve is not contained in the cylindric surface, though it adapts pretty well

One simple choice of the sequence $s_0, \ldots, s_n$ is the parameterization: $s_j = j$ which is called *uniform parameterization* (see below). This choice completely ignores the geometry of the input data points.[29]

Considering the parameter t of an interpolation curve $\mathbf{p}(t)$ as *time* the uniform parameterization implies that a section between two distant points on $\mathbf{p}(t)$ is covered with higher speed than a section between two close points. This also affects the shape of $\mathbf{p}(t)$ (see Fig. 3.68).

In order to achieve a more balanced speed distribution we can allow for the distances $\|\mathbf{c}_{j+1} - \mathbf{c}_j\|$ between consecutive points.

Definition 3.38. Standard types of parameterization. The knot vector $(s_0, \ldots, s_n)$ is called

- *uniform* if

[29] The same thing holds for the first Interpolation Problem 3.1. If we interpret the scalar values c_i as 'points' on the real number line we can apply the following adaption of the parameter sequence $s_0, \ldots, s_n$ correspondingly.

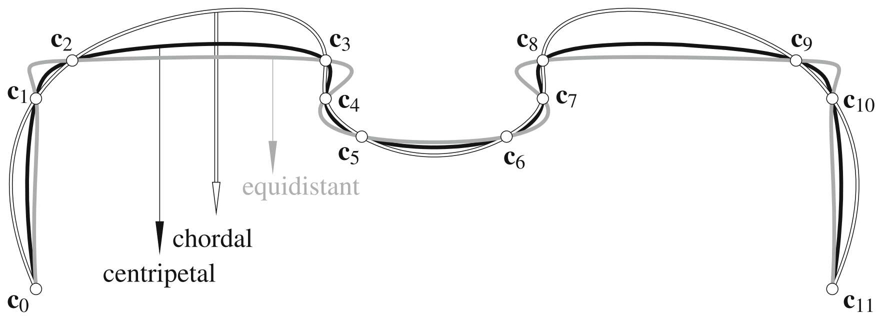


Fig. 3.68 Equidistant parameterization provides a first solution to the interpolation problem (*grey*). As the input points $\mathbf{c}_0, \ldots, \mathbf{c}_{11}$ are not evenly distributed, the chordal parameterization delivers a more balanced result (*double line*). Centripetal parameterization finally conveys an interpolation curve (*black*) which greatly considers the geometry of the input. It depends on the task which of the results is preferable

$$s_{j+1} = s_j + \delta, \quad \text{with } \delta = const. \text{ for } j = 0, \ldots, n-1. \tag{3.140}$$

- *chordal* if

$$\frac{s_{j+2} - s_{j+1}}{s_{j+1} - s_j} = \frac{\|\mathbf{c}_{j+2} - \mathbf{c}_{j+1}\|}{\|\mathbf{c}_{j+1} - \mathbf{c}_j\|}, \quad j = 0, \ldots, n-2. \tag{3.141}$$

- *centripetal* if

$$\frac{s_{j+2} - s_{j+1}}{s_{j+1} - s_j} = \sqrt{\frac{\|\mathbf{c}_{j+2} - \mathbf{c}_{j+1}\|}{\|\mathbf{c}_{j+1} - \mathbf{c}_j\|}}, \quad j = 0, \ldots, n-2. \tag{3.142}$$

Chordal parameterization adapts the parameter intervals proportional to the chord lengths. Centripetal parameterization stems from physical observations regarding the centripetal force of a body moving along the curve.

For other suggestions of appropriate parameterizations we refer to ([7], pp. 201).

The choice of parameterization is an important and influential step within the interpolation job. It finally determines the quality of the resulting interpolation curve. To recognize the great influence of the parameter distribution we solve an interpolation problem to twelve given input points $\mathbf{c}_0, \ldots, \mathbf{c}_{11}$ by cubic spline interpolation. The different shapes of the solutions (see Fig. 3.68) merely stem from different parameterizations.

3.6 Univariate Approximation

Approximation deals with the problem of finding a function (out of a given class) such that a distance function to a set of prescribed data is minimized. Basically, this is an optimization problem. As opposed to the interpolation case the given data do not have to be adopted exactly. The task reads as follows:

Problem 3.5 Univariate approximation of real values. Let $m+1$ real values $c_0, \dots, c_m$ in $\mathbb{R}$ and corresponding parameter values $s_0 < \cdots < s_m$ in an interval $[a, b]$ be given. The vector $(s_0, \dots, s_n)$ is also called *knot vector*. As in Problem 3.1 let $\mathscr{F}$ be a linear space of continuous functions on $[a, b]$. Moreover let[30] $m > n = \dim \mathscr{F} - 1$.

Find a function $p(t) \in \mathscr{F}$ such that the squared error sum (error function)

$$e = \sum_{j=0}^{m} \left| p(s_j) - c_j \right|^2 \tag{3.143}$$

is minimized.

This optimization task is a classical Gauss least squares problem. We address it in the following manner:

Let $F_0(t), \dots, F_n(t)$ be a basis of $\mathscr{F}$. The required approximation function $p(t)$ can be written in the form

$$p(t) = \sum_{i=0}^{n} F_i(t) \cdot a_i \tag{3.144}$$

with coefficients $a_0, \dots, a_n \in \mathbb{R}$. The squared error sum g can be rendered more precisely as the function

$$e(a_0, \dots, a_n) = \sum_{j=0}^{m} |p(s_j) - c_j|^2 \; = \; \sum_{j=0}^{m} \left| \left(\sum_{i=0}^{n} F_i(s_j) \cdot a_i \right) - c_j \right|^2 \tag{3.145}$$

which is quadratic in the yet unknown constant coefficients $a_0, \dots, a_n$. The necessary conditions for a minimum of this function e are that all partial derivatives $\frac{\partial e}{\partial a_i}$ are zero:

$$\frac{\partial e}{\partial a_i} = 0, \quad i = 0, \dots, n$$

These are $n+1$ linear equations for the $n+1$ coefficients a_i. In matrix form these conditions read as

[30] Usually, approximation deals with large data sets: $m > n$; the case $m \leq n$ would offer the opportunity to switch over to an interpolation task (see Sect. 3.5).

$$\mathbf{M} \cdot \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^{m} F_0(s_j) \cdot c_j \\ \vdots \\ \sum_{j=0}^{m} F_n(s_j) \cdot c_j \end{bmatrix} \tag{3.146}$$

where **M** is the symmetric positive definite $(n+1) \times (n+1)$-matrix

$$\mathbf{M} = \begin{bmatrix} \sum\limits_{j=0}^{m} F_0^2(s_j) & \sum\limits_{j=0}^{m} F_0(s_j) \cdot F_1(s_j) & \dots & \sum\limits_{j=0}^{m} F_0(s_j) \cdot F_n(s_j) \\ \sum\limits_{j=0}^{m} F_0(s_j) \cdot F_1(s_j) & \sum\limits_{j=0}^{m} F_1^2(s_j) & \dots & \sum\limits_{j=0}^{m} F_1(s_j) \cdot F_n(s_j) \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{j=0}^{m} F_0(s_j) \cdot F_n(s_j) & \sum\limits_{j=0}^{m} F_1(s_j) \cdot F_n(s_j) & \dots & \sum\limits_{j=0}^{m} F_n^2(s_j) \end{bmatrix}. \tag{3.147}$$

Note that **M** can also be written as

$$\mathbf{M} = \mathbf{F}^{\top} \cdot \mathbf{F}$$

where

$$\mathbf{F} := \begin{bmatrix} F_0(s_0) & \dots & F_n(s_0) \\ \vdots & & \vdots \\ F_0(s_m) & \dots & F_n(s_m) \end{bmatrix}. \tag{3.148}$$

Comparing with (3.99), p. 117, we see that the matrix **F** corresponds to the one in the interpolation problem. However, in the approximation case **F** is not a square matrix. The number $m+1$ of rows exceeds the number $n+1$ of columns.

The matrix **M** is the so-called *Gram matrix* assigned to the matrix **F** and has the following properties: **M** is positive semidefinite[31] and thus has a non-negative determinant. Moreover, $\det \mathbf{M} > 0$ if and only if the rank of **F** is $n+1$: The $n+1$ columns of **F** are linearly independent. We can summarize:

Proposition 3.14 Solvability of the approximation task. *If $\mathscr{F}$ is a Chebyshev space the Approximation Problem 3.5 has the unique solution* $p(t) = \sum_{i=0}^{n} F_i(t) \cdot a_i$

[31] A matrix **M** of dimension $(n+1) \times (n+1)$ is called *positive semidefinite* if

$$[x_0, \dots, x_n] \cdot \mathbf{M} \cdot \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} \geq 0 \text{ for all } x_0, \dots, x_n \in \mathbb{R}.$$

in $\mathscr{F}$ where $\{F_0(t), \ldots, F_n(t)\}$ is an arbitrary basis of $\mathscr{F}$. The coefficients a_i for the solution can be computed via

$$\begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{M}^{-1} \cdot \begin{bmatrix} \sum_{j=0}^{m} F_0(s_j) \cdot c_j \\ \vdots \\ \sum_{j=0}^{m} F_n(s_j) \cdot c_j \end{bmatrix}. \tag{3.149}$$

Whether or not the approximation task has a unique solution only depends on the space of functions $\mathscr{F}$ considered and the given knot vector $(s_0, \ldots, s_m)$. However, it does not depend on the values $c_0, \ldots, c_m$ to be approximated.

The linear space $\mathbb{R}_n[t]$ of polynomials of degree $\leq n$ (see Sect. 3.2) on any interval $[a, b]$ is a Chebyshev space. Hence the Approximation Problem 3.5 always has a unique solution in $\mathbb{R}_n[t]$. Figure 3.69 shows the solutions for one particular input in the spaces $\mathbb{R}_3[t]$, $\mathbb{R}_4[t]$, $\mathbb{R}_5[t]$.

The same holds for the $(2n + 1)$-dimensional space of trigonometric functions spanned by the functions $1, \sin t, \cos t \sin(2t), \cos(2t), \ldots, \sin(nt), \cos(nt)$ on the interval $[0, 2\pi]$.

At this point it is easy to rephrase Problem 3.5 for points (vectors) to be approximated instead of real values. It then reads as:

Problem 3.6 Univariate approximation of points. Let $m + 1$ points $\mathbf{c}_0, \ldots, \mathbf{c}_m$ in the plane or in 3-space and corresponding parameter values $s_0 < \cdots < s_m$ in a closed interval $[a, b]$ be given. Furthermore let $\mathscr{F}$ be a linear space of continuous functions on $[a, b]$ as in Problem 3.5. Let $\{F_0(t), \ldots, F_n(t)\}$ be a basis of $\mathscr{F}$. Moreover let $m > n = \dim \mathscr{F} - 1$. Find a curve $\mathbf{p}(t)$

$$\mathbf{p}(t) = \sum_{i=0}^{n} F_i(t) \cdot \mathbf{a}_i \tag{3.150}$$

such that the squared error sum (error function)

$$e(\mathbf{a}_0, \ldots, \mathbf{a}_n) = \sum_{j=0}^{m} \left\| \mathbf{p}(s_j) - \mathbf{c}_j \right\|^2 = \sum_{j=0}^{m} \left\| \left(\sum_{i=0}^{n} F_i(s_j) \cdot \mathbf{a}_i \right) - \mathbf{c}_j \right\|^2 \tag{3.151}$$

is minimized.

The following proposition covers the question of solvability of Problem 3.6 and also conveys the solution.[32]

Proposition 3.15 Approximation of points. *If $\mathscr{F}$ is a Chebyshev space there is exactly one solution to Problem 3.6. The coefficient vectors $\mathbf{a}_i$ of the solution curve*

[32] Of course, the approximation task for points can also be solved ‘coordinate-wise’ with the help of Proposition 3.14 We still want to note Proposition 3.15 explicitely as a condensed account on the frequently occurring task regarding points.

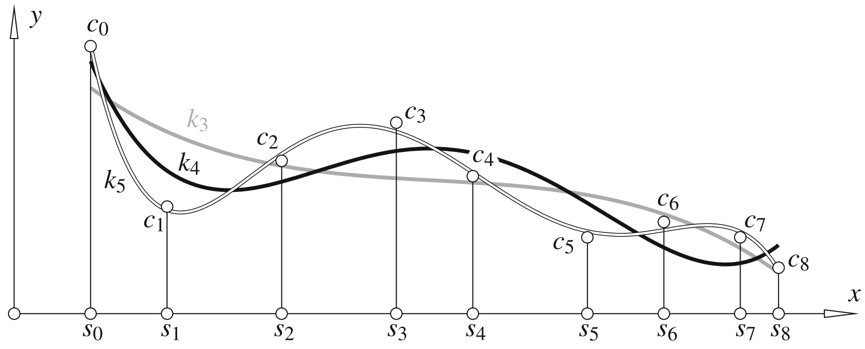


Fig. 3.69 Three different solutions to Problem 3.5. Nine values $c_0, \ldots, c_8$ are to be approximated. The input data are the same as in Fig. 3.57. The approximation problem is solved in different spaces of polynomials. This is why different solutions arise. *Grey curve* k_3: solution in the space $\mathbb{R}_3[x]$. *Black curve* k_4: solution in the space $\mathbb{R}_4[x]$. *Double line* k_5: solution in the space $\mathbb{R}_5[x]$

are[33]

$$\begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \mathbf{M}^{-1} \cdot \begin{bmatrix} \sum_{j=0}^{m} F_0(s_j) \cdot \mathbf{c}_j \\ \vdots \\ \sum_{j=0}^{m} F_n(s_j) \cdot \mathbf{c}_j \end{bmatrix} \tag{3.152}$$

where **M** *is the matrix defined in (3.147).*

3.6.1 Improving the Quality of Approximation

Robust Approximation

Frequently the data points $\mathbf{c}_j$ to be approximated are afflicted with measurement errors. This may render the whole approximation susceptible to coincidence. In order to minimize the influence of single measurement errors we can replace the error sum (3.151) by a weighted variant

$$e(\mathbf{a}_0, \ldots, \mathbf{a}_n) = \sum_{j=0}^{m} w_j \cdot \left\| \left(\sum_{i=0}^{n} F_i(s_j) \cdot \mathbf{a}_i \right) - \mathbf{c}_j \right\|^2 \tag{3.153}$$

where the *weights* w_j are positive real numbers. The value of w_j is chosen small in case of an unreliable point $\mathbf{c}_j$ and large in case of a sound input.

[33] This equation has to be interpreted separately for each component of the vectors $\mathbf{a}_i$, $\mathbf{c}_j$.

In order to allow for the additional parameters (weights) the solution (3.152) to the approximation problem has to be slightly modified:

$$\begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \widetilde{\mathbf{M}}^{-1} \cdot \begin{bmatrix} \sum_{j=0}^{m} w_j \cdot F_0(s_j) \cdot \mathbf{c}_j \\ \vdots \\ \sum_{j=0}^{m} w_j \cdot F_n(s_j) \cdot \mathbf{c}_j \end{bmatrix} \tag{3.154}$$

where

$$\widetilde{\mathbf{M}} := \begin{bmatrix} \sum\limits_{j=0}^{m} w_j \cdot F_0^2(s_j) & \sum\limits_{j=0}^{m} w_j \cdot F_0(s_j) \cdot F_1(s_j) & \dots & \sum\limits_{j=0}^{m} w_j \cdot F_0(s_j) \cdot F_n(s_j) \\ \sum\limits_{j=0}^{m} w_j \cdot F_0(s_j) \cdot F_1(s_j) & \sum\limits_{j=0}^{m} w_j \cdot F_1^2(s_j) & \dots & \sum\limits_{j=0}^{m} w_j \cdot F_1(s_j) \cdot F_n(s_j) \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{j=0}^{m} w_j \cdot F_0(s_j) \cdot w_j \cdot F_n(s_j) & \sum\limits_{j=0}^{m} w_j \cdot F_1(s_j) \cdot w_j \cdot F_n(s_j) & \dots & \sum\limits_{j=0}^{m} w_j \cdot F_n^2(s_j) \end{bmatrix}. \tag{3.155}$$

If the user is aware of possible measurement errors but has no reliable information which of the input data points $\mathbf{c}_j$ are particularly concerned he or she can address the problem as follows:

1. Compute an approximating curve $\mathbf{p}(t)$ without specific weights.
2. Measure the distances of the input points to the resulting curve: If the distance $\|\mathbf{p}(s_j) - \mathbf{c}_j\|$ is particularly large it is likely that $\mathbf{c}_j$ is an unreliable point and thus it gets assigned a small weight, else it gets a larger weight w_j for the next round of computation.
3. Compute a weighted approximation curve using the weights w_j.
4. The whole process could be repeated and the weights could be further adjusted if necessary.

This modified approach increases the overall robustness of the approximation process.

Parameter Correction

For approximation problems the input data $\mathbf{c}_j$ are frequently given without prescribed corresponding parameter values s_j. If so, it is imperative to choose them appropriately.

In the case of interpolation we have already recognized that the choice of the parameter values s_j is important. It makes little sense to choose equidistant values s_j if the distance of consecutive data points $\mathbf{c}_j$ varies a lot. In that case chordal or centripetal parameter distribution (3.141), (3.142), p. 136, is the better choice.

But even if we choose the parameter values s_j sensibly we still must be aware that the distances $\|\mathbf{p}(s_j) - \mathbf{c}_j\|$ occurring in the error sum are not measured along the curve normals. So the point $\mathbf{p}(s_j)$ is not the *pedal point* $\mathbf{l}_j$ of $\mathbf{c}_j$ on the approximation curve c. Measuring these errors on the normals may well improve the result. Unfortunately this is a nonlinear problem. Yet an improvement can be obtained by iterative parameter value correction $s_j \longrightarrow s_j + \sigma_j$ getting the point $\mathbf{p}(s_j)$ gradually closer to the respective pedal point $\mathbf{l}_j$ after a few steps (compare with [7], p. 208).

In order to determine the correcting term σ_j we have to minimize the squared distance function

$$d_j(t) := \|\mathbf{c}_j - \mathbf{p}(t)\|^2$$

for each j where $t = s_j + \sigma_j$. This means that we have to find a zero of its first derivative which after division by 2 reads as[34]

$$f_j(t) = \langle \mathbf{c}_j - \mathbf{p}(t), \dot{\mathbf{p}}(t)\rangle.$$

Expanding $f_j(t)$ by means of a Taylor series at s_j and suppressing higher order terms we obtain a linear approximation of that function:

$$\begin{aligned} f_j(t) &= f_j(s_j + \sigma_j) \sim f_j(s_j) + \dot{f}_j(s_j) \cdot \sigma_j \\ &= \langle \mathbf{c}_j - \mathbf{p}(s_j), \dot{\mathbf{p}}(s_j)\rangle + (-\|\dot{\mathbf{p}}(s_j)\|^2 + \langle \mathbf{c}_j - \mathbf{p}(s_j), \ddot{\mathbf{p}}(s_j)\rangle) \cdot \sigma_j \end{aligned}$$

As a first approach we get

$$\sigma_j \sim \frac{\langle \mathbf{c}_j - \mathbf{p}(s_j), \dot{\mathbf{p}}(s_j)\rangle}{\|\dot{\mathbf{p}}(s_j)\|^2 - \langle \mathbf{c}_j - \mathbf{p}(s_j), \ddot{\mathbf{p}}(s_j)\rangle}. \tag{3.156}$$

This is the quintessence of an iterative procedure which can improve the overall quality of the approximation curve significantly:

Algorithm 3.4. Parameter correction for approximation curves. *To given data points $\mathbf{c}_j$ and appropriately chosen respective parameter values s_j an approximating curve $\mathbf{p}(t)$ is computed (see p. 139). To improve the quality of approximation take the following steps:*

1. *Replace the parameter value s_j by the new value*

$$s_j^* := s_j + \sigma_j$$

   where σ_j is computed via (3.156).
2. *Compute a new approximation curve $\mathbf{p}^*(t)$ with the new parameter values s_j^*.*
3. *If necessary repeat steps 1. and 2.*

The effect of the parameter correction is demonstrated in Fig. 3.70, p. 143.

[34] Here, dots indicate differentiation with respect to the parameter t on the curve.

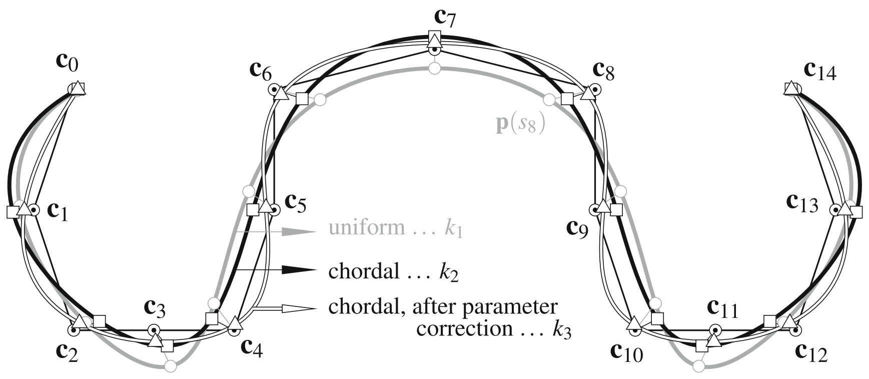


Fig. 3.70 Approximation by cubic ($k = 4$) B-Spline curves; target points $\mathbf{c}_j$ *dotted* in *black*. The *grey curve* k_1 is the variant with uniform parameterization. The points $\mathbf{p}(s_j)$ relating to the parameter values s_j are marked (parameter points, *grey*). The thin grey connection *lines* to the target points symbolize the distances whose square sums have been optimized. The *black* version k_2 (with the squared parameter points) originates from chordal parameterization. Finally, the double *line* variant k_3 has been obtained after 16 rounds of parameter correction according to (3.156), p. 142. It is obvious that the parameter points (marked by triangles) are close to the pedal points of the targets

3.6.2 Approximation with Cubic B-Splines

Cubic B-spline curves are a common tool for approximation tasks. This is because they are easy to compute, behave nicely, are not prone to oscillation and still are sufficiently smooth (C^2-continuity is guaranteed). We want to approximate a given series $\mathbf{c}_0, \dots, \mathbf{c}_m$ of points with corresponding parameter values $s_0, \dots, s_m$ by a cubic B-spline curve

$$\mathbf{p}(t) = \sum_{i=0}^{n} N_{i,4}(t) \cdot \mathbf{a}_i, \quad t \in [t_3, t_{n+1}]. \tag{3.157}$$

with some knot vector $(t_0, t_1, t_2, t_3, t_4, \dots, t_n, t_{n+1}, t_{n+2}, t_{n+3}, t_{n+4})$. As the support interval of the cubic B-Spline curve is $[t_3, t_{n+1}]$ it is necessary to adapt the knot vector to the range of the prescribed parameter values s_i:

$$t_3 \le s_0, \quad s_m \le t_{n+1}.$$

In our examples we put $t_3 = s_0$ and $t_{n+1} = s_m$.

Certainly also the intermediate values $t_4, \dots, t_n$ of the knot vector will affect the shape of the curve—a consistent distribution of the values t_i with respect to the given parameter values s_j is worth considering: One has to avoid the clustering of values

s_j within one interval $[t_i, t_{i+1}]$ as in that case the approximation task might become unsolvable.[35]

Here the matrix $\mathbf{M}$ (see (3.147)) reads as:

$$\mathbf{M} = \begin{bmatrix} \sum_{j=0}^{m} N_{0,3}^2(s_j) & \sum_{j=0}^{m} N_{0,3}(s_j) \cdot N_{1,3}(s_j) & \dots & \sum_{j=0}^{m} N_{0,3}(s_j) \cdot N_{n,3}(s_j) \\ \sum_{j=0}^{m} N_{0,3}(s_j) \cdot N_{1,3}(s_j) & \sum_{j=0}^{m} N_{1,3}^2(s_j) & \dots & \sum_{j=0}^{m} N_{1,3}(s_j) \cdot N_{n,3}(s_j) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=0}^{m} N_{0,3}(s_j) \cdot N_{n,3}(s_j) & \sum_{j=0}^{m} N_{1,3}(s_j) \cdot N_{n,3}(s_j) & \dots & \sum_{j=0}^{m} N_{n,3}^2(s_j) \end{bmatrix}. \tag{3.158}$$

Figure 3.70 shows an example of a cubic B-spline curve approximating a series of 15 target points $\mathbf{c}_j$ in the plane. The different variants arise from different parameterizations as explained above. It is easy to recognize that the choice of parameterization and the iterative parameter correction process are important steps to improve the overall quality of approximation.

3.7 Surfaces

A surface is a 2-dimensional point set in 3-space. Surfaces play an important role in automotive engineering. Not only do they constitute the exterior shell of a vehicle, they also define the shape of virtually each of its components. This section deals with the basic definitions, the mathematical description and the common geometric properties of surfaces.

3.7.1 Parametric Representation of a Surface

A point in 3-space is usually described by a constant vector (position vector). It is sometimes referred to as a *zero-dimensional object*. If a position vector depends on one parameter t we obtain a *curve* (one dimensional object, see Sect. 3.3). A *surface* is a set of points in 3-space described by a vector function in *two* parameters u, v (Fig. 3.71):

[35] Mind that the vector space of cubic splines defined on a given knot vector is—in general—not a Chebyshev space (compare Proposition 3.14, p. 138)!

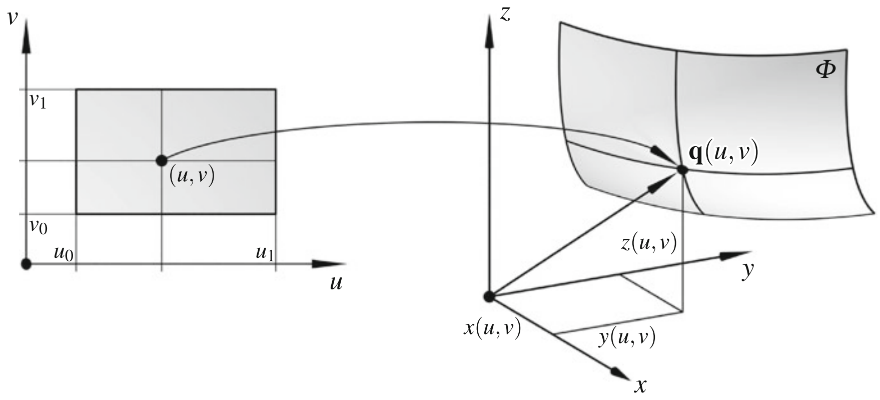


Fig. 3.71 Parameterization of a surface Φ: the mapping on the parameter domain $D = [u_0, u_1] \times [v_0, v_1]$ assigns a point $\mathbf{q}(u, v)$ to every value $(u, v) \in D$. The points $\mathbf{p}(u, v)$ define a surface Φ in 3-space

$$\mathbf{q}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}, \quad (u,v) \in D \tag{3.159}$$

This is called a *parametric representation* or *parameterization* of the surface.

The pairs (u, v) of parameters run in a set $D \subset \mathbb{R} \times \mathbb{R}$ called the *parameter domain* which is frequently rectangular: $D = [u_0, u_1] \times [v_0, v_1]$.

Example 3.12 **Parametric representation of a plane.** Be $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ three non-collinear points in 3-space. Then

$$\mathbf{q}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = u \cdot \mathbf{a} + v \cdot \mathbf{b} + (1 - u - v) \cdot \mathbf{c}, \quad (u,v) \in \mathbb{R} \times \mathbb{R} \tag{3.160}$$

is a parametric representation of the plane ε through $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$. The given points $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ relate to the parameter values $(u = 1, v = 0)$, $(u = 0, v = 1)$ and $(u = 0, v = 0)$. If we substitute $1 - u - v = w$ (compare also Definition 3.66, p. 226) the plane ε is represented by:

$$\tilde{\mathbf{q}}(u,v,w) = u \cdot \mathbf{a} + v \cdot \mathbf{b} + w \cdot \mathbf{c} \ \text{ with } u + v + w = 1 \tag{3.161}$$

Definition 3.39. Admissible parameterization of a surface. A parameterization (3.159) of a surface is called *admissible* if it is a differentiable vector function in u and v and if for all $u, v \in D$ the derivative vectors

$$\mathbf{q}_u(u,v) \;:=\; \frac{d\mathbf{q}}{du}(u,v),\quad \mathbf{q}_v(u,v) \;:=\; \frac{d\mathbf{q}}{dv}(u,v)$$

are linearly independent:

$$\mathbf{q}_u \times \mathbf{q}_v \neq \mathbf{0}$$

Remark 3.13 The parameterization of a surface is not uniquely determined. The same point set could also be parameterized differently. The change

$$\left.\begin{aligned} u &= u(\overline{u},\overline{v}) \\ v &= v(\overline{u},\overline{v}) \end{aligned}\right\} \tag{3.162}$$

from one parameterization to another is called *parameter transformation* (compare also Remark 3.5, p. 72).

The functions (3.162) are called an *admissible parameter transformation* if their first partial derivatives exist and satisfy

$$\det\begin{bmatrix} \frac{\partial u}{\partial \overline{u}} & \frac{\partial v}{\partial \overline{u}} \\ \frac{\partial u}{\partial \overline{v}} & \frac{\partial v}{\partial \overline{v}} \end{bmatrix} = \frac{\partial u}{\partial \overline{u}}\cdot\frac{\partial v}{\partial \overline{v}} - \frac{\partial u}{\partial \overline{v}}\cdot\frac{\partial v}{\partial \overline{u}} \;\neq\; 0 \text{ for all } (u,v)\in D. \tag{3.163}$$

Due to the chain rule of differentiation any admissible parameter transformation (3.162), substituted in an admissible parameterization (3.159) again delivers an admissible parameterization of the same surface.

3.7.2 Surface Curves

If a surface Φ is given by a parametric representation it is easy to describe curves contained in it:

Definition 3.40. Surface curve. Let Φ be a surface with its parametric representation (3.159). We consider two functions

$$\left.\begin{aligned} u &= u(t) \\ v &= v(t) \end{aligned}\right\} \tag{3.164}$$

in one variable t determining a parameterized curve k in the domain D which in turn delivers a curve

$$\mathbf{p}(t) := \mathbf{q}(u(t),v(t)) \;=\; \begin{bmatrix} x(u(t),v(t)) \\ y(u(t),v(t)) \\ z(u(t),v(t)) \end{bmatrix}, \tag{3.165}$$

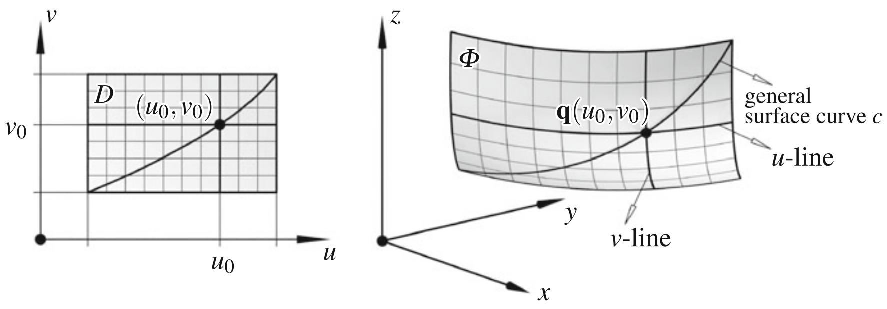


Fig. 3.72 Any curve $u = u(t), v = v(t)$ on the parameter domain D yields a surface curve $c \ldots \mathbf{p}(u(t), v(t))$. As special cases of such surface curves we have the u-lines and the v-lines defined by the parameterization of Φ. Every point $\mathbf{q}(u_0, v_0)$ on Φ is the intersection of a u-line $u = u_0$ and a v-line $v = v_0$

lying on the surface Φ (Fig. 3.72). We call it a *surface curve*.

The very special case of

$$\left.\begin{aligned} u &= t \\ v &= v_0 = const \end{aligned}\right\} \tag{3.166}$$

yields a whole set of such surface curves depending on the constant $v = v_0$. We call them the *u-lines*. In the same way

$$\left.\begin{aligned} u &= u_0 = const \\ v &= t \end{aligned}\right\} \tag{3.167}$$

yields a set of surface curves, called *v-lines*. The u-lines and the v-lines are generally referred to as the *net of parameter lines*.

These two sets of surface curves certainly depend on the given parametric representation $\mathbf{q}(u, v)$ of Φ. If we reparameterize the surface by means of a parameter transformation (Remark 3.13) we—in general—obtain different sets of u-lines and v-lines. The parameter lines are often used to represent a surface. Their shape can convey the shape of the surface Φ.

3.7.3 Derivatives and Tangent Planes

Definition 3.41. Class of differentiability. Let k be a non-negative integer and let Φ be a surface with parameterization $\mathbf{q} = \mathbf{q}(u, v)$, $(u, v) \in D$. We say that the parameterization $\mathbf{q}(u, v)$ is *of class* C^k if all partial derivatives

$$\frac{\partial^k \mathbf{q}}{(\partial u)^i (\partial v)^j}(u, v),\ i + j = k, \tag{3.168}$$

of order k exist[36] and are continuous vector functions. C^k is also called the *class of differentiability* of the given parameterization.

If the k-th partial derivatives of a bivariate vector function $\mathbf{p}(u, v)$ exists for *every* integer $k \in \mathbb{N}$ we say that $\mathbf{q}(u, v)$ is *of class C^∞* or *smooth.*

Let Φ be a surface with a parameterization $\mathbf{q} = \mathbf{q}(u, v)$ of class C^1 and let c be a surface curve parameterized by $\mathbf{p}(t) = \mathbf{q}(u(t), v(t))$ with functions $u = u(t)$, $v = v(t)$ of class C^1. Then the tangent vector to c at $t = t_0$ can be computed via the chain rule of differentiation:

$$\begin{aligned}\dot{\mathbf{p}}(t_0) &= \frac{d\mathbf{p}}{dt}(t_0)\\ &= \frac{d\mathbf{q}}{du}(u_0, v_0) \cdot \frac{du}{dt}(t_0) + \frac{d\mathbf{q}}{dv}(u_0, v_0) \cdot \frac{dv}{dt}(t_0)\\ &= \mathbf{q}_u(u_0, v_0) \cdot \dot{u}(t_0) + \mathbf{q}_v(u_0, v_0) \cdot \dot{v}(t_0)\end{aligned} \tag{3.169}$$

with $u_0 := u(t_0)$ and $v_0 := v(t_0)$. Thus we see that the tangent vector of any surface curve through a point $\mathbf{q}(u_0, v_0)$ of the surface is a linear combination of the tangent vectors $\mathbf{q}_u$, $\mathbf{q}_v$ of the parameter lines intersecting at that point. If the surface parameterization $\mathbf{q}(u, v)$ is admissible (see Definition 3.39, p. 145) the partial derivative vectors $\mathbf{q}_u, \mathbf{q}_v$ at (u_0, v_0) are linearly independent and thus they span a plane τ_Q through the point $Q \dots \mathbf{q}(u_0, v_0)$ containing all such tangent vectors (Fig. 3.73):

Definition 3.42. Tangent plane and normal vector in a point. Let Φ be a surface with an admissible parameterization $\mathbf{q} = \mathbf{q}(u, v)$ of class C^1, and let $Q \dots \mathbf{q}(u_0, v_0)$

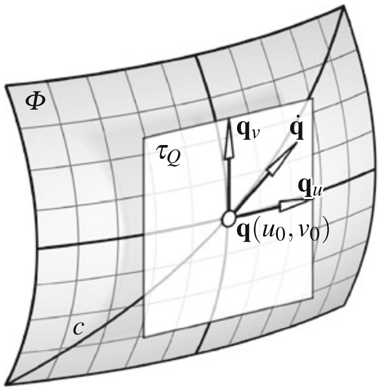


Fig. 3.73 The tangent plane τ_Q at $\mathbf{q}(u_0, v_0)$. τ_Q is spanned by the partial derivative vectors $\mathbf{q}_u, \mathbf{q}_v$; they are the tangent vectors to the parameter lines through $\mathbf{q}(u_0, v_0)$. The tangent to any surface curve c through $\mathbf{q}(u_0, v_0)$ is contained in τ_Q

[36] Of course, this automatically implies that all derivatives of order $< k$ exist as well.

be one of its points. Then the plane τ_Q through Q spanned by the vectors $\mathbf{q}_u(u_0, v_0)$, $\mathbf{q}_v(u_0, v_0)$ is called *tangent plane to Φ* at $\mathbf{q}(u_0, v_0)$. The tangent plane in a point $Q \dots \mathbf{q}(u_0, v_0)$ is also determined by the *normal vector*

$$\mathbf{n}(u_0, v_0) = \mathbf{q}_u(u_0, v_0) \times \mathbf{q}_v(u_0, v_0). \tag{3.170}$$

The plane τ_Q has the equation

$$\tau_Q \dots \langle \mathbf{n}(u_0, v_0), \mathbf{x} - \mathbf{q}(u_0, v_0) \rangle = 0. \tag{3.171}$$

It is a major task in CAD to fit surface patches smoothly together. The following definitions and explanations illuminate the mathematical background of this issue.

Definition 3.43. ***C^k*-continuity of two surfaces in a point.** Let Φ_1 and Φ_2 be two surfaces with the parameterizations $\mathbf{q}_1 = \mathbf{q}_1(u, v)$ and $\mathbf{q}_2 = \mathbf{q}_2(u, v)$, both of class C^k, $k \geq 0$. Then the parameterizations join *C^k-continuously at* (u_0, v_0) if

$$\frac{\partial^l \mathbf{q}_1}{(\partial u)^i (\partial v)^j}(u_0, v_0) = \frac{\partial^l \mathbf{q}_2}{(\partial u)^i (\partial v)^j}(u_0, v_0), \quad \text{for all } l = 0, \dots, k, \ i + j = l.$$

A parameter transformation according to Remark 3.13 only affects the means of representation but not the point set of a surface itself. Definition 3.43, however, relates to the parametric representations of two surfaces. It seems evident that the concept of two surfaces joining C^k-continuously in a point could favorably be replaced by a somewhat weaker definition which is closer to geometry:

Definition 3.44. ***GC^k*-continuity of two surfaces in a point.** Let two surfaces Φ_1 and Φ_2 be given by parameterizations $\mathbf{r}_1 = \mathbf{r}_1(s, t)$ and $\mathbf{r}_2 = \mathbf{r}_2(\sigma, \tau)$ and let them have a common point, i.e., $\mathbf{r}_1(s_0, t_0) = \mathbf{r}_2(\sigma_0, \tau_0) =: \mathbf{a}$; then Φ_1 and Φ_2 are said to *join GC^k-continuously at the point* $\mathbf{a}$ if there exist reparameterizations

$$\mathbf{q}_1(u, v) := \mathbf{r}_1(s(u, v), t(u, v))$$

and

$$\mathbf{q}_2(u, v) := \mathbf{r}_2(\sigma(u, v), \tau(u, v))$$

of Φ_1 and Φ_2 such that the new parametric representations $\mathbf{q}_1(u, v)$, $\mathbf{q}_2(u, v)$ of Φ_1, Φ_2 are C^k-continuous at (u_0, v_0) in the sense of Definition 3.43.

In the reparameterization the common point is $\mathbf{a} = \mathbf{q}_1(u_0, v_0) = \mathbf{q}_2(u_0, v_0)$.

Mind that two surfaces joining GC^k-continuously cannot be distinguished visually from surfaces joining C^k-continuously in that point. In particular, surfaces joining C^0-, C^1-continuously frequently occur in surface modeling:

- Two surfaces Φ_1 and Φ_2 joining C^0-continuously in a point, for one, have a common point at u_0, v_0: $\mathbf{q}_1(u_0, v_0) = \mathbf{q}_2(u_0, v_0)$

- Two surfaces joining C^1-continuously in a point additionally have the same tangent plane at that point. Of course, the same holds for GC^1-continuous surfaces.

Applications in automotive engineering are predominantly C^2-continuous, especially those at the body shell (see also Remark 3.14) and Sect. 4.2.6, p. 274.

Definition 3.45. C^k**- and** GC^k**-continuity along a surface curve.** Let Φ_1 and Φ_2 be two surfaces.

1. If the parameterizations of Φ_1 and Φ_2 join C^k-continuously in every point of the common surface curve c we say that Φ_1 and Φ_2 join C^k*-continuously along* c.
2. If there exist parameter transformations of the two surfaces such that the new parametric representations are C^k-continuous for every point of the surface curve c then we say that Φ_1 and Φ_2 join GC^k*-continuously along* c.

C^k-continuous and GC^k-continuous transitions of surfaces along a curve c frequently appear in Sect. 3.8, p. 180, and generally in CAD applications. The following remark states that in such a case we only have to make provisions that the partial derivatives in the *cross direction* are identical. The derivatives along the common curve c are identical anyway (which is trivial). The mixed derivatives along c are also identical if only the derivatives in cross direction comply. We record:

Remark 3.14 Two surfaces $\Phi_1 \dots \mathbf{q}_1(u, v)$ and $\Phi_2 \dots \mathbf{q}_2(u, v)$ with a common u-line $c \dots v = v_0$ join C^k-continuously along c if only

$$\frac{\partial^l \mathbf{q}_1}{(\partial v)^l}(u, v_0) = \frac{\partial^l \mathbf{q}_2}{(\partial v)^l}(u, v_0) \text{ for all } l = 0, \dots, k \text{ and for all } u.$$

Obviously, an analogous statement holds for a common v-line of two surfaces.

Another interesting phenomenon is described in

Remark 3.15 Let Φ be a surface consisting of two patches Φ_1 and Φ_2 joined C^k-continuously along a transition curve c. Then a smooth space curve reflected in the surface Φ only shows C^{k-1}-continuity at the transition point on c.

We can observe that the continuity of the reflected curves is (by one degree) less than the continuity of the surface transition. This is why regarding reflected curves on a surface unveils a lot more about discontinuities between the two surfaces than just considering ordinary surface curves.

In Fig. 3.74 we show a few examples, particularly for the cases $k = 0, 1, 2$. Figure 3.74a just illustrates the test arrangement for the following Fig. 3.74b–d: A zebra-striped cylindric surface Ψ encircles a given surface Φ. This surface Φ is our sample whose continuity shall be examined visually. Φ is meant to be reflective such that the mirror images of the zebra stripes can be viewed in the glossy surface. Φ consists of two smooth patches Φ_1 and Φ_2. We focus on the reflected image of the stripes, and particularly on their points on the transition curve.

According to Remark 3.15 the behavior of the reflected curves is more sensitive than the transition itself. They are also more sensitive by one degree than a cross-

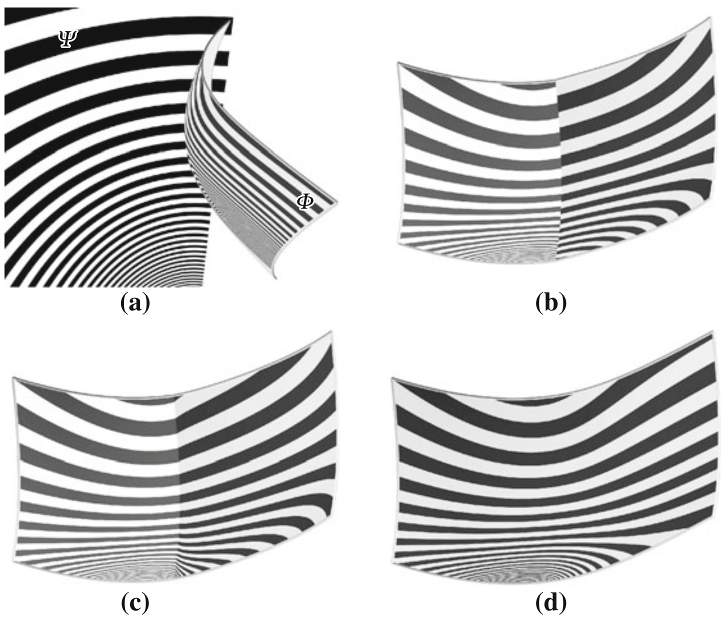


Fig. 3.74 Two patches, joined with different degrees of continuity. In order to visualize the quality of transition a test arrangement (Fig. 3.74a) is readied. The reflections of the striped cylinder on the given patch are shown in the following Fig. 3.74b–d. They are always more sensitive to discontinuities: if the surface patches are joined with C^k-continuity along their transition curve the reflected curves are only C^{k-1}-continuous at the transition point. **a** The pattern of the zebra-striped cylindric surface Ψ is reflected in a patch Φ. The patch consists of two parts joined with different degrees of continuity along the transition curve. The following Fig. 3.74b–d shows the reflected pattern. **b** Two surfaces joined C^0-continuously along a common curve. The reflected stripes show gaps along this curve. **c** Two surfaces with C^1-continuous transition. At any point on the transition curve the tangent planes of both surfaces are the same. The reflected stripes have kinks at the respective transition points. **d** Two surfaces with C^2-continuous transition along the common curve. Mind that even the reflected images of the stripes cross the transition curve C^1-continuously

section of the patch would be. If the two patches are joined C^0-continuously the reflected stripes are not even continuous at all. Moreover, they show gaps along the transition curve (Fig. 3.74b). We can equally observe that mere C^1-continuity of the two patches provokes kinks of the reflected curves along the transition (Fig. 3.74c). C^2-continuity of the two joined patches leads to C^1-continuity of the reflected curves. This is why—in automotive styling—C^2-transitions are widely used; consequently, reflections of smooth lines on a car body are at least C^1-curves (Fig. 3.74d). This way, transitions between adjacent patches can hardly be detected with the naked eye.

3.7.4 Curvature Theory of Surfaces

We consider a surface Φ with an admissible parameterization

$$\mathbf{q}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}, \quad (u,v) \in D \tag{3.172}$$

and one of its surface curves c (cf. Fig. 3.72, p. 147)

$$\mathbf{p}(t) := \mathbf{q}(u(t), v(t)) \;\; = \;\; \begin{bmatrix} x(u(t), v(t)) \\ y(u(t), v(t)) \\ z(u(t), v(t)) \end{bmatrix}. \tag{3.173}$$

We additionally assume that all occurring functions are of class C^2 and, beyond this, that $(\dot{u}, \dot{v}) \neq (0,0)$ for all $(u,v) \in D$. So, the tangent vector

$$\dot{\mathbf{p}} = \frac{d\mathbf{q}}{du} \cdot \frac{du}{dt} + \frac{d\mathbf{q}}{dv} \cdot \frac{dv}{dt} = \mathbf{q}_u \cdot \dot{u} + \mathbf{q}_v \cdot \dot{v}. \tag{3.174}$$

of c is always different from zero. If s is the arc length parameter on c we have (cf. Sect. 3.3.4, p. 76)

$$\begin{aligned} \left(\frac{ds}{dt}\right)^2 &= \|\dot{\mathbf{p}}\|^2 = \|\mathbf{q}_u \cdot \dot{u} + \mathbf{q}_v \cdot \dot{v}\|^2 \\ &= \|\mathbf{q}_u\|^2 \cdot \dot{u}^2 + 2\langle \mathbf{q}_u, \mathbf{q}_v \rangle \cdot \dot{u} \cdot \dot{v} + \|\mathbf{q}_v\|^2 \cdot \dot{v}^2. \end{aligned}$$

With the abbreviations

$$E := \|\mathbf{q}_u\|^2, \quad F := \langle \mathbf{q}_u, \mathbf{q}_v \rangle, \quad G := \|\mathbf{q}_v\|^2 \tag{3.175}$$

we can rewrite this as

$$\left(\frac{ds}{dt}\right)^2 = E \cdot \dot{u}^2 + 2F \cdot \dot{u} \cdot \dot{v} + G \cdot \dot{v}^2. \tag{3.176}$$

If we reparameterize the surface curve c by its arc length parameter s and denote derivatives with respect to s by primes we have

$$\mathbf{p}' = \mathbf{t}, \quad \mathbf{p}'' = \kappa \cdot \mathbf{h} \tag{3.177}$$

where $\mathbf{t}$ and $\mathbf{h}$ are the unit tangent vector and principal normal vector of c (cf. Sect. 3.3.7, p. 79). Recall that the vectors $\mathbf{t}$ and $\mathbf{h}$ span the osculating plane σ of c in the respective point (cf. Sect. 3.3.6, p. 78). Via the chain rule of differentiation we get

$$\mathbf{p}' = \mathbf{q}_u \cdot u' + \mathbf{q}_v \cdot v', \tag{3.178}$$

$$\mathbf{p}'' = \mathbf{q}_{uu} \cdot u'^2 + 2\mathbf{q}_{uv} \cdot u' \cdot v' + \mathbf{q}_{vv} \cdot v'^2 + \mathbf{q}_u \cdot u'' + \mathbf{q}_v \cdot v''. \tag{3.179}$$

If φ denotes the angle between the surface unit normal vector

$$\mathbf{n} := \frac{\mathbf{q}_u \times \mathbf{q}_v}{\|\mathbf{q}_u \times \mathbf{q}_v\|} \tag{3.180}$$

and the principal normal vector $\mathbf{h}$ of c we obtain

$$\begin{aligned}\kappa \cdot \cos\varphi &= \kappa \cdot \langle \mathbf{n}, \mathbf{h} \rangle = \langle \mathbf{n}, \mathbf{p}'' \rangle \\ &= \langle \mathbf{n}, \mathbf{q}_{uu} \rangle \cdot u'^2 + 2\langle \mathbf{n}, \mathbf{q}_{uv} \rangle \cdot u' \cdot v' + \langle \mathbf{n}, \mathbf{q}_{vv} \rangle \cdot v'^2 \end{aligned} \tag{3.181}$$

which follows from (3.177), (3.179) and from the fact that $\mathbf{n}$ is perpendicular to the vectors $\mathbf{q}_u$, $\mathbf{q}_v$. Returning to some parameter t on c we have to allow for

$$u' = \dot{u} \cdot \frac{dt}{ds},\ v' = \dot{v} \cdot \frac{dt}{ds}$$

and Eq. (3.181) reads as

$$\kappa \cdot \cos\varphi = \frac{\langle \mathbf{n}, \mathbf{q}_{uu} \rangle \cdot \dot{u}^2 + 2\langle \mathbf{n}, \mathbf{q}_{uv} \rangle \cdot \dot{u} \cdot \dot{v} + \langle \mathbf{n}, \mathbf{q}_{vv} \rangle \cdot \dot{v}^2}{\left(\frac{ds}{dt}\right)^2}. \tag{3.182}$$

Putting

$$L := \langle \mathbf{n}, \mathbf{q}_{uu} \rangle, \quad M := \langle \mathbf{n}, \mathbf{q}_{uv} \rangle, \quad N := \langle \mathbf{n}, \mathbf{q}_{vv} \rangle \tag{3.183}$$

and substituting (3.176) we finally get

$$\kappa \cdot \cos\varphi = \frac{L \cdot \dot{u}^2 + 2M \cdot \dot{u} \cdot \dot{v} + N \cdot \dot{v}^2}{E \cdot \dot{u}^2 + 2F \cdot \dot{u} \cdot \dot{v} + G \cdot \dot{v}^2}. \tag{3.184}$$

If, especially, $\varphi = 0$ or $\varphi = \pi$—which means that the osculating plane σ of the surface curve c contains the surface normal—we have

$$\kappa = \left| \frac{L \cdot \dot{u}^2 + 2M \cdot \dot{u} \cdot \dot{v} + N \cdot \dot{v}^2}{E \cdot \dot{u}^2 + 2F \cdot \dot{u} \cdot \dot{v} + G \cdot \dot{v}^2} \right|. \tag{3.185}$$

We now introduce the parameter

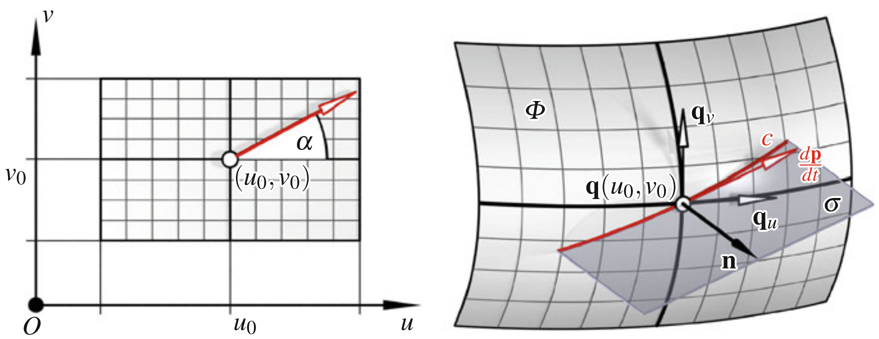


Fig. 3.75 The normal curvature of a surface Φ can be computed to every $\lambda = \tan\alpha$ via (3.188). It coincides with the curvature of the normal section of $c = \Phi \cap \sigma$ in that direction. The intersection plane σ is spanned by the surface normal vector $\mathbf{n}$ and the tangent vector $\frac{d\mathbf{p}}{dt} = \mathbf{q}_u \frac{du}{dt} + \mathbf{q}_v \frac{dv}{dt}$

$$\lambda := \frac{dv}{du} = \frac{\dot{v}}{\dot{u}} \tag{3.186}$$

which equals $\tan\alpha$ where α denotes the angle between the u-axis and the direction $[\dot{u}, \dot{v}]^\top$ in the u, v-parameter plane (Fig. 3.75):

$$\lambda = \tan\alpha$$

Thus (3.185) reads as

$$\kappa = \left| \frac{L + 2M \cdot \lambda + N \cdot \lambda^2}{E + 2F \cdot \lambda + G \cdot \lambda^2} \right|. \tag{3.187}$$

Definition 3.46. Normal curvature. For a C^2-surface Φ given by the parameterization (3.172) the *normal curvature* of Φ in the point $\mathbf{q}(u_0, v_0)$ belonging to the tangential direction $\lambda = \frac{dv}{du}$ (see (3.186)) is defined by

$$\kappa_n(\lambda) = \frac{L + 2M \cdot \lambda + N \cdot \lambda^2}{E + 2F \cdot \lambda + G \cdot \lambda^2} \tag{3.188}$$

evaluated at (u_0, v_0).

Note that the normal curvature κ_n is—up to the sign—equal to the curvature κ of the normal section of Φ with the plane σ through the surface tangent g (Fig. 3.75):

$$g \ldots \mathbf{g}(w) = \mathbf{q} + w \cdot (\mathbf{q}_u \cdot \dot{u} + \mathbf{q}_v \cdot \dot{v}) = \mathbf{q} + w \cdot \dot{\mathbf{p}}$$

For any fixed point $\mathbf{q}(u_0, v_0)$ on the surface Φ the normal curvature $\kappa_n(\lambda)$ is a rational quadratic function in λ.

We record some properties of $\kappa_n(\lambda)$:

1. Due to (3.176) the denominator of (3.188) is always positive which means that $\kappa_n(\lambda)$ is defined for all $\lambda \in \mathbb{R}$.
2. The rational quadratic function $\kappa_n(\lambda)$ is constant if and only if

$$E : F : G = L : M : N. \tag{3.189}$$

A surface point $\mathbf{q}(u_0, v_0)$ with this property is called a *umbilic point*: All normal sections of the surface through such a point have the same curvature. On a sphere this is obviously the case for each point. If Φ is an arbitrary surface and $\mathbf{q}(u_0, v_0)$ is a umbilic point on Φ then we say that *in a neighborhood of* $\mathbf{q}(u_0, v_0)$ *the surface* Φ *behaves like a sphere.*
3. For any *non-umbilic point* $\mathbf{q}(u_0, v_0)$, i.e., $E : F : G \neq L : M : N$ in that point, one can show (see [5], p. 91) that $\kappa_n = \kappa_n(\lambda)$ has exactly two different extremal values $\kappa_{n,1} = \kappa_n(\lambda_1)$ and $\kappa_{n,2} = \kappa_n(\lambda_2)$ where the values λ_1 and λ_2 can be computed as the zeroes of the quadratic polynomial

$$f(\lambda) = \begin{vmatrix} \lambda^2 & -\lambda & 1 \\ E & F & G \\ L & M & N \end{vmatrix} = (FN - GM) \cdot \lambda^2 + (EN - GL) \cdot \lambda + (EM - FL).$$

The two extremal values $\kappa_{n,1}$ and $\kappa_{n,2}$ are called *principal curvatures of* Φ at $\mathbf{q}(u_0, v_0)$. They satisfy the conditions

$$\kappa_{n,1} \cdot \kappa_{n,2} = \frac{LN - M^2}{EG - F^2},$$
$$\kappa_{n,1} + \kappa_{n,2} = \frac{EN - 2FM + GL}{EG - F^2}.$$

The directions of the respective normal sections yielding these extremal curvatures are even orthogonal to each other.

We give the following

Definition 3.47. Gaussian curvature and mean curvature of a surface. Let Φ be a C^2-surface given by the parameterization (3.172). Then the *Gaussian curvature* K and the *mean curvature* H are defined by

$$K := \kappa_{n,1} \cdot \kappa_{n,2} = \frac{LN - M^2}{EG - F^2}, \tag{3.190}$$
$$H := \frac{\kappa_{n,1} + \kappa_{n,2}}{2} = \frac{EN - 2FM + GL}{2(EG - F^2)}. \tag{3.191}$$

The Gaussian curvature enables us to classify the surface points in three categories:

Definition 3.48. Elliptical, hyperbolic and parabolic surface points. Let Φ be a C^2-surface given by the parameterization (3.172). Then a non-umbilic point $\mathbf{q}(u_0, v_0)$ on Φ

$$\text{is called} \left\{ \begin{array}{c} \textit{elliptical} \\ \textit{hyperbolic} \\ \textit{parabolic} \end{array} \right\} \text{point if} \left\{ \begin{array}{c} K > 0 \\ K < 0 \\ K = 0 \end{array} \right\}.$$

To find out the geometric meaning of this categorization we study the normal sections of the surface Φ through the given point $\mathbf{q}(u_0, v_0)$. Obviously, the centers of curvature $\mathbf{c}(\lambda)$ of those normal sections lie on the surface normal n of Φ through $\mathbf{q}(u_0, v_0)$. The point $\mathbf{q}(u_0, v_0)$ splits the normal n into two rays, say n^+ and n^-.

- If $\mathbf{q}(u_0, v_0)$ is an *elliptical point* the principal curvatures $\kappa_{n,1}, \kappa_{n,2}$ are both different from zero and have the same sign. Additionally, one can show that $\kappa = \kappa(\lambda)$ has this sign for all $\lambda \in \mathbb{R}$. This means that all centers of curvature $\mathbf{c}(\lambda)$ are distributed on only one of the two rays n^+, n^-. Locally, the surface lies only on one side of its tangent plane at the point $\mathbf{q}(u_0, v_0)$—the surface is locally convex. An example is the ellipsoid (Fig. 3.79, p. 162) which has elliptical points only.
- If $\mathbf{q}(u_0, v_0)$ is a *hyperbolic point* the principal curvatures $\kappa_{n,1}$, $\kappa_{n,2}$ are both different from zero but have different signs. In this case, one can show that $\kappa = \kappa(\lambda)$ has exactly two zeroes: The centers of curvature $\mathbf{c}(\lambda)$ are distributed on both rays n^+ and n^-. Hence, the surface intersects its tangent plane at the point $\mathbf{q}(u_0, v_0)$ in some curve. Locally the surface is *saddle shaped.* A typical example is the hyperbolic paraboloid (Fig. 3.84, p. 165) which has hyperbolic points only; this, by the way, accounts for its name.
- If $\mathbf{q}(u_0, v_0)$ is a *parabolic point* one of the two principal curvatures $\kappa_{n,1}$, $\kappa_{n,2}$ is zero. A cylinder (cf. Definition 3.52, p. 166), for instance, has parabolic points only.

If a surface Φ contains both, elliptical and hyperbolical points, then in general each sort defines one or more regions on Φ. In this case each boundary curve between an elliptical and a hyperbolic region consists of parabolic points. A typical example is a torus Φ (cf. Example 3.19, p. 172) where the parabolic points set up two circles on Φ separating the region of elliptical points and the one of hyperbolic points (Fig. 3.76).

The importance of the Gaussian curvature K and the mean curvature H is also illustrated by

Theorem 3.8. Characterization of particular surfaces by special Gaussian and mean curvature values.

- Developable surfaces *(cf. Sect. 3.7.10, pp. 167) are characterized by* $K \equiv 0$.
- *If* $K = H \equiv const \neq 0$ *holds for a surface* Φ*, then* Φ *is a sphere and vice versa.*
- *If* $K = H \equiv 0$ *holds for a surface* Φ*, then* Φ *is a plane and vice versa.*
- Minimal surfaces *are characterized by* $H \equiv 0$.

Minimal surfaces are an interesting geometric issue as they minimize the total surface area subject to some constraint: If a wire frame is dipped into a soap solution the soap film assumes a minimal surface whose boundary is the wire frame.

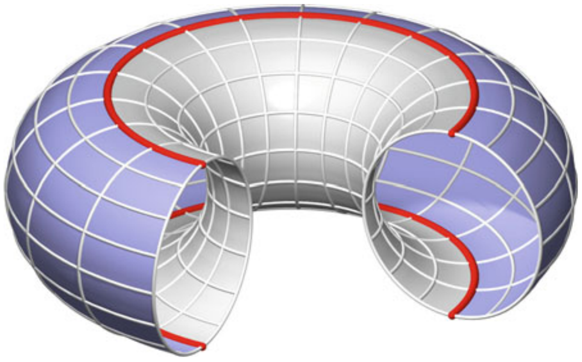


Fig. 3.76 On a torus we have one region of elliptic points (*blue*, Gaussian curvature $K > 0$), another one consisting of hyperbolic points (*gray*, Gaussian curvature $K < 0$). The boundary curves of these two regions are *two circles* (*red*) whose points are parabolic (Gaussian curvature $K = 0$)

Remark 3.16 The differential form

$$I := E \cdot du^2 + 2F \cdot du\, dv + G \cdot dv^2 \tag{3.192}$$

which is based on (3.176) is called the *first fundamental form* of the surface Φ. The Eqs. (3.183) and (3.184) suggest the introduction of a further differential form

$$II := L \cdot du^2 + 2M \cdot du\, dv + N \cdot dv^2 \tag{3.193}$$

which is referred to as the *second fundamental form* of the surface Φ.

The first fundamental form is in charge of the surface metric. Both fundamental forms control the curvature of surface curves in terms of (3.182).

Gaussian and mean curvature also appear in various CAD-applications (compare also Sect. 4.2.6, pp. 274).

3.7.5 Surfaces Represented by Equations

To define a surface by a parametric representation is not the only option we have. A surface Φ in the 3-space might also be determined by some equation

$$F(x, y, z) = 0. \tag{3.194}$$

A point $\mathbf{p}$ is contained in Φ if its coordinates (x, y, z) fulfill the condition (3.194). If one of the coordinates—say z—can be expressed explicitly in that equation we can transform (3.194) into

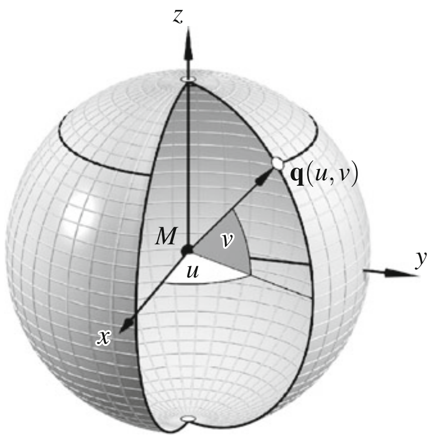


Fig. 3.77 A sphere with its standard (geographic) parameterization. The parameter u is the longitude, v measures the geographical latitude. The u-lines are the circles of latitude, the v-lines are the circles of longitude (meridians). Both, its parameterization (3.196) and its algebraic Eq. (3.197) have their specific merits

$$z = f(x, y). \tag{3.195}$$

Example 3.13 **Sphere.** A sphere (Fig. 3.77) with center $\mathbf{m}(x_0, y_0, z_0)$ and radius r can be parameterized by[37]

$$\mathbf{q}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} = \begin{bmatrix} x_0 + r \cdot \cos u \cdot \cos v \\ y_0 + r \cdot \sin u \cdot \cos v \\ z_0 + r \cdot \sin v \end{bmatrix}, \tag{3.196}$$
$$(u, v) \in [0, 2\pi) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right].$$

On the other hand it can be described by the equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0. \tag{3.197}$$

It is simple to check if a given parameterization $\mathbf{q}(u, v) = [x(u, v), y(u, v), z(u, v)]^\top$, $(u, v) \in D$ and an Eq. (3.194) describe the same surface: One only has to substitute $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$ into (3.194) and verify if the emerging equation is satisfied for all $(u, v) \in D$.

3.7.6 Algebraic Surfaces

Among the surfaces defined by equations there are some which deserve particular attention: those whose equations consist of polynomial expressions only.

[37] This is the so-called geographic parameterization of a sphere as the parameters u and v represent the geographic longitude and latitude, respectively.

Definition 3.49. Algebraic surface. We consider some polynomial equation in the variables x, y, z.

1. A surface Φ is called **algebraic** if it is defined by a polynomial equation:

$$F(x, y, z) \;=\; \sum_{finite} a_{ijk}\, x^i\, y^j\, z^k = 0 \tag{3.198}$$

2. The algebraic surface is called *of order n* if $\deg F(x, y, z) = n$ (cf. Definition 3.11, p. 68).
3. A surface which cannot be described by a polynomial equation is called a *transcendent surface*.

Example 3.14 **Planes.** Any algebraic surface of order 1 is a plane:

$$Ax + By + Cz + D = 0$$

Example 3.15 **Sphere.** The sphere is an algebraic surface of order 2 because in its Eq. (3.196) there are only terms of degree 2 or less.

Example 3.16 **Transcendent surface.** One example of a transcendent surface is

$$z \cos x + \mathrm{e}^y x = 0.$$

The helical surfaces (Sect. 3.7.12, pp. 173) are further examples of transcendent surfaces.

The *order of an algebraic surface* can be interpreted geometrically. Similarly to Theorem 3.4, p. 82 we have:

Theorem 3.9. Intersection of an algebraic surface with a straight line or a plane. *An algebraic surface Φ or order n has the following properties:*

1. *A straight line g which is not contained in Φ, has no more than n points of intersection with Φ.*
2. *A plane ε which is not contained in Φ, intersects Φ in a planar algebraic curve of order n (see Definition 3.22, p. 81).*

3.7.7 Rational Surfaces

Similar to rational curves (Sect. 3.3.9, p. 82) we define rational surfaces:

Definition 3.50. Rational surface. A surface Φ is called a *rational surface* if it can be represented by a rational parameterization

$$\mathbf{q}(u,v) = \begin{bmatrix} \frac{q_1(u,v)}{q_0(u,v)} \\ \frac{q_2(u,v)}{q_0(u,v)} \\ \frac{q_3(u,v)}{q_0(u,v)} \end{bmatrix} \tag{3.199}$$

where $q_i(u, v)$ are bivariate polynomials with[38]

$$\gcd\{q_1(u,v), q_0(u,v)\} = \gcd\{q_2(u,v), q_0(u,v)\} = \gcd\{q_3(u,v), q_0(u,v)\} = 1.$$

If especially $q_0(u, v) \equiv 1$ we call Φ a *polynomial surface*.
$n := \max\{\deg q_0(u,v), \deg q_1(u,v), \deg q_2(u,v), \deg q_3(u,v)\}$ is called the *degree of the rational surface* Φ.

Rational surfaces belong to the family of algebraic surfaces. We have:

Theorem 3.10. *Any rational surface of degree n is an algebraic surface of order $\le n$.*

The proof to this statement is non-trivial. We note that the converse is not true: An algebraic surface need not be rational.

A polynomial surface is a special case of a rational surface. Rational and particularly polynomial surfaces are frequently used in CAD applications. Bézier surfaces (Sect. 3.8.1) are examples of polynomial surfaces, B-Spline surfaces (Sect. 3.8.2) are piecewise polynomial, the components of NURBS surfaces (Sect. 3.8.3) are piecewise rational surfaces.

3.7.8 Quadrics

Algebraic surfaces of order 2 are called *quadrics*. They have polynomial equations of degree 2. Their general form reads as

$$\begin{aligned} &Ax^2 + By^2 + Cz^2 + 2Dxy + 2Exz + 2Fyz + 2Gx + 2Hy + 2Iz + J \\ &= [1, x, y, z] \cdot \begin{bmatrix} J & G & H & I \\ G & A & D & E \\ H & D & B & F \\ I & E & F & C \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = 0 \end{aligned} \tag{3.200}$$

with coefficients $A, \ldots, J \in \mathbb{R}$. Of course we demand that not all coefficients $A, \ldots, F$ be zero.

[38] gcd denotes the *greatest common divisor* of polynomials; see Definition 3.12, p. 69.

Classification of Quadrics

It is a standard exercise in linear algebra to show that—by some appropriate change of the coordinate system—one of the following normal forms of the Eq. (3.200) can be achieved ($a, b, c, p \in \mathbb{R}^+$):

(a) $x^2 = 0$
This is a plane which is doubly counted (*double plane*).

(b) $a^2x^2 + b^2y^2 = 0$
This equation can be rewritten as $(ax + iby) \cdot (ax - iby) = 0$ with $i^2 = -1$. Thus it is a *conjugate complex pair of planes*. The intersecting line of these planes is real (here: the z-axis). All other points of this quadric are non-real.

(c) $a^2x^2 - b^2y^2 = 0$
This leads to $(ax + by) \cdot (ax - by) = 0$ and the surface consists of *two intersecting real planes*.

(d) $x^2 + a^2 = 0$
This amounts to $(x + ia) \cdot (x - ia) = 0$, and we have a *pair of conjugate complex parallel planes*.

(e) $x^2 - a^2 = 0$
yields $(x + a) \cdot (x - a) = 0$ and we have a *pair of real parallel planes*.

(f) $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
turns out to be an *elliptic cylinder* (Fig. 3.78, left).

(g) $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$
is a *cylinder with a hyperbola as its directing curve* (Fig. 3.78, center).

(h) $y^2 = 2px$

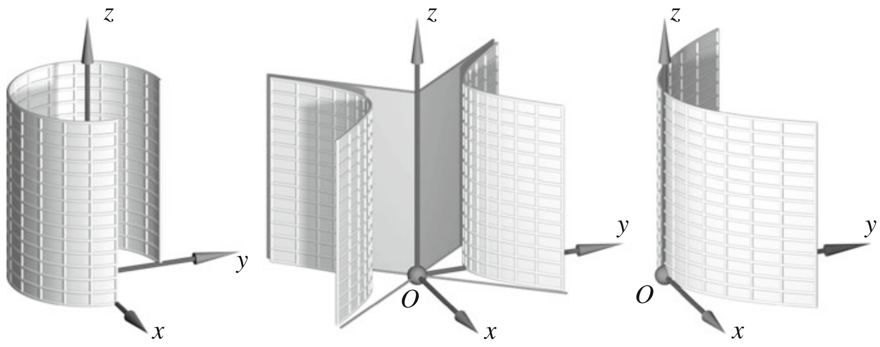


Fig. 3.78 Parallel *straight line* generators characterize a *cylinder*. Planar cross sections of a cylinder (with planes not parallel to the generators) suggest their classification. The three types of quadratic cylinders are distinguished by quadratic curves as their cross sections an ellipse, a hyperbola or a parabola. *Left* elliptic cylinder; here the special case of a right cylinder; the orthogonal cross-section is a *circle*. *Center* hyperbolic cylinder. *Right* parabolic cylinder

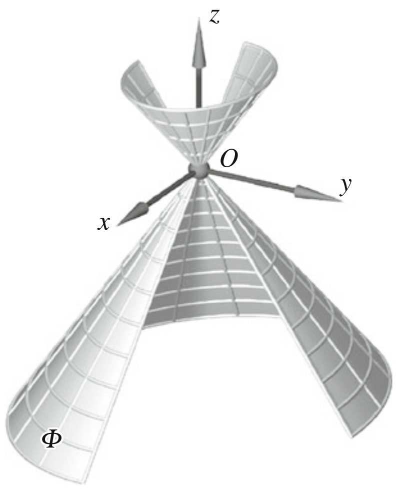


Fig. 3.79 This quadratic cone Φ (case (k)) has its singular point (vertex) in the origin O. The cone consists of *straight line* generators through O. Every planar section $\varepsilon \cap \Phi = k$ where ε does not go through O is a regular curve of 2nd order. The represented parameter *lines* on this example are such planar sections (ellipses) and, on the other hand, *straight line* generators through O

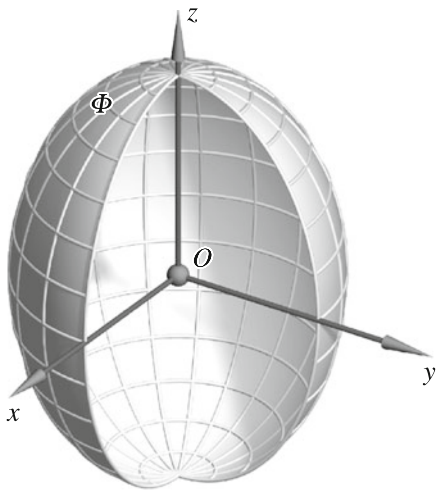


Fig. 3.80 The ellipsoid Φ (case (l)) is one model of a quadric. As a very special case $a = b = c$ we could also obtain the sphere. The example of this figure is an ellipsoid of revolution we have $a = b$; the axis of revolution is the z-axis. The parameter *lines* for the chosen parameterization are shown: circles in parallel planes, orthogonal to the axis of revolution, and meridian ellipses

We have a *parabolic cylinder* (Fig. 3.78, right).

(i) $\frac{x^2}{a^2} + \frac{y^2}{b^2} = -1$
This is a cylindric surface (*virtual cylinder*) which happens to have no real point at all.

(j) $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0$
represents a *real quadratic cone* with its apex in the origin (Fig. 3.79).

(k) $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 0$
This is a *virtual quadratic cone* with the apex in the origin. The apex is its only real point.

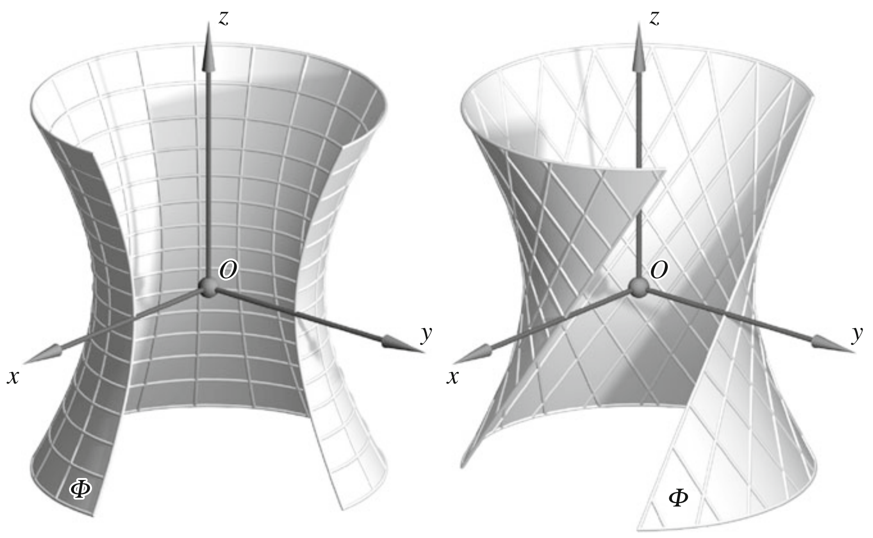


Fig. 3.81 An example of a 1-sheet hyperboloid (case (m)). Here we have $a = b$, a 1-sheet hyperboloid of revolution. Apart from the hyperbolic paraboloid (Fig. 3.84) this is the only regular quadratic surface to carry two families of real *straight line* generators. *Left* the displayed parameter *lines* refer to the generation of the surface as a surface of revolution. The parameter *lines* are the *circles* latitude and the meridian curves (hyperbolae). *Right* the same surface, parameterized according to its *straight lines* generators. The u-lines are one family of *straight lines* on Φ, the v-lines are the other. Two generators of the same family are skew, two generators of different families intersect in a point on the surface. As the formwork for such surfaces is particularly simple (*straight beams*) these surfaces are also used in civil engineering, e.g., for cooling towers of power plants

(l) $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$
is an *ellipsoid* with its center in the origin O (Fig. 3.80).

(m) $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$
This surface is called a *one-sheet hyperboloid* (Fig. 3.81).

(n) $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = -1$
yields a *2-sheet hyperboloid* (Fig. 3.82).

(o) $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = -1$
This is a *virtual ellipsoid*; it has no real point.

(p) $\frac{x^2}{a^2} + \frac{y^2}{b^2} = z$
This surface is called *elliptic paraboloid* (Fig. 3.83).

(q) $\frac{x^2}{a^2} - \frac{y^2}{b^2} = z$
We finally have a *hyperbolic paraboloid*. It is sometimes called a *saddle surface* (Fig. 3.84; compare also Example 3.23 and Fig. 3.115a, pp. 204 and 205).

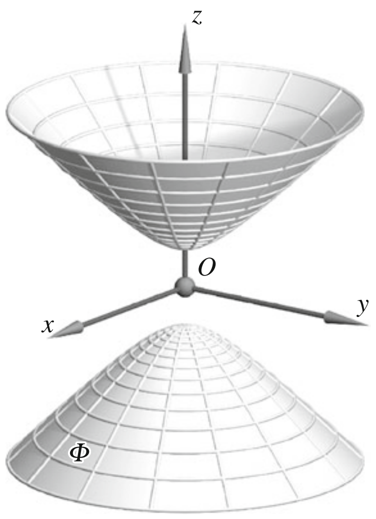


Fig. 3.82 The 2-sheet hyperboloid (case (n)) in a special form; here we have $a = b$: a surface of revolution with a hyperbola as its meridian curve. The main axis of the hyperbola acts as the axis of revolution. The 2-sheet hyperboloid has no real *straight line* generators. According to its name it consists of two separate sheets. The general type of a 2-sheet hyperboloid $a \neq b$ can be created from this surface of revolution by some contraction in y-direction

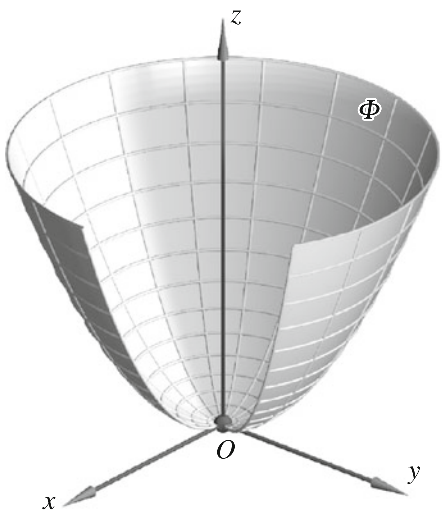


Fig. 3.83 An example of an elliptic paraboloid (case (p)): here we have again selected the special case $a = b$ which leads to a surface of revolution. The meridian curves are parabolae, the *circles* of latitude lie in parallel planes orthogonal to the axis of revolution (z-axis)

The types (a)–(k) are also referred to as *singular quadrics*. The types (a)–(e) among the singular quadrics are called *reducible*. The types (l)–(q) are *regular quadrics*. The normal forms have planes of symmetry in the three coordinate planes, except for the types (p) and (q) which are only symmetric with respect to the xz- and the yz-plane. Admittedly, the types (a)–(e) and the virtual surfaces (k) and (o) are of little relevance for practical engineering. We have just listed them for the sake of completeness.

It can easily be shown that the types (a), (f), (g), (h), (j), (l), (m), (n), (p) and (q) are rational (see Sect. 3.7.7, pp. 159): they can also be represented by rational parameterizations.

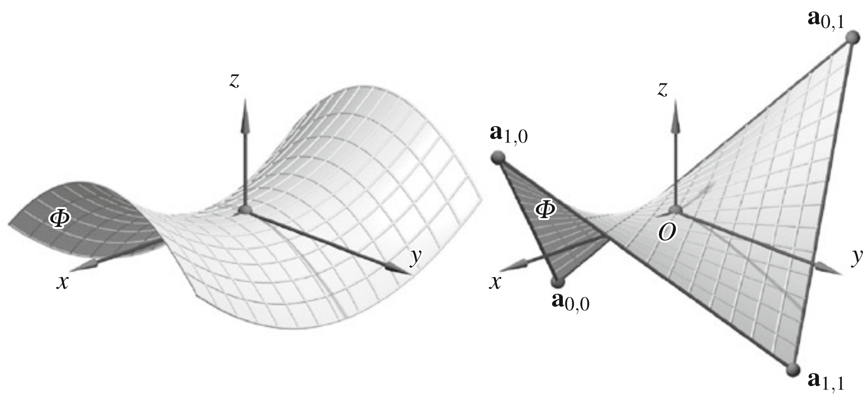


Fig. 3.84 The hyperbolic paraboloid (case (q)) is the classical saddle surface. Apart from the 1-sheet hyperboloid—see above—this is the only surface to carry two families of *straight lines*. In this figure the axis of Φ is the z-axis. The same hyperbolic paraboloid is parameterized in two different ways. *Left* the parameter *lines* are vertical parabola cross-sections. *Right* the two families of *straight line* generators are the parameter lines. This type of surfaces can easily be used in civil engineering for curved roofs, especially if a skew quadrangle has to be covered. The straight beams of the two families make it easy to construct some appropriate scaffold or formwork

It is remarkable that the quadratic surfaces (m) and (q) carry two mutually skew families of straight lines. Hence they can be viewed as *doubly ruled surfaces*.[39] In fact, there are no doubly ruled surfaces apart from these two types. They are particularly interesting for construction engineers because of their statics and their simple scaffolding.

The hyperbolic paraboloid (type (q)), in particular, can be generated by the following simple idea: A skew quadrilateral $\mathbf{a}_{0,0}$, $\mathbf{a}_{0,1}$, $\mathbf{a}_{1,0}$, $\mathbf{a}_{1,1}$ (see also Fig. 3.84, right) can be *filled* by a family of straight lines, each of them connecting two points on the opposite sides $\mathbf{a}_{0,0}\mathbf{a}_{0,1}$ and $\mathbf{a}_{1,0}\mathbf{a}_{1,1}$ belonging to the same ratio. Of course, this can also be performed for the other pair $\mathbf{a}_{0,0}\mathbf{a}_{1,0}$ and $\mathbf{a}_{0,1}\mathbf{a}_{1,1}$ of opposite sides delivering another family of straight lines. It is pretty easy to check that these two families of straight lines lie on the same surface sheet Φ: They are the two sets of generators on one hyperbolic paraboloid Φ. This amazing example will also be referred to in Example 3.23, p. 204.

3.7.9 Ruled Surfaces

Surfaces which are built up by a set of straight lines are well worth mentioning. Not only can they be formed by a formwork consisting of straight planks (construction engineering), they also show up in a lot of mechanical engineering applications.

[39] For the notion of *ruled surfaces* see Sect. 3.7.9.

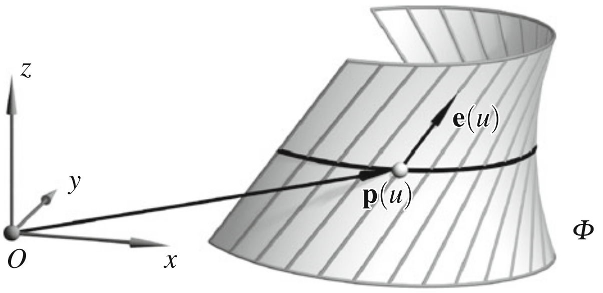


Fig. 3.85 A ruled surface consists of *straight lines* (generators). Each of the generators is defined by a point $\mathbf{p}(u)$ and a direction vector $\mathbf{e}(u)$

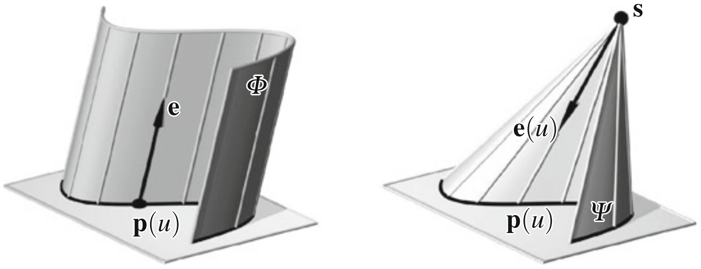


Fig. 3.86 Cylinders and cones are particular types of ruled surfaces. *Left* a cylinder Φ is defined by its directing curve $\mathbf{p}(u)$ and the constant direction vector $\mathbf{e}$ of its parallel *straight line* generators. *Right* the *straight line* generators of a cone go through a vertex $\mathbf{s}$

Definition 3.51. Ruled surface. A surface Φ with a parameterization of the form

$$\mathbf{q}(u,v) = \mathbf{p}(u) + v \cdot \mathbf{e}(u) \quad \text{with} \quad \mathbf{e}(u) \neq [0,0,0]^\top \tag{3.201}$$

is called a *ruled surface* (Fig. 3.85). It consists of a one-parameter set of straight lines, called the *generators of* Φ, whose directions are given by the vectors $\mathbf{e}(u)$. These lines are the v-lines of the parametric representation (3.201). The u-line $\mathbf{p}(u) = \mathbf{q}(u,0)$ is also called the *directing curve* of the parametric representation (3.201).

Two very specific types of ruled surfaces are defined in:

Definition 3.52. Cylinders and cones.

(a) A ruled surface (3.201) where $\mathbf{e}(u) \equiv \mathbf{e}$ is a constant vector is called a *cylindric surface* or a *cylinder*. Its generators are parallel to $\mathbf{e}$ (Fig. 3.86, left). A cylinder is defined by $\mathbf{e}$ and its *generator curve*.

(b) A ruled surface Ψ where all generators run through a given point $\mathbf{s}$ is called a *cone with vertex* $\mathbf{s}$ (Fig. 3.86, right). It is defined by $\mathbf{s}$ and a directing curve $\mathbf{p}(u)$. As $\mathbf{e}(u) = \mathbf{p}(u) - \mathbf{s}$ is a direction vector of the respective generator, the cone can be parameterized by:

$$\mathbf{q}(u, v) = \mathbf{s} + v \cdot (\mathbf{p}(u) - \mathbf{s}) \tag{3.202}$$

3.7.10 Developable Surfaces

In mechanical engineering some surfaces have to be produced from sheet metal without deep-drawing, merely by bending the sheet.

Definition 3.53. Developable surface. A surface is called *developable* if it can be transformed into a plane sheet without stretching. Mathematically this can be expressed by the condition that there exists an isometry—i.e., a mapping that preserves the arc length of any surface curve—from the surface into a plane.

A developable surface is shortly termed a *developable*. We consider a classical result on developables (see [5], pp. 115).

Theorem 3.11. Characterization of developables. *A surface Φ is developable if the following two conditions hold:*

1. *Φ is a ruled surface.*
2. *For each straight line generator e there is only one plane tangent all along e.*

The second condition in this characterization is in general not fulfilled by an arbitrary ruled surface. Which ruled surfaces comply? The answer is given by

Theorem 3.12. Types of developables. *There are 3 types of developable surfaces:*

- *cylindric surfaces*
- *cones*
- *tangent surfaces to spatial curves*

The first and the second type have already been mentioned in Definition 3.52, p. 166. The third is somewhat the generic type of a developable. Such a developable surface is built up by the tangents to a given space curve $c \ldots \mathbf{p} = \mathbf{p}(u)$ (see Fig. 3.87). In this case the direction vector $\mathbf{e}(u)$ of the respective generator (Definition 3.51, p. 78) has to be specified by $\mathbf{e}(u) = \dot{\mathbf{p}}(u) = \frac{dp}{du}$.

A tangent surface Φ to a spatial curve c can be represented by

$$\mathbf{q}(u, v) = \mathbf{p}(u) + v \cdot \dot{\mathbf{p}}(u) \tag{3.203}$$

where the spatial curve $c \ldots \mathbf{p}(u)$ is called the *edge of regression*. Moreover, by using (3.203) one can easily check that the osculating planes of c (see Sect. 3.3.6, p. 78) are the tangent planes to Φ. The following remark is of utmost practicality (Fig. 3.87):

Remark 3.17 **Recognizing a developable.** Developables can visually be recognized:

- A developable has a straight line contour with respect to any (central or parallel) projection.

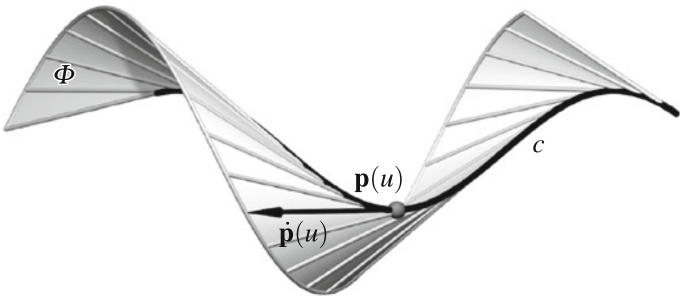


Fig. 3.87 The tangents of a space curve $c \ldots \mathbf{p}(u)$ represent a ruled surface which is developable. In fact every developable which is not a cylinder or a cone can be generated as the tangent surface to some space curve

- A ruled surface with straight contours with respect to every parallel (central) projection is developable (compare Fig. 3.88b,d).

An exception of this rule only occurs if one projects a cylinder parallel to its generators or a cone via a central projection from its vertex—in those cases the cylinder or cone appears as a curved line!

Example 3.17 **A developable connecting two planar curves.** Construct a developable surface Φ through a pair of plane curves $\mathbf{c}_1 \subset \varepsilon_1$ and $\mathbf{c}_2 \subset \varepsilon_2$ (Fig. 3.88a): Imagine a piece of the air duct in an engine, a connecting tube of two convex holes. The core part of the task is to find the generators of the ruled surface Φ such that Φ turns out to be developable. A point $\mathbf{c}_1(u_1)$ on $\mathbf{c}_1$ and a point $\mathbf{c}_2(u_2)$ on $\mathbf{c}_2$ are connected by a generator of Φ if the tangents t_1, t_2 of the two points are coplanar[40]:

$$\det\left[\mathbf{c}_2(u_2) - \mathbf{c}_1(u_1), \frac{d\mathbf{c}_1}{du_1}(u_1), \frac{d\mathbf{c}_2}{du_2}(u_2)\right] = 0. \tag{3.204}$$

This way, for any given parameter u_1 the corresponding parameter u_2 has to be a zero of Eq. (3.204).

In general the outcome will not be a cylinder or a cone, so by rule of exclusion (Theorem 3.12) we can say that it will be the tangent surface (see above) to some spatial curve c.

Remark 3.18 The recipe described in the above example also works for space curves c_1, c_2 instead of planar curves. The zeroes of condition (3.204) can still be evaluated numerically. Again, we arrive at corresponding pairs of parameters (u_1, u_2) and thus pairs of points which are to be connected by straight line generators.

[40] It can be proved: If the tangent planes of two distinct points on a generator of a ruled surface are identical, any other point on this generator will also have this tangent plane.

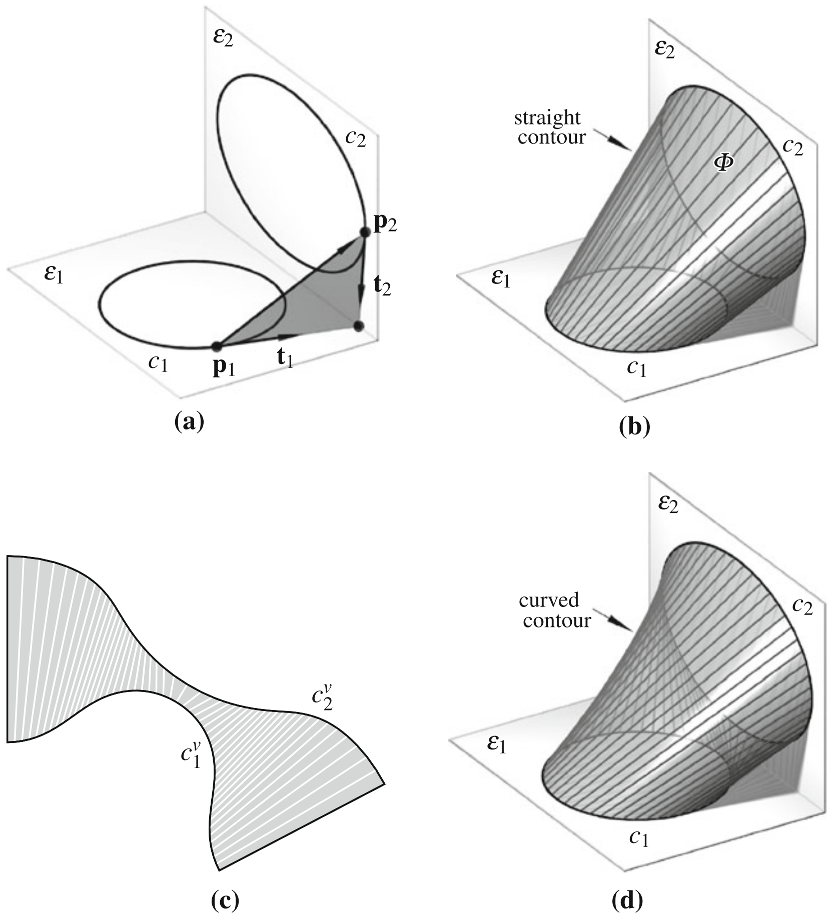


Fig. 3.88 Joining two closed planar curves by a developable surface. Whenever a part has to be produced from a planar sheet by mere bending these geometrical considerations may be valuable. The shape of the constructed net has to be cut out from the planar sheet. **a** Two planar *curves* c_1 and c_2 are to be connected via a developable ruled surface. Each *straight line* generator has to connect points $\mathbf{p}_1 \in c_1$ and $\mathbf{p}_2 \in c_2$ with coplanar tangents t_1 and t_2: the tangents t_1 and t_2 meet the *line* $s = \varepsilon_1 \cap \varepsilon_2$ in a common point. **b** A developable surface connecting c_1 and c_2. In general, this surface will be the tangent surface Φ to some space curve c. Mind that the contour of Φ consists of *straight lines* (Remark 3.17). In order to *develop* Φ we can approximate it by a finite number of *straight line* generators. **c** The segment determined by two neighboring generators is a (albeit slim) skew quadrangle. It can be divided into two triangles by one of its diagonals. Adding such elements in the plane generates an approximating planar net of the surface. **d** If the two planar curves c_1 and c_2 are connected arbitrarily via *straight lines* the emerging ruled surface will, in general, not be developable. In accordance with Remark 3.17 the contour of the non-developable ruled surface does not consist of *straight lines*

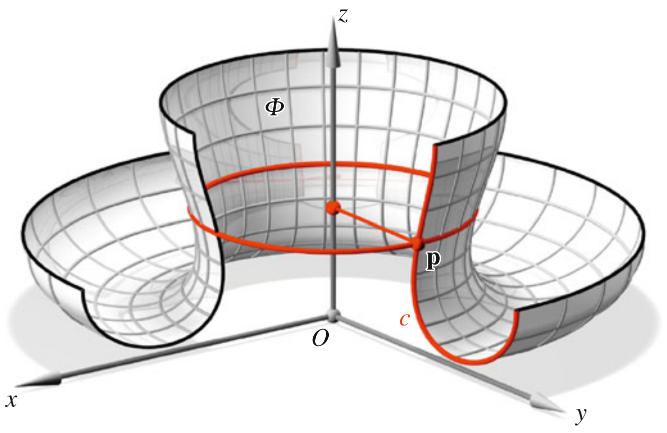


Fig. 3.89 A profile curve c is subjected to a revolution about the axis z. Every point $\mathbf{p} \in c$ runs on a circular trajectory. All such *circles of latitude* lie in parallel planes orthogonal to the axis of revolution

3.7.11 Surfaces of Revolution

Subjecting a curve c to a spatial motion creates a surface. Such surfaces are generally called *surfaces of motion*. The particular type of surface created this way depends on the type of motion and on the generating curve c.

Definition 3.54. Surface of revolution. (Figure 3.89) A continuous revolution about the z-axis as in (3.9), p. 170, is applied to a curve

$$c \ldots \mathbf{p}(v) = [x(v), y(v), z(v)]^\top .$$

The resulting surface Φ has the parametric representation

$$\mathbf{q}(u,v) = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{p}(v) = \begin{bmatrix} \cos u \cdot x(v) - \sin u \cdot y(v) \\ \sin u \cdot x(v) + \cos u \cdot y(v) \\ z(v) \end{bmatrix} \quad (3.205)$$

and is called *surface of revolution*[41] with the *generating curve* (profile curve) c.

Every plane through the axis of revolution a (which—in our case—is the z-axis) is called a *meridian plane* of Φ. Its intersection with Φ is dubbed a *meridian curve* of Φ. All meridian curves of Φ are congruent curves. If we replace the generating curve c by any meridian curve we certainly arrive at the very same surface of revolution Φ albeit with another parameterization. In Fig. 3.89 the curve c is a meridian curve as are the further parameter lines of that type. Some of them are shown in the figure.

[41] This is the kind of surface which can be machined on a lathe.

While being rotated about the axis a every point $\mathbf{p}$ of c is moved on a circular path. The circles all have the same axis a; they are the u-lines in the parameterization (3.205) with constant $v = v_0$. We also call them *circles of latitude* on Φ. The v-lines $u = u_0$—on the other hand—are the generating curve ($c \ldots u_0 = 0$) and its instances while c is being rotated about a. If the generating curve c happens to be a meridian curve (e.g., in the plane $x = 0$) the v-lines coincide with the meridians.

Remark 3.19 **Quadrics of revolution.** The quadrics (see Sect. 3.7.8, pp. 160) include several examples of surfaces of revolution: If we put $a = b$ in the cases (f), (j), (l), (m), (n), (p) we obtain a cylinder of revolution (*right cylinder*), a cone of revolution (*right cone*), an ellipsoid of revolution, a 1-sheet hyperboloid of revolution, a 2-sheet hyperboloid of revolution and a paraboloid of revolution, respectively. In each of these normal forms the axis is the z-axis.

Example 3.18 **A straight line generating a surface of revolution.** To give an example we revolve a simple straight line e about the z-axis. Let e be parameterized as $\mathbf{p}(v) = \begin{bmatrix} \alpha \\ v \cdot \beta \\ v \cdot \gamma \end{bmatrix}$ with $\gamma \neq 0$. Inserted into (3.205) this yields

$$\mathbf{q}(u, v) = \begin{bmatrix} \alpha \cos u - \beta v \sin u \\ \alpha \sin u + \beta v \cos u \\ \gamma v \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.206}$$

Eliminating u and v from (3.206) we arrive at one of the equations

$$\frac{x^2 + y^2}{\alpha^2} - \frac{\beta^2 z^2}{\alpha^2 \gamma^2} = 1 \text{ if } \alpha, \beta \neq 0, \tag{3.207}$$

$$\frac{x^2 + y^2}{\beta^2} - \frac{z^2}{\gamma^2} = 0 \text{ if } \alpha = 0, \beta \neq 0, \tag{3.208}$$

$$x^2 + y^2 = \alpha^2 \text{ if } \alpha \neq 0, \beta = 0. \tag{3.209}$$

By comparison with Remark 3.19 one can easily check that $\left\{ \begin{matrix} (207) \\ (208) \\ (209) \end{matrix} \right\}$ represents a $\left\{ \begin{matrix} 1-\text{sheet hyperboloid} \\ \text{cone} \\ \text{cylinder} \end{matrix} \right\}$ of revolution by putting $\left\{ \begin{matrix} a = \alpha, c = \frac{\alpha\gamma}{\beta} \\ a = \alpha, c = \gamma \\ a = \alpha \end{matrix} \right\}$. The meridian curve of the surface (3.207) is a hyperbola whose minor axis coincides with the z-axis. This accounts for its name: 1-sheet hyperboloid. Figure 3.81 shows this type of surface. The figure on the right shows the generation by a straight line subjected to a revolution. The left figure shows that the same surface can as well be created by revolving a meridian (hyperbola).

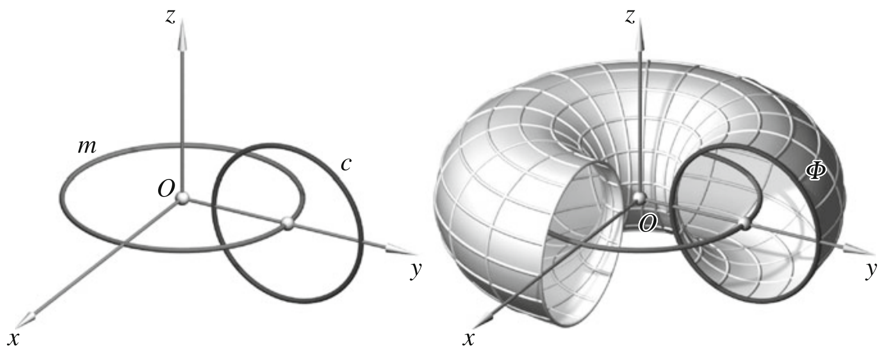


Fig. 3.90 The torus, generated as a surface Φ of revolution. The axis of revolution is the z-axis. The profile c is a *circle* in the yz-plane (meridian plane). The resulting surface is algebraic of order 4 (cf. Remark 3.20)

Example 3.19 **Torus.** (Figure 3.90) We give another example which is equally classical. Let c be the circle

$$\mathbf{p}(v) = \begin{bmatrix} 0 \\ a + b \cdot \cos v \\ b \cdot \sin v \end{bmatrix} \tag{3.210}$$

in the yz-plane. Its center is $(0, a, 0)$ and its radius is b. We apply a revolution about the z-axis and obtain a surface of revolution Φ.
Substitution of (3.210) into (3.205) delivers the parameterization of Φ:

$$\mathbf{q}(u, v) \quad = \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin u \cdot (a + b \cdot \cos v) \\ \cos u \cdot (a + b \cdot \cos v) \\ b \cdot \sin v \end{bmatrix} \tag{3.211}$$

This surface is called *torus*. Eliminating the parameters u and v from (3.211) we obtain the equation

$$(x^2 + y^2 + z^2 + a^2 - b^2)^2 = 4 \cdot a^2 \cdot (x^2 + y^2). \tag{3.212}$$

As this is a polynomial equation of degree 4 we conclude: The torus is an algebraic surface of order 4. A 2nd order meridian curve provided a surface of revolution of order 4. This observation adheres to a general rule:

Remark 3.20 **Algebraic surfaces of revolution.** If a planar algebraic curve c of order n is used as the meridian curve we obtain a surface of revolution Φ which is

- algebraic of degree $2n$ if c is not symmetric with respect to the axis of revolution.
- algebraic of degree n if c is symmetric with respect to the axis of revolution.

As examples for the second case we mention the hyperboloid of revolution, the ellipsoid of revolution and the paraboloid of revolution. Examples for the first case are—apart from the torus and many others—the right cone and the right cylinder.

3.7.12 Helical Surfaces

We proceed creating surfaces by applying some type of spatial motion to a given profile curve c. We now consider a screw motion about the z-axis, with a pitch $h = 2\pi p$ as in (3.10), p. 61.

Definition 3.55. Helical surface. A curve c

$$c \dots \mathbf{p}(v) = \begin{bmatrix} x(v) \\ y(v) \\ z(v) \end{bmatrix}$$

subjected to a helical motion (3.10) delivers a surface Φ with the parameterization

$$\mathbf{q}(u, v) = \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{p}(v) \quad = \begin{bmatrix} \cos u \cdot x(v) - \sin u \cdot y(v) \\ \sin u \cdot x(v) + \cos u \cdot y(v) \\ z(v) + p \cdot u \end{bmatrix}. \quad (3.213)$$

Φ is called a *helical surface* with *generating curve* c.

Every plane through the screw axis a is called a *meridian plane* of Φ. Its intersection with Φ is a *meridian* of the surface. All meridians of Φ are congruent curves. Replacing the generating curve c by a meridian results in the same helical surface Φ.

Helical surfaces are not algebraic. The only exception is trivial: a right cylinder with axis a. The assumption that a helical surface Φ (other than a right cylinder) with the screw axis a and a profile curve c be algebraic, can easily be proven wrong: A line parallel to the axis a through any point $P \in c$ will deliver an infinite number of intersection points with Φ. If Φ were algebraic of any order n the number of intersection points with a straight line would be no more than n (Theorem 3.9, p. 159).

A surface of revolution can be viewed as a special case of a helical surface whose pitch has shrunk to zero. We can equally view any cylindric surface as a special case of a helical surface where the rotational component is suppressed.

Definition 3.56. Helical ruled surface, circular helical surface. If the profile curve c of the helical surface Φ is a straight line we call Φ a *helical ruled surface*. If the profile curve c is a circle the generated helical surface Φ is called a *circular helical surface*.

Figure 3.91 shows a sample of helical ruled surfaces: If the generating straight line c is skew to the screw axis a and not perpendicular to a the surface is called *generic helical ruled surface*. If the profile curve c happens to be a screw tangent (Fig. 3.91, center) the surface is developable (compare with Sect. 3.7.10). It is the

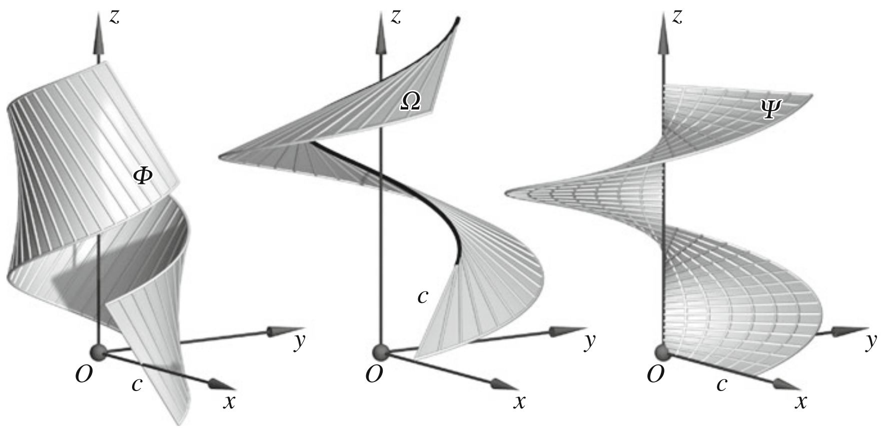


Fig. 3.91 A helical ruled surface is generated by a *straight line* c which is subjected to a helical motion. *Left* the helical ruled surface Φ is determined by a *straight line* generator c which is skew to the screw axis (z-axis). *Center* basically the helical ruled surface Ω is of the same type as Φ (*left*). However, in the case of Ω the generator c happens to be the tangent of a trajectory (*screw line* of this very *screw motion*). This is why we arrive at a developable surface which—by the way—can be recognized by its *straight line* contour (cf. Remark 3.17, p. 167). *Right* the *straight line* generator c intersects the screw axis orthogonally. This surface Ψ is called *right helicoid*

tangent surface to a helix. Apart from the trivial case of right cylinders such tangent surfaces of helices are the only developable helical surfaces.

One interesting application of developable helical surfaces are *helical gears*: These surfaces form the tooth flanks of such cogs. At any moment the two flanks of engaging cogs have line contact along a generating straight line c. Their main advantage is their low noise action.

Figure 3.92 shows a special circular helical surface: If the screw tangent of the profile circle's center is orthogonal to the plane of the circle the emerging surface is a *tubular helical surface*. It can also be generated subjecting a sphere Ω to a screw motion, thus generating a one-parametric family $\Omega(u)$ of spheres. The envelope of this family $\Omega(u)$ is the tubular helical surface. Such surfaces occur as springs in mechanical devices.

Helical surfaces do not only show up in the threads of nuts and bolts, they even play an important role in the design of cars (cf. Sect. 3.13, pp. 235).

3.7.13 *Moving a Curve or a Surface in Itself*

We now define a concept which links spatial motions and surfaces (or curves):

Definition 3.57. Moving a curve or a surface in itself. A curve c (or a surface Φ) is said to be *moved in itself* if every point of c (or Φ) stays on that curve (surface)

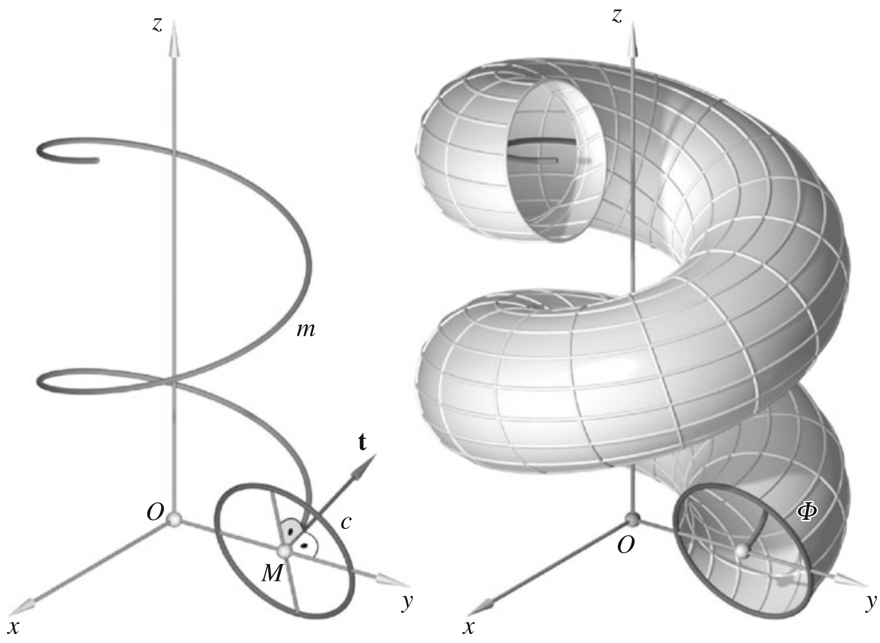


Fig. 3.92 The tubular helical surface is a special case of a circular helical surface. A *circle c* is subjected to a helical motion. The particular thing is that the screw tangent t of the *circle* center is orthogonal to the circle's plane

throughout the motion. As an example we can think of a surface of revolution Φ which is revolved about its own axis, or a helical surface Φ which is subjected to the very screw motion which created it. Overall, the surface stays the same whilst its points are being moved on the sheet.

The following theorem is well-known in the field of mathematics. It is, however, not generally recognized in automotive design.

Theorem 3.13. Classification of curves and surfaces which are moveable in themselves.

1. *The only curves movable in themselves are the straight lines, the circles and the helices. The corresponding motions are appropriate translations, rotations and screw motions.*
2. *The helical surfaces (together with their special cases of cylinders and surfaces of revolution) are the only surfaces which can be moved in themselves. The corresponding motions are appropriate translations, rotations and screw motions.*

This fundamental geometric result can be more than useful for practical applications (see [3, 19–21]). As for automotive design it can be used for the construction of retractable surfaces such as car side windows (see Sect. 3.13, pp. 235 of this book and [22, 23]).

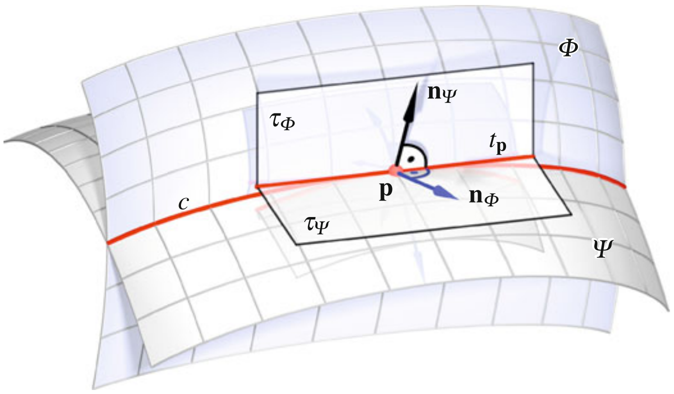


Fig. 3.93 A point $\mathbf{p}$ on the intersection curve $c = \Phi \cap \Psi$ lies on both tangent planes τ_Φ and τ_Ψ. If these planes are different they have a unique intersection $t_\mathbf{p}$ which is the tangent to c in P

3.7.14 Intersection of Surfaces

We now consider *two* surfaces Φ and Ψ. Their intersection $c = \Phi \cap \Psi$ is, in general, a one-parametric set of points, i.e., a space curve or—in some cases—a set of planar curves. We call the outcome *intersection curve c of the surfaces Φ and Ψ*.

In a general point on the intersection curve c we can easily construct the tangent to c. As the curve c lies on both surfaces Φ and Ψ, its tangent has to be contained in both tangent planes (cf. Fig. 3.73, p. 148). If the two tangent planes in the regarded point $\mathbf{p}$ do not coincide we have:

Proposition 3.16 Tangent to the intersection curve. *Let $\mathbf{p} \in c$ be a point on the intersection curve c of the two given surfaces Φ and Ψ (Fig. 3.93). Let moreover τ_Φ and τ_Ψ be the tangent planes of Φ and Ψ in the point* $\mathbf{p}$. *Then we have:*

If $\tau_\Phi \neq \tau_\Psi$ then the tangent $t_\mathbf{p}$ to c at $\mathbf{p}$ *exists and is the intersection of the two tangent planes τ_Φ and τ_Ψ in the common point* $\mathbf{p}$*:*

$$t_\mathbf{p} = \tau_\Phi \cap \tau_\Psi$$

Moreover, as $\tau_\Phi \neq \tau_\Psi$, the normal vectors $\mathbf{n}_\Phi$ *and* $\mathbf{n}_\Psi$ *of Φ and Ψ at* $\mathbf{p}$ *are linearly independent and their cross product*

$$\mathbf{t}_\mathbf{p} = \mathbf{n}_\Phi \times \mathbf{n}_\Psi$$

is a direction vector of $t_\mathbf{p}$.

We add a few examples.

Example 3.20 **Two quadrics of revolution.** Let us regard two quadrics of revolution Φ and Ψ, e.g., two right cylinders as in Figs. 3.94, 3.95, 3.96, and 3.97 or a right

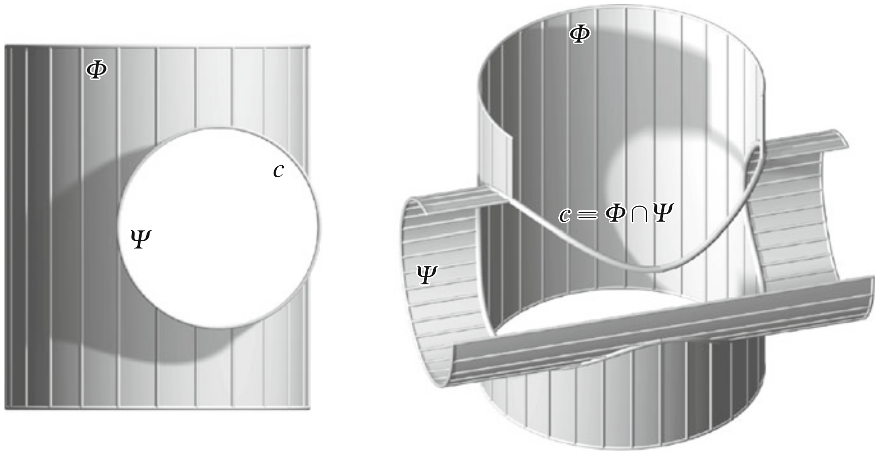


Fig. 3.94 Two *right* cylinders Φ and Ψ. The surfaces are displayed with some of their *straight line* generators. The intersection curve $c = \Phi \cap \Psi$ is a spatial curve. In this particular case c consists of one piece

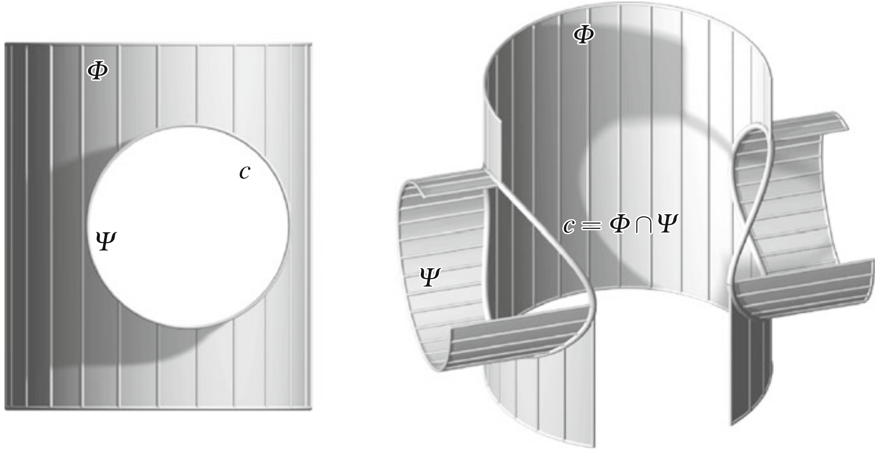


Fig. 3.95 Two *right* cylinders Φ and Ψ. The intersection curve $c = \Phi \cap \Psi$ is a spatial curve consisting of two pieces (two connected components)

cylinder and a right cone as in Fig. 3.98. We can think of two intersecting pipes. Of course, there are a lot of options for the shape of the outcome. Figure 3.94 shows the case where the intersection curve is a spatial curve consisting of one connected component. In Fig. 3.95 there occur two connected components within the space curve c.

If the two surfaces are tangent in a point D (Fig. 3.96), this point is in general a so-called *double point* of the intersection curve c. The tangent to the intersection curve is no more uniquely determined (see Proposition 3.16) as the tangent planes of

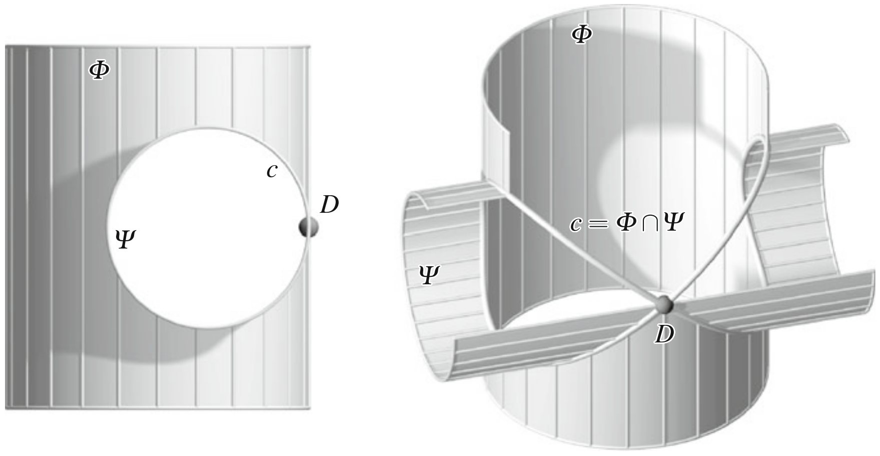


Fig. 3.96 The two *right* cylinders Φ and Ψ share a point D where the two surfaces have the same tangent plane. We say: *Φ and Ψ are tangent at D*. The intersection curve $c = \Phi \cap \Psi$ (again a spatial curve) has a double point D; the curve c goes through D twice

the two surfaces in that point coincide. In general, the curve goes through D twice. Figure 3.96 can be viewed as the border case between the preceding two Figs. 3.95 and 3.94.

The following example again refers to intersecting quadrics of revolution. However, it shows the case when the outcome consists of planar curves.

In practice a pipe junction is much easier to handle (cutting with an angle grinder or a circular saw) if the intersection curve c consists of planar components. When does that happen? The answer takes a little algebraic geometry (e.g., see [24]). We note the result (see Figs. 3.97 and 3.98):

Example 3.21 **A pipe junction with planar intersection curves.** If the axes of the two quadrics of revolution intersect in a point M *and* if there is a common sphere Ω with center M, inscribed tangentially into both surfaces, the intersection c of Φ and Ψ decomposes and consists of two second order curves (in the regarded cases: ellipses). For two right cylinders this means that they have intersecting axes and the same radius. This often happens in pipe knees, in pipe T-knees or pipe crossings. Figure 3.97 shows this case where additionally the two axes intersect orthogonally. Figure 3.98 shows the case of a right cylinder and a right cone. Here, the existence of a common tangent sphere is a welcome criterion for the intersecting curves to emerge as planar curves.

We mention one further useful example.

Example 3.22 **Every skew circular cone carries two families of circles.** We regard a circular cone Φ (not a right cone; such cones are often referred to as *skew circular cones*) with vertex S and a directing circle c lying in a plane ε. (Figure 3.99a). We

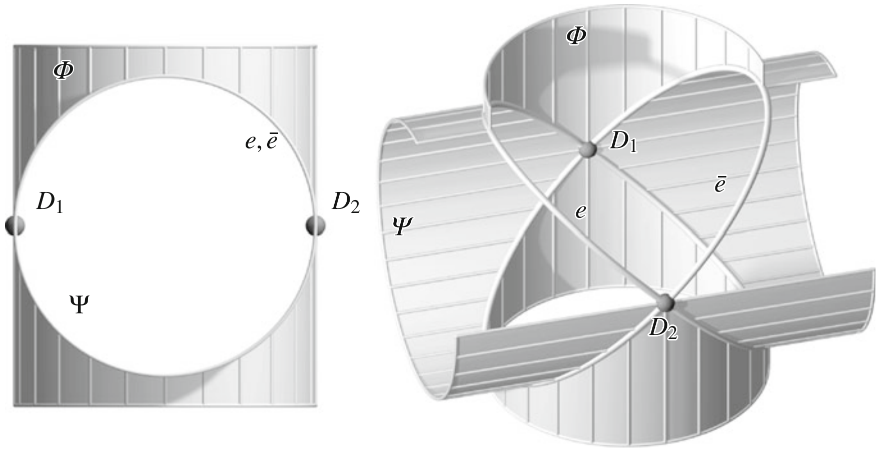


Fig. 3.97 The two *right* cylinders Φ and Ψ are tangent at D_1 and D_2. They have the same radius. Their axes intersect. In that case the intersection curve $c = \Phi \cap \Psi$ degenerates into two (*planar*) curves. It consists of two ellipses: $c = \Phi \cap \Psi = \{e, \bar{e}\}$

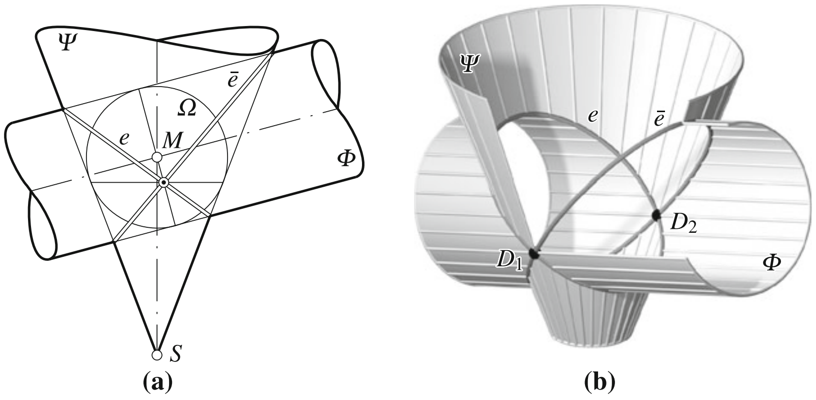


Fig. 3.98 Pipe junctions of cylindric and conical surfaces of revolution. It is sometimes recommendable to dimension the pipes such that this intersection decomposes into planar components. The criterion for this to happen is the existence of a sphere inscribed tangentially to both surfaces of revolution (*pipes*) of 2nd order. **a** A *right cylinder* Φ and a *right cone* Ψ. Each of them is tangent to the same sphere Ω along one of its *circles* of latitude. The intersection curve $c = \Phi \cap \Psi$ has two double points D_1 and D_2 (see *right* figure). It decomposes into planar *curves* (ellipses). **b** A pipe junction can easier be produced if the intersection curve consists of planar components. The existence of a sphere tangent to both pipes (*left* figure) provokes this phenomenon

consider a sphere Ω which also goes through c. Then the intersection curve $\Phi \cap \Omega$ also contains the circle c; the remainder of the intersection is one further circle $\bar{c}$. The respective proofs can be found in books on algebraic geometry (e.g., see [24]). It is interesting to see that the skew circular cone contains another circle $\bar{c}$ whose plane

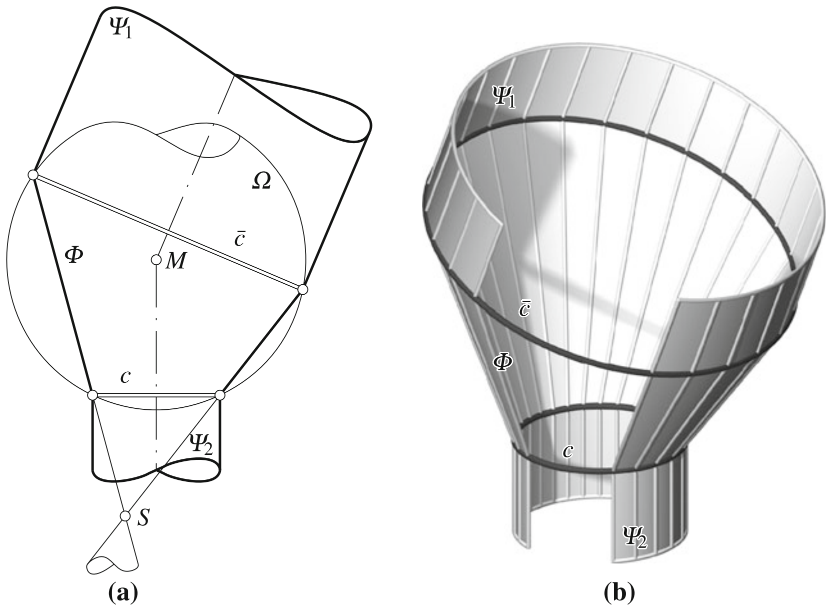


Fig. 3.99 A circular cone Φ—in general—carries two pencils of *circles* if the cone is different from a *right cone*. **a** A circle c on a circular cone Φ. The sphere Ω through c intersects the cone in a further *circle* $\bar{c}$. This way the cone can be used to join two *right* cylinders Φ_2 and Φ_1 with different radii. **b** Under certain preconditions (*left* figure) two *right* cylinders—albeit with different radii—can be joined by a circular cone such that the intersection curve at both ends is a *circle*

is not even parallel to the plane of c. The intersection of the cone and the sphere gets us the position of that circle $\bar{c}$.

As central dilations from the vertex S transform $\bar{c}$ into a pencil (family) of parallel circles on Φ we can summarize:

Proposition 3.17 Existence of two pencils of circles on a circular cone. *On every circular cone (given by a directing circle c and a vertex S) which is not a right cone there exists—apart from the pencil of circles containing c—a second pencil of circles. Intersecting the cone with a sphere through c delivers—apart from c—one circle $\bar{c}$ of the second family of circles.*

3.8 Tensor Product Surfaces

In Sect. 3.4, p. 85, we have defined freeform curves. Such a curve is ruled by a *control polygon*: The curve's shape can be modified by repositioning the vertices of its control polygon. Similar ideas can be used to generate surfaces. In this case the control polygon has to be replaced by a control net:

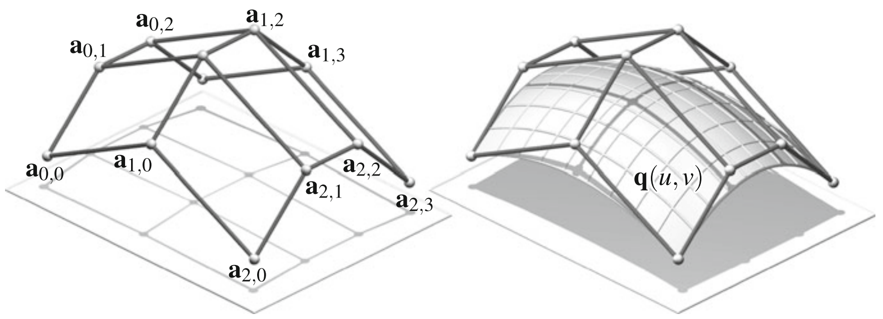


Fig. 3.100 Once the families of functions $F_i(u)$, $i = 0, \ldots, m$ and $G_j(v)$, $j = 0, \ldots, n$ are preselected, the control net $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ determines a tensor product surface. Here, we have $m = 2$ and $n = 3$. *Left* the control net as a *matrix* of points $\mathbf{a}_{i,j}$ in 3-space. The connecting *lines* are also called the u-threads (constant index j) and the v-threads (constant index i). *Right* the surface $\mathbf{q}(u, v)$ as defined in (3.214) or likewise (3.215), together with its defining control net

Definition 3.58. Tensor product surface. Let $(m + 1) \times (n + 1)$ points $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ in 3-space be given (Fig. 3.100). Moreover, let $F_i(u)$, $i = 0, \ldots, m$ and $G_j(v)$, $j = 0, \ldots, n$ be two sets of functions of parameters u and v, respectively. Then the surface with the parametric representation

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{a}_{i,j}, \quad u \in [u_0, u_1], \quad v \in [v_0, v_1] \tag{3.214}$$

is called *tensor product surface* or *tensor product patch*. The point matrix $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ is referred to as the *control net* of the tensor product surface $\mathbf{q}(u, v)$ and the functions $F_i(u)$ and $G_j(v)$ are called *blending functions*.

Remark 3.21 An obvious method of generalizing the concept of freeform curves could be to use only one family of blending functions $H_{i,j}(u, v)$, $i = 0, \ldots, m$, $j = 0, \ldots, n$—albeit doubly indexed and bivariate—and define a freeform surface as

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} H_{i,j}(u, v) \cdot \mathbf{a}_{i,j}.$$

This would, undoubtedly, be a more general approach. The primary reason for sticking to products $F_i(u) \cdot G_j(v)$ of univariate functions instead is that this way one can easily transfer a lot of methods and concepts from the freeform curves.

Remark 3.22 A tensor product surface is a very special type of a so-called *freeform surface*. The notion *freeform surface* refers to a pretty general class including *sub-*

division surfaces, *triangular patches* (cf. Sect. 3.11, pp. 225), *Coons patches* (cf. Sect. 3.9.1, pp. 204) and many more.

One appealing form of representing a tensor product surface (3.214) is:

$$\mathbf{q}(u,v) = [F_0(u), \ldots, F_m(u)] \cdot \begin{bmatrix} \mathbf{a}_{0,0} & \cdots & \mathbf{a}_{0,n} \\ \vdots & & \vdots \\ \mathbf{a}_{m,0} & \ldots, & \mathbf{a}_{m,n} \end{bmatrix} \cdot \begin{bmatrix} G_0(v) \\ \vdots \\ G_n(v) \end{bmatrix} \tag{3.215}$$

The u-lines $v = v_0 = const$ on a tensor product surface $\mathbf{q}(u,v)$, for one, can be recognized as freeform curves in the sense of Definition 3.24, p. 86:

$$\begin{aligned} \mathbf{q}(u,v_0) &= \sum_{i=0}^{m}\sum_{j=0}^{n} F_i(u)\cdot G_j(v_0)\cdot \mathbf{a}_{i,j} = \sum_{i=0}^{m} F_i(u)\cdot \underbrace{\left(\sum_{j=0}^{n} G_j(v_0)\cdot \mathbf{a}_{i,j}\right)}_{=:\ \mathbf{c}_i} \\ &= \sum_{i=0}^{m} F_i(u)\cdot \mathbf{c}_i \end{aligned} \tag{3.216}$$

The points $\mathbf{c}_i$ form a control polygon of that u-line. An analogous result holds for the v-lines.

A tensor product surface is uniquely determined by the given control net and the two families $F_i(u)$ and $G_j(v)$. Moreover, it does not depend on the choice of the coordinate system if each of the families $F_i(u)$ and $G_j(v)$ fulfills the *partition of unity*-condition:

$$\sum_{i=0}^{m} F_i(u) \equiv \sum_{j=0}^{n} G_j(v) \equiv 1 \tag{3.217}$$

3.8.1 Bézier Surfaces

We now particularize the tensor product surfaces by choosing specific sets of blending functions $F_i(u)$ and $G_j(v)$. For the most famous type of tensor product surfaces—the *Bézier surfaces*—these functions are Bernstein polynomials (see Definition 3.25, p. 87).

Definition 3.59. Bézier surface. A *Bézier surface* (or *Bézier patch*) *of degree* (m,n) is a tensor product surface (Definition 3.58) to a given control net $\mathbf{a}_{i,j}$, $i = 0,\ldots,m$, $j = 0,\ldots,n$ where the two sets of functions $F_i(u)$ and $G_j(v)$ consist of the Bernstein polynomials $B_{i,m}(u)$ and $B_{j,n}(v)$ of degree m and n, respectively:

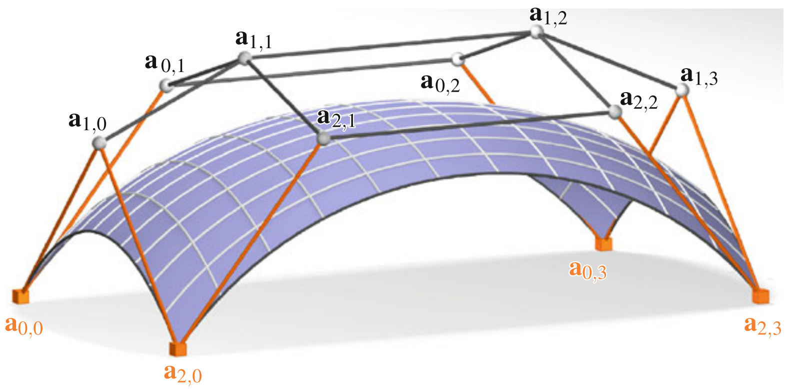


Fig. 3.101 The (2,3)-control net defines a Bézier surface. The corners of the patch coincide with the corners of the control net; even the tangent planes in the corner points (*orange cubes*) are spanned by the respective threads (*orange*). The parameter *lines* are Bézier curves themselves. The control polygons of the four boundary curves are even the boundary threads of the net

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_{i,m}(u) \cdot B_{j,n}(v) \cdot \mathbf{a}_{i,j}, \quad u \in [0, 1], \quad v \in [0, 1] \tag{3.218}$$

Proposition 3.18 Properties of Bézier surfaces. *A Bézier patch is defined by its control net. The following items point out the close connection between this net and the surface itself (Fig. 3.101):*

1. *Any u-line (v-line) of a Bézier surface is a Bézier curve (cf. Definition 3.26, p. 89). Its control polygon can be computed via (3.216).*
2. *The four boundary curves of the Bézier patch (3.218) are Bézier curves. This statement is, of course, a special case of the preceding. Their control polygons, however, are just the four boundary threads of the control net. For example, the control polygon of the boundary curve $\mathbf{p}(u) := \mathbf{q}(u, 0)$ is $\mathbf{a}_{0,0}, \ldots, \mathbf{a}_{m,0}$.*
3. *The Bézier surface and its control net have the same corner points:* $\mathbf{q}(0, 0) = \mathbf{a}_{0,0}$, $\mathbf{q}(0, 1) = \mathbf{a}_{0,n}$, $\mathbf{q}(1, 0) = \mathbf{a}_{m,0}$, $\mathbf{q}(1, 1) = \mathbf{a}_{m,n}$.
4. *The tangent plane $\tau_{0,0}$ to the Bézier surface at $\mathbf{q}(0, 0) = \mathbf{a}_{0,0}$ is spanned by the points $\mathbf{a}_{0,0}$, $\mathbf{a}_{0,1}$, $\mathbf{a}_{1,0}$. This is evident from 2.) and from the basic properties of Bézier curves. The same holds for the other three corners. We can shortly note: The tangent plane in a corner point of the Bézier patch is spanned by the two threads of the control net through this corner point.*
5. *As the Bernstein polynomials of a given degree are a partition of unity (3.217) we have: The surface is invariantly connected with its control net $\mathbf{a}_{i,j}$ with respect to affine transformations.*

6. *Due to the symmetry properties (3.63), p. 87, of Bernstein polynomials we have: The Bézier surface defined by the reversed*[42] *control net is the same as the original Bézier surface. Reversing the net only reverses the orientation of the respective parameter. The Bézier surface, taken as a whole, remains the same.*
7. *The surface is completely contained in the convex hull of the control net (convex-hull property).*

Bézier Surfaces on an Arbitrary Rectangular Domain

Sometimes it is necessary to transform the parameter representation of a Bézier surface to an arbitrary domain. We have introduced Bernstein polynomials on arbitrary intervals (see (3.71), p. 91). They immediately convey the more general parameterization of a Bézier surface:

$$\mathbf{q}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}^{[u_0,u_1]}(u)\cdot B_{j,n}^{[v_0,v_1]}(v)\cdot \mathbf{a}_{i,j}, \quad u \in [u_0,u_1], \quad v \in [v_0,v_1] \quad (3.219)$$

The de Casteljau Algorithm for Bézier Surfaces

Of course, (3.218) or (3.219) perfectly yields the point $\mathbf{q}(u_0, v_0)$ on a Bézier surface $\mathbf{q}(u_0, v_0)$ to any given parameter pair (u_0, v_0). As a favorable alternative to construct the point $\mathbf{q}(u_0, v_0)$ we can also apply the following geometric algorithm (Fig. 3.102) which is greatly based on the de Casteljau Algorithm 3.1 for curves, p. 92.

Algorithm 3.5. De Casteljau algorithm for Bézier surfaces. *We are given the control net $\mathbf{a}_{i,j}, i = 0,\dots,m, j = 0,\dots,n$ of a Bézier surface and a pair of parameters (u_0, v_0). The control net consists of $m+1$ v-threads: The v-thread with index i is $\mathbf{a}_{i,0},\dots,\mathbf{a}_{i,n}$.*

1. *We apply the de Casteljau algorithm for Bézier curves (Algorithm 3.1, p. 92) for the parameter value v_0 to each of the $m+1$ v-threads. This way we obtain a point $\mathbf{c}_i$ for every $i = 0,\dots,m$.*
2. *These points $\mathbf{c}_0,\dots,\mathbf{c}_m$, in turn, form a control polygon to one further Bézier curve which is the parameter line $v = v_0$ of the given Bézier patch.*
3. *We again apply the de Casteljau algorithm for Bézier curves, this time to the control polygon $\mathbf{c}_0,\dots,\mathbf{c}_m$ for the parameter value u_0. We end up at one single point which is the desired surface point $\mathbf{q}(u_0, v_0)$.*

Of course, this algorithm can also be applied the other way round, starting with the u-threads, accordingly. The final result $\mathbf{q}(u_0, v_0)$ remains the same. The proof to

[42] Here, *reversing the net* refers to swapping one index—say i—from i to $(m-i)$ and leaving the other index untouched. Reversing the net with respect to the second index—here, replacing j by $(n-j)$—is a different operation. Both actions combined make for 4 options overall.

this construction lies in formula (3.216) which—for Bézier surfaces—now reads as:

$$\begin{aligned}\mathbf{q}(u_0, v_0) &= \sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u_0)\cdot B_{j,n}(v_0)\cdot \mathbf{a}_{i,j}\\ &= \sum_{i=0}^{m} B_{i,m}(u_0)\cdot \underbrace{\left(\sum_{j=0}^{n} B_{j,n}(v_0)\cdot \mathbf{a}_{i,j}\right)}_{=:\ \mathbf{c}_i} = \sum_{i=0}^{m} B_{i,m}(u_0)\cdot \mathbf{c}_i \qquad (3.220)\end{aligned}$$

The de Casteljau algorithm for surfaces basically amounts to applying *repeated subdivision* to straight line segments, albeit with one additional loop (compared to the Bézier curve case). This also implies that the Bézier surface is affinely connected with its control net: Dividing a straight line segment by a given ratio is an affine construction.

Let us now refer to the standard parameter domain $[0, 1] \times [0, 1]$ as defined in (3.218). For $u_0, v_0 \in [0, 1]$ the segments are always divided in their interior throughout the whole de Casteljau algorithm; therefore, all corresponding points lie within the convex hull of the control net, and so does the resulting Bézier patch itself. This yields a straightforward proof to the convex hull property of Bézier patches (Proposition 3.18, 7.)).

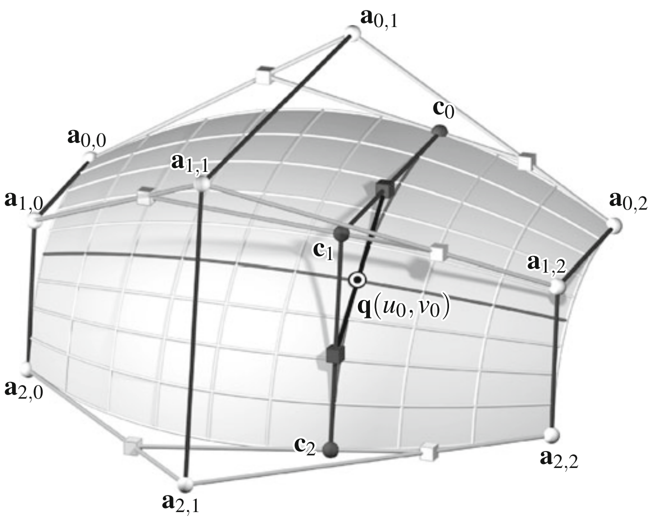


Fig. 3.102 The de Casteljau algorithm for a Bézier surface of degree $(m, n) = (2, 2)$ to the pair of parameters $(u_0, v_0) = (0.66, 0.55)$. Repeated subdivision applied to the edges of the control net leads to the point $\mathbf{q}(u_0, v_0)$

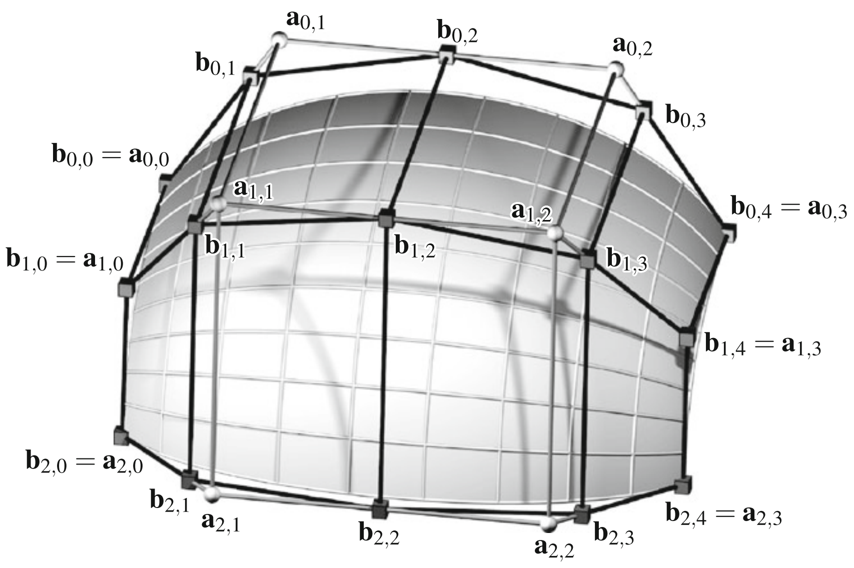


Fig. 3.103 We start with a (2, 3)-Bézier patch (control net *light grey*). Degree elevation, applied in v-direction, delivers a new control net (*dark threads* and *cubes*). This net—formally—determines a (2, 4)-Bézier patch which, however, is the same as the initial one

Degree Elevation

The procedure of degree elevation for Bézier curves (see (3.76), p. 98) can be transferred to Bézier surfaces. Let $\mathbf{q}(u, v)$ be a Bézier patch of order (m, n) with the control net $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$. In order to raise the degree n to $n + 1$ we take the following steps

- We apply the degree elevation for each of the $m + 1$ v-threads $\mathbf{a}_{i,0}, \ldots, \mathbf{a}_{i,n}$, $i = 0, \ldots, m$.
- The emerging new threads add up to a $(m, n + 1)$- control net $\mathbf{b}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n + 1$ of the original Bézier patch $\mathbf{q}(u, v)$.

Figure 3.103 shows a graphic example for a (2, 3)-Bézier patch (control net $\mathbf{a}_{i,j}$). After degree elevation the very same surface is represented as a (2, 4)-Bézier patch with the control net $\mathbf{b}_{i,j}$.

Of course, degree elevation can equally be applied to the u-threads, increasing the degree (m, n) to $(m + 1, n)$. Repeated degree elevation of either sort offers an abundance of options. The procedure does not change the shape of the Bézier patch, but it offers more leverage for further modification and modeling (compare the following section).

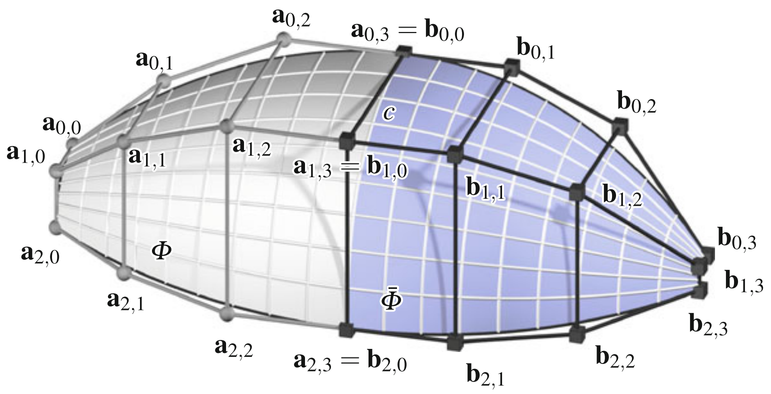


Fig. 3.104 A (2, 3)-Bézier patch Φ (*light grey*) and its continuation $\bar{\Phi}$ (*blue*) in v-direction to a design parameter $v_0 = 1.95$. The surface $\bar{\Phi}$ is a C^∞-continuation to Φ

Modeling Bézier Surfaces

We now want to create a continuation of a Bézier patch across one of its the boundary curves.

Construction 3.6 Continuation of a Bézier patch across a boundary curve. *(Figure 3.104) Let Φ be a Bézier patch with the control net $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$. We continue the patch across the boundary curve $c \ldots v = 1$ in the following way:*

- *We choose a so-called design parameter $v_0 > 1$ which controls the extent of the continuation.*
- *For every $i = 0, \ldots, m$ we apply the continuation algorithm for Bézier curves (Construction 3.2, p. 95) to the i-th v-thread $\mathbf{a}_{i,0}, \ldots, \mathbf{a}_{i,n}$ of the control net.*
- *For every $i = 0, \ldots, m$ we regard $\mathbf{b}_{i,0}, \ldots, \mathbf{b}_{i,n}$, the resulting control polygon of the i-thread continuation. The set of all such continuation polygons constitutes a new control net $\mathbf{b}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ of some Bézier patch $\bar{\Phi}$ which is the desired continuation of Φ across the boundary curve c.*

Figure 3.104 shows one example of such a continuation for the case $m = 2$ and $n = 3$. Here we have to keep in mind that a polynomial surface (e.g., a Bézier surface) can basically have the parameter domain $\mathbb{R} \times \mathbb{R}$. It ranges far beyond the considered parameter domain $[0, 1] \times [0, 1]$ of the original patch Φ. Thus, we can consider the continuation patch $\bar{\Phi}$ as just another part of the same polynomial surface, albeit for the parameter domain $[0, 1] \times [1, v_0]$. The smoothness between Φ and its continuation $\bar{\Phi}$ is GC^∞.

Remark 3.23 **Note on the GC^k- and C^k-continuity of continuation patches.** Similarly as in the case of continuations of Bézier curves via the de Casteljau algorithm (Remark 3.23, p. 187) we have to admit that the new patch $\bar{\Phi}$ only continues Φ with

C^∞-continuity after we have applied the linear parameter transformation $v = \frac{\tau-1}{v_0-1}$. Otherwise all partial derivatives of $\bar{\Phi}$ with respect to the parameter v differ from those of Φ by the factor $v_0 - 1$ all along the common boundary curve. This is why we can only speak of GC^∞-continuity. Analogously, we would only have GC^k- instead of C^k-continuity between Φ and a *modified* continuation patch $\bar{\Phi}^*$ generated via Construction 3.7, if we don't reparameterize accordingly. In the following paragraphs we will imply that this simple substitution has to be carried out after continuation though we will not always mention this step explicitly.

To have a separate control net for the continuation patch $\bar{\Phi}$ is pivotal for the following modeling process. Modifying the points of this control net will only affect the patch $\bar{\Phi}$. The original patch Φ will stay unaltered.

The procedure of modeling can also be transferred from the Bézier curve case (see Proposition 3.3, p. 100) to the case of Bézier surfaces:

Construction 3.7 Modeling the continuation of a Bézier patch. *Let Φ be a Bézier patch with the control net $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$. Moreover, let $v_0 > 1$ be a design parameter. Modeling the continuation patch $\bar{\Phi}$ consists of the following steps:*

- *Construct the continuation $\bar{\Phi}$ belonging to the design parameter v_0 (Construction 3.6).*
- *In each of the v-threads $\mathbf{b}_{i,0}, \ldots, \mathbf{b}_{i,n}$ of $\bar{\Phi}$ replace the last $n - k$ control points by some deliberately chosen*[43] *new points $\mathbf{b}^*_{i,k+1}, \ldots, \mathbf{b}^*_{i,n}$, but leave the first $k + 1$ control points $\mathbf{b}_{i,0}, \ldots, \mathbf{b}_{i,k}$ untouched.*
- *The modified control net*

$$\begin{bmatrix} \mathbf{b}_{0,0} & \ldots & \mathbf{b}_{0,k} & \mathbf{b}^*_{0,k+1} & \ldots & \mathbf{b}^*_{0,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathbf{b}_{m,0} & \ldots & \mathbf{b}_{m,k} & \mathbf{b}^*_{m,k+1} & \ldots & \mathbf{b}^*_{m,n} \end{bmatrix}$$

defines a Bézier surface $\bar{\Phi}^*$ which still continues the original surface Φ with C^k-smoothness.

Figure 3.105 shows an example where the continuation has been modified. The modified patch is still C^2-continuous along the boundary curve as the first 3 threads—counted from the transition tread $\mathbf{a}_{0,3} = \mathbf{b}_{0,0}, \mathbf{a}_{1,3} = \mathbf{b}_{1,0}, \mathbf{a}_{2,3} = \mathbf{b}_{2,0}$—remain unmodified.

Clearly, this construction can be adapted to the case where Φ has to be continued along its boundary curve $v = 0$ for some design parameter $v_0 < 0$. The analogue construction continues the patch across the boundary curve $u = 1$ or $u = 0$.

In the following construction we aim at filling a gap between two given Bézier patches, both of degree (m, n). To achieve this, two continuations of the fore-

[43] The choice of such control points is the job of the stylist. The expression *deliberately chosen* is only meant in terms of geometric correctness.

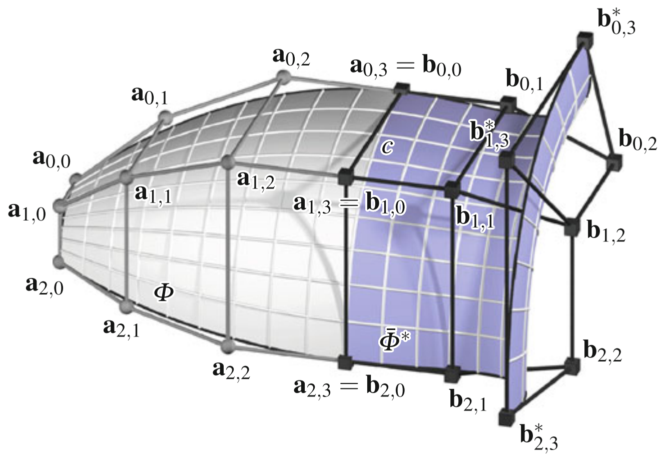


Fig. 3.105 The continuation $\bar{\Phi}$ (see Fig. 3.104) is modified according to Construction 3.7. As the first three threads of $\bar{\Phi}$ remain unaltered the modified patch $\bar{\Phi}^*$ (*blue*) still enjoys C^2-continuity towards Φ (*grey*) along the common boundary *curve*

mentioned type have to be applied. In a way, this is akin to Construction 3.4, p. 97, of a transition curve between two Bézier curves.

Construction 3.8 Filling a gap between two Bézier patches. *Let Φ_1 and Φ_2 be two Bézier patches, both of degree (m, n) and let k be a non-negative integer with $k \leq \frac{n-1}{2}$. To find a Bézier patch filling the gap between Φ_1 and Φ_2 C^k-continuously at either end we obey the following steps:*

- *Construct a continuation $\bar{\Phi}_1$ of Φ_1 along $v = 1$ with some design parameter $v_0 > 1$. Let*

$$\begin{bmatrix} \mathbf{b}_{0,0} & \dots & \mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \mathbf{b}_{m,0} & \dots & \mathbf{b}_{m,n} \end{bmatrix} \tag{3.221}$$

  be the control net of $\bar{\Phi}_1$.
- *Analogously, construct a continuation $\bar{\Phi}_2$ of Φ_2 along $v = 0$ with a design parameter $v_0 < 0$. Let*

$$\begin{bmatrix} \mathbf{c}_{0,0} & \dots & \mathbf{c}_{0,n} \\ \vdots & & \vdots \\ \mathbf{c}_{m,0} & \dots & \mathbf{c}_{m,n} \end{bmatrix} \tag{3.222}$$

  be its control net.
- *Assemble a new control net as follows:*

$$\begin{bmatrix} \mathbf{b}_{0,0} & \dots & \mathbf{b}_{0,k} & \mathbf{d}_{0,0} & \dots & \mathbf{d}_{0,n-2k-1} & \mathbf{c}_{0,k}, & \dots & \mathbf{c}_{0,0} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \mathbf{b}_{m,0} & \dots & \mathbf{b}_{m,k} & \mathbf{d}_{m,0} & \dots & \mathbf{d}_{m,n-2k-1} & \mathbf{c}_{m,k}, & \dots & \mathbf{c}_{m,0} \end{bmatrix} \tag{3.223}$$

The control points $\mathbf{b}_{i,j}$ and $\mathbf{c}_{i,j}$ are taken from (3.221) and (3.222), the points $\mathbf{d}_{i,j}$ can be deliberately chosen.
The Bézier patch Ω with this control net joins the two patches Φ_1 and Φ_2 C^k-continuously. The transition curve $v = 1$ of Φ_1 is identical to the boundary curve $v = 0$ of Ω. In the same way the transition curve $v = 0$ of Φ_2 is identical to the boundary curve $v = 1$ of Ω.

Figure 3.106 shows some examples with different values $n-2k-1$ indicating the number of threads in the control net of the gap filling patch which are still disposable for modeling.

We now return to Construction 3.6 which can be applied in u-direction as well as in v-direction, accordingly. These two continuations of a given patch Φ, however, do not fill a rectangular domain. To complete this we have to deliver some additional construction. Again, the tool is based on the de Casteljau scheme:

Construction 3.9 Filling a rectangular domain C^k-continuously with Bézier patches. *(Figure 3.107) Let Φ be a (m,n)-Bézier patch belonging to the control net $\mathbf{a}_{i,j}$, $i = 0, \dots, m$, $j = 0, \dots, n$. Let moreover u_0, $v_0 > 1$ be two design parameters and let $k \geq 0$ be the desired degree of continuity.*

- *We continue the given patch Φ to a patch Φ_1 in u-direction beyond $u = 1$ (design parameter u_0) and equally to a patch Φ_2 in v-direction beyond $v = 1$ (design parameter v_0) by Construction 3.6, employed twice.*
- *Next we modify the two continuations Φ_1 and Φ_2 such that C^k-smoothness towards Φ is still retained (cf. Construction 3.7, p. 188). We name the modified patches Φ_1^* and Φ_2^*.*
- *We now continue Φ_1^* in v-direction with respect to the design parameter v_0 and Φ_2^* in u-direction with respect to u_0 as design parameter. This way we arrive at two control nets; we now construct the control net of the desired patch Ω^* from these two.*
- *From each of the emerging two control nets we keep the $k+1$ threads adjacent to their respective transition thread. Note that the points within the overlapping area of those threads are identical anyway.*
- *The $(m-k) \times (n-k)$ points in the remote corner of the control net can be chosen deliberately which leaves space for further modeling.*

The Bézier surface Ω^* belonging to the constructed control net continues both adjacent patches Φ_1^* and Φ_2^* C^k-continuously.

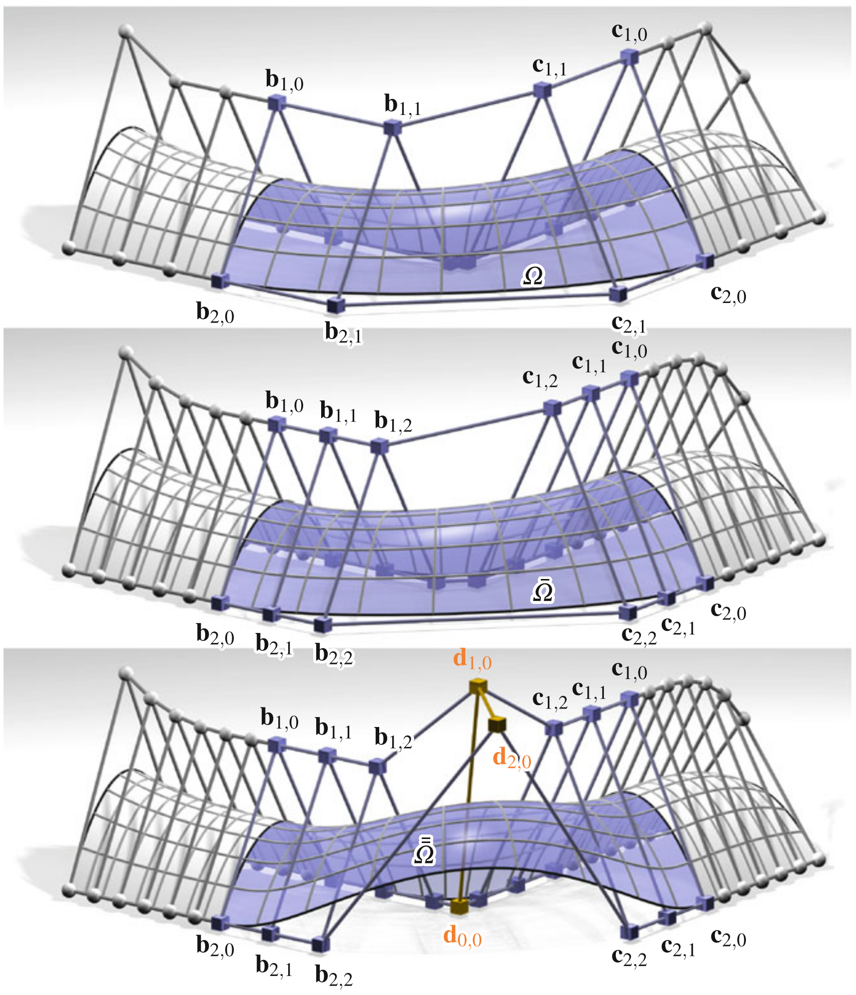


Fig. 3.106 The given (2, 3)-Bézier patches Φ_1 and Φ_2 (*light grey*) are to be connected via a further patch Ω such that the gap between the surfaces is closed. *Top* both patches Φ_1 and Φ_2 are continued towards the gap as described in Construction 3.6. The gap filling patch Ω is created from these two continuations. Each of them contributes the boundary thread and one further thread. The two threads from either continuation constitute the four threads of the (2, 3)-control net (*blue cubes*). The gap filling patch Ω continues Φ_1 and Φ_2 with mere C^1-continuity at either side. We have $n = 3$ and $k = 1$ and, consequently, $n-2k-1 = 0$ there are no remaining points $\mathbf{d}_{i,j}$ in-between which would be free for modification (see (3.223)). *Center* in order to achieve C^2-continuous transitions for the gap filling patch Ω the two given (2, 3)-control nets are subjected to repeated degree elevation in v-direction (see p. 186). We formally arrive at two (2, 5)-Bézier patches Φ_1 and Φ_2. We now demand $k = 2$, so each of the continuations contributes three threads to the (2, 5)-control net of the gap filling patch $\bar{\Omega}$. Here we have $n = 5$ and $k = 2$. The new gap filling patch $\bar{\Omega}$ is a C^2-continuation at either end. As for the number of threads which are free for further modification we again have $n-2k-1 = 0$. *Bottom* in order to retain C^2-continuity *and* to be able to modify the gap filling patch $\bar{\bar{\Omega}}$ we apply the degree elevation to Φ_1 and Φ_2 in v-direction one more time. Continuation as above (see (3.223)) delivers a (2, 6)-patch $\bar{\bar{\Omega}}$ which now offers one thread (control points $\mathbf{d}_{0,0}$, $\mathbf{d}_{1,0}$, $\mathbf{d}_{2,0}$, *orange*) which can be modified by the user without compromising the C^2-continuity at either end

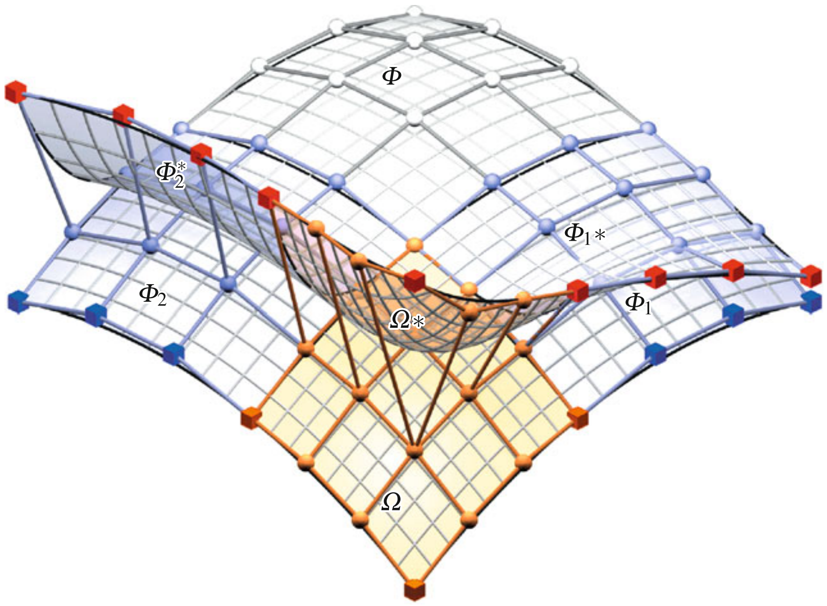


Fig. 3.107 The given (m, n)-Bézier patch Φ (here $m = n = 3$) is to be continued C^2-continuously ($k = 2$) in both directions u and v such that the overall continuation again fills a rectangular domain. The resulting surface is still to be modified such that C^2-continuity is retained. In the first step the two C^2-continuations Φ_1 (in u-direction) and Φ_2 (in v-direction) are each created as in Construction 3.6 (*blue patches*). The *dark blue cubes* denote the control points which are still free for modification. The *red cubes* signify the modified points. The continuation patches modified this way are named Φ_1^* and Φ_2^*. Finally, the patches Φ_1^* and Φ_2^* have to be continued in v-direction and in u-direction, respectively, according to Construction 3.9. We arrive at the patch Ω^*. As $(m - k) \times (n - k) = 1$, in this example there is only one point *left* for modification. It is represented by the *red cube* at the front top corner. For comparison, the figure also shows the unmodified patches Φ_1, Φ_2 and the patch Ω (*yellow*) generated from them

3.8.2 B-Spline Surfaces

After having negotiated Bézier surfaces we now consider another type of tensor product surfaces more closely. We specify the blending functions $F_i(u), i = 0, \ldots, m$ and $G_j(v)$, $j = 0, \ldots, n$ (Definition 3.58, p. 181) as B-spline basis functions (see Definition 3.29, p. 100). This gets us to *B-spline surfaces*. which are widely applied in CAD (see also [10], pp. 486 or [7], pp. 275).

Definition 3.60. Spline surfaces and subspline surfaces. A tensor product surface $\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{a}_{i,j}$ is called *spline surface* (*subspline surface*) if the functions $F_i(u)$, $G_j(v)$ are spline functions (subspline functions).[44]

[44] As for the term *spline function* and *subspline function* see Definition 3.28, p. 99.

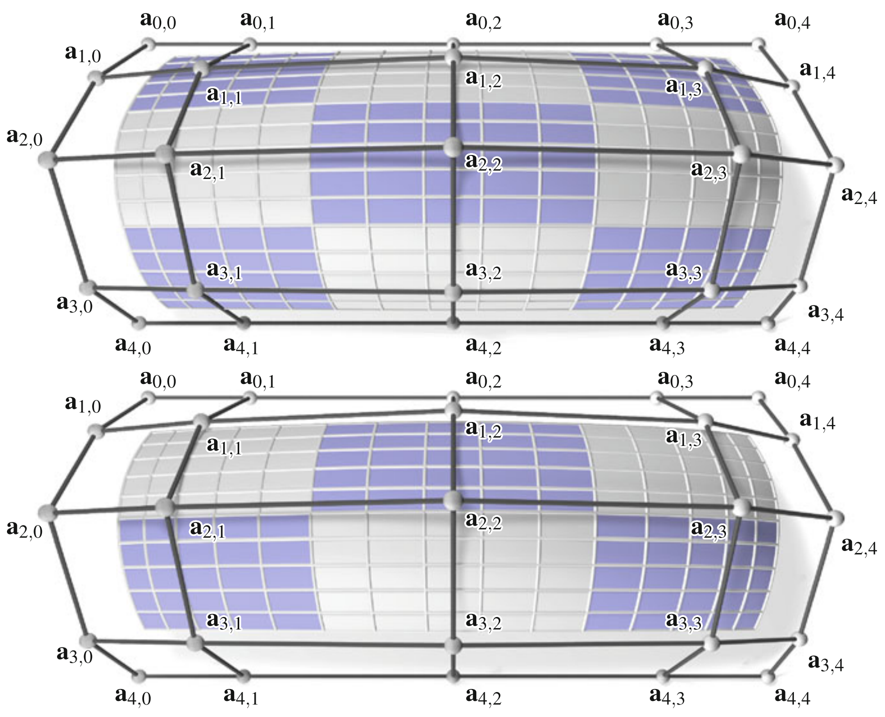


Fig. 3.108 A B-spline surface to a 5×5-de Boor net ($m = n = 4$). The *grey* and *blue* areas mark the segments of the B-spline surface relating to the parameter domains (see Proposition 3.19, 1.)). These surface segments are connected C^{k-2}-continuously in u-direction and C^{l-2}-continuously in v-direction (see Proposition 3.19, 3.)) *Top* $k = l = 3$. Due to $k - 2 = l - 2 = 1$ we have C^1-continuity all over between the segments. *Bottom* the same De Boor net as above, however with $k = 4$, $l = 3$. The number of segments decreases whilst the degree of continuity in u-direction increases to $k - 2 = 2$.

Definition 3.61. B-spline surface. Let k, l, m, n be integers with $2 \leq k \leq m+1$, $2 \leq l \leq n+1$ and let $(u_0, u_1, \ldots, u_m, u_{m+1}, \ldots, u_{m+k})$, $(v_0, v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+l})$ be two knot vectors. Let moreover $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ be a matrix of points in 3-space. Then the surface with the parametric representation

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,k}(u) \cdot N_{j,l}(v) \cdot \mathbf{a}_{i,j},$$

$$(u, v) \in [u_{k-1}, u_{m+1}] \times [v_{l-1}, v_{n+1}] \qquad (3.224)$$

is called *B-spline surface* (see also Fig. 3.108) to the given knot vectors and to the *control net* $\mathbf{a}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$. The latter is also called the *de Boor net* of $\mathbf{q}(u, v)$.

If the knot vectors are uniform (see Definition 3.31, p. 103), we also call the B-spline surface *uniform*.

Proposition 3.19 Properties of B-spline surfaces. *We consider a B-spline surface according to Definition 3.61:*

1. *The segment of the B-spline surface relating to the domain* $[u_r, u_{r+1}] \times [v_s, v_{s+1}]$ *only depends on the control points* $\mathbf{a}_{i,j}$, $i = r - k + 1, \ldots, r$, $j = s - l + 1, \ldots, s$.
2. *The surface is invariantly connected with its de Boor net* $\mathbf{a}_{i,j}$ *with respect to affine transformations.*
3. *A B-spline surface is smooth (i.e.,* C^∞*-continuous) within each domain* $(u_i, u_{i+1}) \times (v_j, v_{j+1})$*. For* $r = min(k, l)$ *it is* C^{r-2}*-continuous even along the boundary curves between such domains.*
4. *For the widely used case* $k = l$ *we can state: A B-spline surface is* C^{k-2}*-continuous all over.*

As is the case for B-spline curves, increasing m or n arbitrarily does not affect the polynomial degree of the B-spline surface. This is one big difference to Bézier surfaces, and one big advantage, too.

The Cox-de Boor Algorithm for B-Spline Surfaces

The de Casteljau algorithm for Bézier surfaces (Algorithm 3.5) has a counterpart for B-spline surfaces, the so-called *Cox-de Boor algorithm* for B-spline surfaces.

The point $\mathbf{q}(u_0, v_0)$ on a B-spline surface can be constructed by means of the Cox-de Boor Algorithm 3.2 for curves, p. 104, applied to the single threads of the de Boor net $m + 1$ times in v-direction, followed by one additional application to the resulting polygon in u-direction.

This algorithm amounts to subdividing straight line segments repeatedly by predefined ratios. Thus the construction is affinely invariant. It tells us that the connection between the de Boor net and the B-spline surface is invariant with respect to affine transformations. The de Boor net subjected to such a transformation (e.g. scalings, isometries, coordinate transformations, ...) delivers the same surface as though we had subjected the original B-spline surface to that transformation.

Corner Interpolation

We remember that Bézier surfaces interpolate the four corners of their control net. Moreover the tangent planes in those four points are determined by the net segments starting at those points (Proposition 3.18, (3. and 4.), p. 183). For B-spline patches this handy attribute does not appear in general, though it can also be imposed if we choose k identical knots at both ends of the u-knot vector, i.e.,

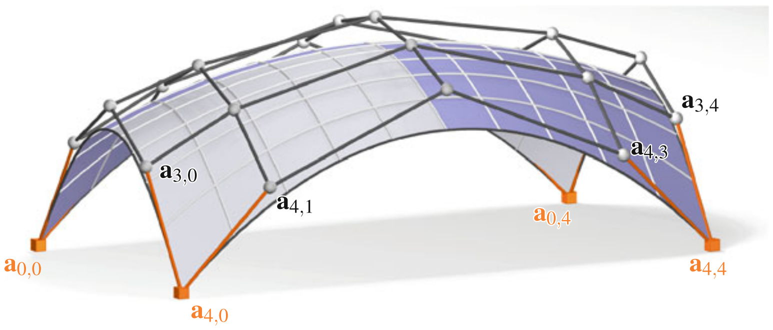


Fig. 3.109 In order to achieve the *corner interpolation property* the knot vectors have to be adjusted (see (3.225) and (3.226)). In this example we have $m = n = 4$ and $k = l = 4$. The tangent planes in the corners $\mathbf{a}_{0,0}$, $\mathbf{a}_{0,4}$, $\mathbf{a}_{4,0}$, $\mathbf{a}_{4,4}$ (*orange cubes*) are spanned by the respective edges of the De Boor net (*orange*)

$$u_0 = u_1 = \cdots = u_{k-1},\ u_k, \ldots, u_m,\ u_{m+1} = u_{m+2} = \cdots = u_{m+k}, \tag{3.225}$$

and, analogously, l identical knots at both ends of the v-knot vector:

$$v_0 = v_1 = \cdots = v_{l-1},\ v_l, \ldots, v_n,\ v_{n+1} = v_{n+2} = \cdots = v_{n+l} \tag{3.226}$$

Using these modified knot vectors the B-spline surface interpolates the corners $\mathbf{a}_{0,0}$, $\mathbf{a}_{0,n}$, $\mathbf{a}_{m,0}$, $\mathbf{a}_{m,n}$ for the parameter pairs (u_0, v_0), (u_0, v_{n+1}), (u_{m+1}, v_0), (u_{m+1}, v_{n+1}), respectively. Moreover, the two segments of the de Boor net emanating from each corner span the tangent plane of the surface in that point (Fig. 3.109).

Closed B-Spline Surfaces

In a similar way as we created closed B-spline curves (cf. Proposition 3.7, p. 108) we can also generate closed B-spline patches by continuing the de Boor net and the knot vectors periodically.

Note that a B-spline surface can be closed either in one direction (in u-direction or, alternatively, in v-direction) or in both directions (see Fig. 3.110).

We start with a B-spline surface defined by a de Boor net

$$\begin{bmatrix} \mathbf{a}_{0,0} & \ldots & \mathbf{a}_{0,n} \\ \vdots & & \vdots \\ \mathbf{a}_{m,0} & \ldots & \mathbf{a}_{m,n} \end{bmatrix}$$

and two knot vectors $(u_0, \ldots, u_{m+k})$, $(v_0, \ldots, v_{n+l})$. To close the patch in u-direction we continue the net $\mathbf{a}_{i,j}$ periodically in column direction:

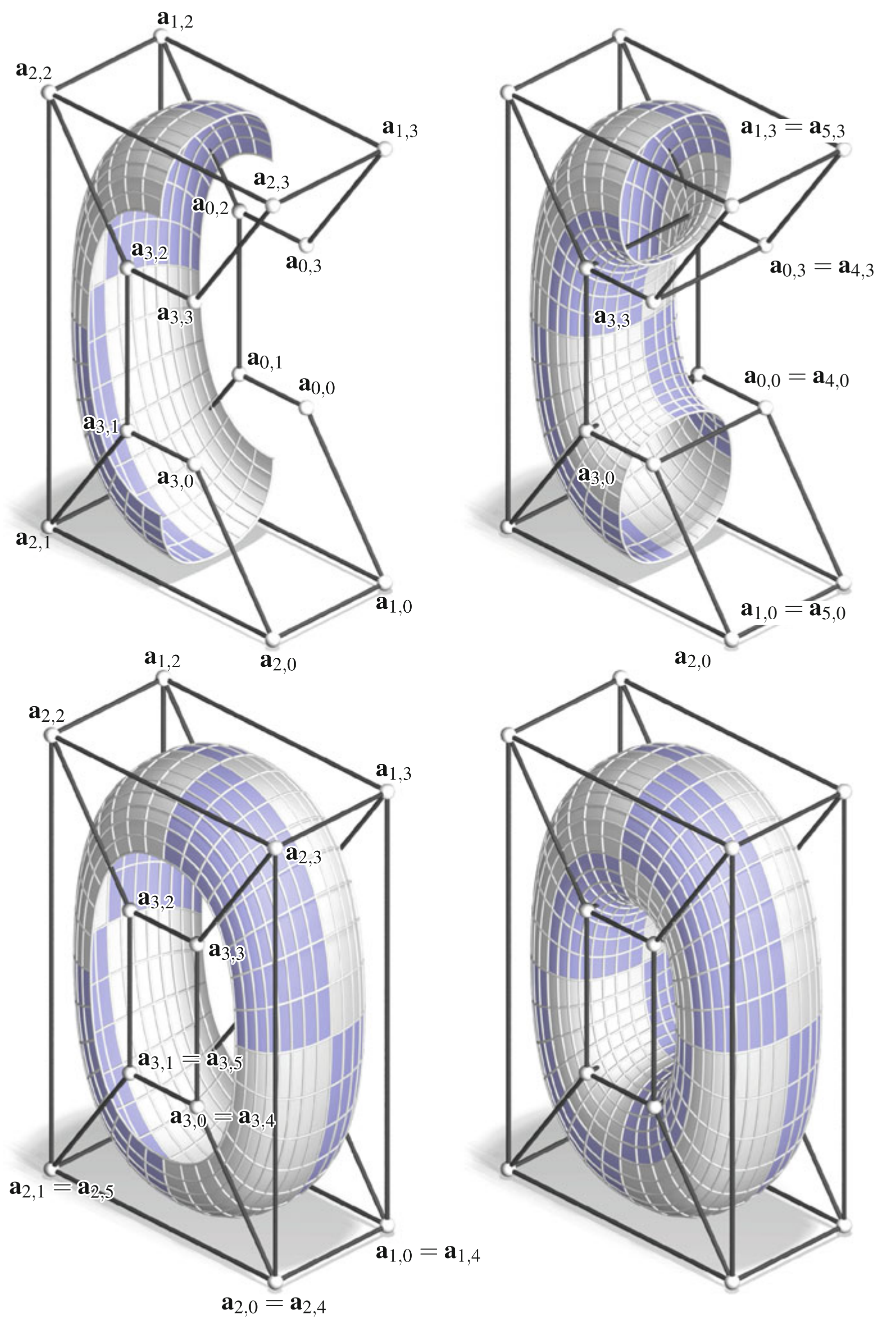


Fig. 3.110 B-spline surfaces can be closed in either direction. *Top left* A B-Spline patch with $m = n = k = l = 3$, not closed. *Top right* closing the patch in u-direction. *Bottom left* closing the patch in v-direction. *Bottom right* applying the closing procedure in u-direction and, after that, additionally in v-direction, delivers a doubly closed patch

$$\begin{bmatrix} \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,n} \\ \vdots & & \vdots \\ \mathbf{a}_{m,0} & \dots & \mathbf{a}_{m,n} \\ \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,n} \\ \vdots & & \vdots \\ \mathbf{a}_{k-2,0} & \dots & \mathbf{a}_{k-2,n} \end{bmatrix} \tag{3.227}$$

Additionally the u-knot vector is continued by adding the $k-1$ new knots:

$$u_{m+k+r} = u_{m+k} - u_{k-1} + u_{k-1+r}, \quad r = 1, \dots, k-1$$

Then the B-spline surface belonging to the de Boor net (3.227) and the knot vectors $(u_0, \dots u_{m+2k-1})$ and $(v_0, \dots v_{n+l})$ is closed in u-direction.

Obviously the analogue procedure could be applied to close the B-spline patch in v-direction.

If we apply the closing procedure in v-direction *after* having already closed the patch in u-direction we obtain a *doubly closed* surface (Fig. 3.110, bottom right). The topology resembles that of a torus (see Example 3.19, p. 172). The de Boor net of this doubly closed surface is

$$\begin{bmatrix} \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,n} & \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,l-2} \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathbf{a}_{m,0} & \dots & \mathbf{a}_{m,n} & \mathbf{a}_{m,0} & \dots & \mathbf{a}_{m,l-2} \\ \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,n} & \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,l-2} \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathbf{a}_{k-2,0} & \dots & \mathbf{a}_{k-2,n} & \mathbf{a}_{k-2,0} & \dots & \mathbf{a}_{k-2,l-2} \end{bmatrix}, \tag{3.228}$$

its v-knot vector is $(v_0, \dots v_{n+2l-1})$ where the additional knots $v_{n+l+1}, \dots, v_{n+2l-1}$ have to be computed via

$$v_{n+l+s} = v_{n+l} - v_{l-1} + v_{l-1+s}, \quad s = 1, \dots, l-1.$$

Local Control

Proposition 3.19, (1.), p. 194 implies that changing any of the specified control points—say $\mathbf{a}_{r,s}$—only influences the patch of the B-spline surface relating to the parameter domain $[u_r, u_{r+k}] \times [v_s, v_{s+l}]$. The sections of the B-spline surface outside this domain remain unaffected. This is generally referred to as the *local control property* of B-spline surfaces. Figure 3.111 illustrates the effect of such a modification.

It is evident that local control is a significant and welcome characteristic of B-spline surfaces.

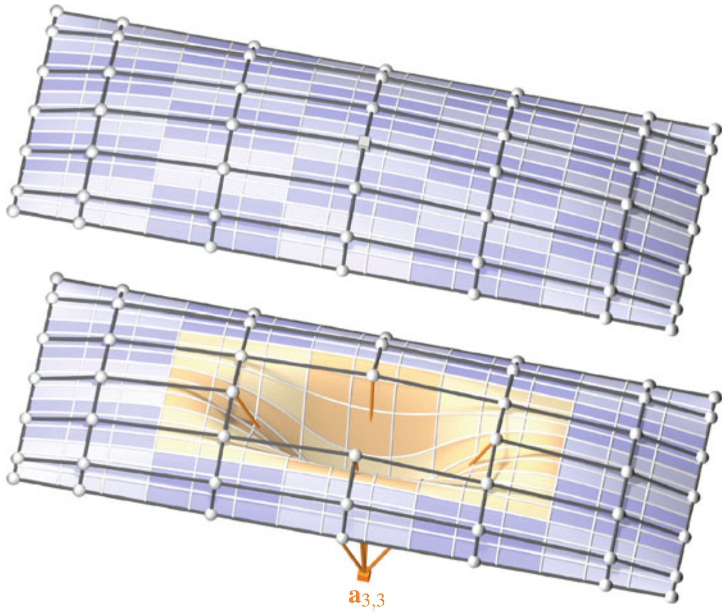


Fig. 3.111 Local control property of B-Spline surfaces. Here we have $m = n = 6$, $k = l = 3$. The segments of the B-spline patch are marked in *light blue* and *blue*. The transitions between them are C^1 all over. *Top* the original patch Φ. *Bottom* the control point $\mathbf{a}_{3,3}$ of Φ has been modified (*orange cube*). The *yellow* segments are affected by the modification, the others remain unaltered. The transitions between segments are still C^1 all over the surface

3.8.3 Rational Tensor Product Surfaces, NURBS Surfaces

Having selected two appropriate families of basis functions $F_i(u), i = 0, \ldots, m$ and $G_j(v), j = 0, \ldots, n$ of parameters u and v, respectively, any control net determines an appropriate tensor product surface (see Definition 3.58, p. 181). We now describe the more general notion of *rational* tensor product surfaces.

We consider a control net of $(m + 1) \times (n + 1)$ points

$$\mathbf{a}_{i,j} = \begin{bmatrix} a_{i,j,1} \\ a_{i,j,2} \\ a_{i,j,3} \end{bmatrix}, \quad i = 0, \ldots, m, \ j = 0, \ldots, n$$

in 3-space. For every control point $\mathbf{a}_{i,j}$ we choose one further value $w_{i,j}$, the *weight of the point* $\mathbf{a}_{i,j}$. We generate a *rational tensor product surface* from the given control net $\mathbf{a}_{i,j}$ and the chosen weights $w_{i,j}$ in much the same way as we generated rational freeform curves in Sect. 3.4.3, p. 109:

We use a coordinate system Σ in 4-space with some origin O^* and coordinates x_0, x_1, x_2, x_3. Let us assume that the 3-space containing the control net $\mathbf{a}_{i,j}$ is embedded as the space $\pi \dots x_0 = 1$. More precisely, a point $\mathbf{p}$ with coordinates x, y, z in 3-space now is viewed as a 4-vector $[1, x, y, z]^\top$. This way the control point $\mathbf{a}_{i,j}$ is represented by the 4-vector $[1, a_{i,j,1}, a_{i,j,2}, a_{i,j,3}]^\top$.

Now, the weight $w_{i,j}$ of the point $\mathbf{a}_{i,j}$ comes into play: It is used as the scale factor of a dilation from the center O^*. The dilation gets $\mathbf{a}_{i,j} = [1, a_{i,j,1}, a_{i,j,2}, a_{i,j,3}]^\top$ into a point $\mathbf{b}_{i,j} := [w_{i,j}, w_{i,j}a_{i,j,1}, w_i a_{i,j,2}, w_i a_{i,j,3}]^\top$. This new point bears the information of both, the coordinates of $\mathbf{a}_{i,j}$ and its weight $w_{i,j}$. The points $\mathbf{b}_{i,j}$ constitute a net in 4-space which can be used as a control net of an ordinary tensor product surface

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{b}_{i,j}, \quad (u, v) \in [u_0, u_1] \times [v_0, v_1] \tag{3.229}$$

with some polynomial basis functions $F_i(u)$ and $G_j(v)$.

We now consider the central projection δ from the center O^* into the 3-space $x_0 = 1$. A point $\mathbf{q} = [x_0, x_1, x_2, x_3]^\top$ in the 4-space with $x_0 \neq 0$ is mapped to its image point $\mathbf{p} = [1, \frac{x_1}{x_0}, \frac{x_2}{x_0}, \frac{x_3}{x_0}]^\top$. We omit the coordinate $x_0 = 1$ and put $\delta(\mathbf{q}) = [\frac{x_1}{x_0}, \frac{x_2}{x_0}, \frac{x_3}{x_0}]^\top =: [x, y, z]^\top$.

This geometric interpretation can be condensed into the following

Definition 3.62. Rational tensor product surface. The *rational tensor product surface* to a given control net $\mathbf{a}_{i,j}$, given weights $w_{i,j}$ and two families of polynomial functions $F_i(u)$ and $G_j(v), i = 0, \dots, m,\ j = 0, \dots, n$, is the surface represented by

$$\mathbf{r}(u, v) = \frac{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot w_{i,j} \cdot \mathbf{a}_{i,j}}{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot w_{i,j}}. \tag{3.230}$$

It is well-defined for all parameter pairs (u, v) except for the zeros of the denominator $\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot w_{i,j}$.

Remark 3.24 Rational tensor product surfaces are actually no tensor product surfaces (compare (3.230) with (3.214)). However, they can be viewed as central projections of tensor product surfaces in a 4-dimensional space. Thus, they share a lot of properties with tensor product surfaces.

It is obvious that putting all the weights $w_{i,j} = 1$ gets us the ordinary tensor product surfaces as defined in Definition 3.58, p. 181: The denominator in (3.230) equals 1. Thus, rational tensor product surfaces can be viewed as a generalization of tensor product surfaces. The latter are also called *ordinary* or *integer tensor product surfaces* as opposed to *rational tensor product surfaces*.

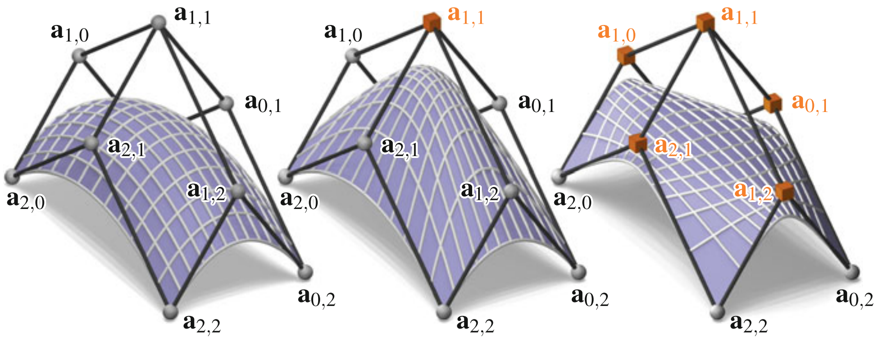


Fig. 3.112 A rational (2, 2)-Bézier patch; *Left* $w_{i,j} = 1.0$ for $i, j = 0, 1, 2$; an integer Bézier patch for comparison. *Middle* all weights unaltered except for $w_{1,1}$ which is changed to $w_{1,1} = 5.0$. Compared to the *left* example all points move towards $\mathbf{a}_{1,1}$ (*orange cube*). *Right* the weights of the marked points (*orange cubes*) are changed: $w_{1,1} \mapsto 0.02$, $w_{0,1} = w_{1,0} = w_{2,1} = w_{1,2} \mapsto 3.0$. The shape adapts accordingly

Our interpretation of such surfaces as central projections easily shows that a lot of properties of ordinary tensor product surfaces can be transferred to the rational case. The new parameters (weights) which can additionally be chosen by the user for each control point offer quite a bit more leverage. Even the range of surfaces which can be represented this way, is considerably wider if we decide to go for rational tensor product surfaces.

In Fig. 3.112 the effect of modifying the weights of rational Bézier patches is demonstrated. The left image shows an ordinary (integer) Bézier patch where the weights all equal 1. In the other two examples the weights of the orange control points (symbolized as cubes) have been modified accordingly (see caption).

The rational tensor product surfaces are a very handy tool for the stylist. The knowledge of their basic properties is essential. The question how the weight factors affect the surface shape, can be answered as in Proposition 3.8, p. 111:

Proposition 3.20 Properties of rational tensor product surfaces.

1. *The shape of a rational tensor product surface can be modified by changing weights, even without altering the control net (see Fig. 3.112).*
2. *Let $\mathbf{a}_{i,j}$ be the control net of a rational tensor product surface $\mathbf{r}(u, v)$. Moreover let us assume that all weight factors $w_{i,j}$ are positive. Increasing one weight w_{i_0,j_0} to $w_{i_0,j_0} + \Delta$ gets us another rational tensor product surface $\mathbf{r}^*(u, v)$. We have: For all (u, v) the points $\mathbf{r}(u, v)$, $\mathbf{r}^*(u, v)$ and $\mathbf{a}_{i_0,j_0}$ are collinear. If the functions $F_i(u)$ and $G_j(v)$ are non-negative within their support interval we can even observe that for a positive Δ every point $\mathbf{r}(u, v)$ moves towards $\mathbf{a}_{i_0,j_0}$. We can conclude:*
   Increasing the weight w_{i_0,j_0} puts more emphasis on the corresponding control point $\mathbf{a}_{i_0,j_0}$.

3. *Multiplying all weights with the same constant factor* $\alpha \neq 0$ *does not change the resulting rational tensor product surface. The common factor* $\alpha \neq 0$ *appears in the denominator and in the numerator of (3.230) and thus can be cancelled.*
4. *If all weights* $w_{i,j}$ *are equal the rational tensor product surface is identical to the ordinary tensor product surface to the same control net.*

Rational Bézier Surfaces

In the last paragraph we were dealing with rational tensor product surfaces in general. We now specify the defining families $F_i(u)$ and $G_j(v)$ of polynomials as the Bernstein polynomials of degree m and n (see Definition 3.59, p. 182). Putting $F_i(u) = B_{i,m}(u)$ and $G_j(v) = B_{j,n}(v)$ we obtain a *rational Bézier surface*:

$$\mathbf{r}(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)\cdot B_{j,n}(v)\cdot w_{i,j}\cdot \mathbf{a}_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)\cdot B_{j,n}(v)\cdot w_{i,j}} \tag{3.231}$$

The range of surfaces covered by rational Bézier representations is considerably wider, compared to ordinary (integer) Bézier surfaces. To give an example, we now can even represent surfaces as Bézier patches whose parameterizations have circular parameter lines. Figure 3.113, for one, shows a rational surface of revolution (compare also Sect. 3.7.11, p. 170) created as a rational Bézier surface. Its meridian curve is some rational Bézier curve, marked in orange. The highlighted part of the surface (blue segment on the right) has been created via Proposition 3.10, p. 113. The angle of the circular arcs is $\varphi = \frac{\pi}{2}$ (quarter circles). The weights $w_{0,0}, w_{1,0}, w_{2,0}, w_{3,0}$ and $w_{0,2}, w_{1,2}, w_{2,2}, w_{3,2}$ of the two boundary u-threads have be set to 1, whereas the weights $w_{0,1}, w_{1,1}, w_{2,1}, w_{3,1}$ of the middle u-thread have been set to $\cos\frac{\pi}{2} = \frac{\sqrt{2}}{2}$. Proposition 3.10 states right away that this way the boundary curves on the top end and bottom end of the patch are circular arcs. It is easy to see, though, that the same holds for all v-lines on the patch and, moreover, that the patch in fact is part of a surface of revolution which could be assembled by four patches of rational Bézier surfaces of this type. The same method also works for arbitrary angles φ.

Further strategies of creating kinematic surfaces[45] as rational Bézier patches can be found in [25] and [26].

Rational B-Spline Surfaces, NURBS Surfaces

Once the general approach of creating a rational tensor product surface from an ordinary one is established it is an easy task to apply it to the case of B-spline surfaces:

[45] A *kinematic surface* is generated by subjecting a curve to a Euclidean motion.

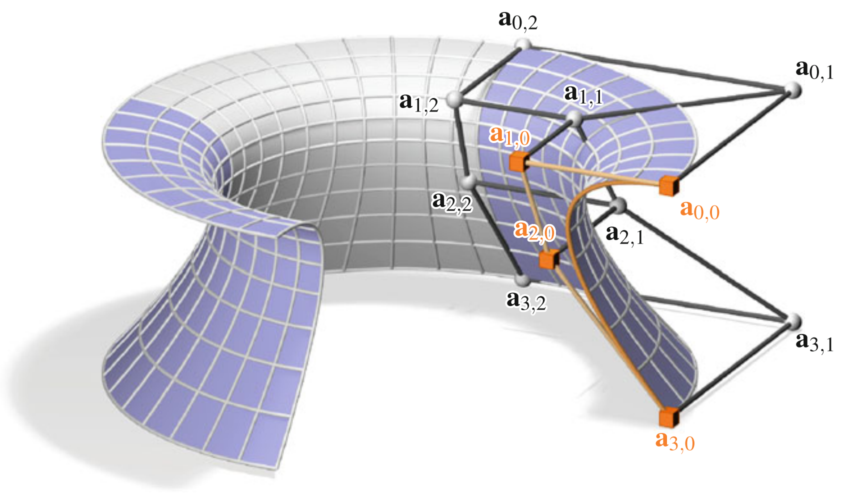


Fig. 3.113 A rational surface of revolution, represented as rational Bézier surface. The control net of one quarter is illustrated

Definition 3.63. Let a control net $\mathbf{a}_{i,j}$ and weight factors $w_{i,j}$, $i = 0, \dots, m$, $j = 0, \dots, n$ be given. Let $N_{i,k}(u)$ and $N_{j,l}(v)$ be the B-spline basis functions belonging to the knot vectors $(u_0, \dots, u_{m+k})$ and $(v_0, \dots, v_{n+l})$, respectively. Then the corresponding *rational B-spline surface* is defined via

$$\mathbf{q}(u, v) := \frac{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} N_{i,k}(u) \cdot N_{j,l}(v) \cdot w_{i,j} \cdot \mathbf{a}_{i,j}}{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} N_{i,k}(u) \cdot N_{j,l}(v) \cdot w_{i,j}}. \tag{3.232}$$

The interpretation as a central projection of an ordinary B-spline surface in 4-space yields the basic properties of rational B-spline surfaces. The decision to switch over from ordinary B-spline surfaces to *rational* B-spline surfaces is an easy one. The rational ones cover the former class completely: Sticking to weights $w_{i,j} = 1$ all over delivers an *ordinary* B-spline patch (cf. Proposition 3.20, p. 200). Hence, the latter is just a special case of a rational B-spline patch.

The knot vectors of a B-spline surface can be uniform or non-uniform (see Definition 3.31, p. 103). The same holds for rational B-spline surfaces.

Definition 3.64. Rational B-spline surfaces with non-uniform knot vectors are generally called *NURBS surfaces* (*N*on-*U*niform *R*ational *B*-*S*pline surfaces).

NURBS surfaces offer lots of parameters to control the shape. This may be the reason why they have developed as a standard in CAD and graphics. Figure 3.114 shows some options of modification. Tinkering with the knot vectors and the weight can have a profound effect on the outcome. Mind that in Fig. 3.114 the control

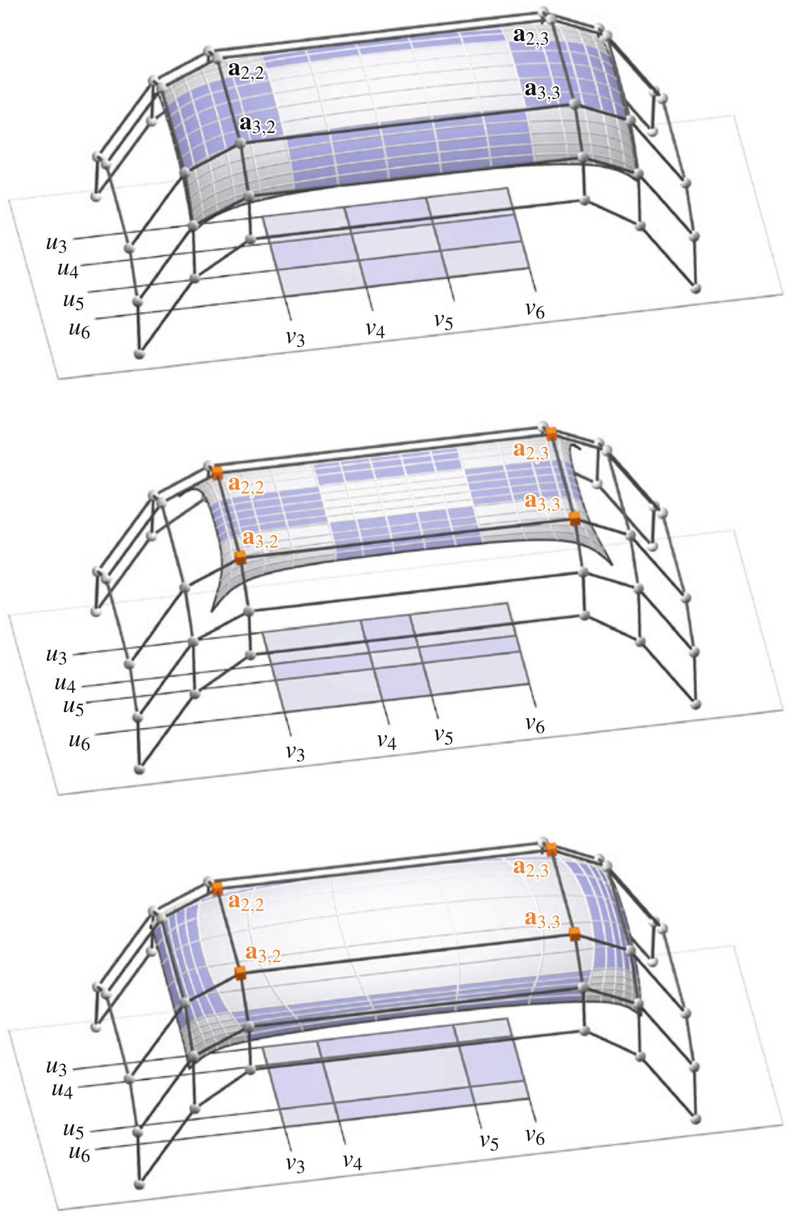


Fig. 3.114 NURBS surface with $m = n = 5$ and $k = l = 4$. Modification options without changing the control net. *Top* all weights are 1; uniform parameterization with $u_i = i$ and $v_j = j$. *Middle* four weights (respective points marked by *orange cubes*) are increased significantly: $w_{2,2} = w_{2,3} = w_{3,2} = w_{3,3} = 16.0$. The knot vectors are modified $\mathbf{u} = \mathbf{v} = (0.0, 1.0, 2.0, 3.0, 4.0, 4.5, 5.5, 6.5, 7.5, 8.5)$; regard the rectangular parameter area. *Bottom* the weights of the marked points (*orange*) are reduced: $w_{2,2} = w_{2,3} = w_{3,2} = w_{3,3} = 0.25$. The knot vectors are set to $\mathbf{u} = \mathbf{v} = (0.0, 1.0, 2.0, 3.0, 4.0, 6.5, 7.5, 8.5, 9.5, 10.5)$

net remains unaltered. Of course, changing the net itself would be an additional opportunity for the stylist.

3.9 Bivariate Interpolation

In Sect. 3.5 we have been dealing with univariate interpolation. Depending on the approach we could achieve C^0-, C^1- or C^2-continuous and—for particular cases—even smooth curves through given data points. Bivariate interpolation delivers a surface through a given set of points.

First we give an example of a tensor product surface (see Sect. 3.8) which is, at the same time, an example of bivariate interpolation. We can interpret it as the interpolation surface to a given skew rectangle.

Example 3.23 **The hyperbolic paraboloid as a tensor product surface.** Let us recall that a hyperbolic paraboloid (see Fig. 3.84, right, p. 165) consists of two families of straight lines. We can easily generate this surface as a (1, 1)-tensor product surface with blending functions[46] $F_0(u) = 1 - u$, $F_1(u) = u$, $G_0(v) = 1 - v$, $G_1(v) = v$ and the control net consisting of the four vertices $\mathbf{a}_{0,0}$, $\mathbf{a}_{0,1}$, $\mathbf{a}_{1,0}$, $\mathbf{a}_{1,1}$ of a skew quadrilateral. In this case the representation (3.215, p. 182) reads as

$$\begin{aligned}\mathbf{q}(u, v) &= [1 - u, u] \cdot \begin{bmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{bmatrix} \cdot \begin{bmatrix} 1 - v \\ v \end{bmatrix} \\ &= (1 - u) \cdot (1 - v) \cdot \mathbf{a}_{0,0} \; + \; (1 - u) \cdot v \cdot \mathbf{a}_{0,1} \\ &\quad + \; u \cdot (1 - v) \cdot \mathbf{a}_{1,0} \; + \; u \cdot v \cdot \mathbf{a}_{1,1}.\end{aligned} \tag{3.233}$$

The two families of straight line generators are the parameter lines of this parameterization.

This pretty simple approach enables us to fill a quadrangular mesh with curved surface patches. The straight line boundary segments between two adjacent patches are edges with mere C^0-continuity.

In the next paragraph we construct an interpolation surface to a given *curved* quadrangle.

3.9.1 Coons Patches

A surface patch $\mathbf{q}(u, v)$ to some rectangular domain $(u, v) \in [0, 1] \times [0, 1]$ has four boundary curves

[46] The blending functions are the Bernstein polynomials of degree 1; the tensor product surface is a (1, 1)-Bézier surface.

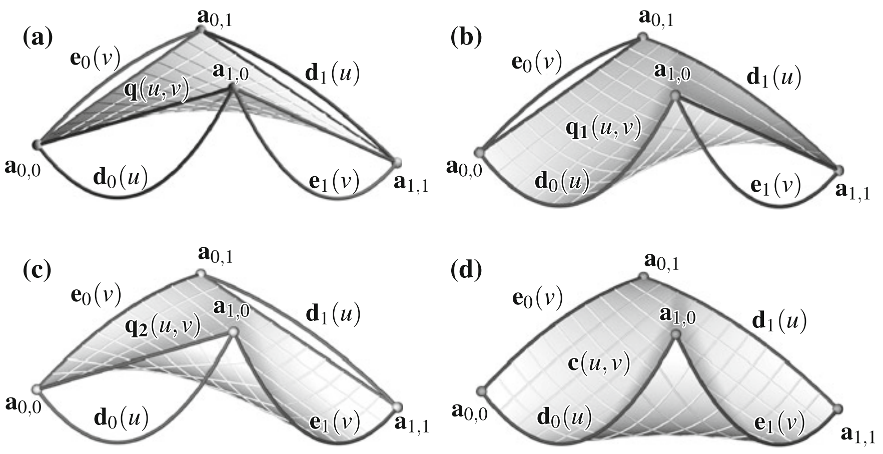


Fig. 3.115 A Coons patch $\mathbf{c}(u, v)$ (Fig. 3.115d) to a given curved quadrangle is made up from the hyperbolic paraboloid $\mathbf{q}(u, v)$ (Fig. 3.115a) and the ruled surfaces $\mathbf{q_1}(u, v)$ (Fig. 3.115b) and $\mathbf{q_2}(u, v)$ (Fig. 3.115c). **a** A suitable curved patch to a quadrangle $\mathbf{a}_{0,0}, \mathbf{a}_{1,0}, \mathbf{a}_{1,1}, \mathbf{a}_{0,1}$ with given curved edges $\mathbf{d}_0(u), \mathbf{e}_1(v), \mathbf{d}_1(u), \mathbf{e}_0(v)$ is to be created. The very first approach to this task is a hyperbolic paraboloid $\mathbf{q}(u, v)$ through the corners of the given quadrangle. It contains neither of the given curved edges. **b** In the following step the ruled surface $\mathbf{q_1}(u, v)$ is created (see (3.236)). It connects the two opposite curved edges $\mathbf{d}_0(v)$ and $\mathbf{d}_1(v)$. **c** The next step towards a Coons patch is the ruled surface $\mathbf{q_2}(u, v)$ as described in (3.237). It connects the two opposite curved edges $\mathbf{e}_0(v)$ and $\mathbf{e}_1(v)$. **d** Finally, according to (3.235) the three approaches are combined to the Coons patch $\mathbf{c}(u, v)$. As desired, it goes through all the given curved edges

$$\mathbf{d}_0(u) := \mathbf{q}(u, 0),$$

$$\mathbf{d}_1(u) := \mathbf{q}(u, 1),$$

$$\mathbf{e}_0(v) := \mathbf{q}(0, v),$$

$$\mathbf{e}_1(v) := \mathbf{q}(1, v).$$

These curve segments meet at their endpoints:

$$\left.\begin{array}{l} \mathbf{a}_{0,0} = \mathbf{d}_0(0) \;=\; \mathbf{e}_0(0) \\ \mathbf{a}_{0,1} = \mathbf{d}_1(0) \;=\; \mathbf{e}_0(1) \\ \mathbf{a}_{1,0} = \mathbf{d}_0(1) \;=\; \mathbf{e}_1(0) \\ \mathbf{a}_{1,1} = \mathbf{d}_1(1) \;=\; \mathbf{e}_1(1) \end{array}\right\} \tag{3.234}$$

If, conversely, such a *curved quadrangle* is given, the problem of creating some appropriate surface patch through these boundary curves has been solved by S.A. Coons[47] in a simple and elegant way [27]. His solution is commonly known as *Coons patch* (Fig. 3.115):

[47] Steven Anson Coons (1912–1979) was an early pioneer in the field of computer graphics at the MIT (Massachusetts Institute of Technology).

Definition 3.65. Coons patch. Let a curved quadrangle with edge curves $\mathbf{d}_0(u)$, $\mathbf{d}_1(u)$, $\mathbf{e}_0(v)$, $\mathbf{e}_1(v)$ and corners $\mathbf{a}_{i,j}$ according to (3.234) be given. Moreover, let $\mathbf{q}(u, v)$ denote the parameterization of the hyperbolic paraboloid through the corner points $\mathbf{a}_{i,j}$ as defined in (3.233). The surface patch

$$\mathbf{c}(u, v) := (1-v)\cdot\mathbf{d}_0(u) + v\cdot\mathbf{d}_1(u) + (1-u)\cdot\mathbf{e}_0(v) + u\cdot\mathbf{e}_1(v) - \mathbf{q}(u, v) \quad (3.235)$$

is called the *Coons patch* to the given curved quadrangle.

This patch (3.235) has boundary curves which indeed follow the given curved quadrangle. Coons' idea was to start with two ruled surfaces

$$\mathbf{q}_1(u, v) := (1-v)\cdot\mathbf{d}_0(u) + v\cdot\mathbf{d}_1(u) \quad (3.236)$$

and

$$\mathbf{q}_2(u, v) := (1-u)\cdot\mathbf{e}_0(v) + u\cdot\mathbf{e}_1(v). \quad (3.237)$$

Each of them connects two opposite curves of the given curved quadrangle. They form the first two summands of the parameterization (3.235). From this we have to *subtract the hyperbolic paraboloid* $\mathbf{q}(u, v)$ as represented in (3.233). These three ingredients perfectly add up to the solution of the task.

Coons patches, though introduced at a very early stage in the history of CAD, are used and appreciated as a singular gem of CAD to this day (cf. *fill surfaces*, p. 264). Sometimes this type of interpolation is termed *transfinite interpolation* as there are not only a number of single points to be interpolated but a quadrangle of curves with their infinite number of points.

If the Coons patch was to join with some given adjacent surface patch C^1-continuously the linear blending functions would have to be replaced by appropriate cubic functions ([7], pp. 377). This way, even a quadrangular gap surrounded by adjacent surface patches can be mended C^1-continuously. As an example of application we refer to Sect. 4.2, p. 256. In that example Fig. 4.20, center, shows the case without continuation constraints (C^0-continuity) and with tangency constraints (joining C^1-continuously all around, Fig. 4.20, right).

3.9.2 Interpolation of a Rectangular Point Set

The classical case of bivariate interpolation deals with the following task:

Problem 3.7 Bivariate interpolation. Let a rectangular point set.[48] The $\mathbf{c}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ in 3-space and two knot vectors $(s_0, \ldots, s_m)$ and $(t_0, \ldots, t_n)$ be given.

[48] The term *rectangular* refers to a matrix of points $\mathbf{c}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$.

Find a surface

$$\mathbf{q}(u, v), \quad u \in [s_0, s_m], \ v \in [t_0, t_n] \tag{3.238}$$

interpolating the given data, i.e.,

$$\mathbf{q}(s_i, t_j) = \mathbf{c}_{i,j}, \ \ i = 0, \dots, m, \ j = 0, \dots, n. \tag{3.239}$$

Clearly, we could modify this task looking for *real-valued* bivariate functions $q(u, v)$ interpolating a rectangular set of *real values*, i.e., $q(s_i, t_j) = c_{i,j}$. The solutions for vector functions which we offer in this section could easily be adjusted to that special case. So, in the following paragraphs we will concentrate on bivariate interpolation of points (vectors).

The formulation of Problem 3.7 is quite a bit general. We now modify the task by confining ourselves to tensor product surfaces:

Problem 3.8 Bivariate interpolation by a tensor product surface. Let a rectangular point set $\mathbf{c}_{i,j}$, $i = 0, \dots, m$, $j = 0, \dots, n$ in 3-space and two knot vectors $(s_0, \dots, s_m)$ and $(t_0, \dots, t_n)$ be given. Let moreover $F_i(u)$ and $G_j(v)$ be two families of linearly independent functions[49] such that each of these families forms a basis to some function spaces $\mathscr{F}$ and $\mathscr{G}$, respectively.

Find a tensor product surface

$$\mathbf{q}(u, v) = [F_0(u), \dots, F_m(u)] \cdot \underbrace{\begin{bmatrix} \mathbf{a}_{0,0} & \dots & \mathbf{a}_{0,n} \\ \vdots & & \vdots \\ \mathbf{a}_{m,0} & \dots & \mathbf{a}_{m,n} \end{bmatrix}}_{=:\mathbf{A}} \cdot \begin{bmatrix} G_0(v) \\ \vdots \\ G_n(v) \end{bmatrix} \tag{3.240}$$

meeting the conditions (3.239).

The particular ansatz (3.240) permits a straightforward solution. The unknown coefficient vectors $\mathbf{a}_{i,j}$ in the matrix $\mathbf{A}$ are to be computed. The conditions (3.239) now read as follows[50]:

$$\mathbf{F} \cdot \mathbf{A} \cdot \mathbf{G} = \mathbf{C} \tag{3.241}$$

where $\mathbf{F}$ is the $(m + 1) \times (m + 1)$-matrix

$$\mathbf{F} := \begin{bmatrix} F_0(s_0) & \dots & F_m(s_0) \\ \vdots & & \vdots \\ F_0(s_m) & \dots & F_m(s_m) \end{bmatrix}, \tag{3.242}$$

[49] The reader might as well imagine the functions $F_i(u)$ and $G_j(v)$ to be the Bernstein polynomials of degree m and n, respectively.

[50] Of course, this equation has to be interpreted for each component of the vectors $\mathbf{a}_{i,j}$, $\mathbf{c}_{i,j}$ separately.

G is the $(n+1) \times (n+1)$-matrix

$$\mathbf{G} := \begin{bmatrix} G_0(t_0) & \dots & G_0(t_n) \\ \vdots & & \vdots \\ G_n(t_0) & \dots & G_n(t_n) \end{bmatrix} \tag{3.243}$$

and **C** contains the points $\mathbf{c}_{i,j}$ to be interpolated:

$$\mathbf{C} := \begin{bmatrix} \mathbf{c}_{0,0} & \dots & \mathbf{c}_{0,n} \\ \vdots & & \vdots \\ \mathbf{c}_{m,0} & \dots & \mathbf{c}_{m,n} \end{bmatrix} \tag{3.244}$$

If **F** and **G** are invertible matrices[51] the unknown coefficient vectors can uniquely be determined via

$$\mathbf{A} = \mathbf{F}^{-1} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}. \tag{3.245}$$

We summarize in

Theorem 3.14. General solvability of the bivariate interpolation problem. *The bivariate interpolation Problem 3.8 has a unique solution if (3.242) and (3.243) are invertible matrices. In this case the solution is the tensor product surface (3.240) where the matrix* **A** *is determined according to (3.245).*

Of course, we cannot leave it there. We now specify the considered space of polynomials and the applied basis which will lead us to a more *down-to-earth*-attitude.

3.9.3 Bivariate Lagrange Interpolation

We pose Problem 3.8 in the spaces $\mathscr{F} = \mathbb{R}_m[t]$ and $\mathscr{G} = \mathbb{R}_n[t]$ of polynomials of degrees $\leq m$ and $\leq n$, respectively.

We still have to specify the bases of $\mathscr{F}$ and $\mathscr{G}$ though we are, of course, aware that the solution to our problem does not depend on the employed bases. The classical Lagrange approach uses the Lagrange bases

$$F_i(u) = L_{i,m}(u) = \prod_{\substack{l=0 \\ l \neq i}}^{m} \frac{u - s_l}{s_i - s_l}, \quad i = 0, \dots, m,$$

[51] This is for instance the case if both, $\mathscr{F}$ and $\mathscr{G}$, are Chebyshev spaces; see Definition 3.36, p. 117.

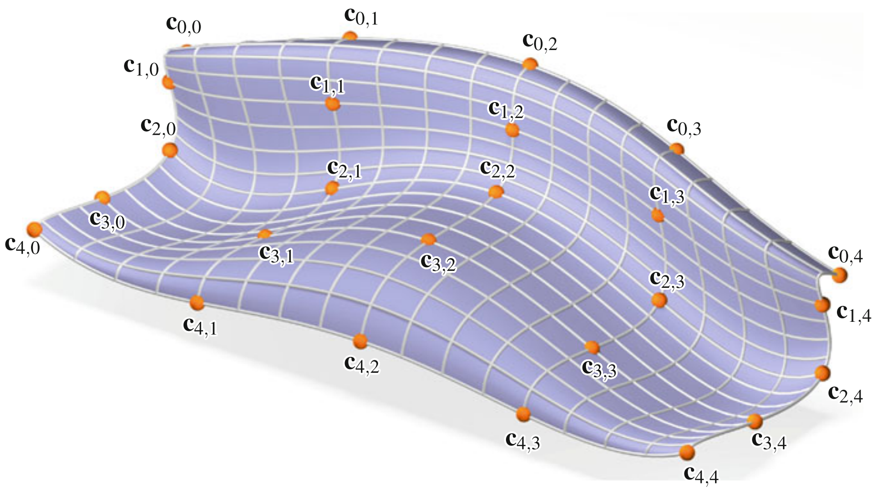


Fig. 3.116 The data points $\mathbf{c}_{i,j}$, $i, j = 0, \ldots, 4$ are to be interpolated. Bivariate Lagrange interpolation for $m = n = 4$ yields an appropriate polynomial surface. It is worth while comparing it with the B-spline solution shown in Fig. 3.118. Both figures start with the same set of input data. The Lagrange solution is prone to *waviness*; a phenomenon which is subdued in the B-spline solution (Fig. 3.118)

$$G_j(v) = L_{j,n}(v) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{v - t_k}{t_j - t_k}, \quad j = 0, \ldots, n$$

belonging to the knot vectors $(s_0, \ldots, s_m)$ and $(t_0, \ldots, t_n)$, respectively. As in the univariate case (see Sect. 3.5.1, p. 119) we enjoy the benefit that $\mathbf{F}$ and $\mathbf{G}$ are unit matrices. As a consequence of (3.245) we have $\mathbf{A} = \mathbf{C}$. Hence, the bivariate Lagrange interpolation surface has the parameterization

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} L_{i,m}(u) \cdot L_{j,n}(v) \cdot \mathbf{c}_{i,j}, \quad u \in [s_0, s_m], \quad v \in [t_0, t_n]. \qquad (3.246)$$

Any other choice of bases in $\mathscr{F}$ and $\mathscr{G}$ would have provided another approach to the same solution. Obviously, opting for the Lagrange bases cuts the whole story short. Figure 3.116 shows the Lagrange solution of a bivariate interpolation task.

Aitken's Algorithm for Bivariate Lagrange Interpolation

Aitken's algorithm for univariate Lagrange interpolation can easily be adapted to the bivariate case. The method resembles the de Casteljau scheme for Bézier surfaces (Algorithm 3.5, p. 184).

Algorithm 3.6. Aitken's algorithm; bivariate case. *Let* $\mathbf{C}$ *(Eq. (3.244)) be the rectangular set of points to be interpolated and let* $(s_0, \dots, s_m)$ *and* $(t_0, \dots, t_n)$ *be the given knot vectors. To compute the point* $\mathbf{q}(u, v)$ *on the Lagrange interpolation surface (3.246) we have to follow the steps:*

1. *We apply the univariate version of Aitken's algorithm (Algorithm 3.3, p. 121) to each column of the matrix* $\mathbf{C}$ *with respect to the knot vector* $(s_0, \dots, s_m)$ *and the value* u*. This gets us one point* $\mathbf{d}_i$ *for each column.*
2. *In the next step we apply the Aitken scheme to this polygon* $\mathbf{d}_0, \dots, \mathbf{d}_m$ *with respect to the knot vector* $(t_0, \dots, t_n)$ *and the value* v*. We end up at one single point which is the desired surface point* $\mathbf{q}(u, v)$*.*

Of course, the point $\mathbf{q}(u, v)$ could also be constructed the other way round, starting with the v-threads, accordingly.

In case of larger numbers of input points $\mathbf{c}_{i,j}$ the emerging Lagrange interpolation surface may be prone to oscillation which—in some cases—might even render the whole method useless. However, it is a perfect tool for reasonably small data sets. On the plus-side of the method we can also point out that the solution surface is C^∞-continuous all over.

3.9.4 Bivariate Hermite Interpolation

For the interpolation of larger rectangular data sets it may be advisable to partition the input into single quadrangles and to solve the interpolation Problem 3.8 for each quadrangle separately. The single solutions are to be created the way that they fit C^1-continuously along the seams. This is what we are aiming at in this section.

In a first step we consider one quadrangle where we additionally prescribe derivative vectors in the four corners:

Problem 3.9 Bivariate Hermite interpolation. Let the following input be given (Fig. 3.117):

- four points $\mathbf{c}_{0,0}$, $\mathbf{c}_{0,1}$, $\mathbf{c}_{1,0}$, $\mathbf{c}_{1,1}$
- corresponding partial derivative vectors $\mathbf{u}_{i,j}$, $\mathbf{v}_{i,j}$, $i, j = 0, 1$
- additional vectors $\mathbf{t}_{i,j}$, $i, j = 0, 1$ (which will be called *twist vectors*)
- a parameter domain $[s_0, s_1] \times [t_0, t_1]$

Find a surface with a bicubic tensor product representation

$$\mathbf{q} = \mathbf{q}(u, v), \quad (u, v) \in [s_0, s_1] \times [t_0, t_1] \tag{3.247}$$

satisfying the conditions

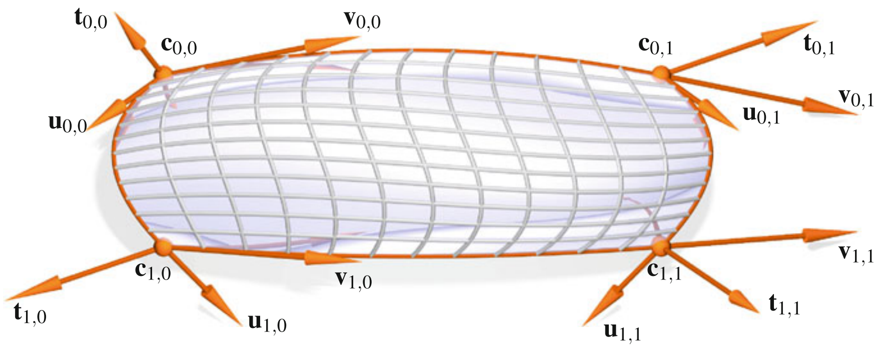


Fig. 3.117 A Hermite patch defined by its partial derivative vectors and twist vectors at the corner points

$$\begin{aligned}\mathbf{q}(s_i, t_j) &= \mathbf{c}_{i,j},\\ \frac{\partial \mathbf{q}}{\partial u}(s_i, t_j) &= \mathbf{u}_{i,j},\\ \frac{\partial \mathbf{q}}{\partial v}(s_i, t_j) &= \mathbf{v}_{i,j},\\ \frac{\partial^2 \mathbf{q}}{\partial u\, \partial v}(s_i, t_j) &= \mathbf{t}_{i,j}\end{aligned} \tag{3.248}$$

for $i, j = 0, 1$.

The Hermite basis (compare with (3.117), p. 126) comes in handy as the solution $\mathbf{q}(u, v)$ to our problem can be written promptly in terms of the input data:

$$\mathbf{q}(u, v) = [H_0^*(u), H_1^*(u), H_2^*(u), H_3^*(u)] \cdot \mathbf{H} \cdot \begin{bmatrix} H_0^\circ(v) \\ H_1^\circ(v) \\ H_2^\circ(v) \\ H_3^\circ(v) \end{bmatrix}, \tag{3.249}$$
$$(u, v) \in [s_0, s_1] \times [t_0, t_1]$$

where $\mathbf{H}$ is the 4×4-matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{c}_{0,0} & \mathbf{v}_{0,0} & \mathbf{v}_{0,1} & \mathbf{c}_{0,1} \\ \mathbf{u}_{0,0} & \mathbf{t}_{0,0} & \mathbf{t}_{0,1} & \mathbf{u}_{0,1} \\ \mathbf{u}_{1,0} & \mathbf{t}_{1,0} & \mathbf{t}_{1,1} & \mathbf{u}_{1,1} \\ \mathbf{c}_{1,0} & \mathbf{v}_{1,0} & \mathbf{v}_{1,1} & \mathbf{c}_{1,1} \end{bmatrix}. \tag{3.250}$$

The functions

$$\left.\begin{aligned}
H_0^*(u) &= \frac{1}{(s_1-s_0)^3}\cdot(s_1-u)^2\cdot(2u+s_1-3s_0),\\
H_1^*(u) &= \frac{1}{(s_1-s_0)^2}\cdot(u-s_0)\cdot(s_1-u)^2,\\
H_2^*(u) &= -\frac{1}{(s_1-s_0)^2}\cdot(u-s_0)^2\cdot(s_1-u),\\
H_3^*(u) &= \frac{1}{(s_1-s_0)^3}\cdot(u-s_0)^2\cdot(-2u+3s_1-s_0)
\end{aligned}\right\} \tag{3.251}$$

are the cubic Hermite polynomials on the interval $[s_0, s_1]$ and

$$\left.\begin{aligned}
H_0^\circ(v) &= \frac{1}{(t_1-t_0)^3}\cdot(t_1-v)^2\cdot(2v+t_1-3t_0),\\
H_1^\circ(v) &= \frac{1}{(t_1-t_0)^2}\cdot(v-t_0)\cdot(t_1-v)^2,\\
H_2^\circ(v) &= -\frac{1}{(t_1-t_0)^2}\cdot(v-t_0)^2\cdot(t_1-v),\\
H_3^\circ(v) &= \frac{1}{(t_1-t_0)^3}\cdot(v-t_0)^2\cdot(-2v+3t_1-t_0)
\end{aligned}\right\} \tag{3.252}$$

are the cubic Hermite polynomials on the interval $[t_0, t_1]$.

The solution (3.249) is called the *Hermite interpolation patch* to the input data given in Problem 3.9. Such a patch with its specified tangent and twist vectors at the corner points is displayed in Fig. 3.117. The following remark bears some significance if we intend to glue together patches of this kind.

Remark 3.25 A Hermite patch (3.249) has the following properties:

1. The boundary curve $u = s_0$ is a Hermite curve (compare with (3.116), p. 126) to the input data $\mathbf{c}_{0,0}$, $\mathbf{v}_{0,0}$, $\mathbf{v}_{0,1}$, $\mathbf{c}_{0,1}$ in the first row of matrix (3.250):

$$\mathbf{q}(s_0, v) = H_0^\circ(v)\cdot\mathbf{c}_{0,0} + H_1^\circ(v)\cdot\mathbf{v}_{0,0} + H_2^\circ(v)\cdot\mathbf{v}_{0,1} + H_3^\circ(v)\cdot\mathbf{c}_{0,1}$$

2. In the same way the cross derivative vectors along the same boundary curve are determined by input data $\mathbf{u}_{0,0}$, $\mathbf{t}_{0,0}$, $\mathbf{t}_{0,1}$, $\mathbf{u}_{0,1}$ in the second row of the matrix (3.254):

$$\frac{\partial\mathbf{q}(u,v)}{\partial u}(s_0, v) = H_0^\circ(v)\cdot\mathbf{u}_{0,0} + H_1^\circ(v)\cdot\mathbf{t}_{0,0} + H_2^\circ(v)\cdot\mathbf{t}_{0,1} + H_3^\circ(v)\cdot\mathbf{u}_{0,1}$$

Obviously, respective statements can be given for the other three boundary curves of the Hermite patch.

We now are ready to construct a C^1-continuous interpolation surface for a whole rectangular net of input data.

Problem 3.10 Bivariate Hermite interpolation of a rectangular point net. Let the following input be given:

- $(m+1) \times (n+1)$ points $\mathbf{c}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$ in 3-space
- corresponding partial derivative vectors $\mathbf{u}_{i,j}$, $\mathbf{v}_{i,j}$
- corresponding twist vectors $\mathbf{t}_{i,j}$
- and two knot sequences $(s_0, \ldots, s_m)$ and $(t_0, \ldots, t_n)$

Find a surface

$$\mathbf{q}(u, v), \quad u \in [s_0, s_m], \quad v \in [t_0, t_n] \tag{3.253}$$

which consists of bicubic tensor product patches $\mathbf{q}_{i,j}(u, v)$ and satisfies the conditions

$$\begin{aligned} \mathbf{q}(s_i, t_j) &= \mathbf{c}_{i,j}, \\ \frac{\partial \mathbf{q}}{\partial u}(s_i, t_j) &= \mathbf{u}_{i,j}, \\ \frac{\partial \mathbf{q}}{\partial v}(s_i, t_j) &= \mathbf{v}_{i,j}, \\ \frac{\partial^2 \mathbf{q}}{\partial u\, \partial v}(s_i, t_j) &= \mathbf{t}_{i,j} \end{aligned}$$

for $i = 0, \ldots, m$, $j = 0, \ldots, n$.

The solution to this problem is straightforward as we can construct a Hermite patch (3.249) for each parameter rectangle $[s_i, s_{i+1}] \times [t_j, t_{j+1}]$. We only have to replace s_0, s_1, t_0, t_1 by $s_i, s_{i+1}, t_j, t_{j+1}$ and the matrix $\mathbf{H}$ by

$$\mathbf{H}_{i,j} = \begin{bmatrix} \mathbf{c}_{i,j} & \mathbf{v}_{i,j} & \mathbf{v}_{i,j+1} & \mathbf{c}_{i,j+1} \\ \mathbf{u}_{i,j} & \mathbf{t}_{i,j} & \mathbf{t}_{i,j+1} & \mathbf{u}_{i,j+1} \\ \mathbf{u}_{i+1,j} & \mathbf{t}_{i+1,j} & \mathbf{t}_{i+1,j+1} & \mathbf{u}_{i+1,j+1} \\ \mathbf{c}_{i+1,j} & \mathbf{v}_{i+1,j} & \mathbf{v}_{i+1,j+1} & \mathbf{c}_{i+1,j+1} \end{bmatrix}. \tag{3.254}$$

We now ask the key question: Do adjacent patches constructed this way really fit C^1-continuously all along their common boundary curve?

As the matrices (3.254) belonging to adjacent patches coincide in the two rows or columns along the common boundary the answer to this question is positive. Remark 3.25 makes sure that all the cross derivatives along the boundary curve are identical. In Remark 3.14, p. 150, we explained that this is already sufficient for the two patches to join C^1-continuously along their common boundary.

One big advantage to the Hermite approach is that it does not lead to a linear system of equations which in case of large data sets might render the problem a numerical challenge.

The Hermite approach also requires derivative vectors $\mathbf{u}_{i,j}$, $\mathbf{v}_{i,j}$ and twist vectors $\mathbf{t}_{i,j}$ apart from the given points $\mathbf{c}_{i,j}$ to be interpolated. The choice of these vectors offers more leverage to the user. On the other hand, though, it may be difficult to choose those vectors reasonably. In order to estimate the derivative vectors $\mathbf{u}_{i,j}$ one can, for instance, compute a curve $\mathbf{p}_j(u)$ interpolating the points $\mathbf{c}_{0,j}, \ldots, \mathbf{c}_{m,j}$ for the knot sequence $s_0, \ldots, s_m$ and then put

$$\mathbf{u}_{i,j} := \frac{\mathrm{d}\mathbf{p}_j}{\mathrm{d}u}(s_i).$$

An analogous construction can be applied to obtain the vectors $\mathbf{v}_{i,j}$.

Several suggestions for the choice of the twist vectors $\mathbf{t}_{i,j}$ can be found in ([8], p. 276). Often the twist vectors are, for simplicity, all chosen zero which was primarily proposed by J. Ferguson [28] (*Ferguson approach*).

3.9.5 Bivariate Cubic B-Spline Interpolation

In the univariate case we have addressed the B-spline approach to the interpolation task (see p. 130). In this paragraph we transfer this method to the bivariate case.

We return to Problem 3.8, p. 207, now specifying the families $F_i(u)$ and $G_j(v)$ as the cubic B-spline basis functions $N_{i,4}(u)$, $N_{j,4}(v)$ (cf. Definition 3.29, p. 100).

Algorithm 3.7. Construction of an interpolating cubic B-spline surface. *Let a rectangular point set*

$$\mathbf{C} := \begin{bmatrix} \mathbf{c}_{0,0} & \cdots & \mathbf{c}_{0,n} \\ \vdots & & \vdots \\ \mathbf{c}_{m,0} & \cdots & \mathbf{c}_{m,n} \end{bmatrix}$$

and two knot vectors $(s_0, \ldots, s_m)$ and $(t_0, \ldots, t_n)$ be given as input data.

1. *For every column of $\mathbf{C}$ we construct an interpolating cubic B-spline curve*

$$\mathbf{p}_j(u) = \sum_{i=0}^{m+2} N_{i,4}(u) \cdot \mathbf{b}_{i,j}$$

   according to the method explained in Sect. 3.5.2, p. 122. Note that we have to add one additional tangent vector at either end, say $\mathbf{u}_{0,j}$ and $\mathbf{u}_{m,j}$. They assume the role of the tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_n$ (see (3.135), (3.136), p. 133). Moreover, an appropriate knot vector $(u_0, \ldots, u_{m+6})$ for the cubic B-spline functions $N_{i,4}(u)$

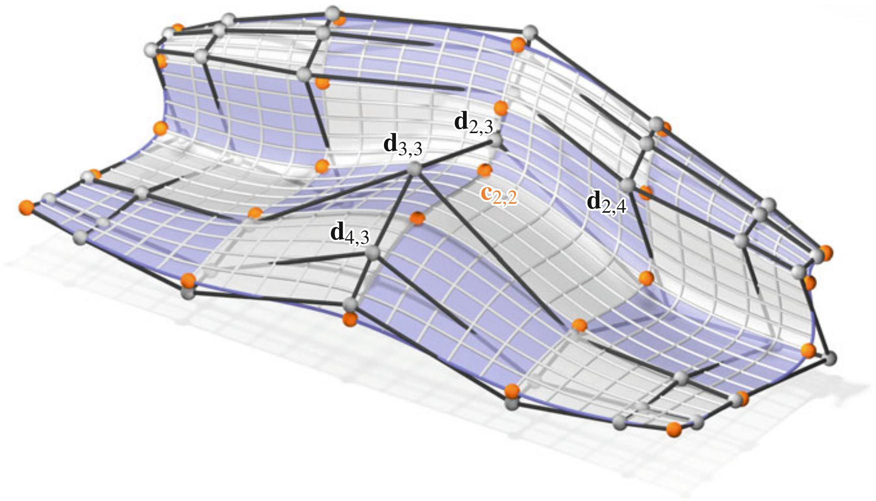


Fig. 3.118 Bivariate interpolation by a cubic B-spline surface for 5×5 input data points $\mathbf{c}_{i,j}$. The de Boor net $\mathbf{d}_{i,j}, i, j = 0, \ldots, 6$ of the appropriate B-spline interpolation patch has been computed via Algorithm 3.7. The set of data points $\mathbf{c}_{i,j}, i, j = 0, \ldots, 4$ (*orange*) to be interpolated is the very same as the one from Fig. 3.116. A comparison with the Lagrange solution in Fig. 3.116 shows that the B-spline method provides a more adaptive interpolation surface which is less susceptible to oscillation

has to be chosen. This vector plays the role of the knot vector $(t_0, \ldots, t_{n+6})$ *defined in the univariate case (see (3.129), p. 131).*
The de Boor points $\mathbf{b}_{i,j}$ *of the B-spline curve* $\mathbf{p}_j(u)$ *are computed via*

$$\begin{bmatrix} \mathbf{b}_{0,j} \\ \mathbf{b}_{1,j} \\ \vdots \\ \mathbf{b}_{m+1,j} \\ \mathbf{b}_{m+2,j} \end{bmatrix} = \mathbf{F}^{-1} \cdot \begin{bmatrix} \mathbf{u}_{0,j} \\ \mathbf{c}_{0,j} \\ \vdots \\ \mathbf{c}_{m,j} \\ \mathbf{u}_{m,j} \end{bmatrix}.$$

where $\mathbf{F}$ *is the matrix given by (3.138), p. 134, albeit with* $n \mapsto m$.

2. *In the next step we apply the same procedure to the rows of the obtained new matrix*

$$\mathbf{B} := \begin{bmatrix} \mathbf{b}_{0,0} & \ldots & \mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \mathbf{b}_{m+2,0} & \ldots & \mathbf{b}_{m+2,n} \end{bmatrix}.$$

Again, an appropriate knot vector $(v_0, \ldots, v_{n+6})$ *has to be determined, this time for the cubic B-spline functions* $N_{j,4}(v)$. *As in step 1.) we add a tangent vector,* *now at either end of each* row *to compute the points of our final de Boor net*

$$\mathbf{D} := \begin{bmatrix} \mathbf{d}_{0,0} & \dots & \mathbf{d}_{0,n+2} \\ \vdots & & \vdots \\ \mathbf{d}_{m+2,0} & \dots & \mathbf{d}_{m+2,n+2} \end{bmatrix}. \tag{3.255}$$

3. *The cubic B-spline surface*

$$\mathbf{q}(u,v) = \sum_{i=0}^{m+2}\sum_{j=0}^{n+2} N_{i,4}(u)\cdot N_{j,4}(v)\cdot \mathbf{d}_{i,j}$$

belonging to the de Boor net (3.255) and the knot vectors $(u_0,\dots,u_{m+6})$ *and* $(v_0,\dots,v_{n+6})$ *solves the interpolation problem.*

Figure 3.118 shows an example of a cubic B-spline interpolation surface created this way.

The biggest advantage of the B-spline approach is its ability to adapt even to larger sets of input points without noticeable oscillation.

3.10 Bivariate Approximation

This section deals with the problem of finding a surface S which fits[52] a given set of points $\mathbf{c}_\alpha$, $\alpha = 0,\dots,\mu$. The simplest case is the one where S is a plane (*plane of regression*). This task can also be referred to as *principal component analysis*, in short PCA (see [1], pp. 398–408). This very special case serves as a preparation for the following more challenging job of finding a tensor product surface fitting the given point set.

3.10.1 A Plane Fitting a Set of Scattered Points

It is a classical problem to determine a plane ε fitting a set

$$\mathbf{c}_\alpha = \begin{bmatrix} c_{\alpha,1} \\ c_{\alpha,2} \\ c_{\alpha,3} \end{bmatrix}, \quad \alpha = 0,\dots,\mu$$

[52] We say that a surface fits a given set of points, if certain distances between the points and some surface points are minimized. At the moment this explanation may seem faint, but it will be substantiated further on.

of points in 3-space. The solution offered below is particularly satisfactory as it does not favor any one of the 3 coordinate directions.

We regard a plane ε with the equation

$$n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + n_0 = 0$$

where

$$\mathbf{n} := \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$$

denotes a normalized normal vector of ε:

$$n_1^2 + n_2^2 + n_3^2 = 1 \tag{3.256}$$

Then the distance of a point $X(x, y, z)$ from the plane ε is computed as

$$\operatorname{dist}(X, \varepsilon) = |n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + n_0|.$$

Hence, we can get down to our optimization task as follows

Problem 3.11 Plane fitting. Let a set of points $\mathbf{c}_\alpha$, $\alpha = 0, \ldots, \mu$ be given. Find a plane

$$n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + n_0 = 0$$

such that n_0, n_1, n_2, n_3 minimize the error function (= sum of squared distances)

$$e(n_0, n_1, n_2, n_3) := \sum_{\alpha=0}^{\mu} (n_1 \cdot c_{\alpha,1} + n_2 \cdot c_{\alpha,2} + n_3 \cdot c_{\alpha,3} + n_0)^2 \tag{3.257}$$

subject to the constraint (3.256).

Introducing the $(\mu + 1) \times 4$ *design matrix*

$$\mathbf{C} := \begin{bmatrix} c_{0,1} & c_{0,2} & c_{0,3} & 1 \\ \vdots & & & \vdots \\ c_{\mu,1} & c_{\mu,2} & c_{\mu,3} & 1 \end{bmatrix}$$

we can rewrite (3.257) as follows:

$$e(n_0, n_1, n_2, n_3) := \mathbf{u}^\top \cdot \mathbf{S} \cdot \mathbf{u} \tag{3.258}$$

where

$$\mathbf{S} := \mathbf{C}^\top \cdot \mathbf{C} \;=\; \begin{bmatrix} \sum_{\alpha=0}^{\mu} c_{\alpha,1}^2 & \sum_{\alpha=0}^{\mu} c_{\alpha,1}\cdot c_{\alpha,2} & \sum_{\alpha=0}^{\mu} c_{\alpha,1}\cdot c_{\alpha,3} & \sum_{\alpha=0}^{\mu} c_{\alpha,1} \\ \sum_{\alpha=0}^{\mu} c_{\alpha,1}\cdot c_{\alpha,2} & \sum_{\alpha=0}^{\mu} c_{\alpha,2}^2 & \sum_{\alpha=0}^{\mu} c_{\alpha,2}\cdot c_{\alpha,3} & \sum_{\alpha=0}^{\mu} c_{\alpha,2} \\ \sum_{\alpha=0}^{\mu} c_{\alpha,1}\cdot c_{\alpha,3} & \sum_{\alpha=0}^{\mu} c_{\alpha,2}\cdot c_{\alpha,3} & \sum_{\alpha=0}^{\mu} c_{\alpha,3}^2 & \sum_{\alpha=0}^{\mu} c_{\alpha,3} \\ \sum_{\alpha=0}^{\mu} c_{\alpha,1} & \sum_{\alpha=0}^{\mu} c_{\alpha,2} & \sum_{\alpha=0}^{\mu} c_{\alpha,3} & \mu+1 \end{bmatrix}. \tag{3.259}$$

is the symmetric and positive semidefinite *scatter matrix* and

$$\mathbf{u} := \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_0 \end{bmatrix}.$$

In vector notation constraint (3.256) reads as

$$\mathbf{u}^\top \cdot \mathbf{E} \cdot \mathbf{u} = 1 \tag{3.260}$$

with

$$\mathbf{E} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.261}$$

In order to complete the Plane Fitting Task 3.11 we introduce a Langrange multiplier λ (see [1], p. 415) to link the error function (3.258) with the constraint (3.260):

$$\tilde{e}(n_0, n_1, n_2, n_3) := \mathbf{u}^\top \cdot \mathbf{S} \cdot \mathbf{u} - \lambda \cdot \mathbf{u}^\top \cdot \mathbf{E} \cdot \mathbf{u}$$

Every solution $\mathbf{u} = [n_1, n_2, n_3, n_0]^\top$ to Problem 3.11 has to satisfy the four equations

$$\frac{\partial \tilde{e}}{\partial n_1} = \frac{\partial \tilde{e}}{\partial n_2} = \frac{\partial \tilde{e}}{\partial n_3} = \frac{\partial \tilde{e}}{\partial n_0} = 0$$

which are linear in n_0, n_1, n_2, n_3. This system of equations can also be written as

$$(\mathbf{S} - \lambda\mathbf{E}) \cdot \mathbf{u} = \mathbf{0} \; = \; [0, 0, 0, 0]^\top. \tag{3.262}$$

At this point we arrive at a generalized eigenvalue problem.[53] The characteristic polynomial

$$p(\lambda) = \det(\mathbf{S} - \lambda\mathbf{E})$$

of the matrix **S** with respect to the matrix **E** is of degree 3 and thus may have up to three zeroes. Its roots can easily be determined, either numerically or by G. Cardano's formula for the zeroes of cubic polynomials.[54] We record some well-known issues:

- The eigenvalues of **S** with respect to **E**, i.e., the zeroes of $p(\lambda)$, are all real and non-negative. (This is due to the fact that both **S** and **E** are symmetric positive semidefinite matrices.)
- **S** is singular if and only if the input points $\mathbf{c}_\alpha$ are coplanar. This case is characterized by the existence of a zero eigenvalue $\lambda_0 = 0$.
- If not all of the input points are coplanar then the solution $\mathbf{u} = [n_1, n_2, n_3, n_0]^\top$ of Problem 3.11 belongs to the smallest eigenvalue λ_0 (= smallest zero of $p(\lambda)$). In general, the eigenspace belonging to λ_0 will have dimension 1 which means that the solution plane $n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + n_0 = 0$ for this point cloud is unique.

We have arrived at the following

Algorithm 3.8. Plane fitting algorithm. *Let a set*

$$\mathbf{c}_\alpha, \alpha = 0, \ldots, \mu$$

of points be given in 3-space. To compute the plane

$$\varepsilon \ldots n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + n_0 = 0 \tag{3.263}$$

which minimizes the sum of squared distances

$$\sum_{\alpha=0}^{\mu} \operatorname{dist}^2(\mathbf{c}_\alpha, \varepsilon)$$

we have to determine an eigenvector $\mathbf{u} = [n_1, n_2, n_3, n_0]^\top$ *belonging to the smallest (positive) eigenvalue of the generalized eigenvalue problem (3.262). The coordinates*

[53] If **E** was the identity matrix, we would have an *ordinary* eigenvalue problem as opposed to the case where **E** is an arbitrary matrix; then the eigenvalue problem is called *generalized*.

[54] Gerolamo Cardano (1501–1576) was an Italian Renaissance mathematician. Cardano's formula for the solution of 3^{rd} order equations can be found in every formulary booklet.

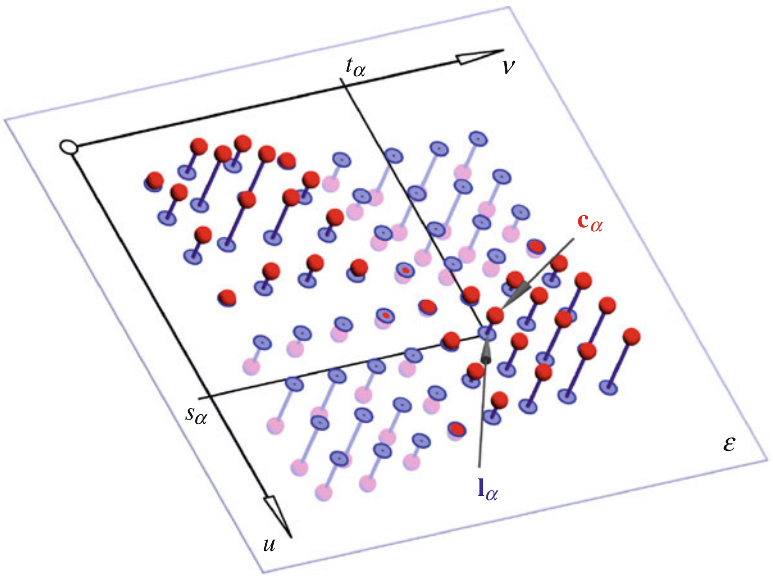


Fig. 3.119 We are given a point cloud of 64 points in 3-space. The Plane Fitting Algorithm 3.8 determines a plane ε such that the squared distance sum to the given points is minimized. The input points $\mathbf{c}_\alpha$ are highlighted in *red*, the pedal points $\mathbf{l}_\alpha$ on the resulting plane ε are marked as *blue* disks. The *black* sticks symbolize the distances. This algorithm is also used to assign parameters to the points of a given point cloud (parameter distribution of a point cloud, Sect. 3.10.2)

n_1, n_2, n_3, n_0 *of this eigenvector* $\mathbf{u}$ *are the coefficients in the linear Eq. (3.263) of the desired fitting plane (plane of regression).*

Figure 3.119 shows the result of the plane fitting algorithm to a given point cloud of 64 points.

3.10.2 A Tensor Product Surface Fitting Scattered Data Points

Let a point cloud

$$\mathbf{c}_\alpha, \alpha = 0, \ldots, \mu$$

and two sets of basis functions

$$\left.\begin{array}{ll} F_i(u), & i = 0, \ldots, m, \\ G_j(v), & j = 0, \ldots, n \end{array}\right\} \tag{3.264}$$

be given. We are to construct a tensor product surface

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{a}_{i,j} \tag{3.265}$$

fitting the given scattered data points. We solve this task in three steps:

(a) First we roughly estimate a suitable parameter distribution ($u = s_\alpha, v = t_\alpha$) for the given points $\mathbf{c}_\alpha, \alpha = 0, \dots, \mu$.
(b) Next we determine the yet unknown vector coefficients $\mathbf{a}_{i,j}$ of the desired solution surface (3.265) by solving a Gaussian least square problem.
(c) Finally we apply some iterative parameter correction routine in order to improve the result.

(a) A Parameter Distribution of a Point Cloud

In order to find a pair ($u = s_\alpha, v = t_\alpha$) of suitable parameters for each of the given points $\mathbf{c}_\alpha$ we determine the plane ε of regression to the given point cloud according to Sect. 3.10.1. Let $\mathbf{l}_\alpha$ be the orthogonal projections of the points $\mathbf{c}_\alpha$ onto the plane ε (Fig. 3.119). If the points $\mathbf{c}_\alpha$ do not deviate too much from ε and moreover, if the mapping $\mathbf{c}_\alpha \longrightarrow \mathbf{l}_\alpha$ is bijective it makes sense to use the points $\mathbf{l}_\alpha$ to find a suitable parameterization: We choose an arbitrary (but not necessarily orthogonal) u, v-coordinate system in ε, determine the coordinates $u = s_\alpha, v = t_\alpha$ of $\mathbf{l}_\alpha$ with respect to that coordinate system and use them as u- and v-parameters for the input points $\mathbf{c}_\alpha$.

Remark 3.26 In many cases the best fitting plane does the job properly. In some cases, however, there may be another simple surface which is better adjusted to the points $\mathbf{c}_\alpha$ (e.g., a sphere, right cylinder, quadric, ...). We can use this surface instead to determine a parameterization of the points.

The parameterization is important even though, in the third step, a round of parameter correction procedure will still fine-tune the parameters.

(b) An Approximating Tensor Product Surface to a Parameterized Point Cloud

Let now $\mu + 1$ points $\mathbf{c}_\alpha$ together with suitable parameter pairs $(s_\alpha, t_\alpha), \alpha = 0, \dots, \mu$ be given. We want to determine a tensor product surface (3.265) such that the squared error sum

$$\begin{aligned} e(\mathbf{a}_{0,0}, \ldots, \mathbf{a}_{m,n}) &= \sum_{\alpha=0}^{\mu} \|\mathbf{q}(s_\alpha, t_\alpha) - \mathbf{c}_\alpha\|^2 \\ &= \sum_{\alpha=0}^{\mu} \left\| \left(\sum_{i=0}^{m} \sum_{j=0}^{n} F_i(s_\alpha) \cdot G_j(t_\alpha) \cdot \mathbf{a}_{i,j} \right) - \mathbf{c}_\alpha \right\|^2 \end{aligned} \tag{3.266}$$

is minimized.

For a minimum of the error function (3.266) it is necessary that all partial derivatives with respect to the components of the vectors $\mathbf{a}_{i,j}$ vanish. In order to write down these conditions concisely we introduce the $(\mu + 1) \times (m + 1)(n + 1)$ *product matrix*

$$\mathbf{P} := \begin{bmatrix} F_0(s_0)G_0(t_0) & \ldots & F_0(s_0)G_n(t_0) & \ldots\ldots & F_m(s_0)G_0(t_0) & \ldots & F_m(s_0)G_n(t_0) \\ \vdots & & & & & & \vdots \\ F_0(s_\mu)G_0(t_\mu) & \ldots & F_0(s_\mu)G_n(t_\mu) & \ldots\ldots & F_m(s_\mu)G_0(t_\mu) & \ldots & F_m(s_\mu)G_n(t_\mu) \end{bmatrix} \tag{3.267}$$

and collect the yet unknown points $\mathbf{a}_{i,j}$ and the input points $\mathbf{c}_\alpha$ in arrays of length $(m + 1) \cdot (n + 1)$ and $\mu + 1$, respectively:

$$\mathbf{a} := \begin{bmatrix} \mathbf{a}_{0,0} \\ \vdots \\ \mathbf{a}_{0,n} \\ \vdots \\ \vdots \\ \mathbf{a}_{m,0} \\ \vdots \\ \mathbf{a}_{m,n} \end{bmatrix}, \qquad \mathbf{c} := \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_\mu \end{bmatrix}. \tag{3.268}$$

This way the vanishing of all partial derivatives of the error function (3.266) with respect to the components of the unknown vectors $\mathbf{a}_{i,j}$ can be written as

$$\mathbf{P}^\top \cdot \mathbf{P} \cdot \mathbf{a} = \mathbf{P}^\top \cdot \mathbf{c}. \tag{3.269}$$

If the quadratic $[(m+1) \cdot (n+1)] \times [(m+1) \cdot (n+1)]$-matrix $\mathbf{P}^\top \cdot \mathbf{P}$ on the left hand side of this equation is invertible our approximation task has the unique solution

$$\mathbf{a} = \left(\mathbf{P}^\top \cdot \mathbf{P}\right)^{-1} \cdot \mathbf{P}^\top \cdot \mathbf{c}. \tag{3.270}$$

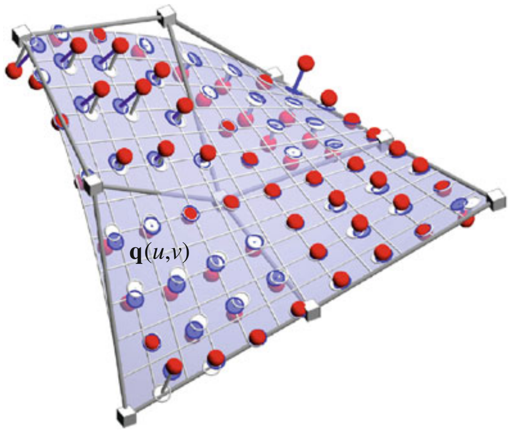


Fig. 3.120 A point cloud of 64 points in 3-space is to be approximated by a (2, 2)-Bézier surface. The input points are highlighted in *red*. The function which is minimized in Algorithm 3.9 is the squared distance sum between the input points (*red*) and the points on the approximating surface belonging to the parameter pairs (s_α, t_α); the latter are marked as white disks. Mind that these points—in general—are not the pedal points on the approximating surface (marked as *blue* disks)

Note that the Eqs. (3.269) and (3.270) are written in *condensed notation*. They have to be interpreted separately for each component of the vectors $\mathbf{a}_{i,j}$ and $\mathbf{c}_\alpha$!

Algorithm 3.9. A tensor product surface fitting a parameterized set of scattered points. *Let $\mu + 1$ points $\mathbf{c}_\alpha$, $\alpha = 0, \ldots, \mu$ in 3-space and corresponding parameter pairs (s_α, t_α), $\alpha = 0, \ldots, \mu$ be given. Moreover, let $F_i(u)$, $i = 0, \ldots, m$ and $G_j(v)$, $j = 0, \ldots, n$ be two sets of basic functions defining a class of tensor product surfaces.*

After having defined the matrix $\mathbf{P}$ *and the arrays* $\mathbf{a}$, $\mathbf{c}$ *according to (3.267) and (3.268) the control net* $\{\mathbf{a}_{i,j}\}$ *of the tensor product surface*

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{a}_{i,j}$$

that minimizes the squared error sum (3.266) can be computed via (3.270) if only the matrix $\mathbf{P}^\top \cdot \mathbf{P}$ *is invertible.*

Figure 3.120 shows a cloud of 64 points and an approximating surface which has been computed according to Algorithm 3.9. In this case the basic functions are the Bernstein polynomials; hence the emerging approximating surface is a Bézier surface. The degree $m = n = 2$ of the approximating Bézier patch has been chosen pretty low for this example in order to visualize the distances between the input points and the approximating surface .

(c) Parameter Correction

We have determined suitable parameter pairs (s_α, t_α) for each point $\mathbf{c}_\alpha$ of the given point cloud (see step (a), on p. 221). This parameter estimate was pretty rough and also widely accidental. If we use those parameter pairs as input in Algorithm 3.9 the error distances $\|\mathbf{q}(s_\alpha, t_\alpha) - \mathbf{c}_\alpha\|$ are not measured on the normals of the resulting tensor product surface

$$\mathbf{q}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} F_i(u) \cdot G_j(v) \cdot \mathbf{a}_{i,j}.$$

This may compromise the result.

To improve the overall result it is helpful to adjust the suggested parameterization subsequently in the following way.[55]

Like in the univariate case the idea is to replace the parameter pairs s_α, t_α by new ones $s_\alpha^* = s_\alpha + \sigma_\alpha$, $t_\alpha^* = t_\alpha + \tau_\alpha$ such that the point $\mathbf{q}(s_\alpha^*, t_\alpha^*)$ lies sufficiently close to the respective pedal point $\mathbf{l}_\alpha$. The values $\sigma_\alpha, \tau_\alpha$ have to be computed by minimizing the squared distance function

$$d_\alpha(u, v) := \|\mathbf{c}_\alpha - \mathbf{q}(u, v)\|^2$$

for each $\alpha = 0, \ldots, \mu$ where $u = s_\alpha + \sigma_\alpha$ and $v = t_\alpha + \tau_\alpha$. The first partial derivatives of the function $d_\alpha(u, v)$ with respect to u and v have to vanish at the minimum, i.e.,[56]

$$f_\alpha(u, v) := d_{\alpha,u}(u, v) = \langle \mathbf{c}_\alpha - \mathbf{q}(u, v), \mathbf{q}_u(u, v) \rangle = 0,$$
$$g_\alpha(u, v) := d_{\alpha,v}(u, v) = \langle \mathbf{c}_\alpha - \mathbf{q}(u, v), \mathbf{q}_v(u, v) \rangle = 0.$$

We expand the functions $f_\alpha(u, v)$ and $g_\alpha(u, v)$ in Taylor series at (s_α, t_α) and—omitting the higher order terms—we resort to the first order Taylor series approach

$$\begin{bmatrix} f_\alpha(u, v) \\ g_\alpha(u, v) \end{bmatrix} = \begin{bmatrix} f_\alpha(s_\alpha + \sigma_\alpha, t_\alpha + \tau_\alpha) \\ g_\alpha(s_\alpha + \sigma_\alpha, t_\alpha + \tau_\alpha) \end{bmatrix} \sim \begin{bmatrix} f_\alpha(s_\alpha, t_\alpha) \\ g_\alpha(s_\alpha, t_\alpha) \end{bmatrix} + \mathbf{J} \cdot \begin{bmatrix} \sigma_\alpha \\ \tau_\alpha \end{bmatrix} \tag{3.271}$$

where $\mathbf{J}$ is the Jacobian

$$\mathbf{J} := \begin{bmatrix} f_{\alpha,u}(s_\alpha, t_\alpha) & f_{\alpha,v}(s_\alpha, t_\alpha) \\ g_{\alpha,u}(s_\alpha, t_\alpha) & g_{\alpha,v}(s_\alpha, t_\alpha) \end{bmatrix}.$$

The entries of $\mathbf{J}$ are

[55] Here we adapt the method for parameter correction which we have already applied in the univariate case; see Sect. 3.6, p. 136.

[56] Subscribed u's and v's indicate differentiation with respect to to u and v.

$$f_{\alpha,u}(s_\alpha, t_\alpha) = -\|\mathbf{q}_u(s_\alpha, t_\alpha)\|^2 + \langle \mathbf{c}_\alpha - \mathbf{q}(s_\alpha, t_\alpha), \mathbf{q}_{u,u}(s_\alpha, t_\alpha)\rangle,$$

$$f_{\alpha,v}(s_\alpha, t_\alpha) = -\langle \mathbf{q}_u(s_\alpha, t_\alpha), \mathbf{q}_v(s_\alpha, t_\alpha)\rangle + \langle \mathbf{c}_\alpha - \mathbf{q}(s_\alpha, t_\alpha), \mathbf{q}_{u,v}(s_\alpha, t_\alpha)\rangle,$$

$$g_{\alpha,u}(s_\alpha, t_\alpha) = f_{\alpha,v}(s_\alpha, t_\alpha)$$

$$g_{\alpha,v}(s_\alpha, t_\alpha) = -\|\mathbf{q}_v(s_\alpha, t_\alpha)\|^2 + \langle \mathbf{c}_\alpha - \mathbf{q}(s_\alpha, t_\alpha), \mathbf{q}_{v,v}(s_\alpha, t_\alpha)\rangle.$$

Due to (3.271) the desired values σ_α and τ_α can then be computed via

$$\begin{bmatrix} \sigma_\alpha \\ \tau_\alpha \end{bmatrix} = -\mathbf{J}^{-1} \cdot \begin{bmatrix} f_\alpha(s_\alpha, t_\alpha) \\ g_\alpha(s_\alpha, t_\alpha) \end{bmatrix}. \tag{3.272}$$

The results σ_α and τ_α are the respective parameter correction values. This is the core point of an iterative procedure:

Algorithm 3.10. Parameter correction for approximation surfaces. *To given data points $\mathbf{c}_\alpha$ and appropriately chosen respective parameter pairs s_α, t_α an approximating surface $\mathbf{q}(u, v)$ is computed (Algorithm 3.9). To improve the quality of $\mathbf{q}(u, v)$ take the following steps:*

1. *Replace the parameter pairs s_α, t_α by the new pairs*

$$s_\alpha^* := s_\alpha + \sigma_\alpha,$$
$$t_\alpha^* := t_\alpha + \tau_\alpha$$

   where $\sigma_\alpha, \tau_\alpha$ are computed via (3.272).
2. *Compute a new approximation surface $\mathbf{q}^*(u, v)$ to the parameter pairs s_α^*, t_α^* according to Algorithm 3.9.*
3. *If necessary repeat steps 1. and 2.*

Figure 3.121 has been created with the same input and the same degree (2, 2) of the approximating surface as Fig. 3.120. However, six rounds of parameter correction have considerably improved the result. Each round of parameter correction has been followed by a complete recalculation of the approximating surface.

In Fig. 3.122 the input data are the same, whilst the degree of the approximating Bézier patch has been raised to $m = n = 3$ which still improves the quality of the result. Of course, we did not miss out parameter correction.

Each round of parameter correction bears some significant influence on the shape of the approximation surface and on the overall quality of the final result.

3.11 Triangular Bézier Patches

Four-sided patches—especially four-sided tensor product patches—are the prevalent type of freeform surfaces. In some cases, however, geometric circumstances require

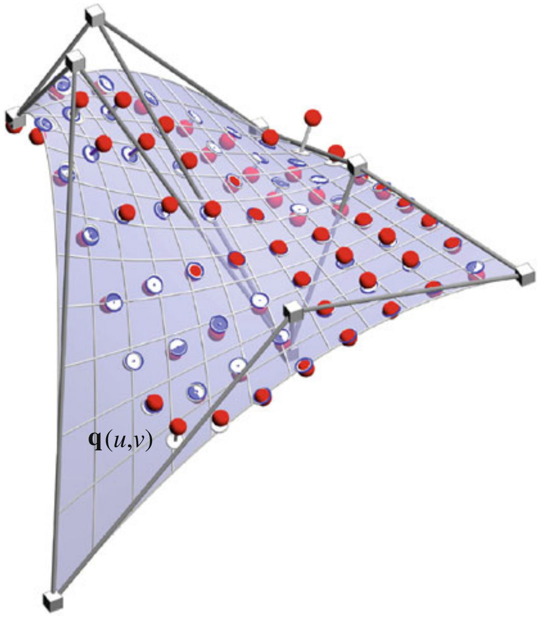


Fig. 3.121 The same set of input points as in Fig. 3.120, again approximated by a (2, 2)-Bézier surface. Here, six rounds of parameter correction according to Algorithm 3.10 have been employed. As a consequence of iterative parameter correction the *white* disks and the *blue* disks are virtually identical all over the surface

triangular patches. On an odd occasion it may be an option to tweak a rectangular patch into triangular shape (see also [7], pp. 289). In the majority of cases, however, a genuine tool for triangular patches will be the better approach which is exactly what we develop in this section. The key instrument for that purpose are barycentric coordinates.

Barycentric Coordinates

A straight line l determined by the two points $\mathbf{u}, \mathbf{v}$ can be parameterized via $\mathbf{p}(u, v) = u \cdot \mathbf{u} + v \cdot \mathbf{v}$ with $u + v = 1$ (cf. Example 3.4, p. 70). Analogously, a plane ε determined by the three non-collinear points $\mathbf{u}, \mathbf{v}, \mathbf{w}$ has a parametric representation of the form $\mathbf{q}(u, v, w) = u \cdot \mathbf{u} + v \cdot \mathbf{v} + w \cdot \mathbf{w}$ with $u + v + w = 1$ (cf. Example 3.12, p. 145).

Definition 3.66. Barycentric coordinates. Let $\mathbf{u}, \mathbf{v}, \mathbf{w}$ be a triangle and let $\mathbf{p}$ be a point in the plane ε determined by $\mathbf{u}, \mathbf{v}, \mathbf{w}$, i.e.,

$$\mathbf{p} = u \cdot \mathbf{u} + v \cdot \mathbf{v} + w \cdot \mathbf{w} \tag{3.273}$$

with

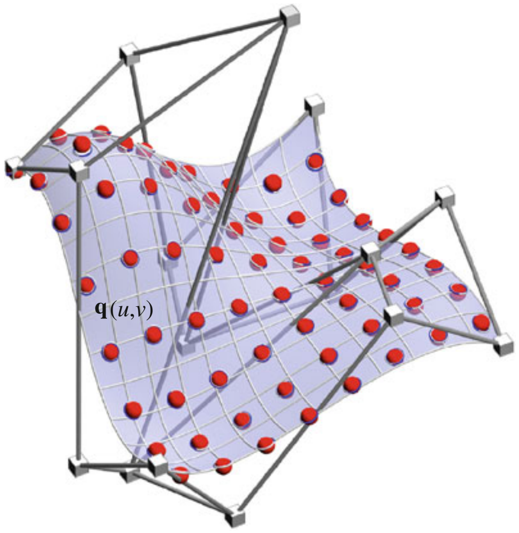


Fig. 3.122 The same set of input points as before, this time approximated by a (3, 3)-Bézier surface. The iterative process of parameter correction (Algorithm 3.10) has been applied 4 times. Bivariate approximation with sufficiently high degree (m, n) together with parameter correction delivers convincingly

$$u + v + w = 1. \tag{3.274}$$

Then (u, v, w) are called the *barycentric coordinates* of $\mathbf{p}$ with respect to $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$. The triangle $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ is called *coordinate triangle* or *barycentric coordinate system* in ε. The point $\mathbf{p}$ is called an *affine combination* of the points $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$.

We compile some basic properties of barycentric coordinate systems (Fig. 3.123):

1. The barycentric coordinates of the coordinate triangle's vertices $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ are $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, respectively.
2. The condition $u = u_0 = const.$ determines a straight line, parallel to the edge $\mathbf{vw}$ called *u-parameter line* or simply *u-line*. We have to keep in mind that along a u-line the parameters v and w vary—albeit under the norming condition $u_0 + v + w = 1$.
3. A special case of a u-line is the straight line $u = u_0 = 0$ which is the edge $\mathbf{vw}$ of the coordinate triangle.
4. In the same way we can define *v-lines* and *w-lines*.
5. Through every point $\mathbf{p}(u_0, v_0, w_0)$ there exists one u-line $u = u_0$, one v-line $v = v_0$ and one w-line $w = w_0$.
6. Coordinates u, v, w with $0 < u, v, w < 1$ refer to points inside the coordinate triangle.
7. We refer to Remark 3.7, p. 86, where we described the concept and the consequences of the partition of unity. In the case of barycentric coordinates condition

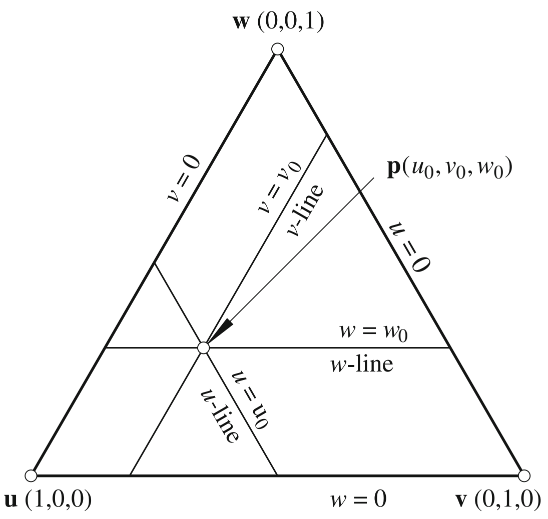


Fig. 3.123 Barycentric coordinates of points in the plane. A triangle **u**, **v**, **w** in the plane determines a barycentric coordinate system. Each point is assigned a triple (u, v, w) of coordinates, subject to $u + v + w = 1$. A u-parameter *line* is determined by the condition $u = u_0$. Two other parameter *lines* arise from $v = v_0$ and $w = w_0$. All three parameter *lines* meet at the point $\mathbf{p}(u_0, v_0, w_0)$

(3.274) claims that the coordinates (u, v, w) in (3.273) are a partition of unity. As a consequence, the point $\mathbf{p} = u \cdot \mathbf{u} + v \cdot \mathbf{v} + w \cdot \mathbf{w}$ is affinely connected with the coordinate triangle.

Bivariate Bernstein Polynomials

Let us recall that the (univariate) Bernstein polynomials $B_{i,n}(t)$ can also be interpreted as the coefficients in the binomial expansion of $1 = (t + (1 - t))^n$:

$$1 = (t + (1 - t))^n = \sum_{i=0}^{n} \binom{n}{i} \cdot t^i \cdot (1 - t)^{n-i} = \sum_{i=0}^{n} B_{i,n}(t)$$

Substituting $u = t$ and $v = 1 - t$ we can rewrite $B_{i,n}(t)$ as

$$B_{i,n}(t) = \frac{n!}{i!j!} \cdot u^i \cdot v^j \quad \text{with} \quad i + j = n.$$

We can easily generalize this to the bivariate case:

Definition 3.67. Bivariate Bernstein polynomials. Let $u, v, w \in \mathbb{R}$ with $u + v + w = 1$; then the functions

$$B_{i,j,k;n}(u, v, w) := \frac{n!}{i!j!k!} \cdot u^i \cdot v^j \cdot w^k, \quad i + j + k = n \tag{3.275}$$

are called *bivariate Bernstein polynomials of degree n* or *Bernstein polynomials of degree n on a triangular domain.*

Note that the Bernstein polynomials (3.275) are in fact only *bivariate* functions because the three parameters u, v, w are linked via $u + v + w = 1$. Moreover it is obvious that they are the summands in the trinomial expansion of $1 = (u + v + w)^n$:

$$1 = (u + v + w)^n = \sum_{\substack{i,j,k=0 \\ i+j+k=n}}^{n} \frac{n!}{i!j!k!} \cdot u^i \cdot v^j \cdot w^k = \sum_{\substack{i,j,k=0 \\ i+j+k=n}}^{n} B_{i,j,k;n}(u, v, w) \quad (3.276)$$

Hence, they are a partition of unity.

Owing to Remark 3.4, p. 69 the dimension of the space $\mathbb{K}_n[u, v]$ of bivariate polynomials of order $\leq n$ is $(n + 1) \cdot (n + 2)/2$. We have:

Proposition 3.21 Basis property of bivariate Bernstein polynomials. *The set* $\{B_{i,j,k;n}(u, v, 1 - u - v) \mid i + j + k = n, \ 0 \leq i, j, k \leq n\}$ *of bivariate Bernstein polynomials of degree n is a basis of the vector space* $\mathbb{K}_n[u, v]$ *of all bivariate polynomials of degree* $\leq n$.

This proposition states that every bivariate polynomial $f(u, v)$ of degree $\leq n$ can be uniquely expressed as a linear combination of the bivariate Bernstein polynomials of degree n, i.e.,

$$f(u, v) = \sum_{\substack{i,j,k=0 \\ i+j+k=n}}^{n} a_{i,j,k} \cdot B_{i,j,k;n}(u, v, 1 - u - v)$$

with uniquely determined coefficients $a_{i,j,k}$. We skip the detailed proof of Proposition 3.21 which splits into two parts: First the linear independence of the bivariate Bernstein polynomials $B_{i,j,k;n}(u, v, 1 - u - v)$ has to be verified. Then the fact that the number $(n + 1) \cdot (n + 2)/2$ of these polynomials coincides with the dimension of $\mathbb{K}_n[u, v]$, finishes the proof.

Barycentric coordinates together with bivariate Bernstein polynomials are the perfect ingredients for the definition of triangular Bézier patches.

Triangular Bézier Patches

Definition 3.68. Triangular net. A *triangular net of order n* or simply *triangular net* (Fig. 3.124) is a set of $(n + 1) \cdot (n + 2)/2$ points $\mathbf{a}_{i,j,k}$, $i + j + k = n$ together with the following three sets of polylines (threads):

$$\begin{array}{lll}
\mathbf{a}_{i_0,0,n-i_0}, \mathbf{a}_{i_0,1,n-i_0-1}, \ldots, \mathbf{a}_{i_0,n-i_0,0}, & i_0 = const, & i_0 = 0, \ldots, n-1 \\
\mathbf{a}_{0,j_0,n-j_0}, \mathbf{a}_{1,j_0,n-j_0-1}, \ldots, \mathbf{a}_{n-j_0,j_0,0}, & j_0 = const, & j_0 = 0, \ldots, n-1 \\
\mathbf{a}_{0,n-k_0,k_0}, \mathbf{a}_{1,n-k_0-1,k_0}, \ldots, \mathbf{a}_{n-k_0,0,k_0}, & k_0 = const, & k_0 = 0, \ldots, n-1
\end{array}$$

Definition 3.69. Triangular Bézier patch. Let a triangular net $\{\mathbf{a}_{i,j,j}\}$, $i, j, k = 0, \ldots, n, i + j + k = n$ be given. The corresponding *triangular Bézier surface* or *triangular Bézier patch of degree n* (Fig. 3.124) is the surface defined by the parameterization

$$\mathbf{q}(u, v) = \sum_{\substack{i,j,k=0 \\ i+j+k=n}}^{n} B_{i,j,k;n}(u, v, w) \cdot \mathbf{a}_{i,j,k} \tag{3.277}$$

with $(u, v, w) \in [0, 1] \times [0, 1] \times [0, 1]$ and $u + v + w = 1$. The given triangular net $\{\mathbf{a}_{i,j,j}\}$ is called the *control net* of the patch.

Triangular Bézier patches have properties analogue to those of their tensor product counterparts. We mention a few of the most important (Fig. 3.124):

1. A triangular Bézier patch and its control net have the same corner points: $\mathbf{q}(1, 0) = \mathbf{a}_{n,0,0}$, $\mathbf{q}(0, 1) = \mathbf{a}_{0,n,0}$, $\mathbf{q}(0, 0) = \mathbf{a}_{0,0,n}$,
2. The tangent plane $\tau_{1,0}$ to the triangular Bézier patch at $\mathbf{q}(1, 0) = \mathbf{a}_{n,0,0}$ is spanned by the points $\mathbf{a}_{n,0,0}$, $\mathbf{a}_{n-1,1,0}$, $\mathbf{a}_{n-1,0,1}$. Analogue statements hold for the two other corners $\mathbf{q}(0, 1) = \mathbf{a}_{0,n,0}$, $\mathbf{q}(0, 0) = \mathbf{a}_{0,0,n}$: The tangent plane in a corner point of the Bézier patch is spanned by the two threads of the control net through this corner point.
3. As the bivariate Bernstein polynomials of a given degree are a partition of unity (3.276) we have: A triangular Bézier patch is invariantly connected with its control net $\mathbf{a}_{i,j,k}$ with respect to affine transformations.
4. A triangular Bézier patch is completely contained in the convex hull of its control net (*convex-hull property*).

The de Casteljau Algorithm for Triangular Patches

Evaluating the bivariate Bernstein polynomials for a triangular Bézier patch according to (3.275) can be avoided in the first place. In much the same way as for standard tensor product Bézier surfaces we can set up the de Casteljau algorithm for the triangular case. It is based on repeated affine combinations of three points. This is a good example to see how smoothly most of the procedures from the standard Bézier patches adapt to the triangular case.

Algorithm 3.11. The de Casteljau algorithm for triangular patches. *(Fig. 3.125) Let the control net $\{\mathbf{a}_{i,j,k}\}$ of a triangular Bézier patch of degree n and a triple of parameters $(u_0, v_0, w_0 = 1 - u_0 - v_0)$ be given. To compute the point $\mathbf{q}(u_0, v_0)$*

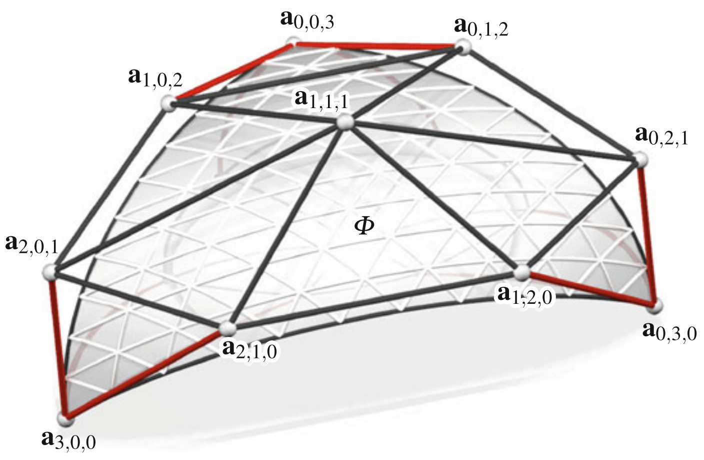


Fig. 3.124 A triangular Bézier patch Φ of degree $n = 3$ and its control net $\{\mathbf{a}_{i,j,k}\}$. The two threads (*red lines*) meeting at a corner of the triangular patch span the tangent plane in that corner point. The figure also shows a couple of parameter *lines* including the boundary curves of the patch. The parameter *lines* form a 3-fold weave

of the corresponding triangular Bézier patch (3.277) one has to take the following steps:

1. *Put* $\mathbf{c}^0_{i,j,k} := \mathbf{a}_{i,j,k}$ *and recursively compute the points*

$$\mathbf{c}^r_{i,j,k} := u_0 \cdot \mathbf{c}^{r-1}_{i+1,j,k} + v_0 \cdot \mathbf{c}^{r-1}_{i,j+1,k} + w_0 \cdot \mathbf{c}^{r-1}_{i,j,k+1} \tag{3.278}$$

   for $r = 1, \ldots, n$ *and* $i, j, k = 0 \ldots, n;\ i + j + k = n - r$.
2. *The point* $\mathbf{c}^n_{0,0,0}$ *is the desired point:*

$$\mathbf{q}(u_0, v_0) = \mathbf{c}^n_{0,0,0}$$

A couple of further procedures (see Sect. 3.8) for (rectangular) tensor product patches like

- continuation of a Bézier patch (see Construction 3.6, p. 187)
- modeling the continuation of a Bézier patch (see Construction 3.7, p. 188)
- degree elevation (see p. 186).

can be adapted to triangular Bézier patches in a straightforward way. For more details we refer the reader to ([7], p. 293).

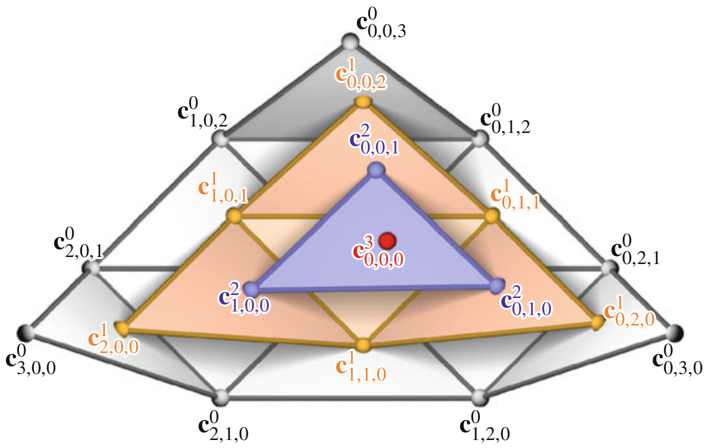


Fig. 3.125 The triangular de Casteljau scheme, applied to a given Bézier patch of degree $n = 3$ and a given parameter triple ($u_0 = 0.25, v_0 = 0.35, w_0 = 1 - u_0 - v_0 = 0.4$). The original control net (*grey*, superscript 0) consists of $n^2 = 9$ triangles, namely $(n + 1) \cdot n/2 = 6$ *upright triangles* and $n \cdot (n - 1)/2 = 3$ *downright triangles*. The corners of every upright triangle within the net are affinely combined to a new point according to (278), $r = 1$. We arrive at the net of the next generation (*orange*, superscript 1). In the same way we take each *upright* triangle of that net and arrive at a point of the following generation (*blue* net, superscript 2) and so forth. Finally we get a net which consists of one single point (here $\mathbf{c}^3_{0,0,0} = \mathbf{q}(u_0, v_0)$) which is the desired point of the triangular Bézier patch, relating to the given parameter triple (u_0, v_0, v_0). We can even point out that the triangle of the last but one generation (*blue* triangle) marks the tangent plane of the considered point

3.12 Tensor Product Volumes

In Sects. 3.4 and 3.8 we have been developing freeform curves and their bivariate counterparts, tensor product surfaces. Basically all the methods and algorithms can as well be extended to multivariate objects. In engineering applications the most common is the trivariate case. It deals with *tensor product volumes*.[57]

Definition 3.70. Tensor product volume. Let $(l + 1) \times (m + 1) \times (n + 1)$ points $\{\mathbf{a}_{i,j,k}\}$, $i = 0, \ldots, l$, $j = 0, \ldots, m$, $k = 0, \ldots n$ in 3-space be given: We call this array of points a *control grid*. We further use three sets of functions $F_i(u)$, $i = 0, \ldots, l$, $G_j(v)$, $j = 0, \ldots, m$ and $H_k(w)$, $k = 0, \ldots, n$ of parameters u, v, and w, respectively. The trivariate vector function

$$\mathbf{q}(u, v, w) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} F_i(u) \cdot G_j(v) \cdot H_k(w) \cdot \mathbf{a}_{i,j,k}, \qquad (3.279)$$

$$u \in [u_0, u_1], \quad v \in [v_0, v_1], \quad w \in [w_0, w_1]$$

[57] For modeling with solids see also Sect. 4.3, p. 276.

represents the *tensor product volume* to the given *control grid* $\mathbf{a}_{i,j,k}$ relating to the families of basis functions $F_i(u)$, $G_j(v)$ and $H_k(w)$.

A tensor product volume is uniquely determined by its control grid and the three function families $F_i(u)$, $G_j(v)$, $H_k(w)$. It does not depend on the chosen coordinate system if each of the families $F_i(u)$, $G_j(v)$ and $H_k(w)$ fulfills the *partition of unity*-condition:

$$\sum_{i=0}^{l} F_i(u) \equiv \sum_{j=0}^{m} G_j(v) \equiv \sum_{k=0}^{n} H_k(w) \equiv 1 \tag{3.280}$$

Here we do not want to particularize all conceivable types and options of tensor product volumes. We rather confine our considerations to Bézier volumes:

Definition 3.71. Bézier volume. A *Bézier volume of degree* (l, m, n) is a tensor product volume (3.279) where the functions $F_i(u)$, $G_j(v)$, and $H_k(w)$ are the Bernstein polynomials of degree l, m and n in u, v and w, respectively:

$$\mathbf{q}(u, v, w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} B_{i,l}(u) \cdot B_{j,m}(v) \cdot B_{k,n}(w) \cdot \mathbf{a}_{i,j,k},$$
$$(u, v, w) \in [0, 1] \times [0, 1], \times[0, 1] \tag{3.281}$$

Of course, it is difficult to display a volume in itself, so we have to gratefully resort to particular surfaces and curves contained in it in a natural way:

Definition 3.72. Parameter surfaces and parameter lines of a Bézier volume. Let a Bézier volume according to Definition 3.71 be given.

1. The Bézier surface defined by the condition

$$u \equiv u_0 = const. \tag{3.282}$$

   is called a (v, w)-parameter surface of the Bézier volume. In the same way the conditions $v \equiv v_0$ and $w \equiv w_0$ lead to a (u, w)- and a (u, v)-parameter surface, respectively.
2. The Bézier curve defined by the condition

$$u \equiv u_0 \text{ and } v \equiv v_0 \tag{3.283}$$

   is called a *w-parameter line* (or a *w-parameter curve*) of the Bézier volume. Similarly, the conditions $w \equiv w_0$ and $u \equiv u_0$ as well as $u \equiv u_0$ and $v \equiv v_0$ lead to a v- and a w-parameter line, respectively.

In much the same way as surfaces are often displayed by means of parameter lines, the tensor product volumes can be illustrated by means of some of their parameter surfaces and parameter lines (Fig. 3.126).

Definition 3.73. Two Bézier volumes $\mathbf{p}(u, v, w) = \mathbf{q}(u, v, w)$ are called C^r-continuous in a point $\mathbf{p}(u_0, v_0, w_0) = \mathbf{q}(u_0, v_0, w_0)$ if all partial derivatives up to order r coincide at this point.

Two Bézier volumes $\mathbf{p}(u, v, w) = \mathbf{q}(u, v, w)$ are called C^r-continuous all along a parameter curve or a parameter surface (or in any other parameter domain) if all partial derivatives up to order r coincide for all respective parameter values (u, v, w) within that domain.

There are a lot of concepts and algorithms for Bézier surfaces (see Sect. 3.8.1, p. 182) which can promptly be inherited to Bézier volumes. We mention some examples:

1. The de Casteljau algorithm for Bézier surfaces (cf. Algorithm 3.5, p. 184) can be adapted to Bézier volumes right away.
2. Tensor product volumes can be continued with respect to any of their three parameters.
3. The continuation via the de Casteljau algorithm delivers a control grid of a continuation volume.

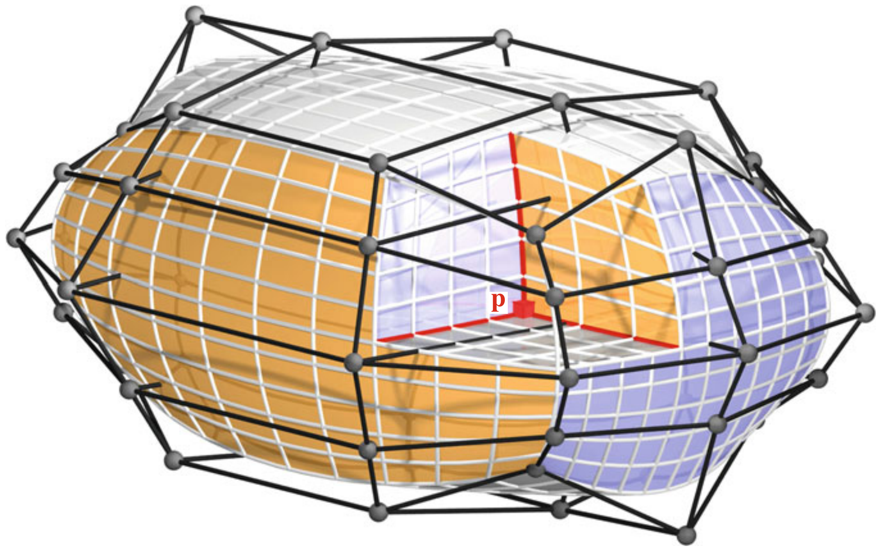


Fig. 3.126 A Bézier volume of degree $(k, m, n) = (2, 3, 4)$ and its control grid. The boundary surfaces $u = 1$ (*orange*), $v = 1$ (*blue*) and $w = 1$ (*grey*) are visible as are a couple of parameter lines (*white*). The point $\mathbf{p} = \mathbf{q}(u_0, v_0, w_0)$ to the parameter triple $(u_0, v_0, w_0) = (0.6, 0.7, 0.5)$ is marked in *red*. The (v, w)-parameter surface $u = u_0$ and the other two parameter surfaces through $\mathbf{p}$ are also displayed

4. The partial derivatives up to order r in a boundary point $u_0 = 0$ only depend on grid points contained in the first $r + 1$ layers $\{\mathbf{a}_{i,j,k}\}$, $i \leq r$.
5. We regard a Bézier volume and its continuation in u-direction with respect to a design parameter $u_0 > 1$. The resulting continuation grid $\{\mathbf{b}_{i,j,k}\}$ obtained via the de Casteljau algorithm defines a Bézier volume which continues the original volume C^∞-continuously in the sense of Remark 3.23, p. 187. Modifying the grid $\{\mathbf{b}_{i,j,k}\}$ by replacing some of its control points will still yield C^r-continuity along the boundary surface (transition surface) as long as the first $r + 1$ layers $\{\mathbf{b}_{i,j,k}\}$, $i = 0, \ldots, r$ of the continuation grid remain unaltered.

This short record on Bézier volumes, their continuation and modification is meant to convey that the basic methods can easily be transferred from the bivariate to the trivariate case. For more details on multivariate methods we refer the reader to ([7], pp. 462–504) and the references cited therein.

3.13 Example: Side Window Kinematics

In this section we discuss a single engineering task which is meant to show a number of geometrical and technical aspects occurring in one job. For more detailed information see [22].

In Sect. 3.7.13, Definition 3.57, p. 174, we have been dealing with surfaces and curves which can be *moved in themselves* which is—by the way—the key property of a car side window surface, and that is for a couple of reasons:

- The window pane has to be movable through the sealing slit along the daylight curve.
- Moreover, in the course of its motion the side window has to vanish in the door body, where the conditions are rather crammed.

Both reasons would highly recommend movability in itself. But there is one compelling argument which is even harder to ignore:

- If the window motion is stopped at some moment during its action, the window—or the part of it which is still visible at that instant—still has to lie on the same surface as formerly in its closed position. Any deviation from that surface would sorely be noticed in the reflected light on the shiny car body shell.

In order to be movable in itself, a surface has to be a helical surface or—as special cases of helical surfaces—a cylinder or a surface of revolution (cf. Sect. 3.7.13, p. 174).

Certainly, the outer shell of a car is mainly driven by styling. And so is the geometry of a car side window. While the stylist creates the window surface he or she may be unaware of movability in itself. It so may happen that the stylist's first suggestion of a side window does not allow for this severe geometric constraint. So, if we put it bluntly: The engineer's task to design the mechanism moving the side window is all but impossible.

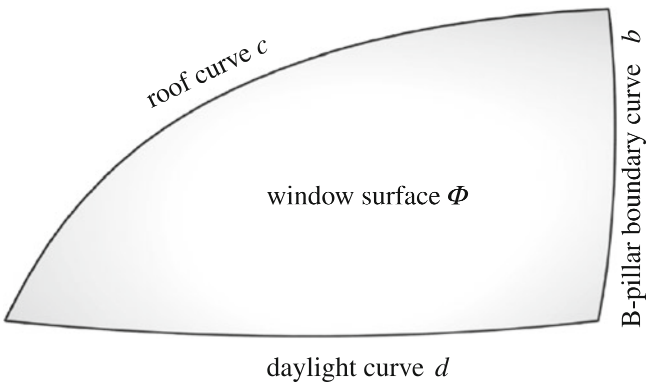


Fig. 3.127 Car side window with its boundary curves: the roof curve c, the B-pillar boundary curve b and the daylight curve d

Figure 3.127 shows the shape of a car side window with its three boundary curves: the B-pillar boundary curve b, the roof curve c and the daylight curve d.

3.13.1 The Appropriate Screw Motion to a Given Surface

Let $\varPhi$ be the surface representing the door window pane as it has been designed by the stylist. Even though $\varPhi$ is—in all likelihood—not a helical surface we can still use the line geometry tools introduced in Sect. 3.1.6, p. 62, to compute a suitable screw motion M fitting $\varPhi$.

We select a number of normals g_i of $\varPhi$ and determine their Plücker vectors $\mathbf{g}_i = [g_{i,1}, g_{i,2}, g_{i,3}]^\top, \overline{\mathbf{g}}_i = [\overline{g}_{i,1}, \overline{g}_{i,2}, \overline{g}_{i,3}]^\top$ (cf. Eq. (3.11), p. 62) arriving at a set of points $G_i \ldots g_{i,1}, g_{i,2}, g_{i,3}, \overline{g}_{i,1}, \overline{g}_{i,2}, \overline{g}_{i,3}$ in 6-space $\mathbb{R}^6$ (see Fig. 3.128).

We already know that the points G_i would lie in a hyperplane H through the origin O of $\mathbb{R}^6$ in case of $\varPhi$ being a helical surface. This is because the normals g_i to a helical surface $\varPhi$ also belong to the linear complex of the corresponding screw motion (cf. p. 64)!

If $\varPhi$ is not actually a helical surface we still can compute the point set $\{G_i\}$ to the normals $\{g_i\}$ of $\varPhi$. In that case they will not be contained in a hyperplane $H \subset \mathbb{R}^6$ through O. Instead, we can determine an *approximating* hyperplane H through O. We have arrived at the standard task of plane fitting, though in a higher dimensional space. The methods presented in Sect. 3.10.1, p. 216 can easily be adapted to this case. They were originally introduced in [19].

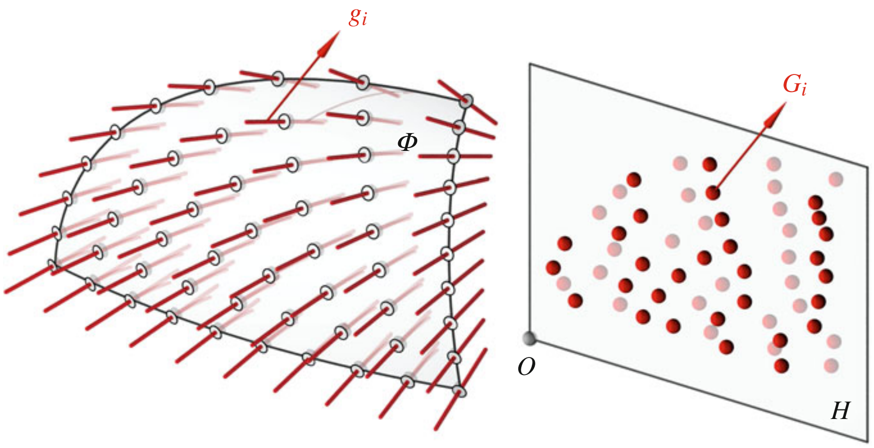


Fig. 3.128 Finding a spatial motion M which is adapted to the given window surface Φ. The input of the optimization job consists of a number of surface normals g_i (*left*, *red*). The optimization boils down to a plane fitting problem of scattered data points G_i, albeit in a higher dimensional space (*right*, symbolical)

Having computed the approximating hyperplane

$$H \ldots \langle \mathbf{v}, \mathbf{g} \rangle + \langle \mathbf{w}, \bar{\mathbf{g}} \rangle = 0$$

in $\mathbb{R}^6$ through O we can easily find the corresponding screw motion M according to (3.15), (3.16) and (3.17), p. 65.

Remark 3.27 The motion M which we have constructed, is adapted to the given window surface Φ. In this very example we additionally have to demand that the B-pillar boundary curve b is a trajectory of the motion M. This further constraint can easily be imposed without problems (for details see [22]).

3.13.2 Constructing an Ideal Side Window Surface

If we had started with a helical surface Φ in the first place the screw motion M constructed above would move the surface Φ perfectly in itself.

Yet, for now we have to put up with the stylist's window pane Φ which—in general—is NOT movable in itself. We have computed the screw motion M which is adapted to Φ in the best possible way.

If we subject the stylist's surface Φ to the motion M we certainly have to be aware that deflections along the seals might be inevitable. It will be alright as long as these deflections do not exceed certain limits.

In order to assess these deflections it would be convenient to compare Φ with some appropriate *helical* surface Φ^* which,

- subjected to the motion M, does not cause any stress on the sealing,
- lies very close to Φ.

To this avail we take the roof-line[58] c and subject it to the screw motion M. Of course, we arrive at a helical surface. We call Φ^* the *ideal window surface*. If we applied the motion M to Φ^* we would not have to face any deflections at all as Φ^* is moved *in itself* all the way.

As a consequence the distance between stylist's surface Φ and the ideal surface Φ^* measures the deflections of Φ. The maximum distance ε between Φ and Φ^* indicates the maximum stress on the rubber seals. So we can easily compute the amount of stress which we have to be braced for. If that stress ε is well below the given limits (imposed by the sealings manufacturer) we can combine the motion M and the given window surface Φ and go on with the construction of the window lifting mechanism.

However, if the maximum stress ε exceeds the limits there are two options for the engineer and the stylist:

1. Replace the window surface Φ by the ideal surface Φ^*. That modification of the overall design will change the window surface by an extent which nowhere exceeds the value ε.
2. Create a blending surface Φ^{**} between the original suggestion Φ and the ideal surface Φ^* such that the maximum stress stays within some desired limits.

Our geometric considerations have delivered a tool which offers more than one benefit.

- We can find the optimal motion of a car side window lifter mechanism.
- We get reliable information about the quality of the suggested window surface Φ and the amount of deflections which are inherent in the geometry of Φ.
- Last, but not least, we get an ideal side window which can—just in case—be a perfect substitute for the originally suggested window surface. Its distance from the original proposal would nowhere be more than the above-mentioned ε, whilst any problems with sealing stress could be avoided in the first place.

References

1. Gallier, J.: Geometric Methods and Applications. 2nd edn. Springer , New York (2011)
2. Chasles, M.: Note sur les propriétés générales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre. Bull. des Sci. Mathematiques Astronomiques Physiques et Chim. **14**, 321–326 (1830)

[58] Note that what we call the *roofline* is actually the boundary curve of the window towards the roof **plus** its boundary curve towards the A-pillar of the car.

3. Pottmann, H., Wallner, J.: Computational Line Geometry. Springer, Berlin (2000)
4. Blyth, T.S., Robertson, E.F.: Basic Linear Algebra. 2nd edn. Springer, Berlin (2002)
5. Stoker, J.J.: Differential Geometry. Wiley, New York (1969)
6. Walker, R.J.: Algebraic Curves. Springer, New York (1978)
7. Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design. Wellesley, Massachusetts (1993)
8. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design. 2nd edn. Academic Press, Boston (1990)
9. Bernstein, S.: Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités. Commun. Math. Soc. Kharkov (2) **13**, 1–2 (1912)
10. Schuhmaker, L.L.: Spline Functions: Basic Theory. Wiley, New York (1981)
11. De Boor, C.: A Practical Guide to Splines. vol. 27, rev. edn. Applied Mathematical Sciences, Springer (2001)
12. Nürnberger, G.: Approximation by Spline Functions. Springer, Berlin (1989)
13. Haase, J.C.F.: Zur Theorie der ebenen Curven n-ter Ordnung mit $(n-1)(n-2)/2$ Doppel- und Rückkehrpunkten. Math. Ann. **2**, 515–548 (1870)
14. Overhauser, A.: Analytic Definition of Curves and Surfaces by Parabolic Blending. Technical report, Ford Motor Company (1968)
15. Röschel, O.: An interpolation subspline scheme related to B-spline techniques. In: B. Werner (ed.) Computer Graphics International '97, pp. 131–136. IEEE Computer Society Press, Los Alamitos (1997)
16. Beresin, I.S., Shidkow, N.P.: Numerische Methoden 1, Hochschulbücher für Mathematik, vol. 70. VEB Deutscher Verlag der Wissenschaften, Berlin (1970)
17. Usmani, R.A.: Inversion of Jacobi's tridiagonal matrix. Comput. Math. Appl. **27**, 59–66 (1994)
18. Ascher, U.M., Greif, C.: A first course in numerical methods, Computational Science and Engineering. In: Society for Industrial and Applied Mathematics (SIAM), vol. 7. Philadelphia, USA (2011)
19. Pottmann, H., Hofer, M., Odehnal, B., Wallner, J.: Line geometry for 3D shape understanding and reconstruction. In: Pajdla, T., Matas, J. (eds.) Computer Vision—ECCV 2004, Part I. Lecture Notes in Computer Science, vol 3021, pp. 297–309. Springer (2004)
20. Odehnal, B., Stachel, H.: The upper talocalcanean join. Technical Report 127, Vienna University of Technology (2004). http://www.geometrie.tuwien.ac.at/odehnal/knochen.pdf
21. Pottmann, H., Randrup, T.: Rotational and helical surface approximation for reverse engineering. Computing **60**, 307–322 (1998)
22. Gfrerrer, A., Lang, J., Harrich, A., Hirz, M., Mayr, J.: Car side window kinematics. Comput. Aided Des. **43**, 410–416 (2011)
23. Harrich, A., Mayr, J., Hirz, M., Rossbacher, P., Lang, J., Gfrerrer, A., Haselwanter, A.: CAD-based synthesis of a window lifter mechanism. In: SAE World Congress, Detroit (2010) doi:10.4271/2010-01-0009
24. Shafarevich, I.: Algebraic Geometry I, II. Springer, New York (1977, 1994)
25. Röschel, O.: Rationale Bézier Schiebflächen. CAD Computergraphik und Konstruktion **13**, 29–33 (1989)
26. Röschel, O.: Kinematic Rational Bézier Patches I, II. Rad YAZU **10**(95–108), 131–138 (1991)
27. Coons, S.: Surfaces for computer aided design. Technical Report VA 22161, MIT, available as AD 663 504 from the National Technical Information Service, Springfield (2001)
28. Ferguson, J.: Multivariable curve interpolation. JACM **2**(2), 221–228 (1964)

Chapter 4
Modeling Techniques in CAD

Computer-aided design in mechanical engineering covers the IT-supported creation of product geometry and the corresponding product structure within a virtual environment. In the automotive industry, three-dimensional product modeling was established in the 1980s, which was significantly supported by the introduction of 3D CAD software. CAD processes focus on the representation of product geometry and topology and include all required geometry modeling operations and structure-related organizational procedures. The geometrical representation of product models can require the creation of complex curves, surfaces and solids within a virtual environment (CAD-working space), whereby different technologies of geometry definition, approximation, interpolation and transformations come into use. Besides the creation of geometry models, CAD provides the basis for subsequently performed engineering tasks and model preparation (e.g. meshing operations).

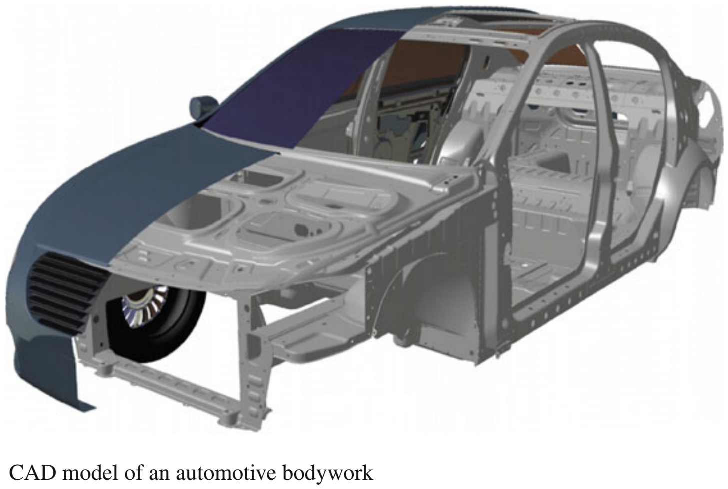


Fig. 4.1 CAD model of an automotive bodywork

In mechanical engineering, geometrical modeling is performed in different ways (e.g. as surface modeling in body-in-white design or as solid modeling for the development of casting components). The integration of geometry information, product structure and additional data (e.g. dimensioning, tolerances, material, masses and moments of inertia, production-related information) significantly enhanced the fields of application and made it necessary to involve CAD models in nearly all of the disciplines in virtual product and production development. In computer-aided design, the product geometry is composed of several basis-elements (i.e. lines, curves, different types of surfaces and volume elements). Depending on the type of product and the desired characteristics of the virtual product model, these elements are combined by applying specific methods and strategies, which are called design rules. In general, complex CAD models consist of numerous components and modules, which are put together and structured in assemblies.

Computer-aided styling (CAS) is a related discipline for the creation of styling-relevant geometry. In contrast to technology-oriented product modeling in CAD-processes, the creation of smooth styling surfaces for vehicle exteriors and interiors places high demands on the quality of the curves and surfaces themselves. Quality criteria include the order and continuation of curves and surface patches, the complexity of surface definition, and the curvature of surfaces and poly-surfaces. Styling surfaces are often based on supporting elements or measured data, such as sketches, point clouds, curves or meshes, which provide the basis for the definition of three-dimensional bent geometrical elements. Because neither product structure nor production related information is considered, CAS provides functionalities purely for geometry creation. Figure 4.1 shows a combination of styling data which define the outline of a car and design data, which represent the technical components of automotive bodywork. Figure 4.2 illustrates automotive exterior surfaces with their segmentations. In this example, several different types of styling-relevant surface patches are used.

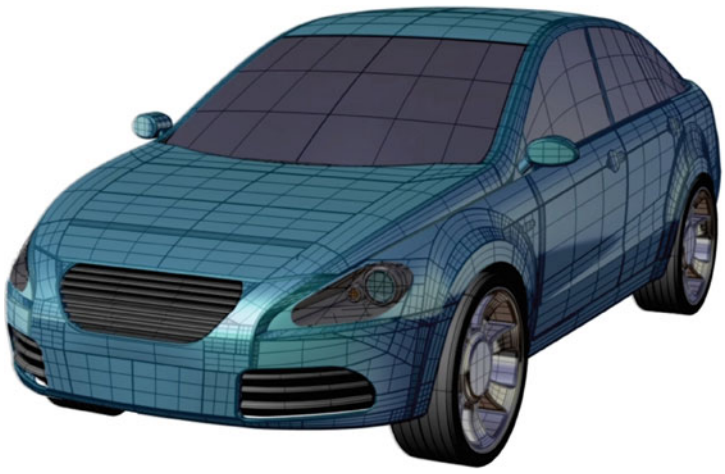


Fig. 4.2 Automotive exterior surfaces with segmentations

The different stages of the development process lead to styling surfaces with different quality levels. These quality levels of surface data are classified as class A, B, or C surfaces. At the beginning of styling development, the concepts are represented with low accuracy as so-called class C-surfaces. These surfaces are used for initial evaluations of the styling concepts and the first engineering-related operations and feasibility studies. In addition, class C surfaces are utilized for concealed areas and components. Class B surfaces are used for hidden components and surfaces, whose appearance is relatively unimportant. Application examples for class B surfaces include the inner areas of body design, the engine compartment, and the under carriage. In general, class C surfaces are characterized by a relatively discontinuous progression, whereby class B surfaces show a tangency continuation (cf. Sect. 3.7.3, p. 148). Finally, class A surfaces are applied for all visually relevant components, such as the visible exterior and interior surfaces. Class A surfaces fulfill high smoothness requirements and are characterized by smooth transitions between individual patches to keep the variation of curvature low. CAS delivers surfaces and boundary conditions for subsequent engineering-based processes, such as vehicle layout and component development. The data transfer between styling software and engineering software is mainly performed via neutral data interfaces, such as IGES, STEP, and JT.

The preparation and maintenance of geometrical data for subsequent calculation and simulation processes are two other important tasks that are highly dependent on CAD. In common development processes, geometrical data delivered by CAD models serve as a basis for the generation of specific geometry models suitable for the projected simulation. This can involve the generation of approximated geometry models by triangulation (surface or volume meshes) or the derivation of geometrically simplified simulation models (e.g. beam models). In both approaches, process-specific geometry models are derived from existing CAD models. Some simulation software supports the development of geometry models for a specific simulation application (e.g. the conceptual creation of the vehicle body structure). In such cases, CAD methods are combined with CAE-related functionalities to enable the efficient generation of simplified product models for subsequent FE-simulation (e.g. [1]). Figure 4.3 shows an example of a surface mesh of an automotive bodywork that is used for finite-element-based crash simulation.

CAD systems use computational graphics algorithms to generate near real-life objects in a virtual environment. The main features of virtual product representation were developed in the 1980s. They provided the foundation for the fast-growing expansion of computer-aided applications in research and development. Currently, a variety of software solutions for computational product generation are available on the market, but these solutions generally use similar algorithms for the representation of geometries. Depending on the specific type of application, styling software enables the creation of smooth surfaces for product shaping. Design software, on the other hand, offers functionalities for engineering-focused product creation processes, and computation software facilitates the derivation of computation models for further simulation procedures. In the automotive industry, there are three main providers of 3D CAD software on the market today [2–4], but the standard algorithms of computer-aided geometry representation are valid for a broad field of applications.

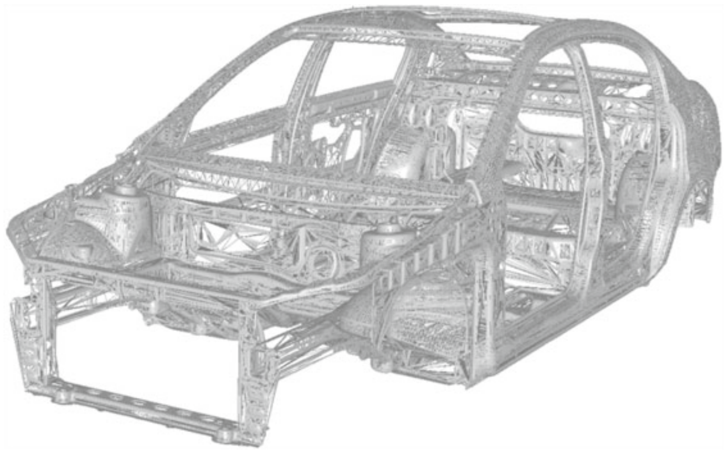


Fig. 4.3 Surface mesh of an automotive bodywork

The representation of geometrical elements in CAD software can be classified in different ways. A simple categorization considers the dimension of the element space. 0D elements are points which merely define a location in space, 1D elements are lines and curves, and 2D elements include surfaces. Earlier CAD systems used 2.5D elements to define the characteristics of two-dimensional sections as a function of a third coordinate. In this way, three-dimensional figures were built up from a number of layers. Finally, 3D representation models cover all three dimensional components which are used for the visualization of product models in space. Mathematical descriptions can include vectors or matrices with more than three dimensions to enable different computations, but in general, the models used in product-design processes are mainly represented in two or three-dimensional (2D or 3D) spaces.

Another way of categorizing geometrical elements is based on their types. Basically, these types are the following:

- **Wireframe models** include points and curves for the definition of two- and three-dimensional objects. Two-dimensional wireframe models are used for the generation of 2D workshop drawings, simple component modeling or for NC-path control algorithms. In design processes, wireframe models provide the basis for the generation of surface or volume features. Wireframe elements are able to define positions, dimensions and some kinds of component extensions, but they lack information about material filling and inside/outside orientations.
- **Surface models** describe planar and cambered faces in three-dimensional space. Apart from standard surface types, Bezier, B-spline and NURBS (non-uniform rational B-splines) models are particular instances defined by specific mathematical relations. Typical applications of 3D CAD surface modeling include body design in automotive and aeronautic engineering and the representation of complex geometries in a wide variety of uses, including even architectural or medical representation tasks. Surface models contain the geometrical information of a

product in terms of its hull geometry, but they do not include information on solid components.

- **Volume (solid) models** enable a complete and compact geometrical product representation in a virtual environment. Alongside the hull geometry, solid models are capable of defining closing conditions, inside/outside information and geometrical consistency specifications. The ability to define material properties supports the realization of several physical simulations (weight, inertia). In subsequent steps, the 3D CAD model serves as a basis for far-reaching technical simulation and computation procedures.
- **Hybrid models** include all types of geometrical representation in a logical order for the definition of a virtual product. The connection of wireframe and surface models and the subsequent derivation of solid objects are state of the art in modern 3D CAD software packages as they are used in the automotive industry.

The design of the kind of complex mechanical products that are used in the automotive industry requires the application of integrated CAD software packages that provide much more than simple functions for digital geometry creation. CAD models can contain additional information and functionalities that join components, parts and modules. Automotive CAD processes are based on the assembly of numerous components, units and groups in DMUs. Thus, it is important to recognize that CAD is used not only for the definition of individual component geometries on the part level, but also to accomplish component positioning and other features on the assembly level. The implementation of parameters, constraints and relations within an assembly structure enables the linkage of components with the goal of a systematic, structured control of their positioning within a superordinated system. The success of CAD-based modeling is due to the design methods and strategies applied. The implementation of interconnections between geometry elements and parameters requires an integrated modeling process which increases the effort of former non-parametric geometry creation. In particular, the development of complex product structures with a large number of various components and modules necessitates the implementation of design-process-related guidelines which include detailed instructions regarding both the geometry definition strategy and the structurization of assemblies.

Non-Parametric CAD

In the early years of 3D CAD, non-parametric design methods enabled product representation in a virtual environment using geometry-focused functionalities. The creation of geometrical objects strictly followed the requirements of geometry creation and enabled a direct access to specific functionalities. However, as the complexity increased, the definition of geometry models became more and more cumbersome and difficult. Since non-parametric design methods are focused only on geometry creation, the part-oriented design processes are accomplished in autonomous operations. These operations have to be coordinated with relatively large effort, which impedes the creation of complex product structures. Since assemblies do not include interac-

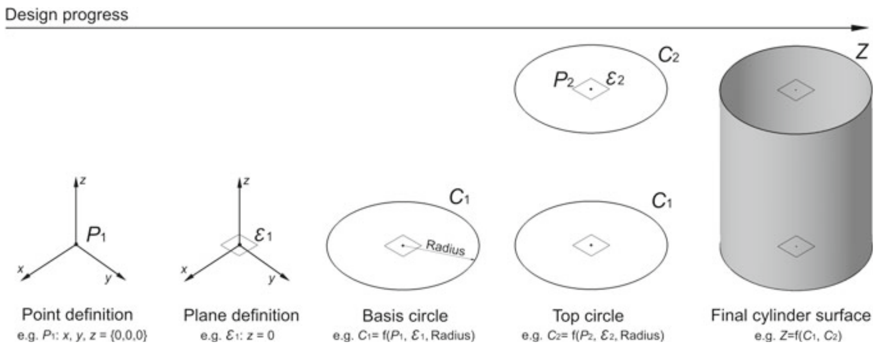


Fig. 4.4 Exemplary non-parametric creation of a cylindrical surface Z

tions between components or modules calling for the implementation of extensive control and monitoring procedures, modifications of product models are often complex and troublesome. Due to the fact that geometry elements are defined with low levels of logical dependencies and associations, single geometrical modifications can lead to instabilities of the entire model structure, which in turn can require complicated revisions of the complete geometry model. Figure 4.4 shows the steps of an exemplary non-parametric creation of a cylinder surface Z. The individual elements are defined separately (e.g. the basis point P_1 might be defined by coordinates). A plane ε_1 in space could be defined by an equation or via other components (e.g. lines, planar curves, etc.). Finally, the first basis circle C_1 is created based on P_1, ε_1 and a radius, whereby the radius could be defined by some appropriate value or the selection of another point. The resulting cylindrical surface Z is created under consideration of the basis circle C_1, a second circle C_2 in a parallel plane ε_1 and a second point P_2.

The relatively simple model structure and lean application of non-parametric CAD makes this technique favorable for initial sketches and drafts. Direct modeling enables the efficient creation of geometry models, independent from any adjacent components and boundary conditions. The disadvantages, however, are the limited number of advanced design methods available with this approach and the increasing effort in the case of complex product models. In particular, the efficient integration of multiple assemblies in automotive engineering is limited when using non-parametric CAD. Since the significant number of independent components and modules led to disadvantages in the management and maintenance of complex product models, nearly all automotive manufacturers and suppliers worldwide switched to parametric CAD at the beginning of the twenty-first century.

Parametric-Associative CAD

Parametric modeling techniques in 3D CAD link geometry objects with geometric constraints and dimensional data. The separation of the geometry elements of the CAD model and the corresponding parameters is one important characteristic of parametric design. Geometry variation is accomplished by changing the input data of the corresponding dimensional constraint associated with a new computation cycle. The parameter values are influenced either through direct data input or by means of equations. In the latter case, parameters are able to supersede the equation arguments. In order for this recalculation to be performed, the model consistency must first be examined to determine if it is suitable for geometrical adaptation. The geometrical degrees of freedom of associative models are only partly ascertained by the direct input of specifications. The remaining requirements for an unambiguous determination are defined by means of relations between geometrical elements.

Associative design operations include direct relations and dependencies between geometrical objects. Geometrical interrelations are defined as a part of the creation process and establish parent-child relations. For example, a direct relationship is established between a given surface (parent) and its offset surface (child); any modification of the parent elements then results in an update cycle of the offset geometry. Modern CAD systems also offer the ability to create multi-model links, which enable the definition of associative functionalities between (formerly independent) parts in assembly structures. This characteristic supports the implementation of far-reaching geometrical connective relations through the use of comprehensive functionalities.

In addition, parametric-associative design combines the possibilities of parametric design and associative functionalities. The realization of parametric geometry control and interlinked geometry elements enables an enhancement of 3D CAD product representation with mathematical and logical functionalities. One further important characteristic of parametric-associative design methods is the unambiguous assignment of geometrical functions and their controlling parameters. Apart from that, logical relations between parameters and geometrical elements form parent-child relations, which lead to comprehensive parametric model structures. In the case of complex product models, geometrical modifications require re-calculation of the entire model structure, which subsequently leads to a logical modification of the components involved. Of course, these various integrated relationships lead to complex model structures, which have to be carefully created and maintained.

Figure 4.5 shows an example of the parametric creation of simple geometry. The geometrical parameters of cylinder Z are defined by inputting the length L_Z and the radius R_Z. The cylinder geometry itself is positioned in relation to a line g, whereby the line definition depends on a point P_1. Due to the logical dependencies, the resulting geometry elements allow for the modification of elementary components (parent-child relations). In the present example, a modification of the parent-element P_1 would lead to a modification of the position of Z, while a modification of the direction of g would lead to a modified orientation of Z. While this strategy supports efficient subsequent variations, in the case of complex parent-child linkages, the interactions have to be monitored and maintained carefully to avoid logical inconsistencies and update problems.

Parametric-associative methods offer several key advantages. First, a clear definition of the geometry, parameters and reference elements in the entire design data structure is essential for the implementation of automated functionalities for subse-

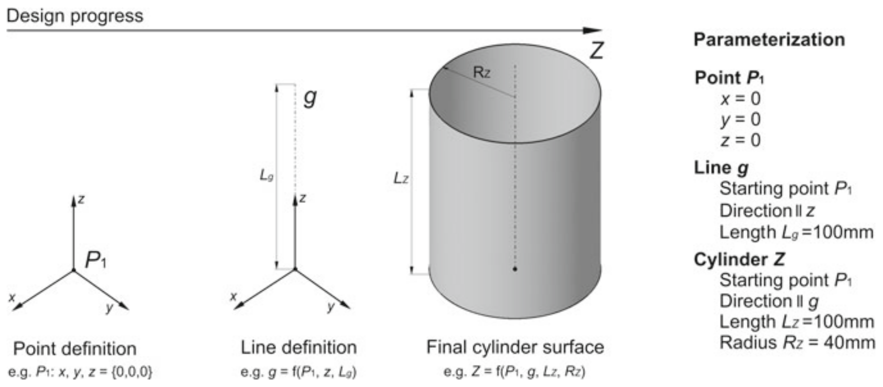


Fig. 4.5 Exemplary parametric creation of a cylindrical surface *Z*

quent or parallel processes, such as component weight calculation, production-related investigations or data quality check functionalities. Second, a continuously compatible model data structure provides the foundation for interconnected working methods that integrate engineering providers and component suppliers. Third, a predefinition of the data setup in CAD parts enables the application of enhanced design methods in assembly development. Finally, the definition of adapter geometries (or adapter models) supports parametric geometry management in several components by using centralized control geometry. This control geometry contains the key information for the creation of geometrical structures, which are then transferred into the corresponding models. As an example, flange geometries are controlled by adapters, which in turn define the geometry of the models at both sides of the flange. A modification in one model triggers an automatic adjustment of the corresponding geometry in the neighboring part. In complex assembly, parametric skeleton models can serve as reference elements for the positioning of modules and components.

Parametric-associative design can be applied on both the part level and the assembly level. The part level includes the implementation of relations, dependencies and mathematical functionalities within the environment of individual components. These components are generated in reproducible historical orders of parameters and geometry elements, which are represented in specific lists or tables (structure trees). Complex products are created in assemblies, which support a classification and structuring into modules and sub-modules.

The application of parametric-associative structures on the assembly level facilitates a highly flexible but complex product representation in a virtual environment. The manifold interactions between components have to be organized by a data management system (PDM-System) for efficient data handling. The integrated consideration of product-structure-related influences supports the continuous cross-linking of components and modules. Using relational design methods, design processes are interlinked with several external activities, such as product calculation and simulation, organizational sequences and production engineering. In this way, the

CAD methods applied have considerable bearing on the entire product development process.

Parametric-associative CAD constitutes a basis for a vision of a highly integrated virtual development that would enable the complete virtual (digital) product description (including process integration, production and manufacturing planning, and the attendant operations) throughout the product life cycle. In this case, the effort required for data management and process organization increases significantly with increasing levels of parameterization. The consideration of different information related to product and process during state-of-the-art 3D CAD routes is leading to an increasing implementation of data and knowledge into the models created. As the complexity of design methods and technologies increases, the complexity of data and process organization increases as well. This leads to changes in the share of workload intensity during the design processes.

Unlike non-parametric geometry creation methods, the implementation of parametric-associative design involves an increased effort for the programming of parametric models. The implementation of logical relations with parameters and the generation of a history-based data structure require additional effort, which leads to higher expenses during the early development phase. Once the parametric-associative models (with all the implemented functions and data) have been created, the geometry creation expenditure decreases considerably, as modifications can be performed more clearly and easily than in direct design processes. Especially during series development, the advantages of the parametric-associative design method offer significant benefits, as the method offers the ability to draw on the knowledge generated previously in the introduction phase of development.

In order to reap the full benefits of parametric-associative design, additional provisions in terms of knowledge integration and automation have to be implemented. Knowledge-based design processes include the use of predefined parametric geometry structures and models (e.g. templates), which can be reused in different applications. In this way, design knowledge gained in former projects is saved and made available for new development cycles. In addition, the establishment of mathematical connections between geometry-driving parameters supports quick and effective geometry creation cycles. In automotive development, the sequences of design are prescribed in so-called ‘startup models’, which include some detailed prediction of the historical order, the relations to reference elements and the marking of different geometrical elements in a 3D CAD model structure.

4.1 Structures of 3D CAD Models

3D CAD models are configured in a logical order of geometrical elements. In state-of-the-art design software, the history-based construction of parametric-associative geometry models is represented in specification trees, which include a detailed description of all geometrical elements and parametric relations. In modern development processes, the 3D CAD geometry generation is based on parametric-associative

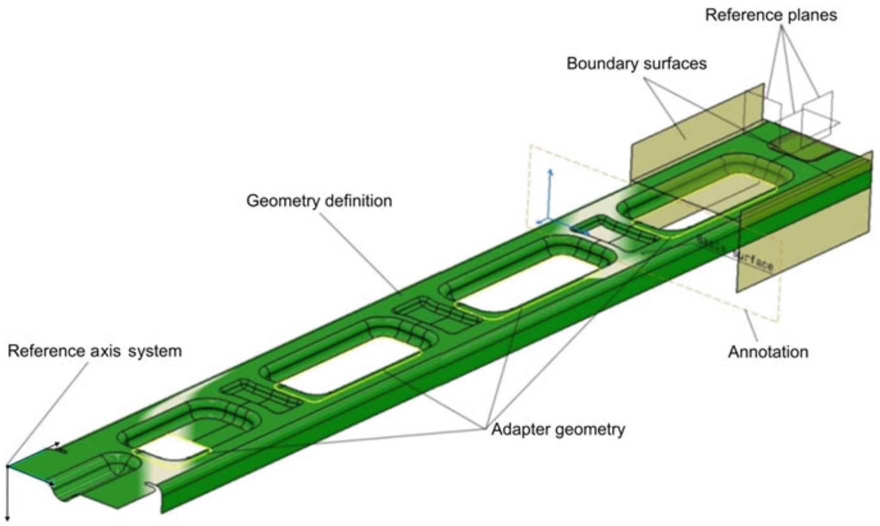


Fig. 4.6 Surface model with different elements necessary for geometry definition

structures, while the virtual models are built up according to predefined orders in so-called startup models. These startup models include various definitions related to the design process of the components and can also include additional functionalities associated with design check features, DMU-relevant information or calculations. In automotive development processes, different kinds of parts call for different design-related boundary conditions. Therefore, the setup of startup models varies according to the requirements of each type of component to be created.

4.1.1 Surface-Based Model Structure

Surface-based models, which occur in car body development, require a predefinition of the sheet metal design process. Depending on which CAD software is applied, startup models of surface-based parts consist of reference geometries, supporting geometries, executive surfaces, green surfaces and the final geometry. Figure 4.6 shows an example of a surface model with the different elements necessary for the geometry definition. A startup model predefines the order of each element of the geometry creation process to support modifications, check operations or upgrade processes. Startup models of components from the same category (e.g. sheet metal parts, plastic parts, cast parts) ensure the consistency of the model structure throughout the entire development project.

Figure 4.7 shows a generic model structure for sheet metal design processes, which was created after detailed studies of body in white startup model configurations in the automotive industry. The generic startup model shows the segmentation of a

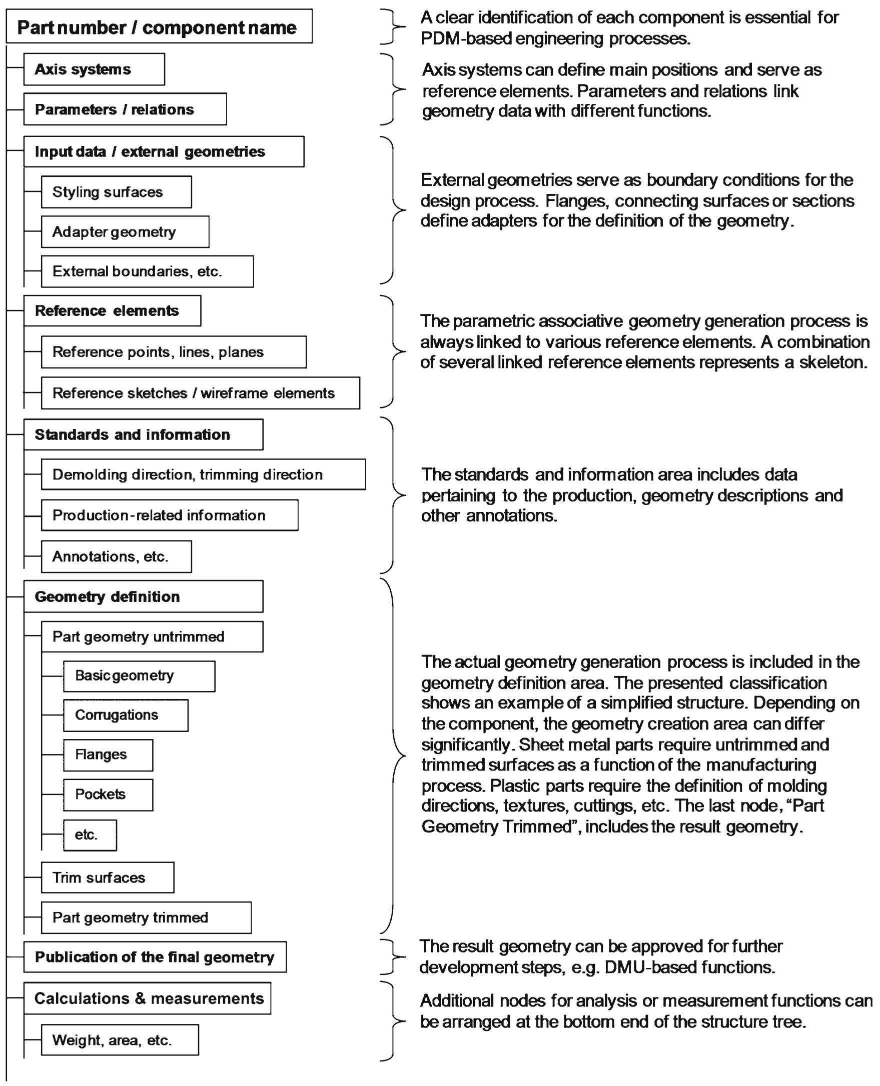

Part number / component name
Axis systems
Parameters / relations
Input data / external geometries
Styling surfaces
Adapter geometry
External boundaries, etc.
Reference elements
Reference points, lines, planes
Reference sketches / wireframe elements
Standards and information
Demolding direction, trimming direction
Production-related information
Annotations, etc.
Geometry definition
Part geometry untrimmed
Basic geometry
Corrugations
Flanges
Pockets
etc.
Trim surfaces
Part geometry trimmed
Publication of the final geometry
Calculations & measurements
Weight, area, etc.
A clear identification of each component is essential for PDM-based engineering processes.
Axis systems can define main positions and serve as reference elements. Parameters and relations link geometry data with different functions.
External geometries serve as boundary conditions for the design process. Flanges, connecting surfaces or sections define adapters for the definition of the geometry.
The parametric associative geometry generation process is always linked to various reference elements. A combination of several linked reference elements represents a skeleton.
The standards and information area includes data pertaining to the production, geometry descriptions and other annotations.
The actual geometry generation process is included in the geometry definition area. The presented classification shows an example of a simplified structure. Depending on the component, the geometry creation area can differ significantly. Sheet metal parts require untrimmed and trimmed surfaces as a function of the manufacturing process. Plastic parts require the definition of molding directions, textures, cuttings, etc. The last node, "Part Geometry Trimmed", includes the result geometry.
The result geometry can be approved for further development steps, e.g. DMU-based functions.
Additional nodes for analysis or measurement functions can be arranged at the bottom end of the structure tree.

Fig. 4.7 Example of a startup model configuration for sheet-metal-based geometry creation

parametric model structure into different modules, which contain specific types of components, functions and relations required for the model generation. Because different OEMs use varying methods and strategies in component design, startup models from different automotive manufacturers differ. However, the main functions shown in Fig. 4.7 can be found in all startup models.

Each component is identified by its part number and/or component name. Unambiguous notations are the basis for the integration of CAD models into product data management systems, which control the data flow during the development processes. Reference elements serve to orient geometrical relations within the entire product structure. There are different types of reference elements, which are organized in prescribed folders within the specification tree.

Axis systems define main positions in the design process. In automotive engineering, the main axis system of a car in the design process is usually placed in the center of the front axis at the mean plane. This is different from the axis system of simulation processes, which is positioned at the car's center of gravity. Based on the requirements of specific component and module design, subsequent axis systems are implemented to define positions and orientations. Besides axis systems, points, lines and planes often serve as reference elements. Combining different geometrical elements in two-dimensional sketches with three-dimensional wireframe configurations enables the definition of skeleton models, which provide integrated functionalities for geometry-related referencing. External geometry elements serve as input data and boundary conditions for the design process. In this way, styling data, adapter geometries and other external geometry-related aspects are imported into the model environment and stored in a predefined folder of the specification tree.

Design-related standards and information enable the implementation of production-related aspects, such as demolding and trimming directions, annotations, and specifications for programming mechanical machining procedures. This information is provided as geometry elements or in the form of information and parameters. The folder *Parameters/relations* includes user-defined parameters and formulas, which control a variety of integrated functions, such as embedded calculation algorithms, geometry control and data exchange procedures. In combination with macro routines, several automated processes can be implemented into the CAD model.

The geometry creation process itself is accomplished in the *Geometry Definition* area, which is divided into different folders based on the requirements of the model which has to be created. In many cases, the operational sequences of geometry creation are not prescribed and therefore provide the creative freedom that is essential for the generation of appropriate geometries. In this area, engineers produce their models based on their knowledge and experience. Finally, the resulting geometry is marked as final component and checked for further processes. The approved geometry is placed in a specific area of the structure tree, for example in the folder *Publication of the final surface*. Supporting calculations and measurement operations are placed in the last folder of the specification tree.

4.1.2 Solid-Based Model Structure

Unlike sheet metal parts, cast metal components are mainly designed in solid structures. Solid structures define the geometry with the help of volume-based functionalities. The main body design is generated with the help of solid-based features and so-called Boolean operations. If required, surfaces are integrated as boundary conditions, reference elements or splitting components.

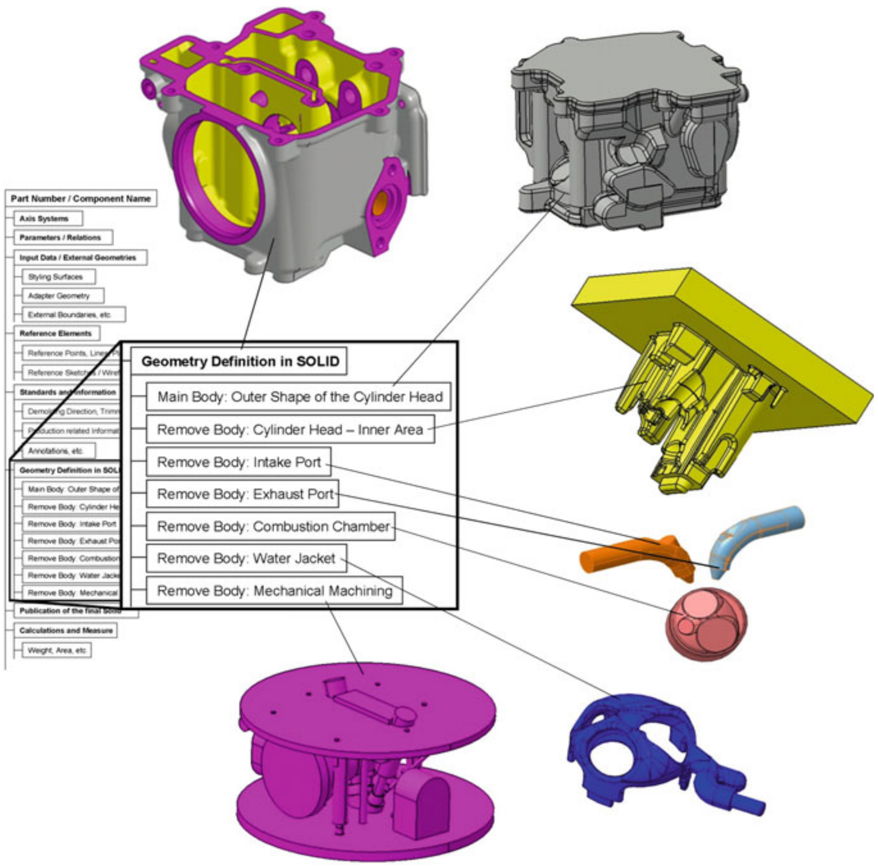

Part Number / Component Name
Axis Systems
Parameters / Relations
Input Data / External Geometries
Styling Surfaces
Adapter Geometry
External Boundaries, etc
Reference Elements
Annotations, etc.
Publication of the final Solid
Calculations and Measure
Weight, Area, etc
Geometry Definition in SOLID
Main Body: Outer Shape of the Cylinder Head
Remove Body: Cylinder Head – Inner Area
Remove Body: Intake Port
Remove Body: Exhaust Port
Remove Body: Combustion Chamber
Remove Body: Water Jacket
Remove Body: Mechanical Machining

Fig. 4.8 Structure of a 3D CAD cylinder head model

As an example, Fig. 4.8 shows a general structure of a cast metal part. The motorcycle cylinder head presented is made of aluminum in a casting process with steel molds and sand cores. In the case of the cylinder head, the manufacturing processes significantly influence the structure of the geometry creation. The principle structure of the startup model corresponds to that of the sheet-metal-based geometry creation. Axis systems, boundary conditions, annotations and reference elements are arranged in the same order, but the geometry creation itself follows the rules of solid-based operations. In this way, the node *Geometry Definition in SOLID* is performed as a logical connection of several bodies. Each body can consist of numerous solid functions and describes a single module. The final geometry results in a logical interaction of these bodies in a coherent order.

The sample cylinder head model depicted in Fig. 4.8 is composed of seven individual elements (solids), whereby each element includes all of the required information related to both its geometry and the production process. The body *Outer Shape of the*

Cylinder Head contains geometry data for the mold manufacturing process, including the parting surfaces, the draft angles and the fillets. In this way, the manufacturer is able to produce the mold directly from the 3D CAD model by separating the body *Outer Shape of the Cylinder Head* from the cylinder head model. The second body *Cylinder Head - Inner Area* also includes the production-related information and is defined as a negative volume.

After removing the second body from the first one with a Boolean operation, the geometry of the cylinder head in the upper area is created. In the production process, the intake and the exhaust port are defined as sand core parts, which are fixed in the steel mold. The corresponding bodies in the virtual geometry creation process include all of the information required for the production and positioning of the core parts. The last body describes the mechanical machining of the cylinder head. This component represents the basis for the programming of NC-controlled working machines and can also include 3D tolerance-related data.

4.1.3 The Role of CAD Models in Product Development

CAD models play an important role in modern product development processes. Because product representation is accomplished by comprehensive geometry models within a CAD environment, different product-related information is developed, stored and maintained in parametric-associative design processes. The design model itself is composed of a number of sub-modules, which are arranged in logical orders. These sub-modules can serve as a basis for simultaneous, linked CAE-processes, which cover specific calculation and simulation processes. The assemblies of modules and sub-modules result in CAD-models of specific components, which can serve as an information source for different operations performed in parallel or subsequently (e.g. production-related development processes or DMU-based investigations).

A parametric-associative structure of virtual product models provides the basis for highly flexible development processes. Parameterized design modules can be linked with each other or linked with modules of external geometries, thereby supporting automatic update functions during optimization cycles. Figure 4.9 shows the interactions of a virtual engineering process using the example of a cylinder head development. The parametric-associative design process is divided into several sections, which are simultaneously linked with corresponding CAE processes. A smart structuring of the CAD model enables a relatively simple exchange between of individual modules. In this way, the results of simulation processes (e.g. FE optimization, CFD calculations) of specific areas can be continuously implemented.

In the cylinder head presented, the geometry of the intake and exhaust ports, as well as the geometry of the water jacket, are first built into the 3D CAD model as a rough estimation. These initial geometry models serve as placeholders in early design phases and also as input data for extensive computational flow-dynamics simulation. The findings of the CFD-simulation are transferred into design processes

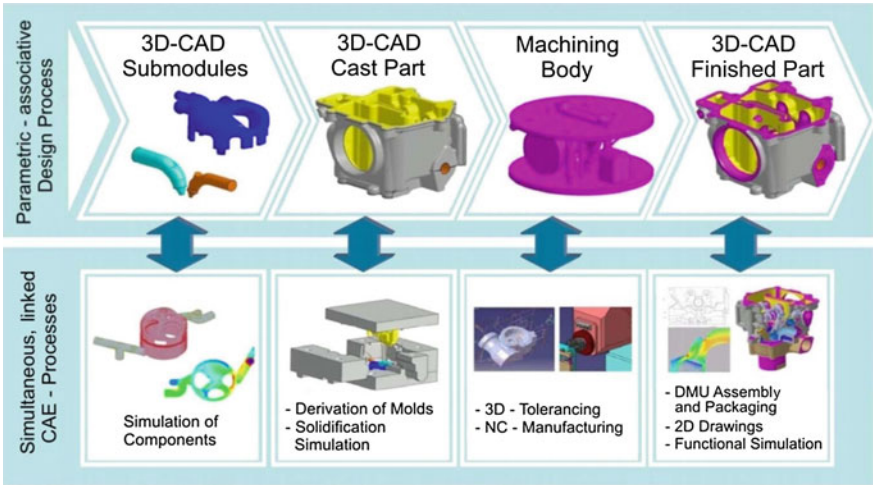


Fig. 4.9 Parametric-associative design and simultaneous, linked CAE operations in a cylinder head development process

and result in modifications of the initial geometries. During the design process, the model shapes are revised under consideration of several additional characteristics (e.g. space requirements or production-related aspects).

A logical arrangement of modules and sub-modules leads to a final CAD-model, which represents the virtual geometry model of a component. In the present example, the design process results in a cylinder head model cast part. This cast part provides the basis for the design of casting molds and sand cores. Besides the geometrical information of the cylinder head, the design of molds and cores has to consider several production-related aspects, such as the requirements of production machines and other casting-technology-related properties. Simultaneously performed solidification simulation can support both detailing of the cylinder head geometry (especially in areas that are relevant for the flow of liquid aluminum) and the actual design of the molds.

In the case of cast parts, mechanical processing operations (e.g. drilling and milling) are designed as a separate module. This so-called *machining body* includes all of the surfaces of mechanical processing and different manufacturing-related information (e.g. the type of mechanical treatment, cutting speed and feed rate, tolerancing). In addition, the geometrical information contained in the machining body can support the programming of production machines. Finally, the finished CAD part is used in assemblies and subsequent DMU processes, the derivation of workshop drawing and in kinematics or other functional simulation processes. The periodic interaction of parametric-associative design and simultaneously linked CAE-processes forms virtual optimization phases, whereas geometry generation sections and simulation steps are linked using CAD product models.

4.2 Wireframe and Surface Design

Commercial CAD software applications provide user-oriented functionalities for the definition and modification of geometrical elements, which support efficient design processes. Unlike the fundamental mathematical description of points, curves and surfaces introduced in Chap. 3, the handling of functions and operations within a CAD software environment is object- and problem oriented and supports user-friendly access to the geometrical elements. These methods enable efficient design processes because the engineers can focus on product-oriented development, while the software manages the mathematical definition of geometry in the background. In general, modern CAD software provides several functions for the definition of geometrical basis elements (e.g. points, lines, curves or different types of surfaces and solids). Once defined, these elements can be modified or combined in a logical order to form the desired geometry of the product which is being developed.

The definition of geometry differs between styling and design software. Due to the specific requirements of styling surfaces, CAS software provides functionalities for an efficient conversion of sketches or point clouds into smooth surfaces. In addition, the creation of curves and surface patches is performed via control polygons and control grids. In combination with different geometrical weighting factors, the control elements enable a direct access to the shape of curves and surfaces (cf. Sects. 3.4, p. 85 and 3.8, p. 181). Unlike CAS, CAD requires the technology-focused definition of geometry as well as a logical and comprehensible structuring of product models. This is the reason why CAD software manufacturers are increasingly implementing parametric-associative functionalities in combination with feature-based modeling. These utilities provide the foundation for object-oriented design processes, which push the development of the product itself into the foreground, while the mathematical geometry definition is performed simultaneously by software-embedded automated routines in the background. The following sections give a software-manufacturer-independent overview of state-of-the-art functionalities and methods in the application of 3D CAD.

4.2.1 Reference Elements

Reference elements serve as a basis for dimensioning or for the definition of specific boundary elements. They also enable efficient design processes by supporting geometry creation and transformation features. In general, coordinate systems can serve as reference elements, which are related to a specific position and orientation in the working space. In modern CAD systems, a coordinate system with an origin P and three axes x, y, z also defines three planes, ε_{xy}, ε_{yz} and ε_{xz}, which can be used for different purposes. Besides coordinate systems, single planes often serve as reference elements (e.g. to define a distance or an orientation). Planes can be defined as an offset element from another plane or planar surface, by two lines or a

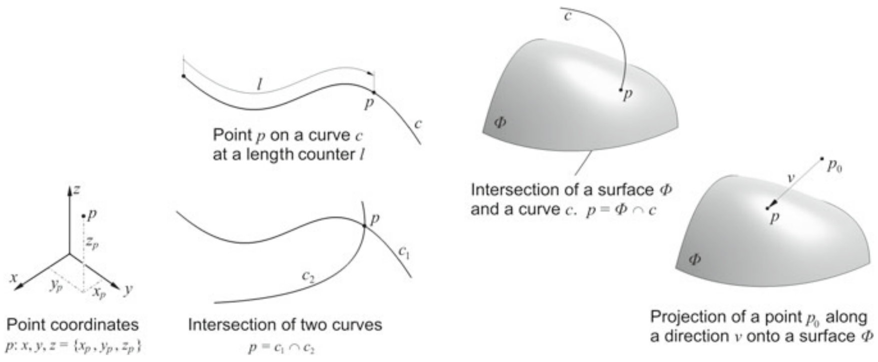

Point p on a curve c at a length counter l
Intersection of a surface Φ and a curve c. $p = \Phi \cap c$
Point coordinates $p: x, y, z = \{x_p, y_p, z_p\}$
Intersection of two curves $p = c_1 \cap c_2$
Projection of a point p_0 along a direction v onto a surface Φ

Fig. 4.10 Examples of point definition in CAD

planar curve, as well as a normal plane to a curve in space. In addition, a plane is defined via its equation $Ax + By + Cz + D = 0$ (cf. Example 3.14, p. 159), whereby $\mathbf{n} = [A, B, C]^\top$ is a normal vector of the plane and $\frac{|D|}{\sqrt{A^2+B^2+C^2}}$ is its distance from the coordinate system's origin.

In general, all geometry elements can serve as reference elements. Besides the two types mentioned above, points, lines and regular surfaces can be used because of their relatively simple definition and high stability. Complex geometries should not be used as reference elements because they tend to lead to instability in the case of geometrical or structural modifications of the CAD model.

4.2.2 Wireframe Design

Wireframe elements include points, lines and curves, which are defined in two-dimensional or three-dimensional working spaces. In the case of two-dimensional definitions, the elements can be created as independent elements within a plane or as components of a sketch. In all instances, wireframe elements can have non-parametric or parametric-associative characteristics.

Points often serve as initial elements in CAD. In parametric design processes, points can be created in different ways. Besides inputting the point coordinates, it is possible to create a point on a curve or as an intersection of two curves or a surface and a curve. The projection of an existing point in space onto a curve or a surface also serves for the definition of point elements. Due to the fact that a point represents a zero dimensional location in space, point elements can serve as reference elements, as the basis for the creation of curves or lines, or originators for parametric dimensioning. In addition, points can serve as elements for measurement procedures in virtual product models. Figure 4.10 shows a selection of examples of point creation in CAD.

Lines often serve as reference elements or as supporting elements for the design of surfaces and solids. In parametric CAD, lines are created based on parent elements (e.g. points, vertices, curves or surfaces). The simplest definition of a line considers two points as start and end elements. Another possibility is the definition of a start point, a direction and a length, whereby the direction vector can be defined by different elements (e.g. a line, an edge or a normal direction to a plane or a surface). Another possibility is the extraction of the axis from surfaces of revolution. In general, CAD systems enable the extraction of lines from the boundary elements of existing elements (e.g. a cube or a block). Figure 4.11 includes a selection of examples of the creation of lines in CAD.

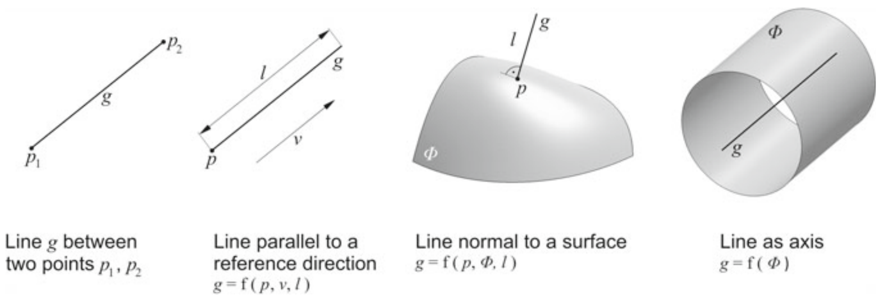

p_2
g
p_1
l
g
p
v
l
g
p
Φ
Φ
g
Line g between two points p_1, p_2
Line parallel to a reference direction $g = f(p, v, l)$
Line normal to a surface $g = f(p, \Phi, l)$
Line as axis $g = f(\Phi)$

Fig. 4.11 Examples of line definition in CAD

Curves, which serve as a basis for the creation of simple and complex surfaces, solid models and as reference elements for different additional operations (e.g. as translation paths in kinematics simulation), are important geometrical elements in CAD processes. Curves can be defined within planes or planar surfaces as 2D elements or in the 3D working space.

Regular curves are curves that can be defined by mathematical equations (e.g. circles, conic sections, helical curves or spirals; cf. Sect. 3.4, p. 85). CAD softwares provide additional possibilities for the creation of regular curves by different types of operations (e.g. using geometrical elements as parent elements and boundary conditions). As an example, a circle c can be defined by selecting a supporting plane ε, a point p_1 as circle center and a point $p_2 \in \varepsilon$ for the definition of the radius. Another possibility is the consideration of a center point p_1 and a tangent curve c_T. There are other commonly applied methods of circle definition that are similar to these examples, and analogous functionalities are provided for other regular curves. With all of these approaches, regular curves are created using geometrical elements for their unambiguous definition. Screw lines (helices) represent a special case of regular curves because of their three dimensional extension. A helix c displays the path of a point which rotates around an axis g while simultaneously proceeding a pitch h (cf. Example 3.3, p. 61 and Example 3.5, p. 71). Besides a helix, Fig. 4.12 shows examples of the definition of two selected regular curves, a circle and a parabola.

Splines represent a general, so-called freeform type of curve in space (cf. Sect. 3.4, p. 85 and, in particular, Definition 3.28, p. 99). While their mathematical definition takes into account different complex interpolation and weighting function-

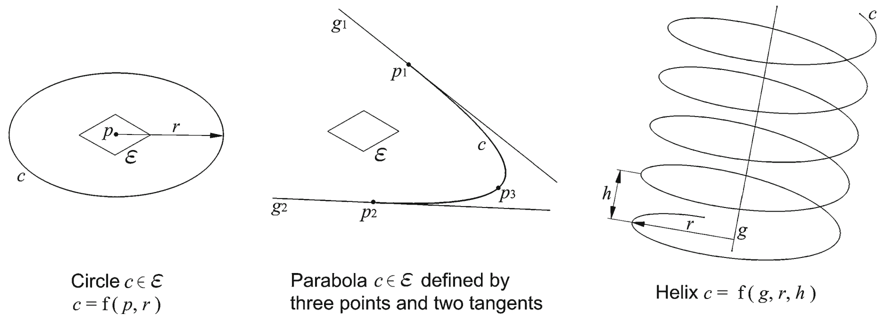
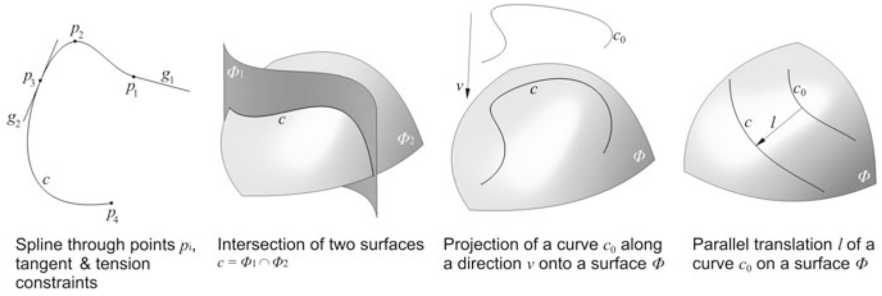


Fig. 4.12 Examples of regular curves



p_2
p_3
p_1
g_1
g_2
c
p_4
Φ_1
c
Φ_2
c_0
v
c
Φ
c_0
c
l
Φ
Spline through points p_i, tangent & tension constraints
Intersection of two surfaces $c = \Phi_1 \cap \Phi_2$
Projection of a curve c_0 along a direction v onto a surface Φ
Parallel translation l of a curve c_0 on a surface Φ

Fig. 4.13 Exemplary definitions of free-form curves in CAD

alities, CAD systems offer geometry-based operations for a user-friendly creation. In common CAD systems, splines can be defined by passing points p_i, tangent directions g_i and tension factors (Fig. 4.13, left). Tension factors are factors that influence the progression of curves as a function of the curvature and the distance from a specific point on the curve. In CAD processes, splines often result from intersection or projection operations. Figure 4.13 shows a selection of operations for the creation of free-form curves. The intersection of two surfaces Φ_1, Φ_2 leads to a curve c, which is represented by some spline curve in the CAD environment. The projection of a curve c_0 along a direction vector v onto a surface Φ also results in a curve c. The last example shows the translation l of a curve c_0 on a surface Φ, resulting in a curve c. With all of these operations, the resulting curve is related to its associated parent elements.

In parametric-associative designprocesses, wireframe elements often serve as the basis for the creation of complex surface or solid models. The simple and lean structure of wireframe geometry makes this type favorable for application as basic elements, control geometry or dimensioning components. In addition, wireframe geometry is often used for the parametric positioning of components and modules in assembly structures. In this way, components are positioned in relation to a so-called skeleton model, which provides exact information about the location and arrangement of each component in the 3D space. If the skeleton model has a parametric structure,

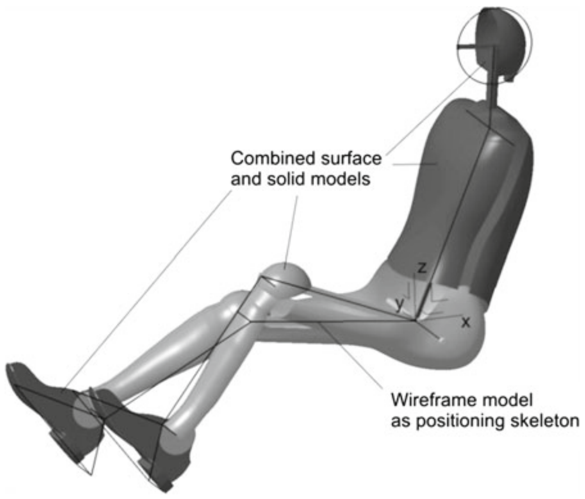


Fig. 4.14 Wireframe elements as a skeleton model for the positioning of surface and solid geometries in a simplified human manikin model

it can be used to control movements and dimensional variations as well. Figure 4.14 shows an example assembly of a simplified human manikin model with wireframe, surface and solid geometries. The manikin model consists of several parts that are assembled in relation to a wireframe model, which is used as a positioning skeleton.

4.2.3 Surface Design

The modeling of surfaces was a main development goal of CAD software suppliers in the 1980s. Since that time, the implementation of multifarious functionalities has broadened the application of surface design significantly, enabling the virtual creation of complexly styled product models. With the introduction of solid-modeling in the early 1990s, the application of surface-based design in mechanical engineering was reduced, whereas some area of business, and especially the aeronautic industry, always has used surface based geometry creation. Since the beginning of the twenty-first century, the combination of parametric-associative design methods and surface-based product modeling has undergone a strong revival. The high styling and shape requirements in modern product design led to an intensified re-integration of surface modeling into development processes.

Especially in the automotive industry, the application of surface modeling is not restricted to sheet metal design, but also concerns the design and development of nearly all visible components. Thus, exterior components and interior car surfaces (e.g. seats, dashboards, panels) are designed with surface modeling, as well as the complete vehicle body structure. Besides these applications, surface modeling has a strong relevance in the development of components and modules that are related to fluid dynamics. Examples include the intake and exhaust systems and ports of internal combustion engines, their cooling systems, and components for brake cooling and vehicle air conditioning. The following section provides an overview of the main functionalities of surface modeling in modern CAD systems and uses application examples to show some important surface-based design methods.

Regular surfaces are surfaces that can be defined by the motion of a curve without surface-based interpolation or weighting functionalities. This basic type of surfaces consists of *extruded surfaces* (also called *cylinders*; cf. Definition 3.52, p. 166), surfaces of revolution (see Sect. 3.7.11, p. 168), *translational surfaces*, *helical surfaces* (cf. Sect. 3.7.12, p. 172) and *pipes*. An extruded surface is based on a profile curve c_P, which is displaced along a direction vector v under consideration of a length l. In general, the basis curve c_P is defined as a 2D-curve within a plane (e.g. in a sketch), but this is not a limitation. A surface of revolution is defined in terms of a curve c, a revolution axis g and an angle of revolution α (Fig. 4.15).

Multi-section surfaces are based on a set of curves, which define sections of the surface to be created. These sections can be positioned in the 3D working space in any way, but should be positioned in an expedient configuration, such that the multi-section surface can be created clearly with no ambiguity or twisted areas. The sections are connected by a surface, whereby the surface continuation can be adjusted by applying several additional functionalities. One possibility is the definition of coupling points at each section. These points serve for the definition of surface segments, which enables the computation of even characteristics for each segment. The segmentation of the surface is performed by coupling functionalities, which link the corresponding points of each section. Figure 4.16 shows an example of a multi-section surface, which is based on two sections, c_1 and c_2. The unregulated surface Φ_U has a twisted and unsmooth progression, which is not useful for the desired application. The regulated surface Φ_R shows a smooth progression and no sharp edges. This behavior is achieved by coupling predefined corresponding points of each section, which significantly improves the surface quality.

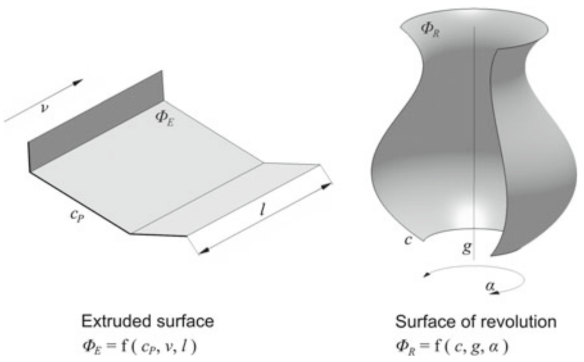


Fig. 4.15 Extruded surface and surface of revolution

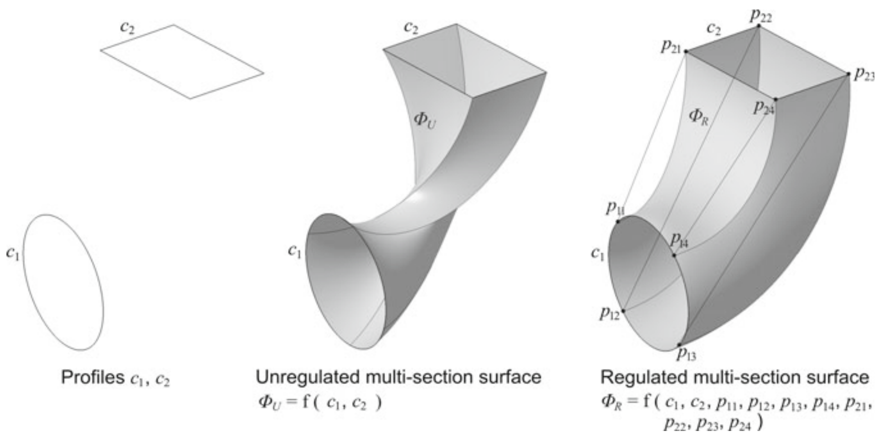

c_2
c_1
Profiles c_1, c_2
c_2
Φ_U
c_1
Unregulated multi-section surface
$\Phi_U = \mathrm{f}(c_1, c_2)$
p_{22}
c_2
p_{21}
p_{23}
Φ_R
p_{24}
p_{11}
c_1
p_{14}
p_{12}
p_{13}
Regulated multi-section surface
$\Phi_R = \mathrm{f}(c_1, c_2, p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24})$

Fig. 4.16 Multi-section surface with coupling function

Besides the adjustment of multi-section surfaces using coupling points, additional functionalities enable extensive modifications. For example, guide curves serve as shaping elements, which are positioned on the surface itself to define its shape directly. Figure 4.17 shows an example of the application of multi-section surfaces based on more than two sections. This part of a longitudinal beam consists of a surface which changes its cross sections as a function of the beam length. The sections c_1, c_2, c_3, c_4 and c_5 are positioned in parallel planes ε_1, ε_2, ε_3, ε_4, and ε_5. In the present example, additional guide curves g_1, g_2 and g_3 support the computation of the resulting surface to achieve a smooth continuation. Important functionalities include the ability to define tangency and curvature continuations of the surface patches, interpolation and extrapolation operations, and the ability to define spine curves as additional shaping elements.

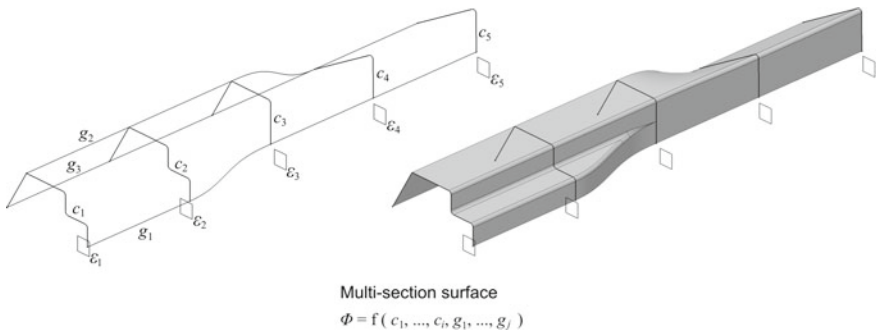
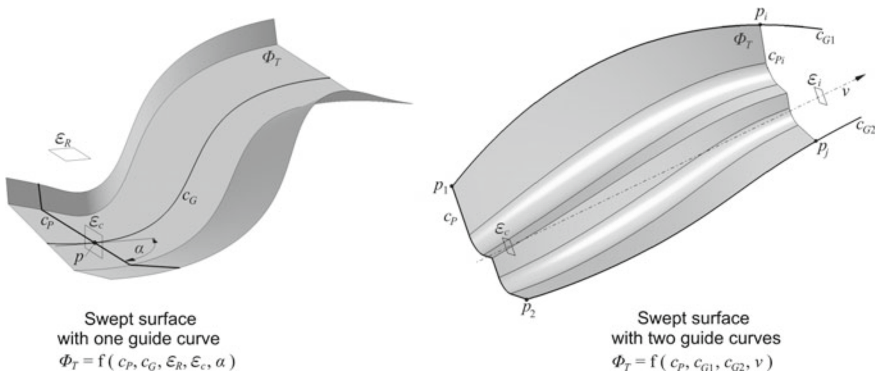


Fig. 4.17 Longitudinal beam created as a multi-section surface



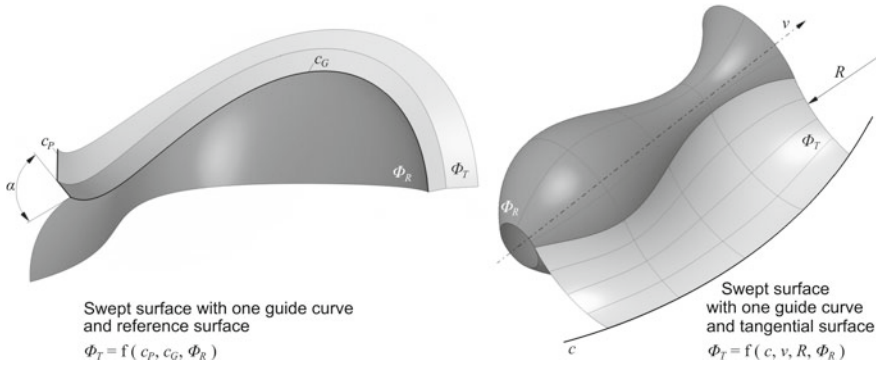
Φ_T
ε_R
c_G
c_P
ε_c
α
p
Swept surface with one guide curve
$\Phi_T = f(c_P, c_G, \varepsilon_R, \varepsilon_c, \alpha)$
p_i
c_{G1}
Φ_T
c_{Pi}
ε_i
v
c_{G2}
p_j
p_1
c_P
ε_c
p_2
Swept surface with two guide curves
$\Phi_T = f(c_P, c_{G1}, c_{G2}, v)$

Fig. 4.18 Translational surfaces with guide curves

The functionalities of multi-section surfaces can be applied at open or closed profiles. In this way, they are used for the design of sheet metal components as well as for the creation of tubes and pipes with variable cross-sections. The manifold options for varying and adjusting surface progression and continuation under different quality criteria make this type of surface suitable for the creation of complex geometries.

Translational surfaces concern a broad family of surfaces, which can be defined in different ways. One possibility is the translation of a profile c_P along a guide curve c_G, whereby the angle α between the profile and the guide curve remains constant. The rotational position of the swept profile as a function of the guide curve length can be constant or variable. In the left-hand example in Fig. 4.18, the profile c_P is a 2D curve within a plane ε_c, and the rotation angle is set to zero in relation to a reference plane ε_R. The right-hand example in Fig. 4.18 shows a translational surface which is defined by a profile curve c_P, two guide curves c_{G1} and c_{G2}, and a computation direction v. The surface Φ_T is created by a displacement of c_P along two guide curves, whereby the distances of two sectioning points p_i and p_j serves as a scaling factor for each section of the surface. The sectioning points are created by a set of intersections of planes ε_c, which have a perpendicular position to the computation direction v. In this way, every incremental position in the computation direction between a starting plane ε_c and an ending plane ε_i results in two sectioning points (from p_1, p_2 to p_i, p_j), which define the related scaled profile curve c_{Pi} in each section. A swept combination of a quantity of incremental sections leads to the final swept surface Φ_T. The application of swept surface creation based on profile and guide curve based enables a parametric definition of multifarious surface models by utilizing complex interpolation procedures, whereby the geometry-based definition of input parameters and boundary conditions supports a user-friendly handling within common CAD software.

Figure 4.19 includes two examples of swept surfaces which use other options for the creation of such surfaces. In the example on the left, a translational surface is defined by a profile curve c_P, one guide curve c_G and a reference surface Φ_R containing c_G. In this case, the angle α between the profile curve and the reference



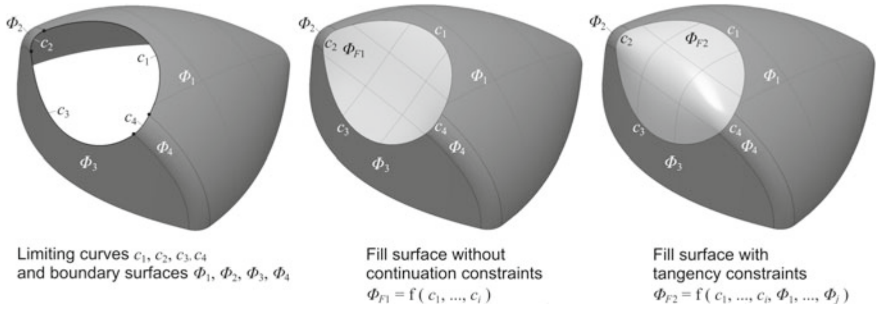
c_G
c_P
α
Φ_R
Φ_T
Swept surface with one guide curve and reference surface
$\Phi_T = f(c_P, c_G, \Phi_R)$
v
R
Φ_T
Φ_R
c
Swept surface with one guide curve and tangential surface
$\Phi_T = f(c, v, R, \Phi_R)$

Fig. 4.19 Examples of swept surfaces with reference surfaces

surface is kept constant, so the resulting surface Φ_T has a constant orientation in relation to the reference surface. One other common option, which is not shown in the figure, is to use a variable profile angle α, which results in a bended surface Φ_T in relation to the reference surface Φ_R. The example on the right of Fig. 4.19 shows an exemplary application of tangential surfaces for the creation of swept surfaces. In this example, a surface Φ_T is defined by a guide curve c, a constant circular profile radius R and a tangency constraint in relation to a reference surface Φ_R. There are several other ways to define swept surfaces using geometrical boundary conditions, such as using two tangential surfaces, variable profile radii or the implementation of specific functions for the definition of the profile curves (e.g. conic sections). These approaches have two things in common: the sliding of a (possibly variable) profile curve in a prescribed direction, and the application of one or several guide curves or one or several tangency surfaces.

Because the features for the creation of multi-section surfaces and swept surfaces offer a wide range of functionalities for the creation of complex surface models, these types of geometrical elements play an important role in automotive design processes. An efficient application of functions for complex surface definition provides the foundation for efficient design processes but also requires well-educated design engineers who are able to manage the extensive functionalities offered by state-of-the-art CAD software.

A *fill surface* is a special type of surface, which is based on a group of boundary elements. In general, fill surfaces fill in a set of wireframe elements, which have to be arranged in an unbroken configuration. The mathematical background of fill surfaces (see also Sect. 3.9.1, p. 205) contains complex interpolation functionalities, which are applied automatically within the CAD environment. The uncomplicated user handling simulates a simple operation, but in some cases, fill surfaces show an extraordinary behavior. Due to the fact that wireframe elements in a 3D-working space can have extensive progressions and sharp corners, the continuations of fill surfaces may have uneven characteristics. To avoid this, the wireframe elements must be carefully defined. Modern CAD systems enable additional functionalities for controlling the shape of fill surfaces (e.g. by applying tangency constraints in



Φ_2
c_1
c_2
c_3
c_4
Φ_1
Φ_3
Φ_4
Φ_{F1}
Φ_{F2}
Limiting curves c_1, c_2, c_3, c_4 and boundary surfaces $\Phi_1, \Phi_2, \Phi_3, \Phi_4$
Fill surface without continuation constraints
$\Phi_{F1} = f(c_1, ..., c_i)$
Fill surface with tangency constraints
$\Phi_{F2} = f(c_1, ..., c_i, \Phi_1, ..., \Phi_j)$

Fig. 4.20 Sample of fill surfaces

relation to adjacent surface patches). Figure 4.20 shows an example of the application of two different types of fill surfaces. Φ_{F1} represents the result of a fill operation of the limiting curves c_1, c_2, c_3, c_4 without consideration of tangency constraints. Φ_{F2} shows the result in the case of tangency constrains in relation to adjacent surfaces Φ_1, Φ_2, Φ_3 and Φ_4.

Finally, offset surfaces represent parallel surfaces of an existing set of closed surface elements. Their definition requires parent surfaces and the specification of an offset distance. In the case of radii or restricted curvatures which are smaller than the transformation distance l, the derivation of offset surfaces may lead to sharp edges g. Figure 4.21 shows and example of such a case by creating an offset surface Φ_O based on the reference surface Φ_R.

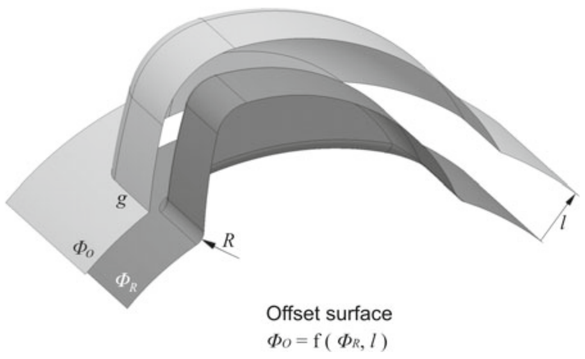
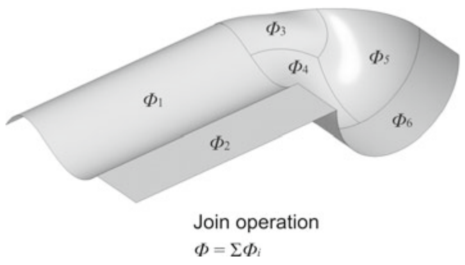


Fig. 4.21 Exemplary creation of offset surfaces

4.2.4 Operations in Wireframe and Surface Design

Wireframe and surface elements can be manipulated by different types of operations, which enable modeling by taking into consideration and/or modifying two or

Fig. 4.22 Joining of surface patches



more geometrical components. These functionalities are based on the interactions of individual elements and result in composed configurations. In parametric design, the resulting geometry is associated with parent elements, whereby the specific functions execute the corresponding mathematical operations in the background. The following section introduces and briefly explains selected main operations in wireframe and surface design. In addition to the functions described here, modern CAD systems offer other possibilities, which are often composed of the main operations presented below. Besides functionalities for geometry manipulation, these extended features enable several adjustments and configurations in terms of tolerances, continuations, and other factors.

One important operation in wireframe and surface design enables the merging of several geometrical elements (e.g. the composition of surface patches to one cumulative surface model). This so-called join function can be applied to surfaces as well as to lines or curves, but it requires a clear definition of the elements to be considered. In this operation, some specific handling errors can lead to geometrical problems. Sources for errors include gaps or overlapping areas between single elements, as well as computation tolerances and deviations in tangency or curvature continuation. Figure 4.22 shows an example of a result of a join operation of several surfaces $\Phi_1 \cdots \Phi_6$.

Another family of operations handles split and trim functionalities. Split operations use splitting geometry (e.g. planes, lines, curves, surfaces) to cut wireframe elements. The resulting elements show a modified geometrical extension such that the intersecting curve of the geometrical elements involved represents the partitioning component. If the intersecting curve is shorter than the surface which has to be split, modern CAD systems offer extrapolation functionalities to enable an easy handling. In the case of split using wireframe elements, it has to be ensured that the lines or curves lie completely on the surfaces treated. The middle illustration in Fig. 4.23 shows an example of the result of a split operation of two surfaces Φ_1 and Φ_2.

In contrast to split operations, trim operations modify the treated surfaces and join them at the same time. Trim operations offer different settings for the orientation of resulting surfaces, their continuation and the extrapolation of trim curves. Similar to split operations, the intersecting curve of the surfaces involved represents the trim curve. In the case of split- or trim operations of wireframe geometries, the intersection results in a point. The right-hand illustration in Fig. 4.23 shows a possible result of a trim operation of two surfaces Φ_1 and Φ_2. In this case, the left side of Φ_2 is joined with the lower part of Φ_1.

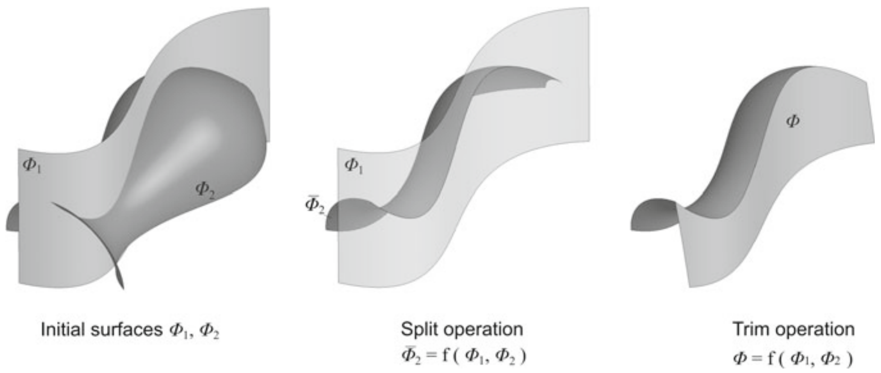

Φ_1
Φ_2
Initial surfaces Φ_1, Φ_2
Φ_1
$\bar{\Phi}_2$
Split operation
$\bar{\Phi}_2 = f(\Phi_1, \Phi_2)$
Φ
Trim operation
$\Phi = f(\Phi_1, \Phi_2)$

Fig. 4.23 Split and trim operations

Rounding and fillets are applied by using specific features that enable an easy handling of these complex mathematical operations. In general, fillet operations enable the application of corners with a constant or variable radius at edges or between wireframe or surface elements. In the case of wireframe elements, the operation results in a curve; in the case of surface elements, the operation results in additional surfaces. These fillet surfaces can be trimmed with the adjacent parent surfaces automatically or can be kept separate for further operations.

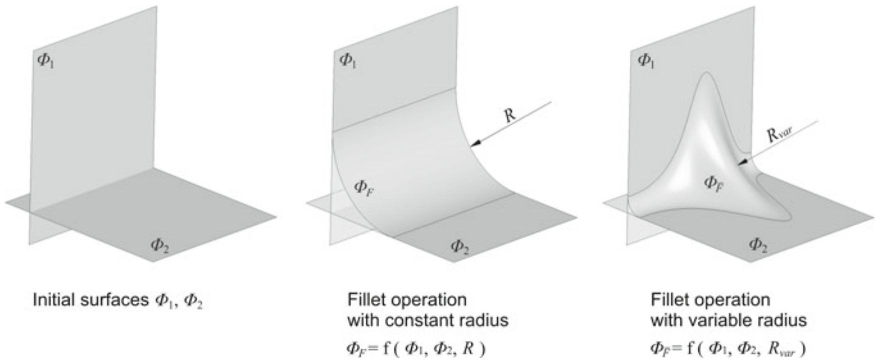
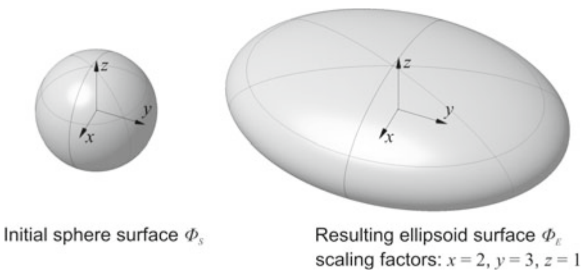

Φ_1
Φ_2
Initial surfaces Φ_1, Φ_2
Φ_1
R
Φ_F
Φ_2
Fillet operation with constant radius
$\Phi_F = f(\Phi_1, \Phi_2, R)$
Φ_1
R_{var}
Φ_F
Φ_2
Fillet operation with variable radius
$\Phi_F = f(\Phi_1, \Phi_2, R_{var})$

Fig. 4.24 Fillet operations

Figure 4.24 shows examples of the application of two different types of fillets with two initial surfaces Φ_1 and Φ_2 (here planes) as boundary elements. The application of a fillet surface with constant radius R leads to a simple cylindrical surface Φ_F (middle), whereas the application of a fillet surface with variable radius R_{var} results in a more complex surface. The continuation characteristic of R_{var} as a function of the length of the intersection curve of the two initial surfaces Φ_1 and Φ_2 directly defines the shape of the resulting surface. In standard applications, the resulting fillet surface has a tangent continuation (GC^1-continuity; cf. Sect. 3.7.3, p. 148) to the

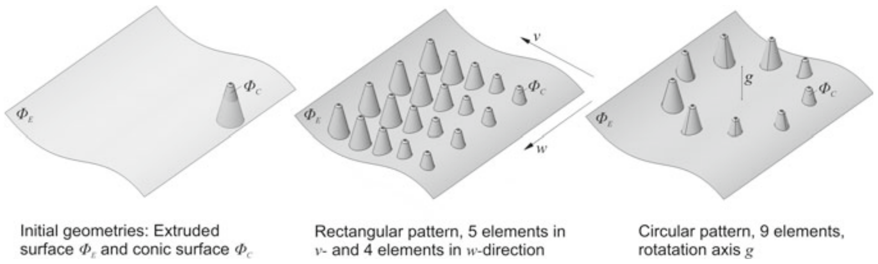


z
y
x
Initial sphere surface Φ_S
Resulting ellipsoid surface Φ_E
scaling factors: $x = 2$, $y = 3$, $z = 1$

Fig. 4.25 Exemplary application of a scale operation with different factors for the x, y and z directions

adjacent surfaces. Advanced functionalities enable the definition of more complex continuation behavior, as well as the implementation of enhanced curves as section profiles instead of circular radii. These so-called styling fillets facilitate a smooth curvature continuation (GC^2-continuity) from one surface to another and are used in the case of vehicle body design or the creation of other visually relevant components.

Another family of operations includes extrapolation, transformation, scaling and pattern functions. Extrapolations enable an extension of existing wireframe or surface elements under consideration of continuation conditions (e.g. tangency or curvature). Transformation functions cover the movement of geometrical elements, including translation, rotation and symmetry functions. These operations often enable the creation of clone elements from existing parent geometries. Scaling operations handle the modification of geometry extension under consideration of scale factors, whereby these factors can be defined separately for the x-, y- and z-direction.[1] This possibility supports a simple creation of warped surface models (Fig. 4.25).



Φ_E
Φ_C
v
w
g
Initial geometries: Extruded surface Φ_E and conic surface Φ_C
Rectangular pattern, 5 elements in v- and 4 elements in w-direction
Circular pattern, 9 elements, rotatation axis g

Fig. 4.26 Pattern operations

Figure 4.26 shows an example of the application of pattern operations. In the case of a rectangular pattern, the initial geometry Φ_C is transferred and copied into the v and the w directions by defining the desired numbers of elements. The circular pattern concerns a rotational transformation of initial geometry in reference to a rotation

[1] These operations are affine transformations in the sense of Definition 3.1, p. 54.

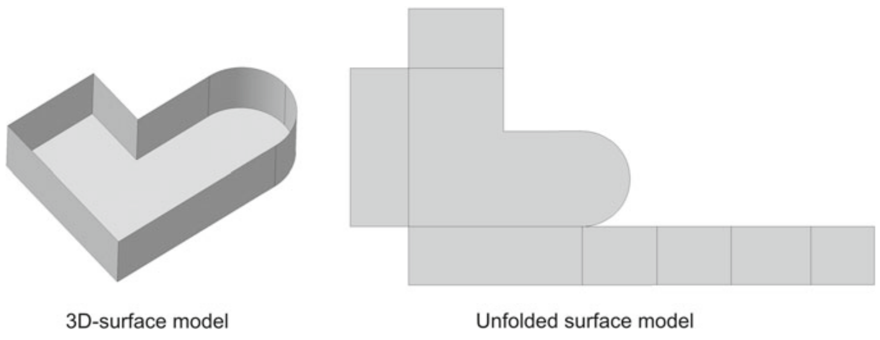


Fig. 4.27 Example of an unfolded sheet metal

axis g. Pattern operations can be combined with different subsequent operations like trimming by a surface Φ_E.

Besides fundamental functionalities for geometry creation, modern CAD systems also offer far-reaching operations for the support of production-related product modeling. These features often combine several basic functionalities for the formulation of problem-oriented and user-friendly operations in wireframe and surface modeling. Figure 4.27 shows one example of the automated development of unfolded sheet metals. In general, an existing surface model provides the basis for the creation of an unfolded model, which can be directly transferred into production engineering processes. One prerequisite for a failure-free development of unfolded surface models is the preparation of foldable parent surfaces. It is known that not all types of surfaces are able to be transferred into an unfolded planar surface model (cf. Sect. 3.7.10, p. 167). For the treatment of 3D-bent or free-form surfaces, modern CAD systems offer approximation techniques, which perform a segmentation of the 3D-surface patches and a subsequent transformation into 2D-unfolded models. These approximations always result in distortions, which have to be monitored and evaluated carefully for each application case.

Another important function handles the application of draft angles. Draft angles concern a specific orientation of surfaces for production purposes. The manufacturing process of deep-drawn parts, some types of cast parts and forged components makes the application of angled surfaces necessary. This is based on the molding process and depends on several production-related factors, such as the type of manufacturing, the component material, materials of molds, and mold lifetime. Typically, draft angles are 0–2° for plastic die-cast parts, 1.5–3° for aluminum die-cast parts, 3–5° for forged steel parts, and 1–5° for deep-drawn steel metal sheets. The bigger the draft angle, the better the mold durability and surface quality of the components, but increasing draft angles lead to increasing modification of the model geometry itself. Thus, the application of draft angles is always a compromise between the product geometry shape and the different requirements from production engineering. Modern CAD systems offer several features for the application of draft surfaces, but in general, a thorough knowledge of the manufacturing process is required for the proper design of these types of components. Figure 4.28 shows an example of a

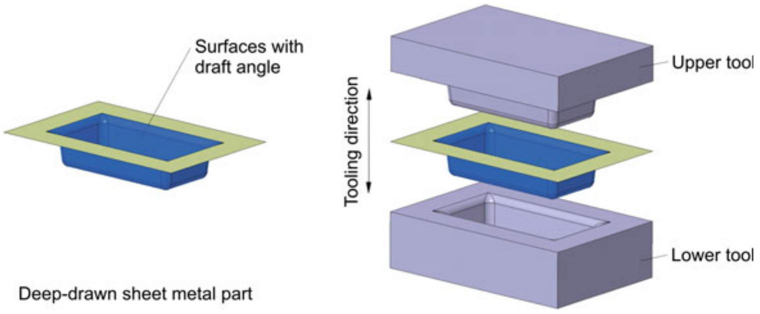


Fig. 4.28 Example of draft application

deep-drawn sheet metal part and the simplified mold components. Corresponding to the tooling direction, draft angles are applied on the affected surfaces.

4.2.5 Modeling in Wireframe and Surface Design

Depending on the CAD software package applied, the workflow in parametric-associative wireframe and surface design will vary, but the general modeling processes follows similar steps of geometry creation. The following sequence gives a software-independent overview of the design process in wireframe and surface geometry modeling using two examples. In these examples, the main steps are shown and briefly explained to introduce the structuring of surface models in automotive engineering. The examples are related to sheet metal body components. The design of plastic parts or aluminum cast components may differ significantly because of fundamentally different modeling strategies, which include solid bodies. The different possibilities for the integration of volume and solid models in surface design are discussed in Sect. 4.5.

In addition, the methods of parametric-associative surface design introduced here do not correspond with the techniques of CAS. Whereby CAD encompasses the technology-oriented engineering of product models, CAS is focused on the creation of styling surfaces for the definition of the product outline and its shape. In general, the processes can be distinguished as the development of product appearance on the one hand (CAS) and the development of everything that is *behind the skin* on the other hand (CAD). Of course, the two disciplines influence each other significantly, and it is clear that there are many instances of multiple functions and processes. In general, however, CAS uses non-parametric geometry creation for the development of smooth free-form surfaces.

Figure 4.29 shows an exemplary parametric-associative design process for a sheet metal cover. In the first step, the basis profile is defined in a two-dimensional wireframe model c_1. This wireframe model consists of lines and circles and can be

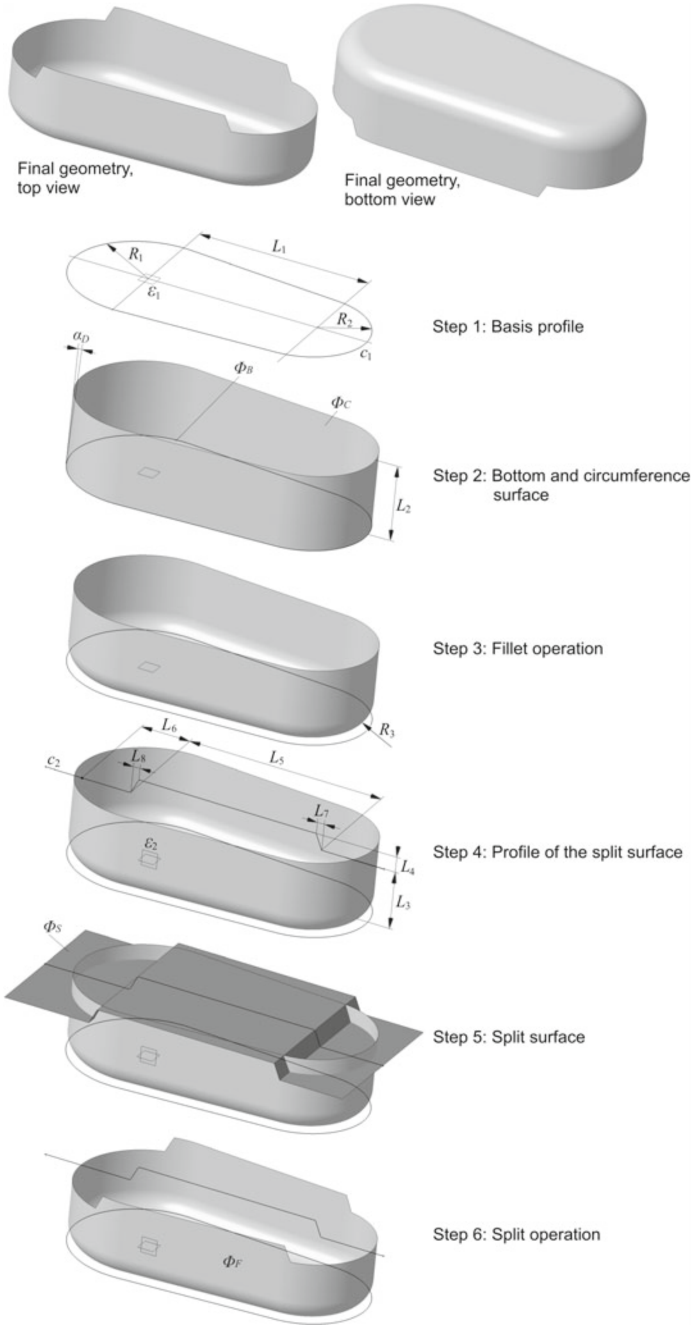


Fig. 4.29 Exemplary design steps of a sheet metal cover

defined using elements in 3D-working space or using sketch-based functions within a selected plane ε_1. In both cases, the corresponding dimensional parameters R_1, R_2 and L_1 control the geometrical extension of c_1. The basis profile is used to define the bottom and circumference surfaces in step 2. The bottom surface Φ_B is a planar surface in ε_1, which is framed by c_1. The circumference surface Φ_C is defined by an extension L_2 and a draft angle α_D, which takes the tooling process of the sheet metal part into account. Φ_C can be defined via different operations, which depend on the CAD software applied. Some software enables the creation of draft surfaces, which are built on basis surfaces. In the present case, the circumference surface is created by using an extrusion function, and in a subsequent step, the application of the required draft angle onto this extrude leads to the final drafted surface. One alternate possibility applies a translational surface operation, which sweeps a profile along the basis curve c_1. In this case, the profile is represented by a line with an angle of $90° + \alpha_D$ in relation to the bottom plane ε_1.

Step 3 shows the result of the application of a rounding surface with a constant radius R_3 and the subsequently performed trim operations. In step 4, the profile curve c_2 of the split surface Φ_S is defined. The definition follows the strategy of description of c_1 and is performed within a plane ε_2. The dimensions of c_2 are described by the parameters $L_3, \ldots, L_8$. This profile curve serves as a basis for an extrude operation, which defines the split surface Φ_S, as shown in step 5. The final step 6 includes the split operation with the goal of creating the final surface Φ_F. In a subsequent step, the surface model can serve as input for a thick operation, which converts it into a solid model under consideration of a desired wall-thickness. This operation is discussed in detail in Sect. 4.5.

The next example covers the general design steps for the creation of a more complex sheet metal panel. This panel is part of an engine carrier beam and is used in a personal car. Figure 4.30 shows the main steps of design but does not go into detail in terms of parameterization or the individual operations of surface creation. Step 1 handles the definition of reference elements, which characterize the general extensions of the panel and serve as a basis for different operations. These reference elements consist of an axis system and planes. Depending on the specific task and the modeling strategies applied, additional elements (e.g. lines, points, curves) may be used. Step 2 shows the basic surfaces of the relevant model, which are trimmed in step 3. Draft angles in relation to a predefined tooling direction along the z- axis are considered. In general, basic surfaces are designed somewhat larger than the final geometry to support some geometrical operations, such as split or trim functions. The trimmed basic surface provides a source for several subsequently performed operations, which finally form the resulting surface model. Due to the symmetry of the panel, only one half is designed and then mirrored in a final step.

In step 4, the deep drawn trays of the panel are created under consideration of the required draft angles for production engineering. Each element is composed of several surface patches, which are created using common functionalities (e.g. extrude, revolute, and sweep operations). In the present example, some of the trays have the same geometrical shape, so these can be designed individually and copied using transfer functions. The deep drawn trays are integrated into the surface model

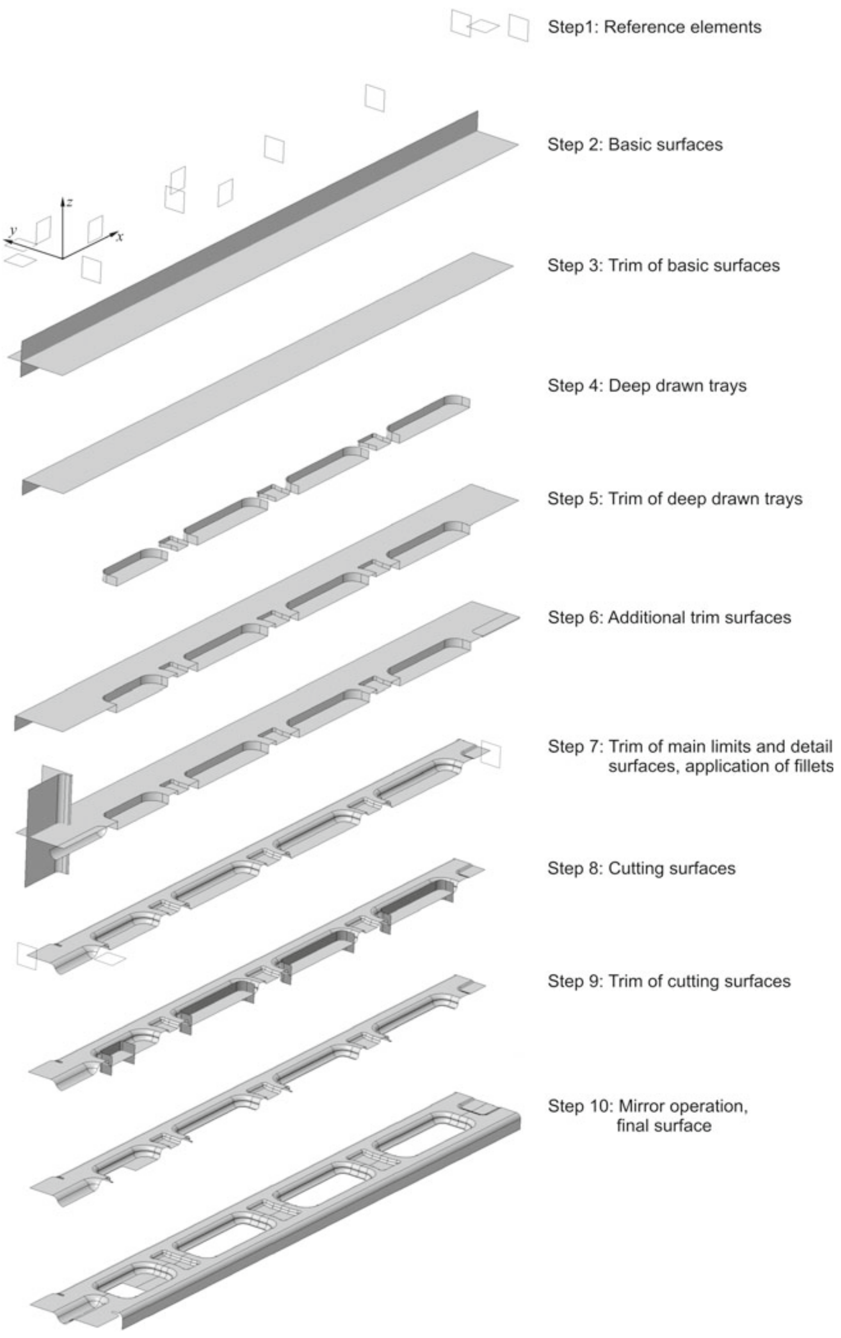

Step1: Reference elements
y
z
x
Step 2: Basic surfaces
Step 3: Trim of basic surfaces
Step 4: Deep drawn trays
Step 5: Trim of deep drawn trays
Step 6: Additional trim surfaces
Step 7: Trim of main limits and detail surfaces, application of fillets
Step 8: Cutting surfaces
Step 9: Trim of cutting surfaces
Step 10: Mirror operation, final surface

Fig. 4.30 Exemplary design steps of a sheet metal panel

by applying trim operations in step 5. Step 6 includes the definition of additional trim surfaces, which characterize several details of the panel. These trim surfaces are created under consideration of the desired draft angles and are positioned using the corresponding reference elements. After that, they are integrated into the panel model using trim operations.

At this point, the geometric main characteristics of the panel are implemented. The following steps handle the final detailing operations. Step 7 includes the result of several operations. Firstly, all the relevant shaping surfaces are integrated by split and trim operations, and secondly, the actual main dimensions of the panel are fixed by trim operations with the corresponding reference planes. Finally, step 7 also includes the application of fillet surfaces at the corresponding edges of the model.

The next step concerns the cut-out operation of inside holes. In the present example, specific surfaces serve as cutting elements. These surfaces simulate the stamping process in production and are integrated into the panel model using a split function (step 8). In step 9, the panel is completely defined and ready for the final symmetry operation. Step 10 shows the resulting panel, which includes all required information for a complete geometrical product description. The surface model can be converted into a solid model by applying a wall thickness. During further processes, additional information can be applied, such as the material characteristics and supplementary production-related information. The panel model itself is implemented within an assembly structure and serves as a basis for DMU-based investigations and crash and stiffness simulation. The panel model also serves as a geometrical boundary for the design of adjacent components and modules.

4.2.6 Surface Analysis Functions

Modern CAD software offers different analysis functions which enable extensive evaluation and optimization of wireframe, surface and solid models. In general, these functions are related to geometrical product characteristics. However, as the product geometry includes knowledge of different disciplines, the design analysis also covers topics of styling evaluation and production-related aspects.

An assessment of geometry within a CAD-environment begins with the visualization techniques. There are different types of visualization enabling diverse types of product display and thus several variants for the recognition of defects. Figure 4.31 shows a selection of different types of visualization in CAD.

For the following considerations on surface continuity we also refer to Definition 3.44, p. 149. In mechanical design the analysis of surface and wireframe geometry often focuses on the quality of geometry assembly and continuity. This way the distances between curve elements or surface patches are checked to avoid unwanted holes, gaps or overlapping wireframe or surface areas. The continuity characteristics represent a special case. Closed surface patches with edges in-between fulfill the lowest standard of continuity, the so-called GC^0-continuity which only ensures that the joined surface elements meet along an edge. The second degree GC^1 is represented

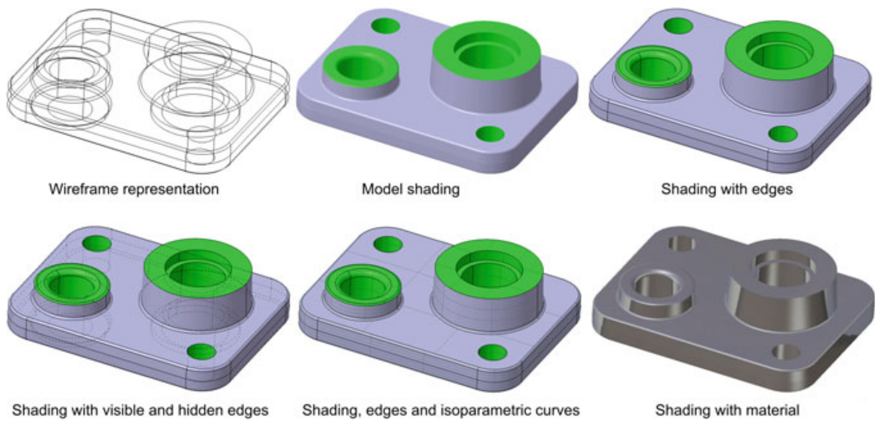

Wireframe representation
Model shading
Shading with edges
Shading with visible and hidden edges
Shading, edges and isoparametric curves
Shading with material

Fig. 4.31 Selected different types of visualization in CAD

by tangency continuity which is often applied in technology-based design processes. All standard operations of modern CAD systems work with GC^1-continuity, such as the application of fillet surfaces or the extrapolation of surface patches. Curvature continuity GC^2 describes the higher order of wireframe or surface continuity. Surfaces with curvature continuity are used in styling surfaces or in the surfaces of visible components. Due to their high visual quality, GC^2-surfaces show a smooth behavior even in the reflection of light. In addition to visual purposes, curvature continuity is applied in the design of components relevant to fluid dynamics, which require a smooth change of flow sections. Finally, the GC^3-continuity represents the highest degree of smoothness in wireframe and surface design employed in automotive engineering. GC^3 characterizes a smooth change of curvature all along a transition curve. Surfaces with GC^3-quality are used in some areas of automotive body design with particularly high claims to visual quality.

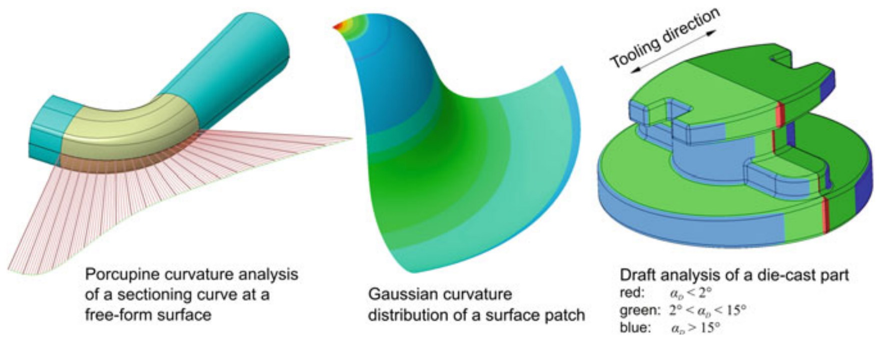

Tooling direction
Porcupine curvature analysis of a sectioning curve at a free-form surface
Gaussian curvature distribution of a surface patch
Draft analysis of a die-cast part
red: $\alpha_D < 2°$
green: $2° < \alpha_D < 15°$
blue: $\alpha_D > 15°$

Fig. 4.32 Examples of surface analysis functions

There are different functions for the analysis of curvature continuity. One type is the porcupine curvature analysis which enables a detailed observation of the behavior of curves or internal curves of surfaces by using a colored or a spike representation along the curve length. In Fig. 4.32, the image on the left shows an example of a porcupine curvature analysis of a section curve of a connection tube, which has been created as a multi-section surface. The length of the spikes represents a value for the curvature behavior. Another option for assessing the quality of surface progression is the application of surface-based curvature analysis, such as the Gaussian curvature distribution of a surface patch (cf. Definition 3.47, p. 155). This so-called surfacic curvature analysis displays different areas of curvature in different colors under consideration of the curvature direction (Fig. 4.32, middle). Similar to these exemplary functionalities, there are several additional features for surface analysis which support the evaluation of surfaces by displaying minimum or maximum radii, inflected areas, and others.

4.3 Solid Design

Solid modeling deals with the generation of virtual figures of products as compact three-dimensional models in a CAD environment. Unlike wireframe or surface models, which are based on sets of curves or surfaces, solid models are characterized by their representation as closed volumes (see also Sect. 3.12, p. 233). This enables a detailed description of the product specification beyond the purely geometrical aspects of the outer shape. For example, the definition of material characteristics enables a weight calculation, and the volume geometry can serve as input data for meshing operations or for direct derivations of tool path computation for numerically controlled production machines. In addition, the design process considers manufacturing-related requirements for the model structure, such as the subdivision of model components for casting or forging processes. Compared to wireframe and surface generation, the definition of solid models requires more complex computational procedures. It is not sufficient to define the outer geometry of a model; the hull surfaces have to satisfy closing conditions. In addition, geometry inconsistencies, such as overlapping surfaces or manifold solutions, have to be prevented.

To support solid modeling processes, modern CAD systems offer various functionalities which automatically include the complex mathematical algorithms for the definition of solid bodies. Solid models enable the calculation of volume properties directly from the geometry model. The volume properties are the volume, mass (as a function of the density), center of gravity, moments of inertia and products of inertia. The volumetric property calculation with a general function Ψ of the triple integral of a function $F(x, y, z)$ can be defined as

$$\Psi = \iiint F(x, y, z)\, dV, \tag{4.1}$$

which can be applied for the calculation of density distribution in a closed volume, for example.

Since the 1980s, different solid modeling methods and procedures have been introduced and discussed in the literature, while the fast-growing capabilities and frequently expanded functionalities of 3D CAD design software have led to a logical progression of software capabilities. Depending on the specific requirements of working fields in automotive engineering, solid modeling is combined with wireframe and surface design, thereby enabling a thorough virtual product representation. State-of-the-art design software packages enable a fusion of wireframe, surface and solid modeling techniques, which supports the application of integrated design methods. The following sections provide a brief overview of the main functionalities of solid design in modern 3D CAD software packages.

4.3.1 Modeling of Basis Solids

The composition of basis solids enables the development of the raw product geometry, which is defined in the initial steps. In subsequent detailing operations, this basis geometry is refined and detailed. Depending on which CAD software package is applied, there are different functionalities available for the formation of basis solid models.

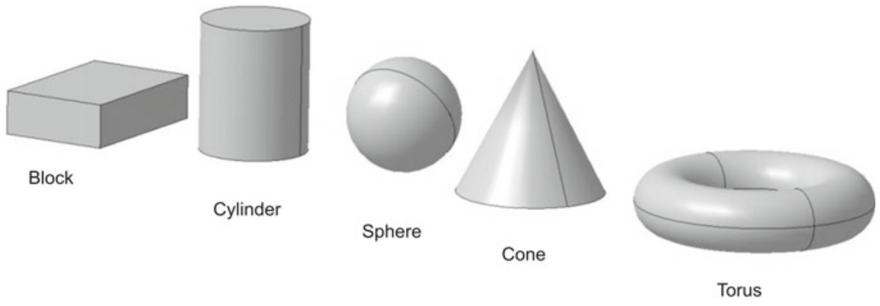


Fig. 4.33 Examples of primitives in solid design

The creation of primitives enables a simple and quick generation of solid models using predefined geometries. As an example, cuboids, cylinders, cones, torus etc. are created by inputting the corresponding geometrical parameters. Figure 4.33 shows examples of primitives in solid design. The computation procedure for these objects includes a user-defined input of the geometrical dimensions and a subsequent automated calculation of the closed outer surface, including identification as a solid model.

One alternative technique is the definition of primitive objects using two-dimensional sketches. In this technique, sketches describe the geometry of a section, which is used for the definition of three-dimensional geometries. The sketch-based

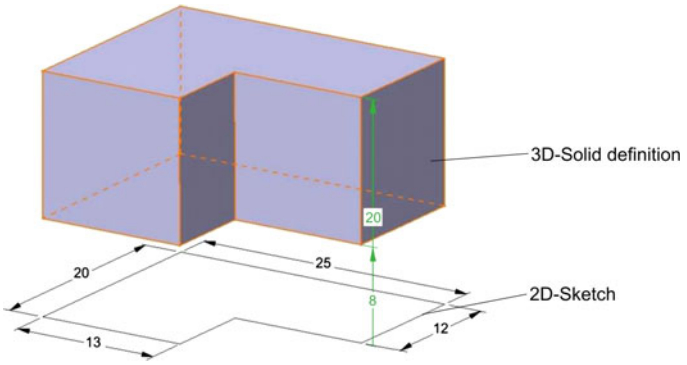


Fig. 4.34 Sketch-based solid definition

definition method offers a wide range of possibilities for geometry creation via user-friendly operations. Besides simple outlines for the generation of raw geometries, sketches can be used to create complex shaped drawings, which serve as input data for subsequent translational or rotational solid definition. Figure 4.34 shows an example of a sketch-based solid creation using a translational function. The object measurements are defined in a sketch (black dimensions) as well as in the translational function (green dimensions), which form a 3D solid from the 2D drawing.

The combination of several 2D sketches enables an easy-to-handle creation of basis solids with more complex hull geometries. Figure 4.35 shows an example of a multi-sectional solid definition. A closed volume is defined by a skin surface over a number of cross sections. Each cross section includes the geometry information defined in planar sketches or wireframe arrangements, and a trajectory gives the computation direction in the form of a center line. The skin surface is created by connecting the cross section using coupling points, which can be positioned individually to influence the shape and smoothness of the hull surface. A closing condition is fulfilled under consideration of the two cross-sectional surfaces at the ends. Depending on the applied 3D CAD program, there are different sketch-based functionalities available for the modeling of basis solids, but all of them refer to the same methods of translational, rotational, sweeping or skinning operations. Similar functionalities are available in surface design. The so-called multi-section surfaces are created in a comparable procedure by defining the outline geometries of the presented functionalities without considering the requirements of solid creation. In this way, there is no closing condition required, and surfaces can be created as separate elements and used for subsequent operations.

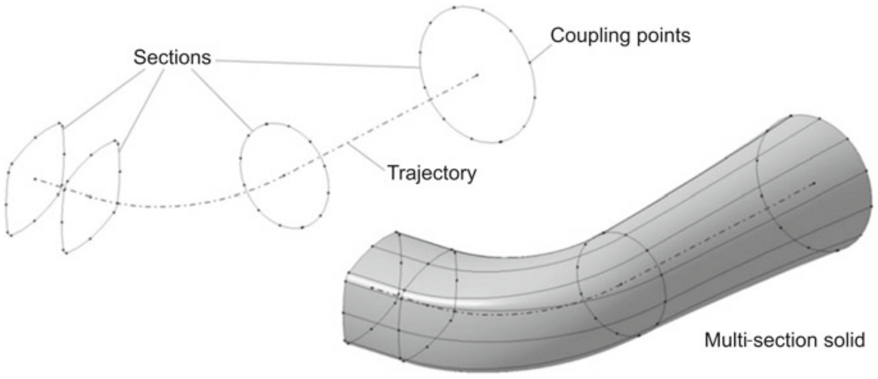


Fig. 4.35 Multi-section solid

4.3.2 Boolean Operations

Boolean operations make it possible to combine independent geometrical objects. In the case of solid modeling, the Boolean operations enable the union, subtraction and intersection of volume models. The union operation combines two existing solids into a resulting solid with the extension of the outer hull geometry of the initial solids. The resulting solid object has a closed character, which means that it can be handled as a solid primitive in further operations. The subtraction operation removes the volume of the subtracted model from that of the remaining model. The resulting solid has a reduced geometrical extension, as compared to the initial primitives. The premise for a failure-free application of a subtraction operation is the geometrical intersection of both treated solid models. Finally, the intersection operation calculates the conjoint volume of both contributing objects. Figure 4.36 shows examples of the application of Boolean operations in solid design with primitives.

Boolean operations with primitives are performed to enable the creation of complex geometry structures. Besides purely geometrical aspects, Boolean operations

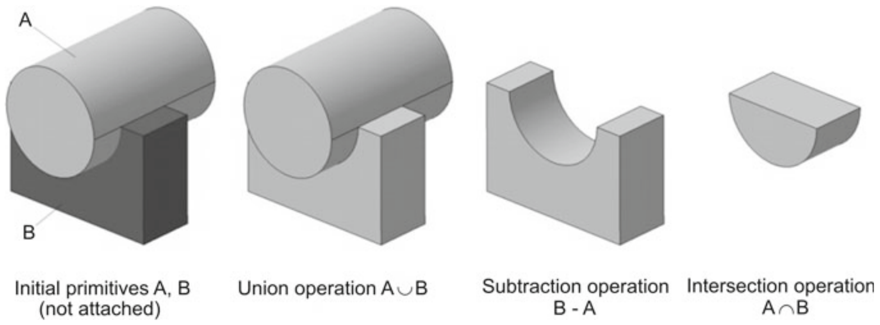

A
B
Initial primitives A, B (not attached)
Union operation A∪B
Subtraction operation B - A
Intersection operation A∩B

Fig. 4.36 Modeling of primitives with Boolean operations

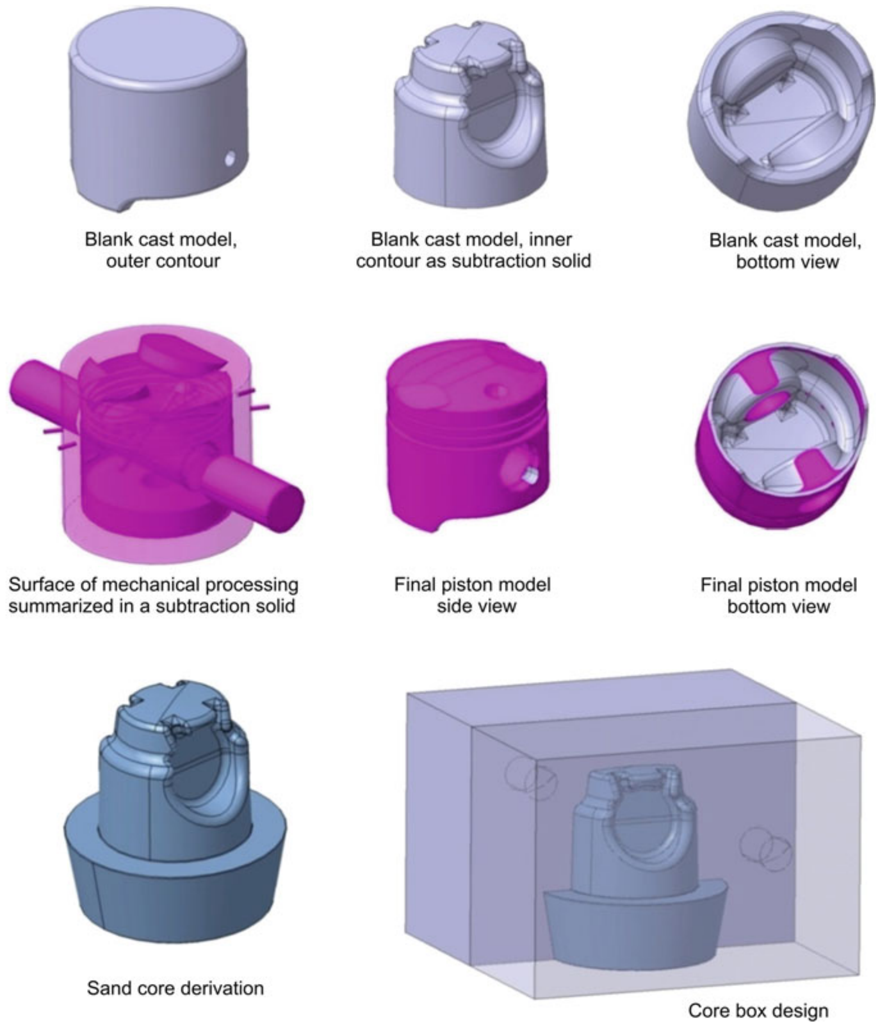

Blank cast model, outer contour
Blank cast model, inner contour as subtraction solid
Blank cast model, bottom view
Surface of mechanical processing summarized in a subtraction solid
Final piston model side view
Final piston model bottom view
Sand core derivation
Core box design

Fig. 4.37 Application of Boolean operations in production-related modeling of a piston

can be used to create specific model structures, which are often required for the consideration of production-related viewpoints. For example, the design process of die casting or forging parts has to consider the factors influencing mold design, including draft angles, rounding and of course the configuration of the mold components. Thus, the design of a product (or components of a product) which is produced in a die casting process has to consider several production-related aspects, and particularly the partitioning surfaces of the molds.

Figure 4.37 shows an example of a piston model, which will be produced by a die casting process and subsequent mechanical machining operations. Due to its relatively simple geometry, the piston shown can be cast in a two-parted mold with slider technology or by applying a sand core. One component of the mold defines the outer (cylindrical) shape, and the other component forms the inner shape. In Fig. 4.37, the inner shape of the piston is designed as a subtraction solid, which means that the geometry represents a *negative volume*. The inner shape of the piston is designed completely as an independent component, including all necessary geometrical details, such as draft angles and fillets. This solid is subtracted from the primary solid object, which defines the cylindrical outline. After performing the Boolean operation, the final piston model shows the characteristics of the mechanical part. For the completeness of the presented example, it must be mentioned that in the industrial engineering of a piston, the outline-defining solid is also created as a die cast mold.

The presented design method enables a direct derivation of the molds from the product model. The subtraction solid can be extracted in the design software and provided for subsequent design processes of molds and machines. In this way, the design process not only considers the product geometry creation, but also involves important aspects that have far-reaching influences on the entire product generation cycle. Besides the standard requirements of geometry and function, today's design engineers have to consider a variety of product-related information. The presented design method enables the consideration of draft angles, fillets and material addition for the subsequent mechanical machining processes. In the present example, the mechanical treatment is represented as an additional solid, which is subtracted in a final step.

4.3.3 Editing and Detailing Functionalities

The functionalities described above mainly serve for the creation of general geometry and for the application of production-related design strategies in principle. To enable a detailed geometry definition, modern CAD systems offer several additional functionalities. One main group of editing and detailing operations deals with the application of detailing-production-related aspects, such as draft angles, rounding and filleting, as well as the definition of threads. These operations are always related to existing geometries and modify them according the specified input data. While different software packages offer different user handling procedures, their functionalities are all quite similar.

Figure 4.38 shows an example of a sequence of detailing functions applied to a primitive solid. In the first step, a draft surface is attached to a basis surface by defining a draft angle. Draft operations apply inclined surfaces to existing geometries to enable demolding in casting, forging or deep drawn manufacturing processes (see also Fig. 4.28). Adding draft surfaces always modifies the outer contour of the model. Based on the draft angle and the model dimensions, the geometrical changes have to be considered and evaluated. The next step includes the application of rounding and fillets at specified edges. Rounding functions replace sharp edges with smooth,

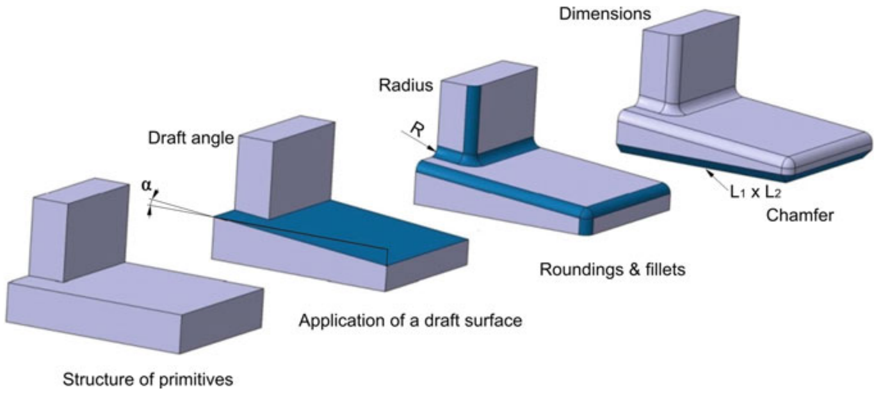


Fig. 4.38 Selection of editing and detailing functionalities

curved surfaces. In the design processes of mechanical products, the surfaces have a tangency continuation with a constant or variable radius. In the case of styling surfaces, rounding surfaces sometimes require enhanced shapes, such as curvature continuation surfaces (e.g. Bézier patches; cf. Sect. 3.8.1, p. 183), to achieve attractive outline shapes. The third example in Fig. 4.38 shows the application of chamfers, which are applied on predefined edges. In the case of mechanical processing, chamfers are used to avoid sharp edges. There are various additional editing and detailing functionalities available, such as for defining threads and taps, for implementing pattern, and for the scaling, translation or rotation of components.

4.3.4 Feature-Based Geometry Modeling

Feature-based modeling in 3D CAD uses predefined combinations of functionalities to create geometrical components. Features are created as model-building primitives, which are related to reference systems or to existing solid geometry. In this way, feature-based modeling uses associative design methods to create pockets, holes, ribs, slots, etc., whereby both the referencing behavior and the geometrical characteristics can be controlled by associated rules and attributes. In feature-based modeling, the creation of geometrical components is performed by interlinked sequences of functions, which include the definition of the feature type, the selection of reference elements, the input of required dimensions and the composition of the desired geometry models (surfaces and solids). Unlike stepwise design methods, features enable the definition of geometrical objects within one process. In addition to the geometrical characteristics, features can include additional information for organizational, manufacturing or distribution processes and support the derivation of data and information for subsequent engineering processes. In general, modern CAD systems offer two types of features. The first type includes standard functionalities offered by the software package, which enable a user-friendly creation of different geometrical

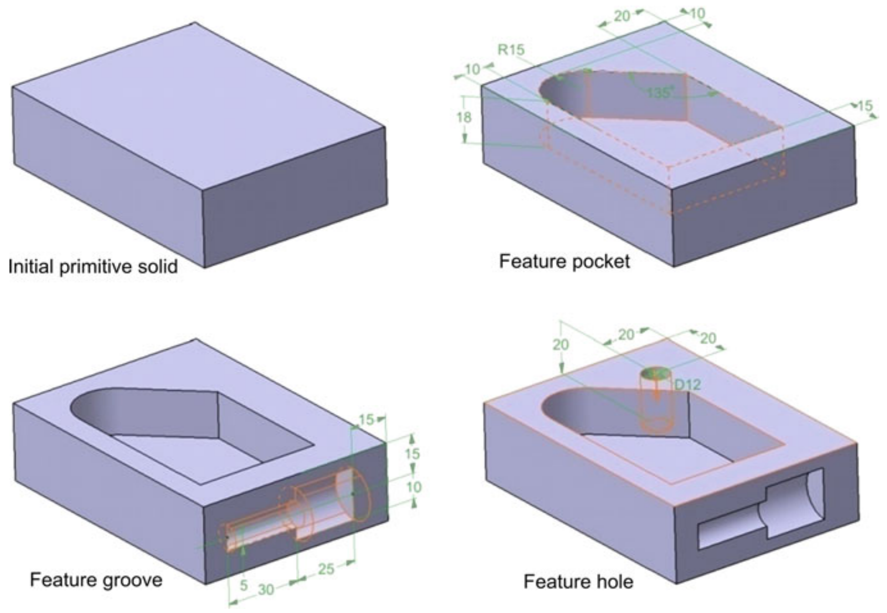


Fig. 4.39 Sample applications of feature-based modeling

elements (e.g. primitives, pockets, grooves, holes). The second type includes user-defined features, which are problem-oriented operations for the support of specific product design processes. Such user-defined features often include knowledge of design methods and product-related information.

Figure 4.39 includes examples of feature-based modeling. In the case of creating a pocket or a groove, the geometric characteristics are defined in two-dimensional sketches using wireframe elements and attached dimensions. Depending on the type of feature, these contours are used in subsequent operations, such as translation for the creation of a pocket or rotation for the creation of a groove. The example at the bottom right shows the application of a hole, which is positioned on a surface of the primitive solid. The borehole position is related to the existing edges of the primitive, and its dimensions (length and diameter) are specified within the feature-creation process. In the CAD program, the borehole geometry elements are marked as a borehole, which enables a direct data transfer to subsequent NC-production planning processes. Feature-based modeling is not restricted to solid design. State-of-the-art CAD systems also offer various methods for generating surface-based feature modeling to support integrated development processes.

A smart combination of functionalities in solid design supports an efficient definition of complex product models, which include more than purely geometrical information. In particular, the creation of cast and forged solid components requires that production-related aspects be taken into account during the design process. In mechanical product development, and especially in automotive development, this type of parts plays an important role, alongside sheet metal parts and plastic

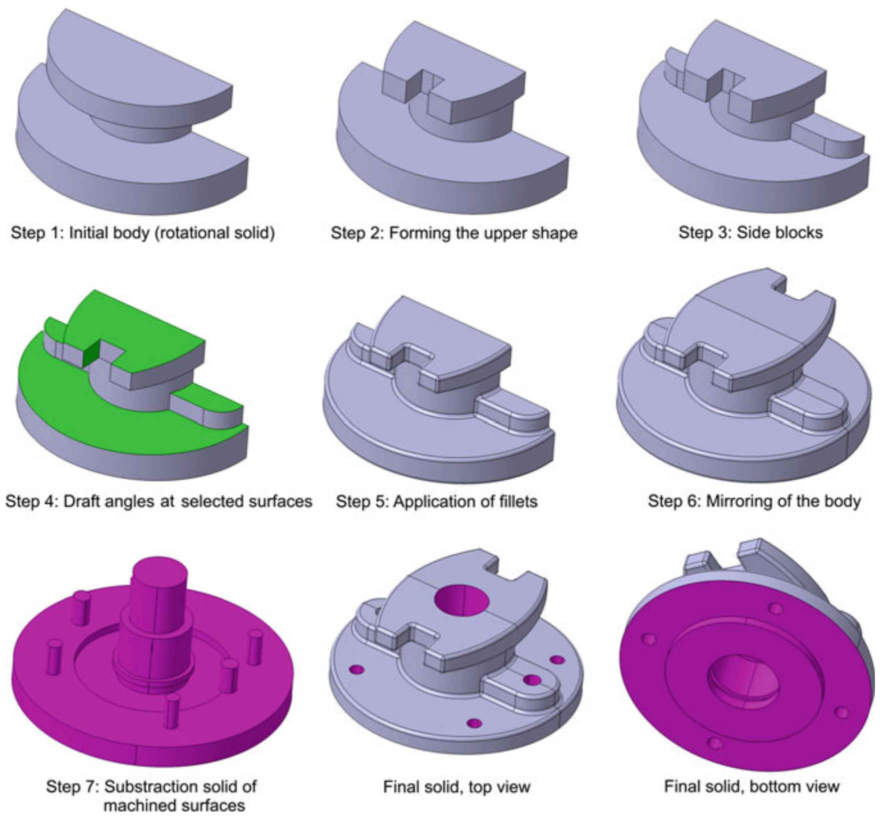

Step 1: Initial body (rotational solid)
Step 2: Forming the upper shape
Step 3: Side blocks
Step 4: Draft angles at selected surfaces
Step 5: Application of fillets
Step 6: Mirroring of the body
Step 7: Substraction solid of machined surfaces
Final solid, top view
Final solid, bottom view

Fig. 4.40 Exemplary design steps of a valve cover

components. Metal cast-parts or forged parts are used in drivetrains (e.g. cylinder head, engine block, transmission casing or others), as well as being applied as main components of suspension systems and as other load-related elements of a car. Plastic parts are often used in car interiors or as components of front or rear-end modules.

Incorporating production-related requirements of casting and forging processes in the design of mechanical products directly influences the geometrical shape of the models and thus the entire design process. Since the molds in modern development processes are directly derived from the product models, various required information for mold design has to be considered during product design. In this context, the appropriate application of draft angles and fillets at the corresponding faces and edges plays an important role. In mold design processes, these draft angles and fillets are directly integrated into the corresponding tools (Fig. 4.9, p. 255).

As a representative example, Fig. 4.40 shows the design steps of a valve cover. This valve cover is made of aluminum and manufactured by a combination of die-casting and subsequent machining operations. The design process starts with the definition of an initial body, which represents the primitive geometry. For symmetry reasons, several design steps are applied on a half-side model, which is mirrored in a later step. In the present example, the primitive geometry is described by a rotational solid, which serves as the raw shape of the model. Step 2 includes the forming of the upper area. This is performed by applying a pocket-operation, which removes material from the upper primitive solid. The next step applies the side block by adding and blending two extruded solids. In step 4, the draft angles are applied at selected surfaces. In the present example, a two-part mold is used for the die-casting process. In this way, the valve cover is separated into two main solids, whereby each of them is designed under consideration of a corresponding draft direction, which is normal to the symmetry plane of the valve cover. All surfaces with draft angles are colored green. After applying the draft angles, step 5 includes the application of fillets and rounding. The final raw cast model results from a mirror operation (step 6). It must be stated that the cast model considers different aspects of production (e.g. draft angles, fillets and additional wall-thickness) for subsequent mechanical machining operations. Normally, the specific measures required for the cast process itself (e.g. ducts for pouring in the liquid aluminum and holes to allow gas to escape during the casting process) are defined during production engineering processes. In this way, the CAD model shown in step 6 represents the raw geometry of the valve cover cast part as it results from the casting process and arrives in the machining department.

In the solid-based design of mechanical components, surfaces that result from machining operations are created separately from the geometry which results from non-cutting manufacturing processes. In many cases, all the surfaces that result from mechanical treatment are integrated into one solid and subtracted from the initial cast-part. This strategy enables a clearly arranged structure of design-related components within complex models. Step 7 shows a solid that includes all of the geometrical elements necessary for the definition of mechanical machined surfaces. This solid is designed as a comprehensive closed body and subtracted from the cast-part using a Boolean operation, which results in a final geometry of the valve cover.

4.4 Combination of Wireframe, Surface, and Solid-Based Functions

The design of the complex geometries and component assemblies that occur in automotive development requires the integration of wireframe, surface and solid geometries into one modeling process, which makes it possible to use each type based on its specific attributes. Thus, wireframe elements are used as reference elements (points, lines or curves and planes), which define positions in the working space exactly or serve as skeleton models for the assembly of modules and components. In mechanical design, surface elements are used for various purposes. Sheet metal parts and automotive body components, which are characterized by thin walls, are mainly designed as surface structures. The high requirements for the surface shape and its quality call for flexible geometry generation possibilities, which are available in surface design. Once surface models have been created, they can be converted into solid models by applying a wall thickness. This leads to the generation of three-dimensional models,

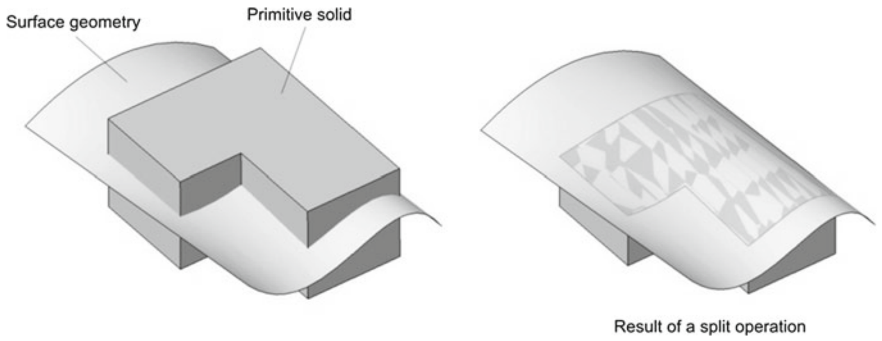


Fig. 4.41 Example of split operation of a solid model

which are suitable for subsequent analysis, such as accurate DMU studies and the application of material properties for weight calculation and subsequent CAE simulations.

Styling software often works with surface functionalities due to the high degree of freedom for the creation of smooth free-form surfaces with a high visual quality. In technical engineering processes, styling surfaces are delivered in neutral data formats and integrated into component design. For example, surface models can serve as split and trim objects for modifications of solid geometries. In state-of-the-art CAD systems for automotive development, solid models represent the basis for product representation in a virtual environment. In recent years, solid models have replaced the formerly preferred surface models in nearly all applications. The advantageous, *near-real* product representation of solid models enables a direct integration of DMU processes into the 3D CAD software environment. Figure 4.41 shows an example of a split operation of a solid model with a surface geometry. After it has fulfilled its function, the surface can be hidden, and the solid model can be used for further operations.

Thick operations enable another important integration of surface models into solid design. This type of operation is applied to transfer surface models (e.g. sheet-metal models) into solid models to enable the definition of wall thickness and to handle other related information (e.g. material) as well as to compute mass and center of gravity. Figure 4.42 shows an exemplary application of a thick operation onto a surface model (left). The final solid (right) considers the geometrical shape of the surface model, but has been modified by adding a wall thickness. The wall thickness is applied by an offset operation with a constant distance, and the space between the initial surface and the offset surface has been filled and converted into a solid. The application of thick operations places high demands on the quality of the underlying surface models. Smooth surfaces with no discontinuities or sharp edges are essential for failure-free transformation. In addition, the radii of the initial surfaces must have values below the desired wall thickness of the solid model to avoid geometrical errors.

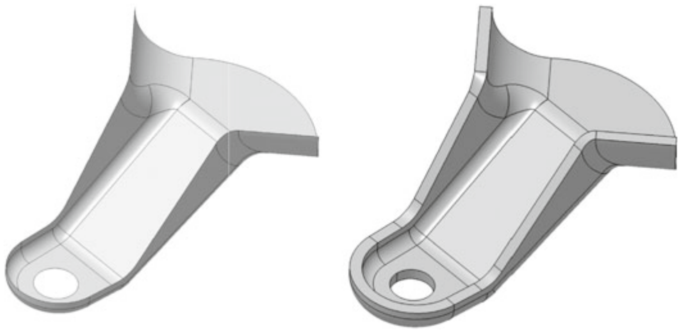


Fig. 4.42 Exemplary thick operation applied onto a sheet-metal model

The consideration of production-related aspects in sheet metal design requires the application of specific methods. Sheet metal parts are manufactured by applying deep-drawing operations, which form raw sheets into the desired shape (Fig. 4.28, p. 270). After the forming process, the excess material is cut away by punching machines. The punching machine stamps have working directions, which emboss the cutting edges of the sheet metal parts. Because the thick operations of CAD systems work with offset surfaces, which are created by perpendicular transformation, the areas of three-dimensional shaped edges are created as non-manufacturing-related solutions. Figure 4.43 shows an example of sheet metal design that does not take manufacturing-related edges into account. The closing surfaces of sheet metals at three-dimensional bent edges show a unidirectional progression, which does not consider the punching directions of cutting processes (arrows in the detail views of Fig. 4.43). If the CAD process has to consider production-related aspects, the design process for the sheet metal part must include several additional steps.

Unlike conventional design processes, the consideration of production-influences requires a completely different model structuring. In standard sheet metal design, a comprehensive surface model includes all geometrical information of the final geometry, except the wall thickness. The wall thickness is applied in a final step using thick

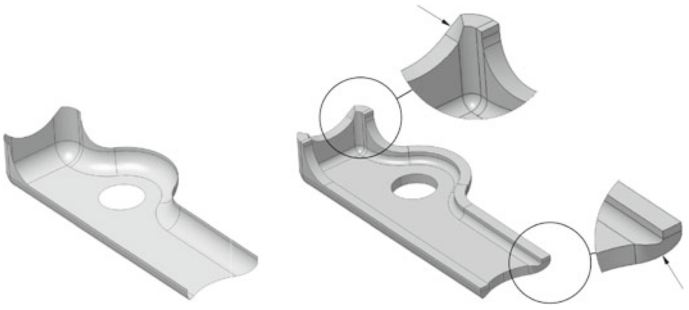


Fig. 4.43 Sheet metal design without consideration of manufacturing-related edge design

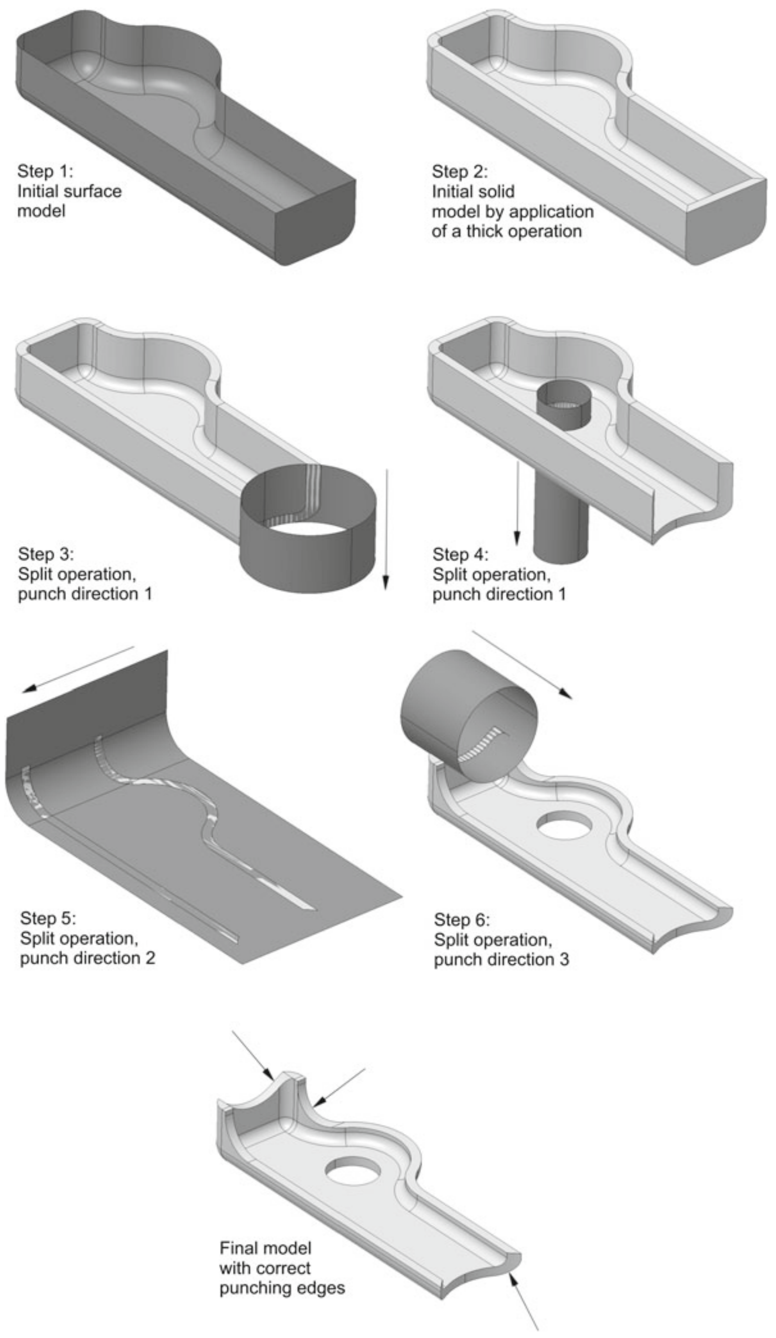


Fig. 4.44 Sheet metal design considering manufacturing-related edge design

operations, which convert the surface model into a solid model. A consideration of punch directions requires a stepwise design of the final geometry. Figure 4.44 shows an example of a method for sheet metal design that takes into account manufacturing-related edge design. In this case, an initial surface model defines the uncut geometry (step 1). The appropriate wall thickness is then applied to convert the surface model into a solid model (step 2), which in turn is split using separate surfaces, representing the geometry of the applied tooling. In addition, predefined punch directions define the cutting directions for split operations of the initial solid geometry. In the present example, the split operations of steps 3 and 4 cut away the right side area and the hole of the sheet metal part using cylindrical surfaces. Step 5 then forms the upper area by removing the corresponding material with a split operation supported by a translational surface, which represents punch direction 2. Step 6 produces the final geometry using a split operation that considers punch direction 3, represented by a cylindrical surface. The final model features correct punching edges. The arrows show the sensitive areas of the model, and a comparison with the model in Fig. 4.43 reveals the differences.

4.5 Assembly Design

Complex mechanical products are composed of several modules and components, which are arranged in logical sequences in so-called product structures. These structures define the order of different main and sub-modules and manage linkages and relationships between the components. Besides geometrical design-related aspects, product structures provide information for the generation of organizational lists and tables (e.g. the bill of material (BOM)). Modern CAD systems include specific assembly-related workbenches for the creation and organization of product structures. In this way, modules and components in 3D CAD are managed within an assembly-design environment. The product structures created contain links between the individual components and their relations to each other. An assembly-oriented design strategy in automotive engineering is based on a division of the product structure into several sub-assemblies, which supports an easy examination of component positions and intersections.

In general, assembly design includes the following tasks:

- Arrangement of modules and components within 3D CAD
- Definition of product structures for the organization of modules and components
- Positioning of modules and components in the working space and in relation to each other
- Definition of relations and links between modules and components, including the creation of geometrical dependencies (so-called multi-model links) and the implementation of logical interconnections
- Implementation and organization of component-comprehensive parameter structures

- Provision of geometry and structure information for subsequent digital mock-up processes
- Provision of geometry data for assembly-based simulation processes (e.g. multibody simulation, MBS)
- Interaction with product data management (PDM) systems

Digital mock-up investigations are carried out on the basis of 3D CAD product structures, which include the placement of all involved components within the working space. The 3D CAD assembly model provides all geometrical information for the subsequent derivation of DMU-models, which represent a simplified shape of the modules and components involved, including the product structure. For example, a sub-assembly of a vehicle DMU can represent the body, including all movable and fixed parts. The described sub-assembly of an automotive body itself consists of several sub-assemblies (e.g. side panel modules, the roof module, doors, and some other products and parts). Stripping down complex structures into sub-assemblies consisting of multiple components provides a basis for simultaneous design processes that take into account various functional and spatial requirements.

4.5.1 Organization of Product Structures

The organization of product structures consists of several levels, which include specific combinations of autonomic or linked components and modules. These levels are defined under consideration of reasonable compositions of CAD models. In most cases, a technical background prescribes the assembly structuring. Some examples of factors that influence the distribution of components and modules are the classification into main and sub-modules, assembly sequences in the production process and other manufacturing-related aspects (e.g. the integration of supplier deliverables). Figure 4.45 shows an exemplary product structure that includes several hierarchical levels. The main product is composed of components and modules which are arranged within the main level 1. Components are single parts that are not split any further. In 3D CAD, these components mainly represent indivisible elements, such as cast parts, machined elements or standard parts. In many cases, the assembly of components results in a module or, in the case of a simple product, in a main product structure. Complex mechanical products are composed of numerous elements, whereby modules and sub-modules include separate configurations of components and sub-modules in reasonable configurations.

In the present example, module 2 consists of a component 2.1, a sub-module 2.2 and other components and sub-modules within level 2. Sub-module 2.2, in turn, is composed of component 2.2.1, sub-module 2.2.2 and other components and sub-modules within level 3. This method can be expanded such that even very complex products can be assembled in a logical and clearly arranged order. It is important that each level of the product structure is completely defined, including the creation of geometrical positioning constraints between all components and modules, as well as

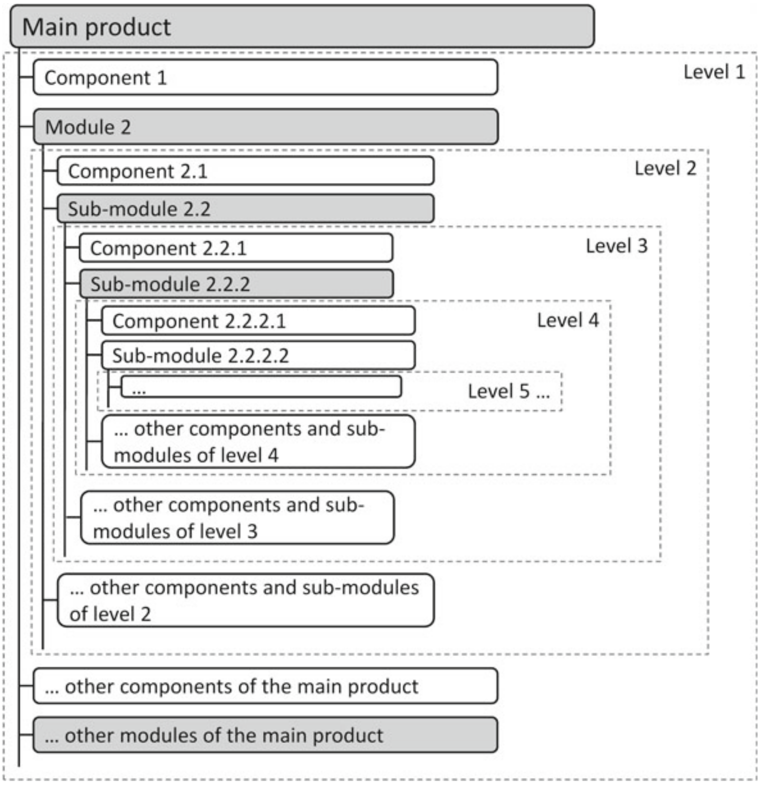

Main product
Component 1
Level 1
Module 2
Component 2.1
Level 2
Sub-module 2.2
Component 2.2.1
Level 3
Sub-module 2.2.2
Component 2.2.2.1
Level 4
Sub-module 2.2.2.2
...
Level 5 ...
... other components and sub-modules of level 4
... other components and sub-modules of level 3
... other components and sub-modules of level 2
... other components of the main product
... other modules of the main product

Fig. 4.45 Exemplary product structure including several hierarchical levels

the definition of linkages and logical relations between the elements. Similar to the complex hardware models that occur in the automotive industry, the product structure has to represent a meaningful distribution of individual elements with the target of defining the complete system. Although the product structure often represents assembling sequences in the manufacturing process, there are several exceptions, such as the automotive wiring harness, which is structured in a separate sub-module of the main vehicle product structure. In contrast to this separate structuring, hardware wiring harness elements are assembled according to the meaningful configuration in the relevant modules and are interconnected with the main wiring at the vehicle assembly line.

In addition to the theoretical background of product structuring, the following example describes a practical application of assembly design in automotive engineering. Figure 4.46 shows an assembly of an automotive front axis composed of a McPherson suspension strut and a lower suspension arm, a wheel hub, a rim and the tire, as well as the brake system, the steering system and the drive shaft. Altogether, this exemplary front axis consists of 6 modules. Figure 4.46 shows the front axis

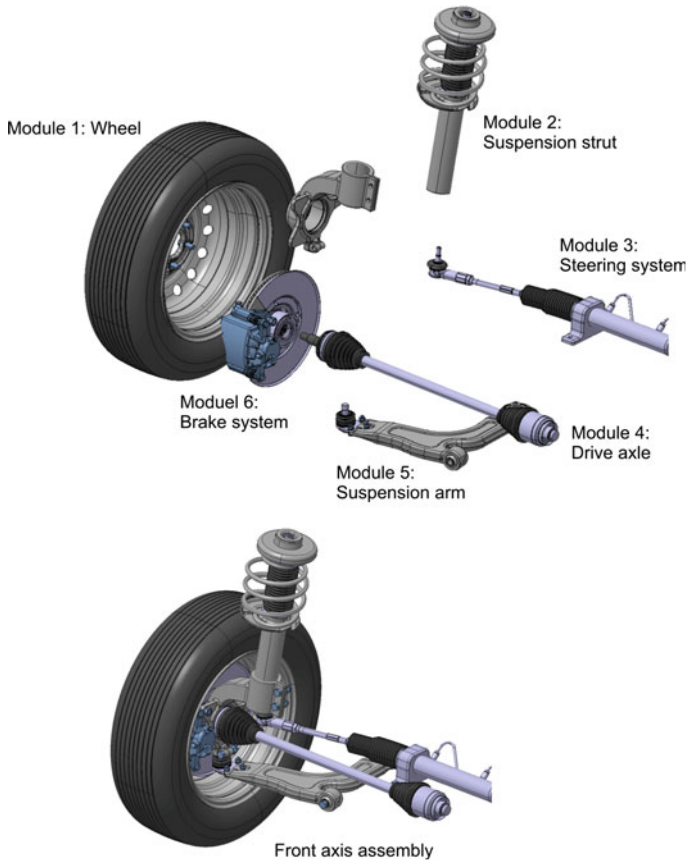


Fig. 4.46 Front axis assembly

assembly and an expanded view of the 6 main modules. Module 1 represents the wheel unit, consisting of the wheel itself, the rim, the valve, the balancing weights and the wheel bolts. As in the other modules, these components are positioned in relation to each other using geometrical constraints. Module 2 includes the complete suspension strut as it is delivered by a supplier. Module 3 consists of steering system components, including the actuator, the link lever and the joints. Module 4 represents the complete drive axle, including pivots, bearings and sealing. Module 5 represents the lower suspension arm, including all fixed and movable parts. Finally, module 6 represents the brake system, including the wheel hub and the wheel carrier. Module 7, which is not shown in Fig. 4.46, includes all screws and standard parts for the assembly of the 6 main modules. These modules are arranged in the main level of the product structure (see Fig. 4.48).

To explain a breakdown of main modules into sub-divisions, Fig. 4.47 shows the assembly of module 6, the brake system with the hub and wheel carrier. Module 6

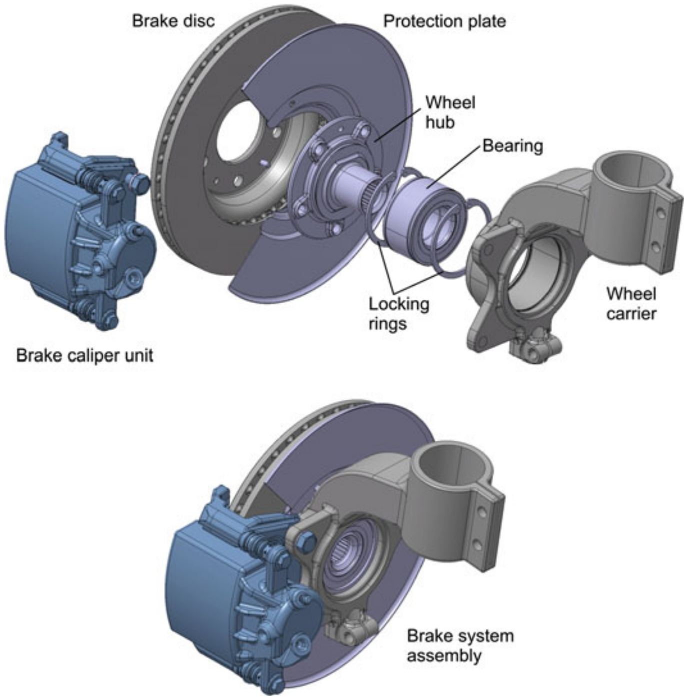


Fig. 4.47 Assembly of module 6, brake system including hub and wheel carrier

consists of components of both the brake system (e.g. the brake disc, the brake protection plate and the brake caliper unit) and the wheel hub (e.g. the bearing and locking rings and the wheel carrier). The brake caliper unit itself, which represents a further sub-module, is also composed of several components, which are listed in Fig. 4.48 under the corresponding header *Brake caliper unit*. Finally, Module 6, which includes a set with the screws required for assembly, is represented in the product structure as *Brake system screws*. The geometrical positioning of the components and sub-modules is accomplished using different types of constraints, which are explained in Sect. 4.5.2.

Figure 4.48 shows the product structure of the exemplary automotive front axle assembly. For a better understanding of the structuring, white boxes indicate components, and grey boxes indicate modules. The front axis assembly represents the main level, which is divided into 7 sub-modules. With the exception of module 7 (*Screws*), all sub-modules consist of several components and modules, which are arranged in a logical order. As an example, the configuration of module 6, *Brake system and hub*, is illustrated in detail. Module 6 consists of 7 components (*Brake disc*, *Protection plate*, *Wheel hub*, *Bearing*, *Locking ring 1*, *Locking ring 2* and *Wheel carrier*) and two modules (*Brake caliper unit* and *Brake system screws*). The sub-module *Brake caliper unit*, in turn, is composed of 15 components, which are also listed in Fig. 4.48.

Finally, the sub-module *Brake system screws* includes all screws and standard parts for the assembly of module 6.

In a complete automotive product structure, the front axis assembly discussed above might represent one sub-module of the Suspension module. The Suspension module, in turn, could be a main module in an assembly structure, which contains the main modules of a car. This so-called automotive main structure could consist of the main modules Body, Exterior components, Interior components, Propulsion system, Energy storage system, Suspension, Electronics and Others. It is evident that the overall product structure of a product such as a car reaches a very high degree of complexity, which has to be managed carefully during the development process of a new vehicle, and also during the rest of its life cycle.

Besides the nomenclature of different modules, the structure contains a clearly dedicated numbering of each element. This is very important for the product structure management, the unambiguous definition of the corresponding CAD data, and a clear definition of the complete assembly in a PDM-system. Of course, the form of numbering and element title can vary based on the different requirements of the specific project, the software environment, the manufacturer regulations or other influencing factors, but the general structure includes a clearly defined name and number of all elements. Some PDM systems work with element numbers only, but the corresponding names are stored in a database to support the user-orientation.

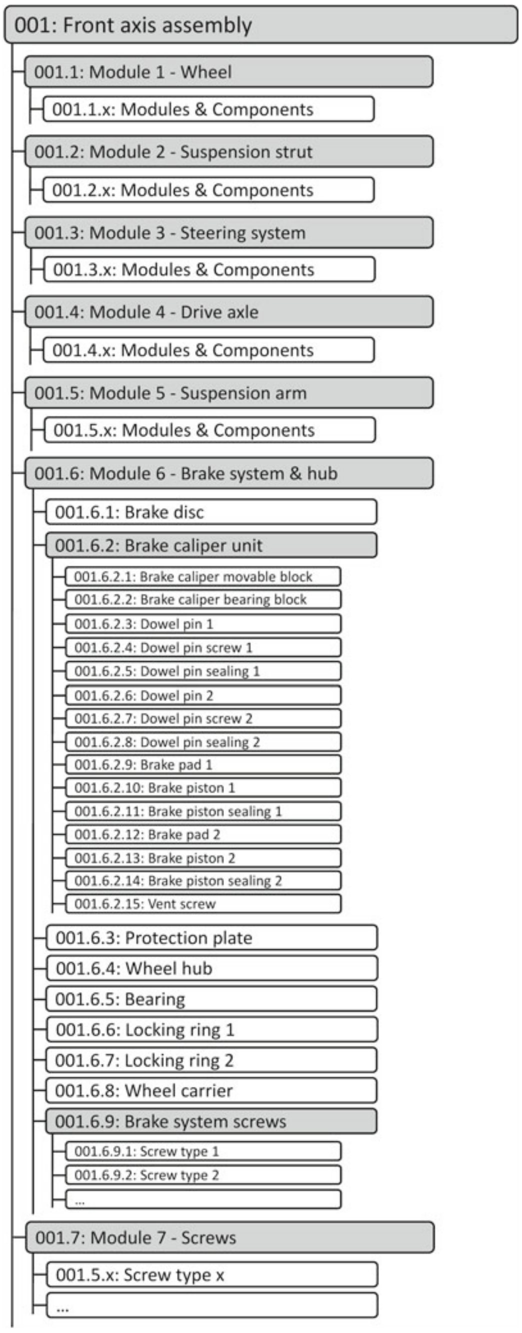
4.5.2 Methods of Component Positioning

The geometrical positioning of components and modules within the working space and in relation to each other is an important task in 3D CAD. In general, individual elements or groups of elements can be translationally and rotationally moved in space by applying different types of functions. However, these simple movements do not define any geometrical relationships of positioning. Geometrically binding positioning can be accomplished in different ways.

In the case of small assemblies, it makes sense to define geometrical constraints directly between the corresponding components. Typical constraints define distances, coincidences or angles between geometrical elements (e.g. points, lines, planes or surfaces). Thus, the components of a product can be positioned with reference to each other, with the constraints being subject to parametric-associative laws.

More complex assemblies can be built up by using dedicated positioning components within the product structure. The so-called skeleton method does not use constraints between the components. Instead, the positions of the components are defined relative to an auxiliary construction, the skeleton. The geometrical elements of the skeleton model therefore define the positions of each component in space and relative to the other components. The reference elements of skeletons are typically lines, planes or points. Tall assemblies use several sub-products geometrically combined with the help of a number of skeletons, thereby generating a modular design.

Fig. 4.48 Product structure of an automotive front axis assembly


001: Front axis assembly
001.1: Module 1 - Wheel
001.1.x: Modules & Components
001.2: Module 2 - Suspension strut
001.2.x: Modules & Components
001.3: Module 3 - Steering system
001.3.x: Modules & Components
001.4: Module 4 - Drive axle
001.4.x: Modules & Components
001.5: Module 5 - Suspension arm
001.5.x: Modules & Components
001.6: Module 6 - Brake system & hub
001.6.1: Brake disc
001.6.2: Brake caliper unit
001.6.2.1: Brake caliper movable block
001.6.2.2: Brake caliper bearing block
001.6.2.3: Dowel pin 1
001.6.2.4: Dowel pin screw 1
001.6.2.5: Dowel pin sealing 1
001.6.2.6: Dowel pin 2
001.6.2.7: Dowel pin screw 2
001.6.2.8: Dowel pin sealing 2
001.6.2.9: Brake pad 1
001.6.2.10: Brake piston 1
001.6.2.11: Brake piston sealing 1
001.6.2.12: Brake pad 2
001.6.2.13: Brake piston 2
001.6.2.14: Brake piston sealing 2
001.6.2.15: Vent screw
001.6.3: Protection plate
001.6.4: Wheel hub
001.6.5: Bearing
001.6.6: Locking ring 1
001.6.7: Locking ring 2
001.6.8: Wheel carrier
001.6.9: Brake system screws
001.6.9.1: Screw type 1
001.6.9.2: Screw type 2
...
001.7: Module 7 - Screws
001.5.x: Screw type x
...

In addition to the two positioning strategies mentioned above, a different method has been established in automotive development. This strategy consists of arranging each component relative to one pre-defined main coordinate system. The positions are defined in the course of the part design process using a startup model, which includes the main coordinate system. Sub-modules can be placed by creating sub-coordinate systems, which are related to the main coordinate system. In automotive development, the main vehicle coordinate system for design and assembling processes is placed in the symmetry plane near the front axis in most cases. Of course, it is also possible to combine the different positioning methods. To avoid problems during the design process, it is essential to clearly predefine the component positioning strategies in the course of the project planning.

Positioning Using Constraints

Geometrical constraints for positioning include different types of functions. The first group, the so-called coincidence conditions, defines the congruent alignment of two geometrical elements. This can relate to the concentricity of two circles or the coaxiality of two rotational surfaces (e.g. for the assembly of a bolt in a bore-hole). In addition, coincidence conditions cover the placement of two geometrical elements exactly at the same place (e.g. the alignment of two points or two planes directly upon each other). The application of coincidence constraints to two coordinate systems covers both the placement of the points of origin and the association of the coordinate system orientations.

The second group, known as offset conditions, enables the positioning of two geometrical elements with a distance in between. These conditions are applied to points, planes or planar surfaces under consideration of orientation vectors, which define the direction of distance. The third group, angular conditions, covers the creation of angles between planes or lines. In both cases, the orientation angle between the elements must be clearly defined. In the case of two lines, it has to be ensured that the corresponding lines meet at one point. Parallel elements are defined using offset conditions.

In order to position components and modules within 3D CAD working spaces, the reference elements must be clearly defined. This can be accomplished by assigning at least one fixed part in space, which then represents the basis for the subsequently performed geometrical assembly process. This fixed part should be positioned unambiguously and non-relocatably in the working space to avoid problems with the location of assemblies and sub-assemblies in later steps. Modern CAD systems provide specific functions for defining geometrically fixed parts and creating combinations of rigid configurations.

Figure 4.49 shows an exemplary application of positioning constraints for the assembly of a crank-train model. The left side shows the individual components, while the right side shows the combined crank-train model after assembly. In this example, the rack is the fixed reference element. All other components are then placed using coincidence constraints and offset constraints based on the logical order of assembly. The configuration shown places the elements such that a rotation around

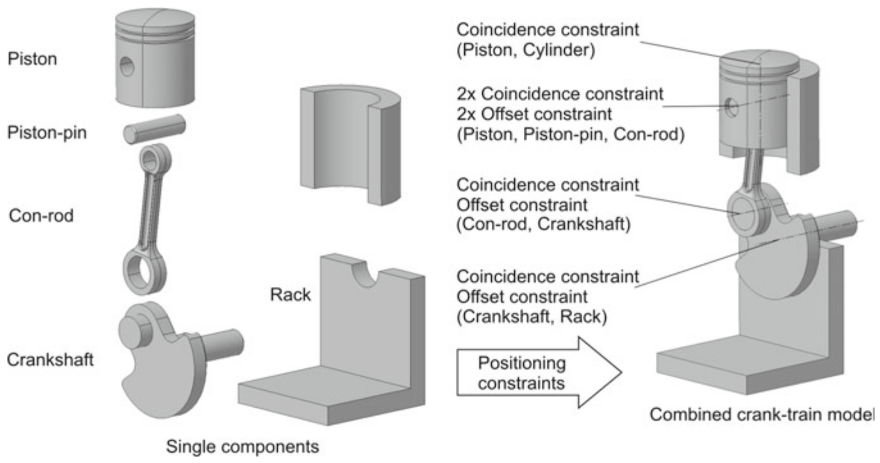


Fig. 4.49 Exemplary application of positioning constraints for the assembly of a crank-train model

the crankshaft axle is still possible. Therefore, the crank-train could execute a translational movement of the piston along the cylinder axis.

Figure 4.50 shows a detailed view of the coaxiality constraint and offset constraint for the assembly of a crankshaft and bearing block. In the first step, the axis a of the crankshaft is aligned with the axis b of the bearing cylinder using a coincidence constraint (coaxiality). The second step defines the axial position of the crankshaft bearing pivot in relation to the outer bearing cylinder of the rack using an offset constraint, which is defined between the two planes ε_1 (crankshaft) and ε_2 (rack). The positioning method does not restrict the rotation of the crankshaft along its axis a, so this degree of freedom could be used in later steps to simulate the crank-train mechanism.

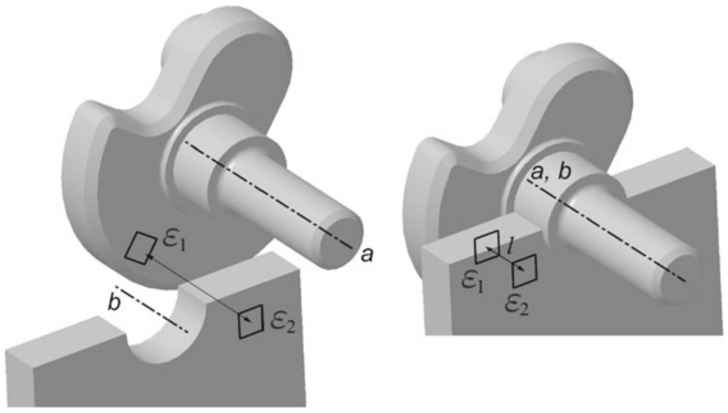


Fig. 4.50 Exemplary application of positioning constraints for the assembly of a crank-train model

Positioning Using Skeleton Models

Skeleton models are separate components in product structures which enable an efficient positioning and control of other components and modules. The positioning process is accomplished by applying constraints, whereby the constraints are not defined between the corresponding components and modules, but rather link each element with the skeleton model. In this way, all positioning constraints are related to the skeleton model. A geometrical modification of the skeleton model (e.g. lengths or distances) leads to a geometrical modification of the product structure positioning and, in special cases, can lead to a modification of the geometry characteristics of the components themselves.

In the product structure, the skeleton model is first placed in the corresponding level and clearly named. The skeleton itself is not a design model in the classic sense because it does not include the geometry of a part or a module. For this reason, this type of model has to be managed separately. By providing the ability to predefine the positions and some geometrical extensions of several parts within an assembly, skeleton models offer centralized functionalities for the design process. The applications of skeleton models range from the positioning of elements in simple models to the management of complex assemblies. Tall product structures use several skeleton models, which are interlinked using parametric-associative connections.

Theoretically, skeleton models can include any geometrical elements. However, to provide an efficient setup, they are mainly composed of points, lines, planes and coordinate systems, which define the geometrical positions of all the components and modules of the product structure. Figure 4.51 shows the application of a skeleton model of a scooter assembly. For better visualization, the geometrical elements of the skeleton model are displayed in red. In this example, the skeleton model consists of lines and planes only. These lines and planes define the positions of all components, which are mounted on the main part, the scooter frame. In addition to the positioning function, the scooter frame itself is related to the skeleton model using parametric-associative design methods. In this way, the main dimensions of the scooter frame (e.g. the steering head angle, the swing arm mounting position or the suspension frame position) can be modified by adapting the skeleton model.

Positioning Using Main Coordinate Systems

In automotive full-vehicle development, the positioning of components and modules using main coordinate systems has been established for many years. This method supports the efficient creation of complex assemblies by simply loading the corresponding elements into the product structure. The positioning process is carried out automatically because all components and modules are designed in the right place in relation to a main coordinate system. In this way, all elements automatically take their correct positions in the 3D working space and in relation to each other. The advantage of this method is obvious: Because every element is placed in relation to a main coordinate system, there is no need for a separate positioning effort.

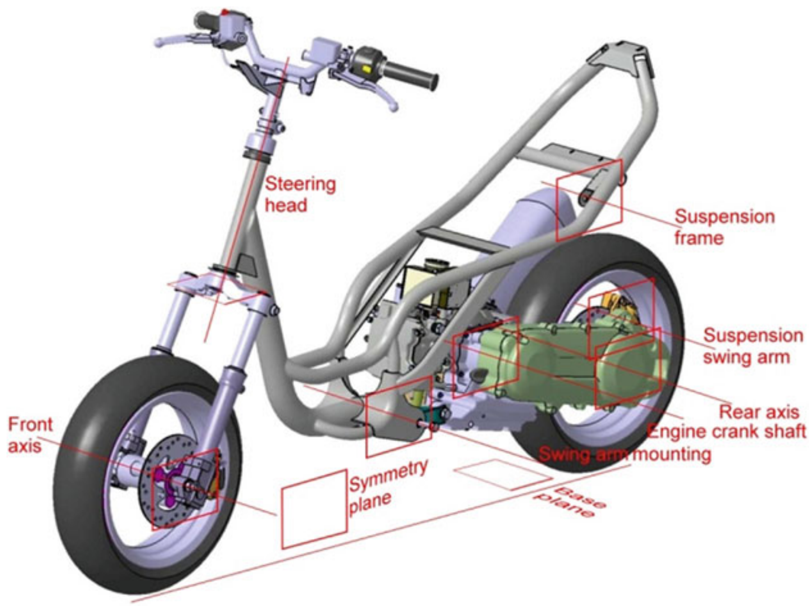


Fig. 4.51 Elements of a skeleton model of a scooter assembly

The management of components and modules within the product structure is lean because there are no additional relations or links between the elements. In general, a very stable and safe data structure is generated, which can be relatively easily managed by PDM-systems. The disadvantages are the very complex handling in the case of design modifications or later geometrical adjustments. In such cases, every element of the product structure has to be moved to its new position, which requires a substantial effort. In addition, this method prohibits any logical or geometrical links and relations between the components, which are state-of-the-art in modern parametric-associative design processes. In this way, the full potential of modern CAD systems cannot be exploited dues to restrictions in data linkage across the product structure.

In general, the user can define the location of the main coordinate system of a vehicle, but in most cases, it is positioned in the middle of the front axis. Because right-handed coordinate systems are used, the x-direction points to the back side, the y-direction to the right side (in the direction of travel) and the z-direction to the top of the car. This so-called design coordinate system differs in its position and in its orientation from the standard automotive coordinate system, which is positioned at the full-vehicle center of gravity (according to ISO 70000 [5] or SAE J670e [6]). Due to advantages for full-vehicle-related simulation and calculation procedures, several CAE processes use the automotive standard coordinate system, which makes it necessary to transform the coordinates between the main coordinate system for design and the general standard coordinate system.

In the product structure, the main coordinate system represents both the origin for the design process and the reference orientation for component positioning. Every component and module which is positioned in the main product structure has its own main coordinate system as a reference element. The design process itself is accomplished using different additional coordinate systems and wireframe elements, but all of them refer to the main axis system. During the assembly process, the main coordinate systems of all elements are aligned to each other to achieve the correct positions.

Figure 4.52 shows an exemplary full-vehicle assembly of a heavy truck model. The full vehicle model has its main coordinate system in the middle of the front axis, x_0, y_0, z_0. The positions of all main modules are related to this coordinate system. In the example, the rear axis assembly, which has its own reference coordinate system $\overline{x}$, $\overline{y}$, $\overline{z}$, is positioned in relation to the main coordinate system x_0, y_0, z_0.

Advantages and Disadvantages of the Three Positioning Methods

The three positioning methods have their specific pros and cons, and automotive development processes therefore combine the three methods to achieve optimal assembly configurations. Small assemblies with limited numbers of elements are often assembled using positioning constraints. Because it defines unambiguous

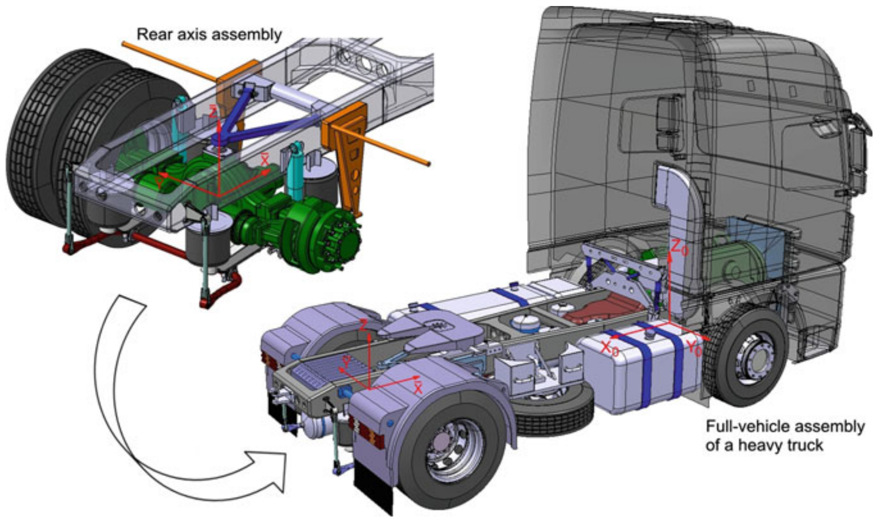


Fig. 4.52 Exemplary full-vehicle assembly of a heavy truck model including a rear axis assembly [7]

joints, this method is also applied when kinematics functionalities have to be taken into account. However, in the case of large product structures with numerous elements, the efficiency of positioning using constraints has limitations: the effort required to interlink all elements geometrically expands; and the management of elements and constraints becomes difficult. On the other hand, the introduction of skeleton models into complex product structures supports both the positioning of numerous elements and the structuring of complex products. Since both the location and the order of each element are related to the corresponding skeleton model (using constraints), the organization of tall assemblies follows the order of the applied skeleton models. A careful structuring of complex assemblies and a division into reasonable sub-modules at the beginning of the design process provides the foundation for an effective application of the skeleton method.

Finally, positioning in relation to a main coordinate system has the advantage of lean data structures, which are controlled by the product structure itself. Since there are no geometrical constraints or complex skeleton model structures, this method enables an easy assembly of nearly any desired complexity of product structures. However, and here is the main limitation, every elements must be designed in its correct place. In automotive engineering, main coordinate systems are applied for the positioning of components and modules in the case of full-vehicle assembling, which involves a significant number of main modules. The main modules represent closed units, which define the vehicle body, the engine, transmission, suspension, etc. These main modules, in turn, are composed of several sub-modules and components, which are positioned using different strategies (e.g. the use of geometrical constraints on the application of skeleton models).

4.5.3 Geometry-Based Interlinks in Assembly Design

Geometry-based interlinks in assembly design enable a direct control of geometry between different parts. Within the design process, a spanned connection of geometry can be carried out in different ways. To avoid potential problems due to the complex interconnection of geometrical characteristics, a careful planning and management of interlinks are essential. Modern CAD systems support the creation of interlinks using several functions. In general, they can be divided into direct geometry derivation, the creation of geometry references and the implementation of adapter models. All these functions are based on the parametric-associative relationship of the relevant geometry, such that any modification of the initial elements leads to an adaptation of the derived geometry as well. These so-called parent-children relations increase the complexity of the design process, but they offer significant potential for the efficient creation of complex product models.

As an example, consider the cylinder and piston of an internal combustion engine, which are two separate parts, each of which is characterized by its diameter. In the case of the piston, the piston outer diameter defines the piston size, and in the case of the cylinder, the inner diameter defines the bore of the engine. For a failure-free operation, the piston should fit perfectly into the cylinder bore, meaning that both diameters should have the same value. In a parametric-associative design process, the outer diameter of the piston can be derived from the inner diameter of the cylinder using a geometrical connection. If the cylinder diameter is then modified during the design process, the piston diameter will reflect the modification and be adapted automatically.

This simple example highlights the complexity of geometrical interlinks in the design of complex products. Because product models contain numerous geometrical characteristics that could be connected, the complexity of parametric-associative CAD models increases rapidly, which leads to a decrease in their stability in the case of modifications. To avoid complexity problems, the implementation of parent-children relations should be planned carefully, and the geometrical linkage between components should be kept to a reasonable level. The lower the number of interlinks, the lower the model complexity, which normally leads to a higher model stability. Nevertheless, this technology offers far-reaching possibilities for the improvement of design processes, and it will therefore become more and more important in the future design of complex mechanical products.

Geometry-based interlinks enable the linking of different parts such that modifications to one part will automatically result in modifications to the other part(s). In addition, these functions are used to define the relative positions between the relevant components, since the geometry derivations are performed within the 3D working space. When geometrical interlinks are applied, it is important to consider the comprehensive dependencies of the parts involved. Problems can occur if both the derivation of geometrical elements and the creation of binding positioning strategies (constraints or skeleton models) are applied to the same components.

Besides a parametric-associative derivation of geometrical dependencies, modern CAD systems enable the development of non-associative geometry. Non-associative geometry represents elements which are not parametrically defined and are not related to any other geometrical object. These elements are used as reference elements or for the representation of independent geometry within parametric-associative model structures. In general, all types of geometrical elements (i.e. wireframe, surface and solid models) can be defined as non-parametric, but the restricted possibilities of this type of geometry within modern design methods limit its application in CAD. In automotive engineering, some guidelines prescribe the application of non-parametric geometry in the case of geometry-based interlinks or geometry derivation between different parts in order to reduce the model complexity and to avoid unwanted dependencies in complex model structures. In such cases, all external geometry references are performed with no parametric-associative linkage. Geometries derived from external sources appear as frozen snap-shots, which cannot be parametrically modified. These geometrical elements are imported into the design environment of the relevant part and can then be used for further design steps. If these frozen geometries are modified, they must be imported again from the supplying part(s).

Figure 4.53 shows an example of the implementation of external geometry into part design by direct geometry derivation. Part 1 represents a simplified cylinder of an internal combustion engine, and Part 2 represents the corresponding piston

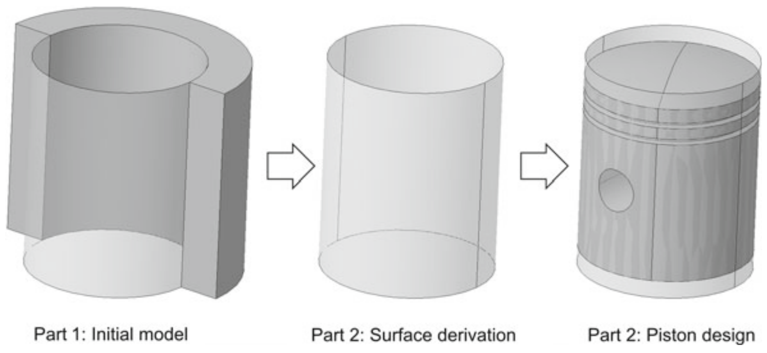


Fig. 4.53 Cross-part geometry derivation

model. With the goal of controlling the diameter of the piston, the cylinder surface is extracted from Part 1 and imported into Part 2. Within the environment of Part 2, the piston is designed using the derived cylinder surface as a reference. If the cylinder diameter is modified, the associative reference surface in the piston model is adapted, and the related piston design is modified as well. The applied design method must be related to the reference surface in such a way that the geometry model will remain stable if modifications are made. This behavior should be ensured for a defined range of adaptation. In the present example, the piston model can handle modifications of the cylinder diameter between 40 and 80 mm. Below or above these values, the parametric-associative configuration of the piston would lead to update errors because of geometrical miscalculation. It is advisable to include variation ranges when planning the design process of parametric-associative models, in order to avoid potential problems in modification cycles or optimization loops.

Figure 4.54 shows an example of design using imported geometry references within the assembly of an internal combustion engine. The exemplary engine has an inline 6-cylinder configuration with an aluminum cylinder block and a magnesium bed plate. The one-piece forged crankshaft is made of steel and carries 6 units of con-rod and piston. At the rear end of the crankshaft, a flywheel is mounted. In the design process of the crankshaft model, several geometry references have been adopted from the cylinder block model to support the positioning in relation to the bearing surfaces of the crankshaft and con-rods. This is accomplished by transferring specific geometry reference elements from one model to the other. In the present example, the engine centerline and several reference planes are imported into the crankshaft model to serve as indicators for the design process. In this way, the positions of significant components of the crankshaft can be directly controlled by the crankcase model. The part-comprehensive import of reference elements is somewhat similar to the application of skeleton models. Similar to skeleton models, wireframe elements (points, lines, planes, etc.) are used to control geometry. However, unlike the implementation of a skeleton into an assembly, the application of interlinked reference

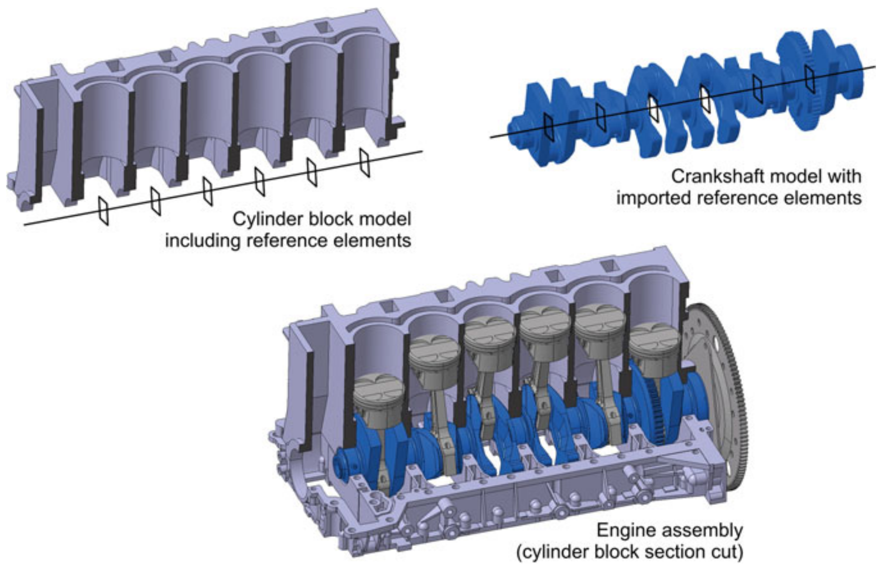

Cylinder block model
including reference elements
Crankshaft model with
imported reference elements
Engine assembly
(cylinder block section cut)

Fig. 4.54 Inline 6-cylinder engine assembly

geometry is always restricted to the components concerned. No separate model is needed in the product structure.

Similar to skeleton models, adapter models are separate components in assembly structures and enable an efficient control of geometrical information in several parts. However, unlike skeleton models, adapter models do not define the positioning of components, but rather have direct access to the geometrical characteristics of the components. In this way, adapter models serve as geometric control elements for components within a hierarchical sub-structure in assemblies. Adapter models can include different types of geometries (i.e. wireframe, surface, and solid elements) and can be a part of the assembly as available geometrical component.

Adapter models summarize geometry data from the sub-ordinated structure and provide this information for subsequent processes by publishing predefined geometrical elements. In general, they are used to control geometrical elements in the concerned parts, but they can also be involved in component positioning and the structuring of design processes.

Figure 4.55 includes an exemplary application of an adapter model in the design process of engine components. In this example, the adapter model includes the geometrical information of the sealing flange for the crankshaft bearing unit. This geometrical information is provided to the two concerned components of the bearing unit, the cylinder block model and the bedplate model. In the assembly, the lower flange of the cylinder block is bolted to the upper flange of the bedplate. The adapter model is positioned in between these two components and defines the geometrical extension of the flange. Due to the parametric-associative linkage of both concerned

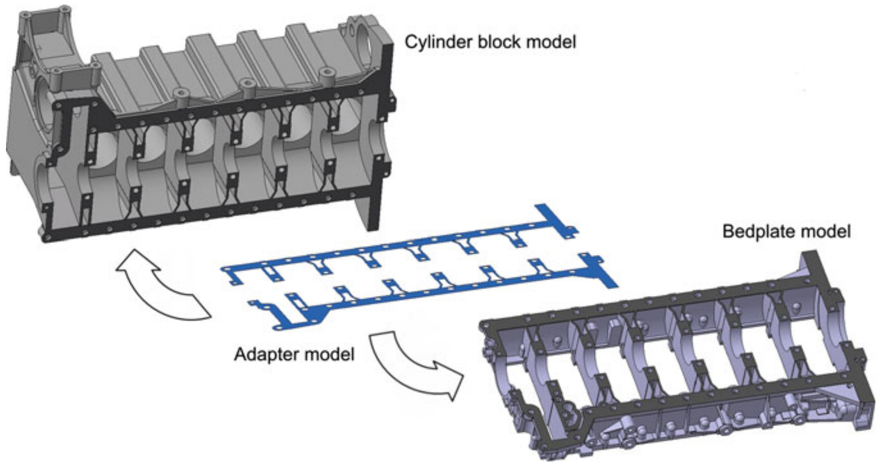


Fig. 4.55 Exemplary application of an adapter model in engine design

flanges with the adapter model, the flange geometry of the cylinder block and the bedplate fit perfectly to each other. Modifications to the flange geometry are made in the adapter model, which leads to adjustments in the corresponding components.

4.6 Derivation of 2D Drawings

Modern CAD processes derive two-dimensional (2D) drawings directly from the corresponding 3D models. 2D drawings serve as product documentation, for the development and evaluation of different product characteristics, and of course as workshop drawings for production purposes. The derivation of drawings from 3D CAD data is accomplished by creating views and sections. In addition, specific functions enable the creation of 2D geometry for additional drawings of supporting geometry, but these tools are not designed to create 2D drawings manually.

Once views or sections have been created, they can be used for dimensioning, tolerancing, the application of additional information and text boxes, etc. Modern CAD systems provide several functions for the efficient creation of workshop drawings, including frame, title block, bill of material (BOM), and all the other elements of standardized 2D documentation. Besides efficient support for the creation of drawings, different operations enable an automated application of standardized features, such as the application and adjustment of dimensions, the definition of standards, and the calculation of weight, volume and other relevant information. Besides native data formats, different types of neutral data formats are used for the exchange of drawing data, including Drawing Interchange File Format (DXF), Initial Graphics Exchange

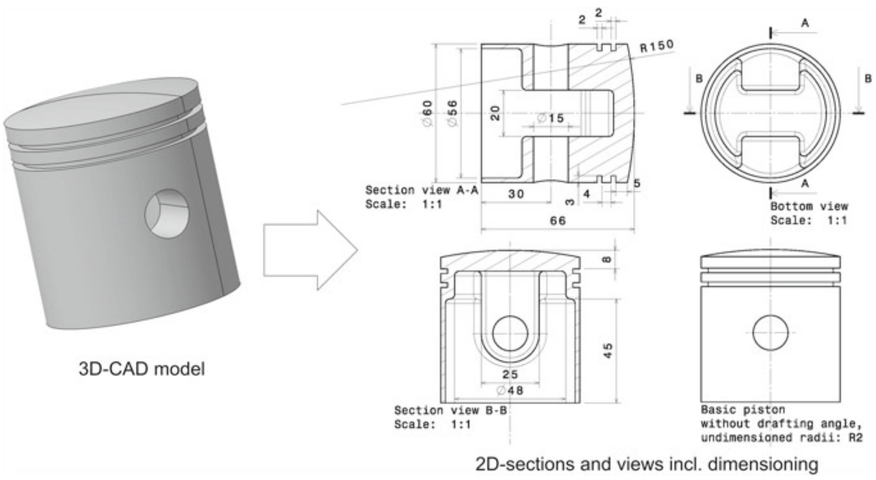


Fig. 4.56 Derivation of a 2D drawing from a 3D model

Specification-2D (IGES-2D), Post Script (PS), Portable Document Format (PDF), and others.

In recent years, the trend has moved in the direction of 3D tolerancing and annotations, which incorporate complete product-related information into the corresponding 3D CAD models in order to avoid additional 2D drawings. Although this trend should continue in the coming years, the application of complete 3D product description will be restricted to specific areas of products that allow for a consistent digital product and production description. In cases where manual working steps are a part of engineering processes, the derivation of 2D drawings will continue to be an attractive technology.

Figure 4.56 shows an exemplary derivation of 2D views and sections from a 3D piston model. The drawing includes several dimensions and annotations for manufacturing. A complete workshop drawing for production would contain some additional information (e.g. material, specific surface treatment information, part number). These data are displayed in an additional title block and/or a BOM.

Figure 4.57 shows an actual example of a workshop drawing including different types of information (e.g. dimensioning, tolerances, surface treatment). This drawing is one of several drawings of a motorcycle cylinder head, which has been developed and prepared for serial production and shows the complexity of manufacturing-related 2D dimensioning.

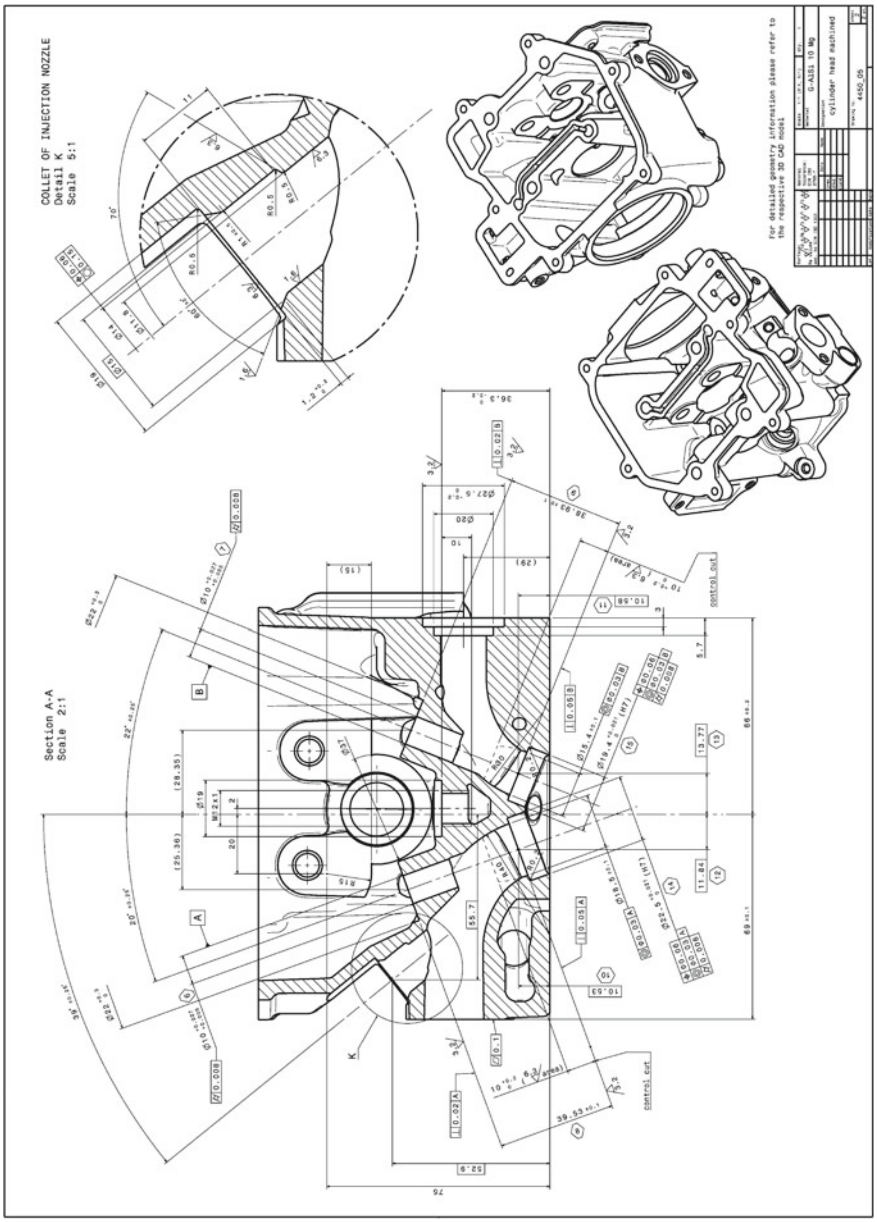

COLLET OF INJECTION NOZZLE
Detail K
Scale 5:1
Section A-A
Scale 2:1
control cut
For detailed geometry information please refer to the respective 3D CAD model
G-AlSi 10 Mg
cylinder head machined
4450_05

Fig. 4.57 Detail Workshop drawing of a motorcycle cylinder head [8]

References

1. SFE - SOLUTIONS FOR EXCELLENCE, Gesellschaft für Strukturanalyse in Forschung und Entwicklung mbH/SFE CONCEPT: date of access: 2010–07-01. http://www.sfe-berlin.de
2. Dassault Systems: CATIA V6. Date of access: 2009–11-10. http://www.3ds.com/products/catia
3. Parametric Technology Corporation: date of access: 2010–04-29. http://www.ptc.com/products/proengineer
4. Siemens PLM Software: date of access: 2010–04-29. http://www.plm.automation.siemens.com/
5. International Organization for Standardization: Road vehicles, vehicle dynamics and road-holding ability, vocabulary. ISO 70000 (1994)
6. Society of Automotive Engineers: SAE Recommended Practice: Vehicle Dynamics Simulation Terminology. SAE J670e (1952)
7. Stadler, S.: Aerodynamische Optimierung von Fernverkehr Sattelzügen. Diploma Thesis, Graz University of Technology, Austria (2010)
8. Korman, M., Hirz, M., Kirchberger, R.: Low Emission High Performance 4 Stroke Scooter Engine - PartII. In: Research Report at the Institute for Internal Combustion Engines and Thermodynamics at Graz University of Technology, Graz (2006)

Chapter 5
Knowledge-Based Design

Knowledge-based engineering as a part of knowledge management includes a technology-oriented focus on methods and tools for the support of product development. Knowledge-based design concentrates on product design and its related procedures. In essence, knowledge-based design supports design processes by re-using predefined methods, algorithms or results, and it is integrated into specific tasks or workflows that are involved in the design processes. In addition, since the knowledge-related design methods and tools applied often contain company-specific information and knowledge, confidentiality must be maintained. This chapter focuses on the possibilities for the development and integration of knowledge-based methods and tools into design processes. A detailed introduction and discussion of knowledge, knowledge management as well as the corresponding strategies and systems is included in Chaps. 6, 7, 8 and 9.

The application of a specific knowledge-based engineering solution within an existing project environment follows the sequences of input, (partially) automated development procedures and output. In many cases, the output includes new information for the extension of the method or tool applied. The re-implementation of previously generated knowledge (experience) and the implementation of automated routines within design processes improve product development and can lead to a significant reduction in development effort.

Knowledge-based design methods and tools can include rigid or variable geometry data, the integration of calculation and simulation procedures into the design process, or the application of problem-oriented software solutions that can be integrated into the design environment. Although one main advantage of knowledge-based design is that it makes existing, proven solutions available for specific tasks, this also involves some drawbacks that must be acknowledged. For example, an initial effort is required to create template models, algorithms or program sequences, and continuous maintenance and update procedures are then needed to keep knowledge-based design tools current with the latest state of development. Finally, the use of previously created solutions to generate knowledge models may reduce the potential for creativity in some cases. In particular, the re-use of existing geometry models affects several design-related aspects and can inhibit the development of outstanding new solutions. Nevertheless, in the long run, the potential gains in power and efficiency that knowledge-based design offers outweigh the drawbacks.

Knowledge-based engineering applications are provided as independent software solutions or integrated into the relevant design or simulation software. In the field of knowledge-based design, the trend is moving towards the direct integration of solutions into the applied CAD software. These solutions are often based on existing functionalities of the design environment and can include specially created or programmed features. Besides geometrical modeling tasks, knowledge-based applications can provide functionalities that integrate procedures which were previously accomplished via separate calculation or simulation software. For example, integrated computation algorithms can be used to pre-calculate the dimensions of mechanical components under consideration of load conditions and material characteristics. To provide another example, a kinematic mechanism can be integrated into the design model to enable the layout of movable machineries and the computation of motion characteristics and space requirements.

Due to its wide range of functionalities and applications, one finds different definitions of knowledge-based design in the relevant literature. On the one hand, knowledge-based design starts with the parameterization of geometrical objects in the course of the design process. On the other hand, the creation of extensive, problem-oriented simulation algorithms within a design environment represents a transition into complex software applications. In this way, the creation of different types of knowledge-based design methods and tools enables the handling of the specific task. Figure 5.1 shows different types of knowledge-based design applications grouped in terms of the effort required for creation and maintenance, as well as the level of complexity. The class of rigid geometry models includes different types of 2 or 3-dimensional non-parametric geometry models, which are provided for re-use in

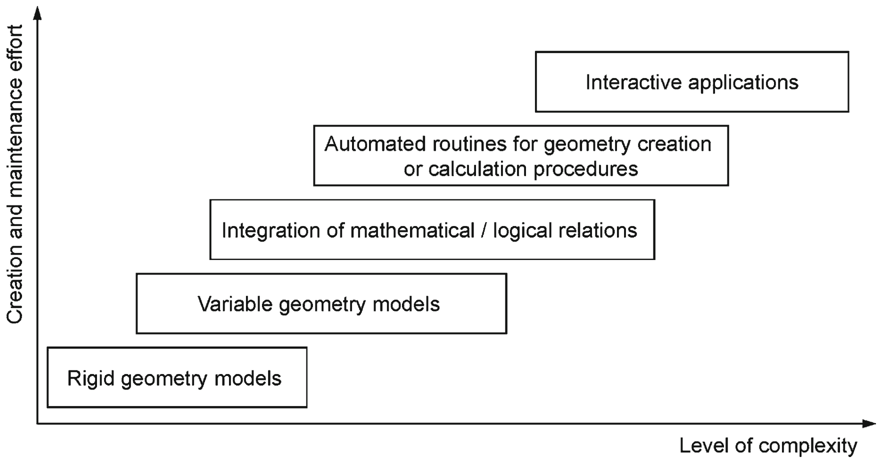

Creation and maintenance effort
Interactive applications
Automated routines for geometry creation or calculation procedures
Integration of mathematical / logical relations
Variable geometry models
Rigid geometry models
Level of complexity

Fig. 5.1 Different types of knowledge-based design applications

databases. These models often represent components from previous development projects or standard components. Rigid models can be provided in native or neutral data formats. Because rigid models represent carry-over parts or simple standard components, the creation and maintenance effort is limited to organizational tasks and data provision. Variable geometry models, also known as template models, include the predefined configuration of geometrical driving parameters to achieve variability of geometrical characteristics. Besides the effort for geometry-related design, which mainly addresses the development of the product shape and function, the creation of variable templates is more complex. In addition, variable geometry models include parametric control of geometry-defining parameters, which are open for user editing. The stability of variable geometry models has to be ensured by taking the desired variation ranges into account within the model structures. All of these points lead to an increased creation and maintenance effort.

Integrating mathematical or logical relations into variable geometry models by implementing formulas, rules, reactions or check operations provides a significant potential in terms of supported geometry creation. Logical relations and the mathematical combination of parameters, which define the dimensions of the design model, expand the control of geometry by input parameters. Besides geometrical characteristics, other parameters (e.g. material, weight calculation) can be calculated automatically within the design process. Automated routines for geometry creation or calculation procedures are performed by integrated scripts (e.g. Visual Basic for Applications (VBA) scripts), which are configured according to the requirements of logical software development. They follow the sequences of input, processing and output, whereby the input section is based on the definition of input parameters or geometrical input elements. The processing section can contain automated calculation sequences or the automated generation of geometrical elements, and the output section displays computation results and/or the created geometrical elements. Automated routines are implemented in template models and require some maintenance and update efforts. Unlike automated routines within CAD models, interactive applications are programmed within the applied design software and are therefore supplied independent from the actual loaded geometry models. Interactive applications represent problem-oriented software solutions for specific tasks. These software solutions are characterized by graphical user interfaces (GUI) for user-friendly handling, professional parameter management and integrated calculation or simulation procedures. If necessary, CAD-external software packages or databases can be integrated using bi-directional data interfaces. In this way, interactive applications enable an integration of CAE into CAD processes.

The integration of knowledge-based methods and tools into design processes can support product design significantly while simultaneously improving development efficiency, as well as product and process-related know-how. The different stages of knowledge-based design, which feature different levels of complexity, enable a problem-oriented selection of appropriate solutions. It is important to remember that the implementation of new methods and tools always requires a certain effort for creation and support. In addition, complex models can lead to arrangements that are not clearly defined (due to overloaded functionalities) and enlarged computation durations. Efficient structures and intelligent programming are necessary to avoid these problems.

5.1 Parameterization as a Basis for Knowledge-Based Design

The application of parameterization for geometry and model structuring provides an important basis for knowledge-based design methods and tools. Chapter 4 includes a detailed explanation of parametric design and some sample applications. In general, the parameterization of a geometrical object leads to a separation of geometry and its defining parameters. Besides a direct access for the definition of geometrical characteristics, additional parameters can be implemented for calculation or structuring purposes. All parameters can be accessed by specific functionalities to support a user-friendly handling. In this way, the parameter structure can be divided into different sections (e.g. an input parameter section, a parameter section for values that drive geometry, and a section for calculation parameters).

Figure 5.2 shows an example of the use of dimensional parameters to control parametric geometry. A modification of the parameters shown leads to an adjustment of the corresponding geometry. Parameterization of geometrical elements is not limited to three-dimensional models. Depending on the functionalities provided by the CAD software, two-dimensional models (e.g. so-called sketches) can be built up in a similar structure. Complex parametric template models include a wide variety of highly complex geometrical elements and therefore a large number of parameters and associations. To avoid complex and inefficient model structures, these templates

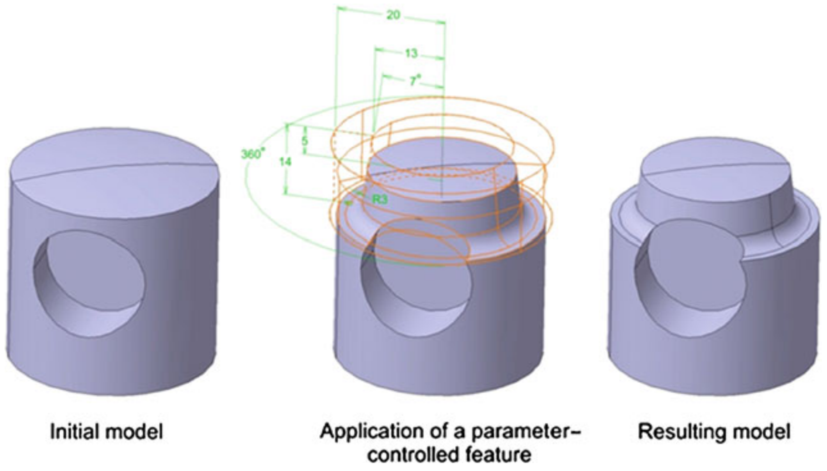


Fig. 5.2 Parametric design of a rotational groove [1]

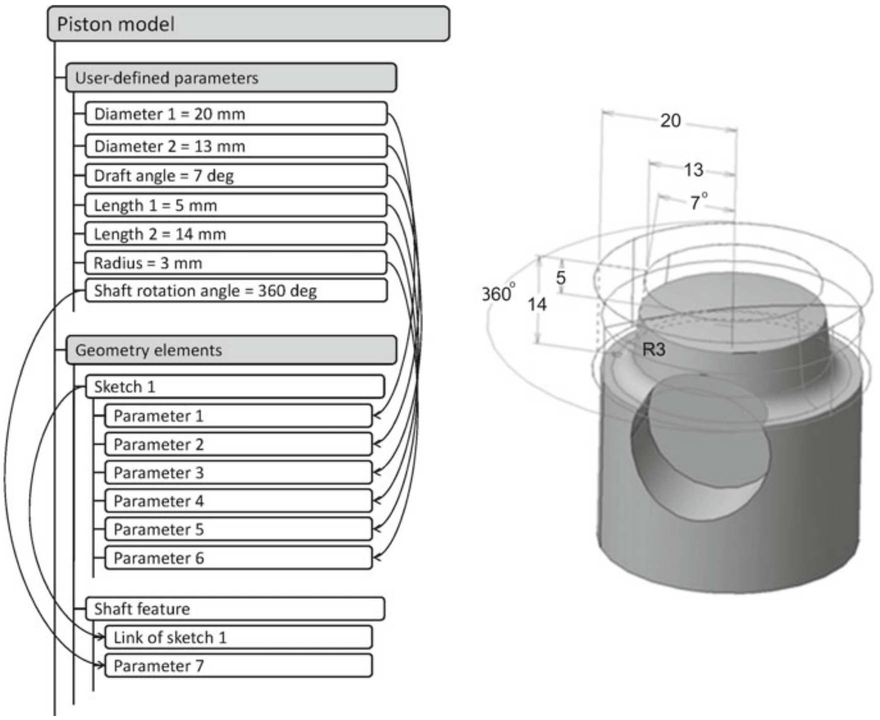


Fig. 5.3 Implementation of user-defined parameters for geometry control

must be carefully configured. Figure 5.3 shows a possible parameterization strategy for the example from Fig. 5.2. The user can enter dimensional values for a set of pre-defined parameters, which are linked with the standard CAD parameters that are used to define the corresponding geometrical elements or functionalities. Thus, the piston template geometry can be modified by altering the user-defined parameters. This method combines an efficient, user-friendly handling and a clearly arranged template structure, which enables subsequent reconstructing, modification or maintenance of the template model.

Beyond the parameterization of CAD models of single components (parts), a parametric structuring of assemblies in combination with skeleton and/or adapter models supports the creation of complex template assemblies. Such comprehensive product templates can consist of a large number of individual components and models, which are then logically interlinked and variable controllable. In order to be user friendly and to avoid circular references, over-constraint geometries and parameterization errors, these complex product templates must be clearly arranged and linked.

Figure 5.4 shows two of the most commonly applied linking strategies in assembly design. The geometry-based relations (left figure) are based on a direct linkage of geometrical elements (e.g. the use of a flange in a component as driving geometry

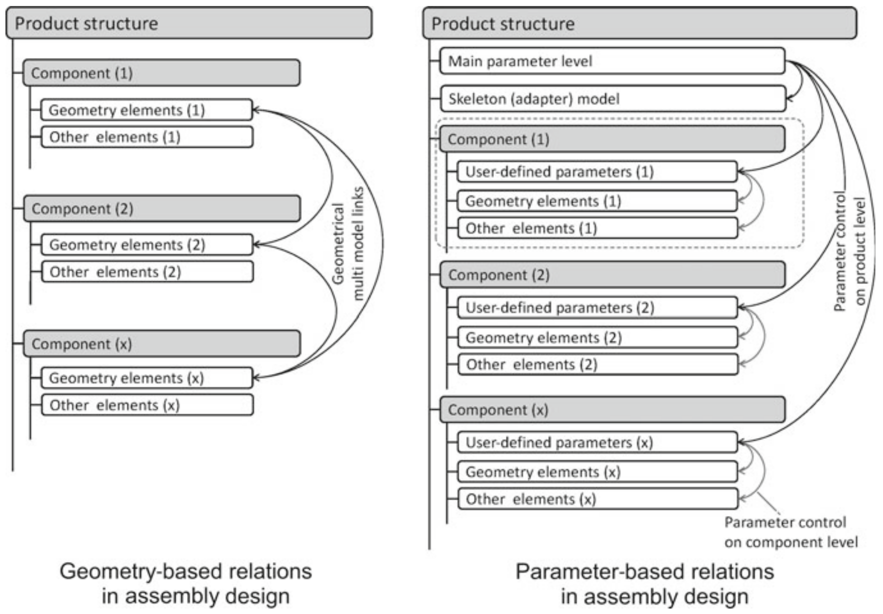


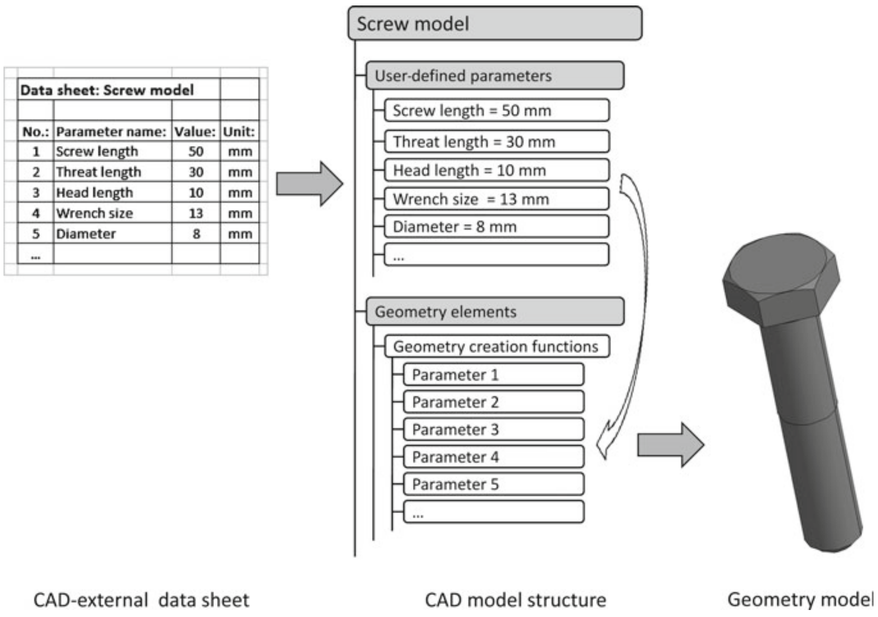
Fig. 5.4 Linkage strategies in assembly design

for the definition of a second components shape). In general, these multi-model links can include different types of geometrical links and can be applied between all components of an assembly. However, complex assembly structures, which contain numerous geometrical links, tend to yield somewhat confusing and overcharged relations, which are difficult to understand and to reproduce. For this reason, complex product template models, which must be sufficiently stable and user-friendly to be re-used, should be parameterized in a clear structure.

To this end, the example on the right in Fig. 5.4 shows a product structure with parameter-based relations. Within this assembly, direct (geometrical) linkage is replaced by parameter control mechanisms. The main parameter level includes a set of user-defined parameters and serves as an input box. The user-defined parameters on the main level are linked with corresponding user-defined parameters on the module or component level. As the example shows, on the module level, the sub-assembly structure has the same arrangement, whereby the main parameter level is supplied by the superordinate level. On the component level, the parameterization follows the strategy shown in Fig. 5.3. Complex product templates can also include skeleton and/or adapter models, which serve as positioning elements or for the definition of superordinate geometrical characteristics. The parameterization of these models follows the strategy of a main parameter level, which supplies specific user-defined parameters for each component.

5.1.1 External Parameter Control

In order to make work routines both user-friendly and highly automatic, modern CAD software offers the ability to control the internally used parameters externally. An interface to a spread-sheet or text processor enables the external control of the parameters that drive the geometry within the CAD model. Figure 5.5 shows an example of an external geometry control procedure. A set of user-defined parameters in a CAD model is supplied with corresponding values from a data sheet containing a predefined parameter structure.



Data sheet: Screw model			
No.:	Parameter name:	Value:	Unit:
1	Screw length	50	mm
2	Threat length	30	mm
3	Head length	10	mm
4	Wrench size	13	mm
5	Diameter	8	mm
...			

Fig. 5.5 External parameter control of a simple geometry

These features make it possible to define design-relevant parameters in a database. The database can be integrated into the 3D CAD model and thereby control functions of geometry-relevant parameters, such that a definition of values in the external data collector controls the geometry of the CAD model. The CAD-independent character of external databases opens up a wide range of possibilities for implementing additional applications. The geometry-based data storage of existing components that are not connected to a CAD system can be linked with other external data collectors. During the design process, the required data can be selected and incorporated into the 3D CAD model (Fig. 5.6). All of the geometry data of the virtual model that are not controlled by an external data link can be modified at any point in the design process. If the external data collector is implemented in a commercially available spreadsheet software package, additional mathematical connections and functions

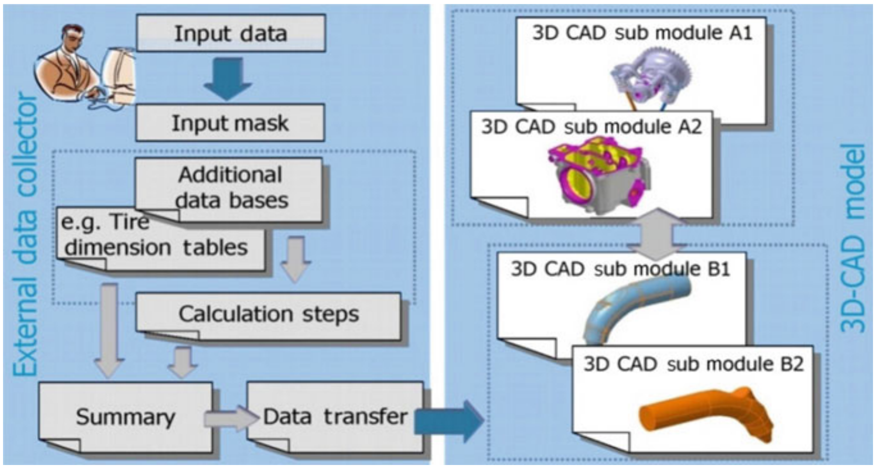


Fig. 5.6 Procedure for an external, parameter-controlled 3D CAD model [2]

can be performed beyond the CAD system, in order to prepare the data flow for the geometrical parameter control in the model.

5.1.2 Implementation of Non-CAD Data

Modern 3D CAD software packages enable the incorporation of pictures, 2D studies, sketches or drawings into the 3D model. This possibility moves the engineering-based construction and DMU development close to the styling process.

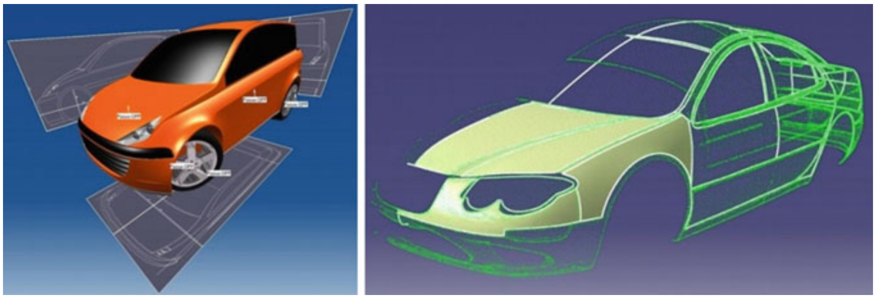


Fig. 5.7 Example of sketch-based and measurement-point-based surface creation [3]

As an example, 2D-based studies can be integrated into a virtual car model to perform different checks pertaining to ergonomic viewpoints (passengers), packaging boundaries (e.g. drivetrain or chassis components) or legislation-based influences (e.g. safety and crash regulations). Advanced design software packages offer an additional ability to generate 3D surfaces from 2D sketches, which allow for a direct implementation of studies into the automotive 3D CAD model. In the case of provided 3D hardware (style studies, clay models or scaled detail models), scan or measurement data can be directly imported into the CAD software to serve as a basis for surface-generation processes (Figs. 5.7 and 5.8).

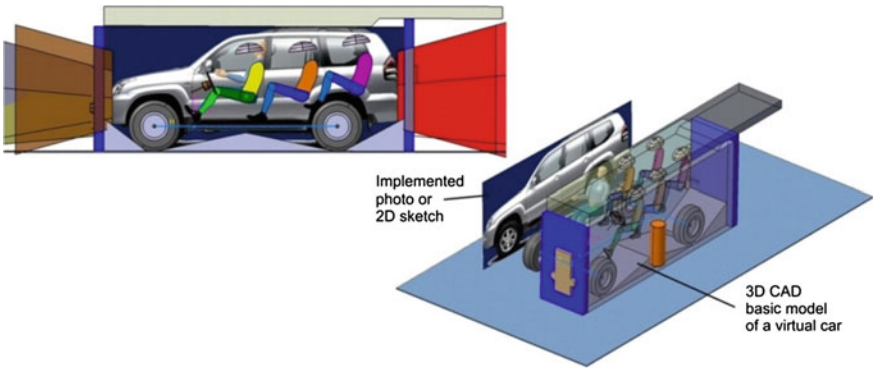

Implemented photo or 2D sketch
3D CAD basic model of a virtual car

Fig. 5.8 Combination of a 3D CAD model and a picture [4]

5.2 Knowledge Integration Using Template Models

Template models represent a kind of master models that can be integrated into development processes. Because templates are prepared for re-use, they include specific know-how. In this way, the application of templates transfers knowledge from former development projects into current product-generation processes. Template-based development methods are used in various domains (e.g. design, simulation, software development). The present book focuses on design processes, whereby the ability of modern CAD software to perform calculation and simulation procedures using design templates enables an integration of CAD and CAE. Here, template models are understood as predefined design model structures or geometry models, which can be augmented with additional functionalities.

The diagram in Fig. 5.9 presents a classification of CAD templates. Process templates support the development and management of different types of processes (e.g. for production engineering or cost calculation). The product templates discussed below focus on product development and can include a variety of knowledge that supports design-related tasks. Product templates can be divided into structure templates, geometry templates and functional templates.

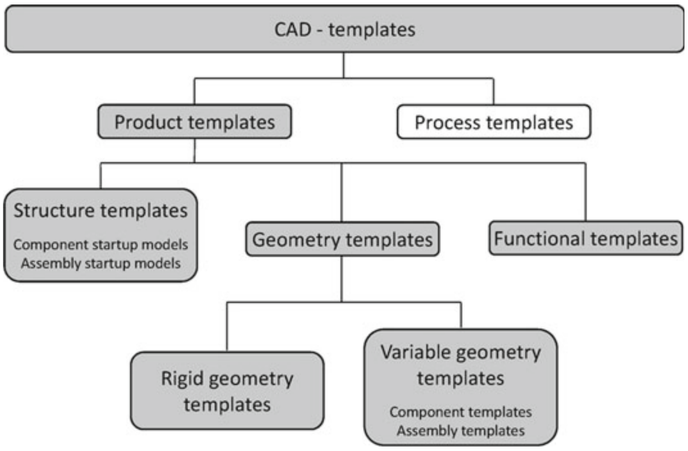


Fig. 5.9 Categories of CAD templates [5]

Structure templates address the predefinition of the internal design model arrangement. This is mainly accomplished by the introduction of startup models, which prescribe the sequences of geometry creation and the integration of different additional design-related aspects and automated functionalities (e.g. mass calculation, surface area computing). The application of structure templates ensures that design rules and specifications are considered throughout the entire development project. This plays an important role, especially in the case of complex product development, in which numerous engineers and departments are involved. There are two types of structure templates, startup models for component design and startup models for assembly design (Sects. 4.1 and 4.5). Assembly startup models define the structuring of a product by predefining groups, modules and components with the goal of generating a predefined product structure that persists through the entire development process. Complex products require the implementation of several levels, which can lead to a multifaceted assembly structure. For example, an assembly startup model of a car can be divided into five main groups, vehicle body, carriage, interior, electrics and drivetrain. The drivetrain group could be further divided into several modules (e.g. engine, transmission and axis). The engine module, in turn, can be divided into the sub-modules cylinder head, motor block, crank train, cooling system and further components. This model is thus broken down into individual components or small modules. The complex configuration of tall assembly models makes it logical to link the structuring directly with the applied engineering data management system (EDM), in order to support efficient data management. In automotive development, assembly startup models are often similar because car types of the same class have comparable architectures. This makes the integration of several functionalities to support the design process desirable (e.g. the automated loading of DMU, automated clash analysis, mass calculation procedures or the organization of a bill of material (BOM)).

Geometry templates include rigid and variable geometry models. In the case of rigid geometry templates, the geometry of the models is not adjustable by direct access. Rigid templates are not normally built up by parametric-associative design methods, and they include no enhanced functionalities for geometry creation. In general, rigid templates are provided in native data formats, which restricts the application to a specific CAD environment, or in neutral data formats, which enables a broad distribution independent from a specific software. This type of template is used to represent carry-over parts (COP), which are delivered from former (car) models and integrated into new vehicles (e.g. engines, suspension components). Besides 3D components, 2D models are also used to define the design or reference sections. One major advantage of rigid templates is their ease of reproduction and re-use in different types of development cycles, as well as the relatively simple data management, since there are not relations or linkages to other components, modules or assemblies.

Variable geometry templates represent predefined component, module or assembly models, which can include several functionalities for supporting the design process. The separation of geometry from underlying parameters enables the definition of flexible models, which can be modified by simply changing the parameter values. These highly variable templates include all structural and geometric information and are controlled by input parameters. The geometry creation process for these templates has to offer a universal usability, so that changing lengths or distances has no negative influence on the stability of the model. When generating variable template models, it is essential that both the range of possible parameter values and the flexibility of the created geometry fulfill the requirements of the intended application. Mathematical connections of parameters and restrictions of input values to reasonable rates support the definition of expandable templates for many standard components. Every variant of a template component represents a variation of the basic model, including the same design methods and rules. In this way, variable templates support the collection of expert knowledge and integrate know-how into the design processes.

The application of variable templates covers a broad field in product design. The range of complexity starts with simple geometry models (e.g. standard parts), which are adjusted based on the desired dimensions. Popular applications address the reproduction of business-specific components, which include knowledge regarding modeling structure and product characteristics (e.g. for the consideration of production-related aspects). The geometry is adjusted by inputting relevant dimensional parameters, which define the geometry model dimensions. Examples for this template type are piston models, gearwheels and different types of shafts. One special case of variable template models concerns predefined geometry creation sequences, which are saved in libraries and provided for re-use in different types of applications. These so-called power copies support quick and efficient creation or recurrent or standard geometries and also include functionalities for geometrical variations. Automotive body design offers several examples of such applications, such as seams, bird picks, flange geometries and other form shapes.

Beyond components, variable template models can also include several parts, which are (variably) organized in assembly structures. This provides flexibility in the geometry characteristics, as well as in positioning and structural configurations. In the case of complex variable assembly templates, skeleton or adapter models can support the control and linkage of different model structures. As mentioned above, a clear and logical structuring of the assembly templates is essential in order to avoid application problems that might be caused by complexity. Integrating the parameter structure into the applied EDM system enables a direct integration of automated functionalities. Thus, variable assembly templates can provide significant support for conceptual development. Comprehensive variable product models can include high level of details, which incorporate know-how from former projects into initial concept investigations, variant studies and geometrical optimization cycles.

Finally, functional templates in a CAD environment include specific, problem-oriented calculation and simulation procedures, which can support different areas of product development (e.g. design and dimensioning, production engineering, cost calculation). Functional templates can exist as special programs or may be integrated into simulation or design software. In the case of CAD-external software, data transfer from calculation to design and vice versa is accomplished by integrating data interfaces. This approach can be applied for everything from simple computations up to integrated software solutions, which enable complex product layout. In many cases, functional templates are integrated into development processes and serve as problem-oriented tools. Embedding them into variable geometry models makes it possible to integrate them into the (existing) parameter structure of CAD models. In this way, an unambiguous parameterization strategy supports efficient data transfer between design and calculation cycles. The combination of variable geometry templates and functional templates within one data structure enables the generation of efficient tools for product development. Because of their integrated programmed functions and algorithms, such tools are advanced knowledge carriers that include company-specific know-how. The following sections present some sample methods for creating integrated templates in automotive development.

5.2.1 Template-Library-Based Design

As an advancement of conventional template methods, library-based design methods facilitate the configuration of knowledge-related models, which are composed of different preselected parametric modules that are imported from a 3D CAD model database. This method supports a quick and efficient generation of conceptual geometries that takes into account a wide range of influencing factors. Beyond the use of complete predefined models as templates, an enhanced application supports the generation of new geometries using predefined modules, which are selected by the user and assembled via automated functions. Due to the remarkable advantages of

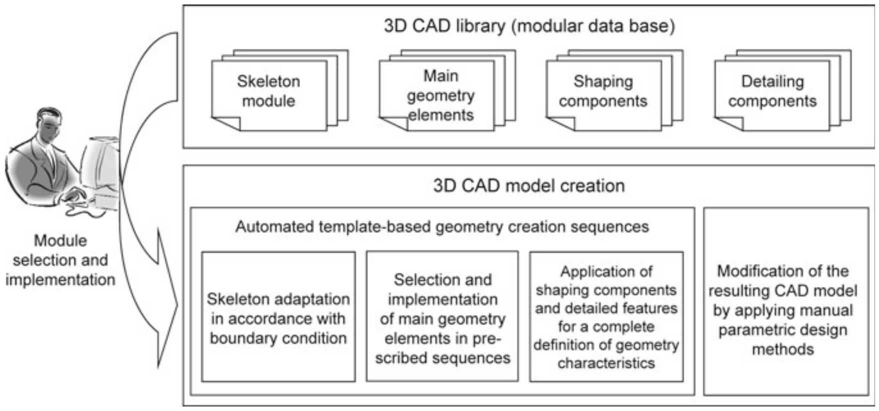

3D CAD library (modular data base)
Skeleton module
Main geometry elements
Shaping components
Detailing components
Module selection and implementation
3D CAD model creation
Automated template-based geometry creation sequences
Skeleton adaptation in accordance with boundary condition
Selection and implementation of main geometry elements in pre-scribed sequences
Application of shaping components and detailed features for a complete definition of geometry characteristics
Modification of the resulting CAD model by applying manual parametric design methods

Fig. 5.10 Semi-automated design process using a modular template library [6]

library-based geometry creation methods, this approach has a high potential for a further increase of efficiency in modern design processes.

The product geometry is divided into several modules, which are saved in a database. During the design process, these modules are loaded in a predefined sequence and integrated into the 3D CAD model. After a user selects the desired components from the library, the geometry is built up step by step by applying automated loading and assembling procedures. Figure 5.10 shows an example of a semi-automated design process using a modular template library. In this example, the library includes several modules for the design of a complex product, which includes different levels of detail. A predefinition of production-related features (e.g. mold configuration, draft direction, draft angles and rounding characteristics) considers the requirements of mass production processes from initial design phases on. Each parametric-associative geometry template module is available in a number of options with different characteristics and details; thus, a broad combination variety enables the generation of numerous different concept geometries.

Figure 5.10 displays the general strategy of a library-based design procedure. The database structure of the library includes a skeleton module, a module with main geometry elements, a module with shaping components and finally a module with detailing components. The first step of semi-automated geometry creation includes an adaptation of a variable skeleton module to conform with geometric boundary conditions. The variable skeleton serves as a reference for the subsequent process of positioning the geometry templates. In the next steps, the user selects the main geometry modules, which are then implemented in prescribed sequences. All main modules include sub-elements, which are available in the database as parametric-associative template geometries. The automated assembling process of geometry templates can include Boolean operations or split and trim operations, as well as the application of fillets and draft angles.

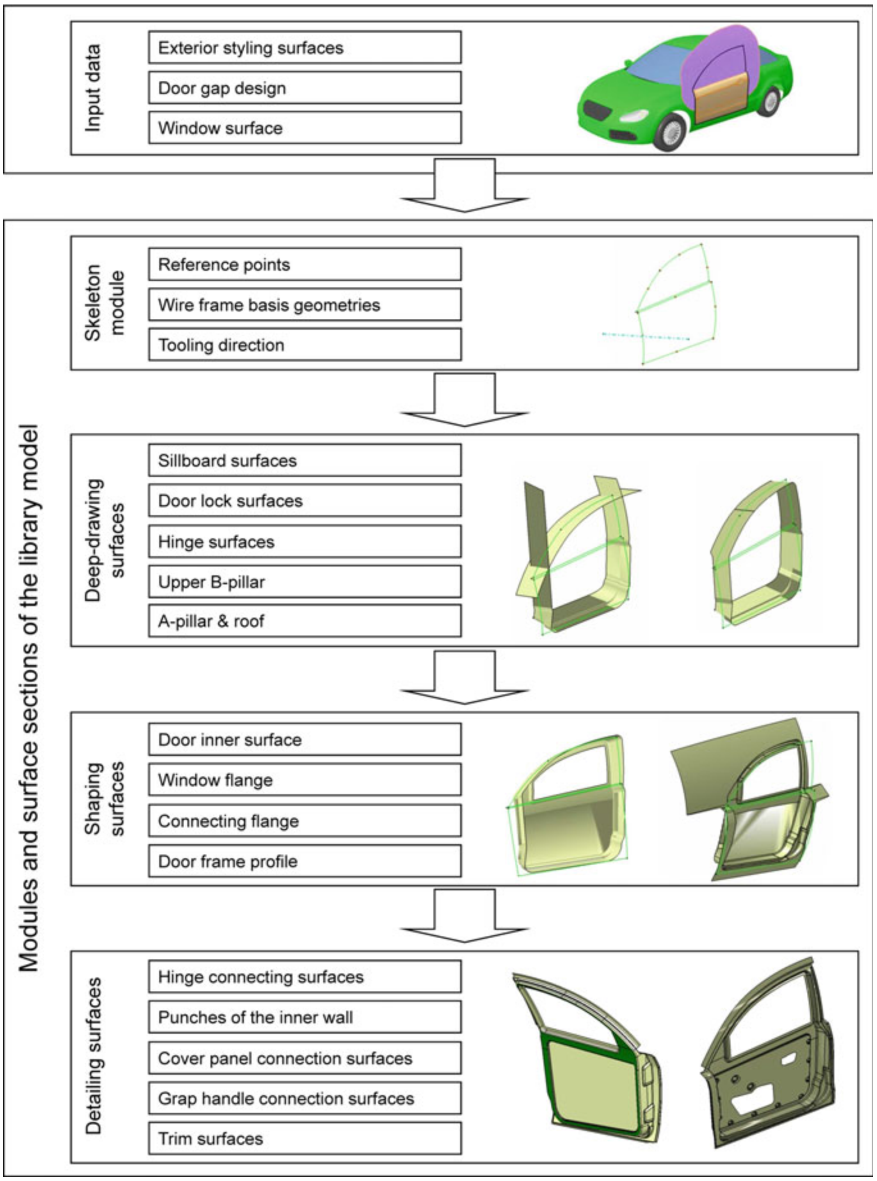


Fig. 5.11 Modular configuration of library-based parametric-associative door design [7, 8]

The sample geometry generation process above has been fully implemented into a commercial 3D CAD environment, such that all standard functionalities for geometry generation and modification can be combined within the automated design process, thereby enabling subsequent manual modifications. A direct linking of the skeleton module to boundary conditions (e.g. geometries of surrounding components or imported styling surfaces) supports adjustments and optimization cycles. If these boundary conditions change during the development process, the existing references are exchanged, and the geometry model is reconnected automatically. In this way, the application of template-based geometry modules for the generation of parametric-associative concept geometries supports a user-friendly definition of flexible 3D CAD models. A high level of detail is achieved by using a hierarchical model architecture, which includes rough geometry templates for initial steps, as well as detailed models for the creation of connecting surfaces, punches, flanges and trims.

Figure 5.11 shows an application of library-based design using the example of parametric-associative door development. A stepwise creation of the product geometry in combination with a continuously increasing portion of verified data leads to an increase of product maturity throughout the entire development process. The product maturity itself is defined by the fulfillment of clearly prescribed milestones, which enables a procedure for observing and reporting on the achievement of objectives. Every optimization step in the workflow is accomplished by a coupled reaction of the 3D CAD model, which leads to a successive increase of knowledge in all involved areas, as level of detail in geometry creation and the amount of information contained in the technology concept definition increase steadily. Besides the purely technology-oriented procedures in virtual product development, the integrated architecture of the presented approach supports communication between all parties involved and the required knowledge transfer. This is achieved by a user-friendly product representation via 3D CAD geometry data, as well as by the universal parameterization strategy and a data-based parameter management.

5.2.2 Implementation of Mathematical and Logical Relations

The definition of mathematical connections between parameters enables the implementation of logical geometrical dependencies. In this way, formulas, relations, rules and reactions can be integrated into the 3D CAD model environment and support automated geometry definition processes. This can lead to a reduction in design effort in the case of variant studies or the re-use of parametric models. Besides purely geometry-related characteristics, these enhanced parametric models can include additional knowledge about dependencies and relations between different design-related aspects.

Figure 5.12 shows an example of function-oriented geometry creation within a common 3D CAD environment. The definition of mathematical connections between parameters forms the basis for the creation of rules that can be used to control geometric functionalities in the applied CAD system. In the present example, the equation $x_2 = Asin(kx_1)$, with $[0 \leq x_1 \leq k\pi]$ and the real parameter A (amplitude), defines the progression of a curve in a Cartesian coordinate system. The application of this curve progression onto a user-defined curve in space as a leading element results in the corresponding three-dimensional spline, which can be used to create complex technical surfaces (e.g. sine-based tubes).

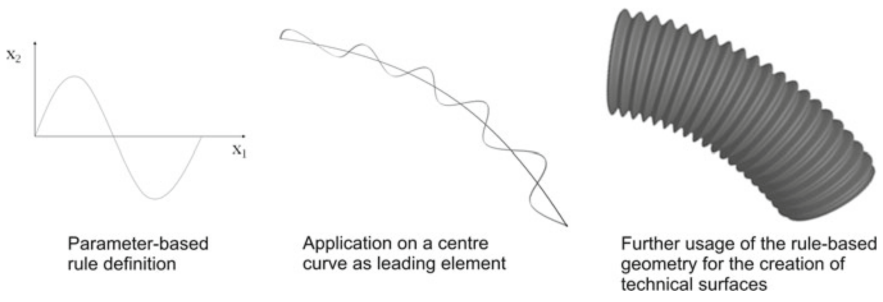

x_2
x_1
Parameter-based rule definition
Application on a centre curve as leading element
Further usage of the rule-based geometry for the creation of technical surfaces

Fig. 5.12 Exemplary function-based geometry creation [6]

Besides standard functionalities for the implementation of mathematical and logical coherences, the integration of (internal or external) problem-oriented solving procedures into the design process supports recurrent operations in the course of development and optimization cycles. In the case of internal calculation procedures, the mathematical algorithms are embedded in the CAD environment, which facilitates a direct access to the implemented functionalities. Internal solving algorithms use the functionalities of the CAD system to specify user-defined procedures and calculations. These can be simple linear calculations of dimensional values or enhanced mathematical algorithms, which are embedded in macro-controlled software sequences.

External solving algorithms use specific programs and/or simulation procedures for computation in complex tasks. A bi-directional data transfer between the design

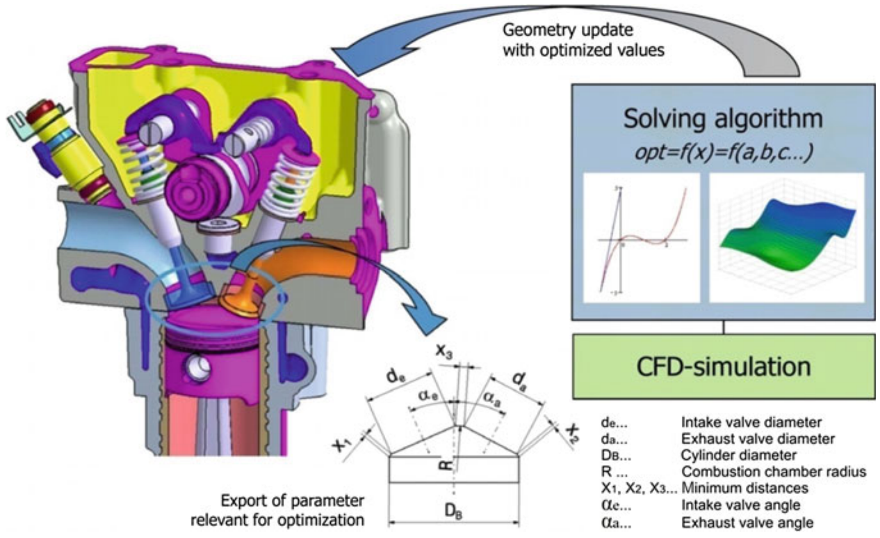

Geometry update with optimized values
Solving algorithm
opt=f(x)=f(a,b,c...)
CFD-simulation
d_e
d_a
x_1
x_2
x_3
α_e
α_a
R
D_B
Export of parameter relevant for optimization
de... Intake valve diameter
da... Exhaust valve diameter
DB... Cylinder diameter
R ... Combustion chamber radius
X1, X2, X3... Minimum distances
αe... Intake valve angle
αa... Exhaust valve angle

Fig. 5.13 Application of external solver algorithms in cylinder head development [9]

software and the simulation programs provides the parameter values required for the calculation. For this purpose, specific parameter sets are defined in the CAD software and handed over to the simulation program. The subsequent external computation procedures can include a variety of procedures, such as complex calculations of dimensional characteristics, optimization cycles or the simulation of influencing physical processes.

Figure 5.13 shows an example of an application of external solver algorithms in cylinder head development. The target of the calculation is to optimize the intake and exhaust valve diameters for a motorcycle engine. The relevant geometrical parameters are the valve diameters, the cylinder diameter, the combustion chamber radius, minimum distances between valves and cylinder wall and the valves. Additional CDF-based simulation procedures, which take into account the dimensional effects on the intake and exhaust flow situation, consider physical influences. All optimization-relevant parameters are exported from the 3D CAD model into an external solver algorithm, which combines the influencing aspects into mathematical optimization cycles. The computation yields parameter values, which are imported to drive a geometry update of the cylinder head model and a subsequent evaluation.

The ability to create macros can be very helpful for enabling automatic sequences of features and actions. Most of the advanced CAD software packages offer programming languages or editors, which support the creation of effective and versatile routines (e.g. VBA, VB.Net). Macros can control recurrent operations in virtual development processes. The integration of these programs into the CAD software enables their integration into the virtual model, while the data flow in assembling structures and between other types of CAD files supports the generation of efficient tools for specific problems in the development process. An automated handling of problem-oriented mathematical connections, formulas, rules and algorithms can be integrated into the corresponding product model to provide significant support for the layout and design phase. In addition, the creation of graphical user interfaces and macro-specific toolbars in the design environment supports a user-friendly operation.

Figure 5.14 shows the general architecture of a macro-based business-oriented software application for the automated generation of sections in 3D CAD product models. Graphical user interfaces make it easy to input data. In the present example, the interface prompts the user to enter the path to the relevant product geometry, as well as the positions and orientations of sections, in prescribed boxes. Next, the program loads the specified geometry models and automatically performs the operations required for the sectioning process. Finally, the produced 3D and 2D sections are saved in predefined folders. Based on the automated functionality, a large number of sections can be created with relatively little manpower (e.g. as an autonomous overnight work package).

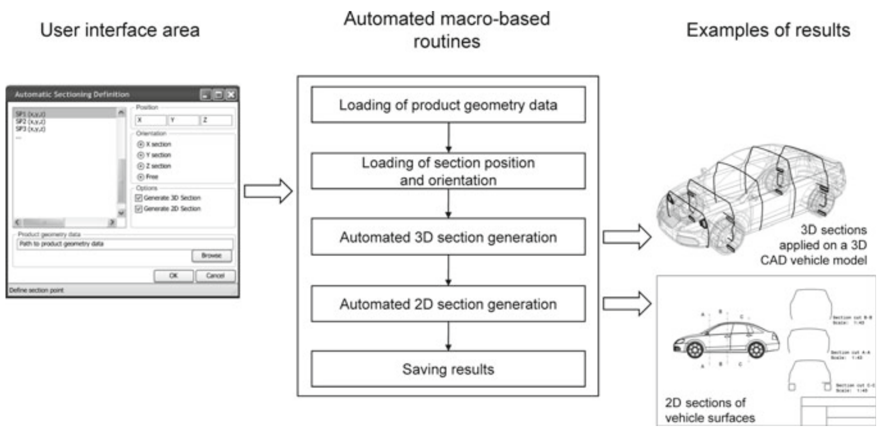

User interface area
Automatic Sectioning Definition
SP1 (x,y,z)
SP2 (x,y,z)
SP3 (x,y,z)
Position
X
Y
Z
Orientation
X section
Y section
Z section
Free
Options
Generate 3D Section
Generate 2D Section
Product geometry data
Path to product geometry data
Browse
OK
Cancel
Define section point
Automated macro-based routines
Loading of product geometry data
Loading of section position and orientation
Automated 3D section generation
Automated 2D section generation
Saving results
Examples of results
3D sections applied on a 3D CAD vehicle model
2D sections of vehicle surfaces

Fig. 5.14 Example of a CAD-system-integrated, automated sectioning application [6]

5.2.3 *Integrated Virtual Product Development Using Centralized Master Models*

One key to an effective coupling of design-related aspects and simulation-related tasks can be the use of a centralized master model, which contains the product geometry, as well as additional information. In principle, the components to be developed form the point of intersection between all engineering disciplines. On the one hand, the design department has to develop the product geometry (including the component structure) under consideration of several design-related aspects. To guarantee an efficient production engineering process, this development must be performed in the context of many boundary conditions (e.g. from calculation, simulation) and under consideration of production requirements. On the other hand, the production engineering must consider the product characteristics, which means the product is the focal point of several processes. Thus, the use of a geometry model as center of development can be the optimum solution. This master geometry model can be managed in a superordinated product data management structure, which takes into account all required data flows.

Figure 5.15 shows the configuration of an integrated master model, which can be used for the development of complex mechanical products. The geometry section includes all components and modules of the product as 3D CAD components. The product structure is arranged and managed in an assembly structure. In addition to the tasks of positioning, packaging and functional development (e.g. kinematics), the assembly includes a hierarchical configuration of the sub-modules and components. For archival purposes and to support production engineering, parametrically derived two-dimensional (2D) drawings are created and administered. One important feature is the organization of geometry parameters in a predefined order, which includes a consistent parameter structure in each component and assembly level, as well as a centralized parameter structure in the main level of the product assembly. This centralized parameter structure is used for direct information exchange with an associated database.

The database section also contains the product structure to enable a thorough organization of different types of product data. These data are stored in different areas of the database and supply a broad field of development disciplines with required information. Besides this role, the database serves as a central unit for storage and tracking functionalities throughout the entire virtual product development process. Depending on the type of product being developed, different disciplines are taken into account. The exemplary configuration in Fig. 5.15 shows a selection of main working tasks and data groups in an automotive full-vehicle development process.

In many cases, the product development starts with a design process within a CAD environment. In the initial phase, the model structure has to be adapted to the requirements of the development status and is displayed in separate, prescribed configurations. Product data management (PDM) systems for series production apply their management strategies in relatively rigid structures, which are created to fulfill the demands of design, simulation, production and administration, whereas PDM configurations for the concept and pre-development phases have to consider the highly flexible processes and requirements of initial engineering. For the engineers who create new development processes and strategies, the challenge is to create methods that are able to combine the flexible work characteristic of concept phases with the rigid, manpower-intensive work in series development. The target is to transfer the knowledge from concept phases directly into the series development process. A direct adoption of geometry models coming from the concept phase as

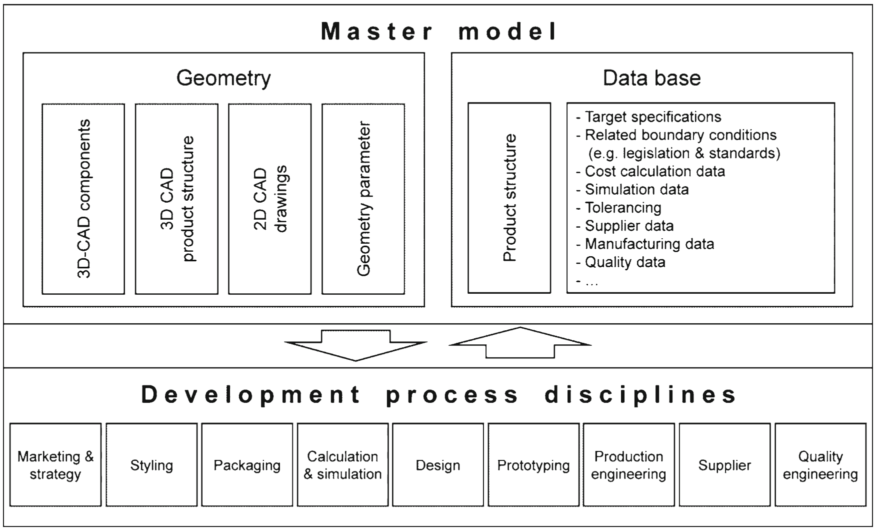


Fig. 5.15 Configuration of a sample integrated master model [1]

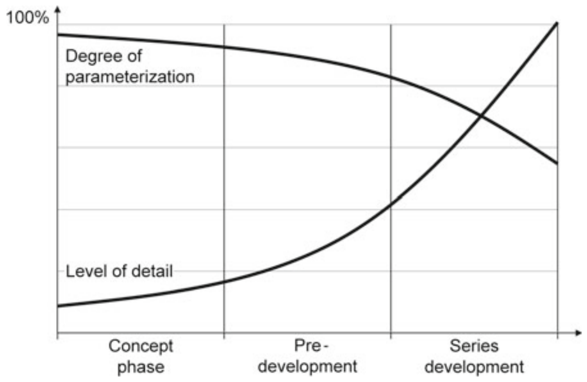


Fig. 5.16 Relationship between degree of parameterization and level of detail during the development phases (*trend lines*) [10]

start-up models in the series development cycle can lead to a data flow integration of both phases. This method has to meet the subsequent challenges (Fig. 5.16).

The use of the same parametric-associative geometry model structure in both pre-development and series development requires a flexible model architecture which is able to fulfill the requirements of both process phases. Currently, different start-up models are used to meet the different requirements. The different demands related to geometry creation are mainly based on a high flexibility during the concept phase and a high degree of detailing during series development. Normally, the degree of parameterization is much higher in initial development phases and decreases significantly in series development. On the other hand, the level of detail in component geometry is relatively low in the early phases, but has to be high in the production-related development of the final geometry creation process. The development of future design processes must solve this contradiction by implementing flexible geometry model structures for both phases. Two things must first be clarified: the degree of parameterization that is reasonable for each process step, and how a changing level of model parameterization can be handled in the same environment to ensure an effective data and knowledge transfer between the different disciplines and departments. In this context, research work in the coming years will take advantage of the increasing abilities of CAD software combined with rising hardware performance to enable a high degree of parameterization of the 3D CAD models throughout the entire development process.

5.3 Example: Integrated Design in Automotive Bumper System Development

This application shows a current challenge for the development of integrated, business-specific design methods using advanced CAD technique in automotive development. A master model includes a full-vehicle configuration and serves as a

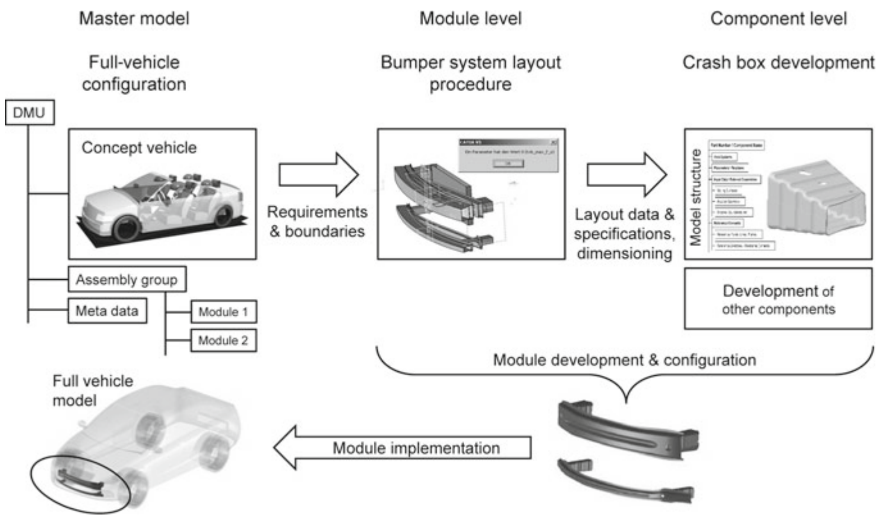

Master model
Full-vehicle configuration
Module level
Bumper system layout procedure
Component level
Crash box development
DMU
Concept vehicle
Requirements & boundaries
Layout data & specifications, dimensioning
Model structure
Assembly group
Meta data
Module 1
Module 2
Development of other components
Module development & configuration
Full vehicle model
Module implementation

Fig. 5.17 Workflow of an automotive bumper system development using integrated design methods [6]

central development platform. Within this configuration, a conceptual vehicle model covers all full-vehicle-related boundary conditions and requirements, such as legislative prescriptions, requirement specifications, ergonomic demands, packaging-relevant aspects, and styling data. This conceptual vehicle model consists of several parametric CAD templates and supports the entire full-vehicle layout phase of a new car model. Section 9.3 describes the set-up of the integrated vehicle model and its possible applications. In addition, the master model DMU contains different design-related modules for the development of a full-vehicle model, including assembly groups and meta-data (e.g. technological requirements). The entire master model structure is linked with a CAD-external database, which manages product structure, as well as implementing and organizing simultaneously performed workflows. For an improved user handling, macro-based program routines and graphic user interfaces are implemented.

Figure 5.17 shows the workflow of an exemplary bumper system development as a part of the full-vehicle development process. Specific requirements and boundary conditions for the development of a front-end crash structure (e.g. packaging-relevant data, limitations imposed by standardized crash tests or legislation, styling demands) are transferred from the master model structure into the targeted module level. The layout of the front-end crash system is performed in a module level using specific layout algorithms, which are implemented in an integrated CAD template model. Within this model, the dimensions of the bumper and deformation elements are conceptually calculated using empirically determined, experienced values and integrated computation routines. As a result, the main dimensions of the bumper system are transferred to a parametric-associative skeleton model, which displays the packaging-relevant space requirements of this module. The verified design space provides the basis for a subsequent conceptual geometry creation process of the respective components.

The geometry is created at the component level using parametric templates for each component, which are included in start-up model configurations. The advantage of the start-up model approach is the predefined component structure, which leads to predefined geometry creation methods and facilitates the reuse of template models and different automation functionalities. Thus, adaptations and changes of the geometry can be arranged flexibly and quickly. Furthermore, the prescribed model structure enables geometry creation procedures using library-based geometry creation features. In this way, a predefined geometry, for example of a crash box, is implemented and automatically adjusted to the embedded module skeleton. The components are assembled on the module level and parametrically associated to the module skeleton model. Finally, the conceptual bumper system geometry is implemented into the full-vehicle configuration of the master model.

This procedure makes it possible to create conceptual component and module geometries quickly and efficiently in initial product development phases. The early concept geometries are used for initial investigations and simulation procedures; so they are continuously improved or replaced by models with higher maturity. Due to the flexible data configuration of the centralized master model, product-related data can be handled efficiently, an advantage which is not limited to the design process, but also comes into play in many related working fields.

References

1. Hirz, M.: Advanced Computer Aided Design in Conceptual Automotive Development. Habilitation Thesis at Graz University of Technology, Graz (2011)
2. Lang, M., Göber, T., Hirz, M.: Macro - based CAD- and DMU- Analysis Methods. Lake Chiemsee, Germany (2007)
3. Dassault Systems: CATIA V6. Date of access: 2009–11-10. www.3ds.com/products/catia
4. Göber, T., Hirz, M., Krammer, S.: A Method for Externally Controlled Parameterized Automotive Design at an Initial Development Stage. Lake Chiemsee, Germany (2006)
5. Harrich, A.: CAD basierte Methoden zur Unterstützung der Karosseriekonstruktion in der Konzeptphase. Phd Thesis, Graz University of Technology, Austria (2012)
6. Hirz, M., Harrich, A., Rossbacher, P.: Advanced computer aided design methods for integrated virtual product development processes. Comput. Aided Des. Appl. **8**(6), 901–913 (2011). doi:10.3722/cadaps.2011.901-913
7. Harrich, A., Mayr, J., Hirz, M., Haselwanter, P., Lang, J., Gfrerrer, A., Haselwanter, A.: Unterstützung der Türenkonstruktion in der Konzeptphase durch parametrisch-assoziative Konstruktionsmethoden. In: OEM Forum Fahrzeugklappen und -türen, no. 2064 in VDI Berichte, pp. 21–37. Verein Deutscher Ingenieure (2009)
8. Hirz, M., Harrich, A., Mayr, J., Rossbacher, P., Haselwanter, A.: The potential of parametric design methods in automotive door development. In: Proceedings of the FISITA World Congress. FISITA, Budapest (2010)
9. Hirz, M., Kirchberger, R., Göber, T., Lang, M., Tromayer, J.: Integrierte 3D CAD Konstruktionsstrategien im Motorenentwicklungsprozess. In: Symposium Konstruktionsmethodik Graz. Graz (2006)
10. Anderl, R.: Virtuelle Produktentwicklung. Lecture Script at Technische Universität Darmstadt (2007)

Chapter 6
Engineering Data Management

This section will describe Engineering Data Management (EDM) as a concept involving the interdepartmental and interdisciplinary integration of data and workflows in automotive product development. More specifically, the fundamental principles of EDM data, basic functional modules and typical CAD and CAE use cases are described based on process-oriented PLM approaches.

Product development in the automotive industry presents several challenges in the area of EDM, including:

- Insufficient transmission of knowledge gained in process and knowledge management to engineering data management
- Systematic approach to the design and development of EDM processes and systems (functional strategy, roadmap)
- Few use cases for operative and project-oriented application of EDM
- The implementation of PLM approaches involves some difficulties, especially in larger companies with complex structures
- Data management activities can lack a process orientation
- Excessive orientation towards systems and software in EDM designs

Engineering data management supports an integrated view of product development that fosters a synergy between static results and a dynamic approach. It facilitates the management of master data and development data functionally and even supports and improves product development processes. Consequently, EDM aim is not only to integrate technical data and processes, but also to link and correlate them to economic parameters. Thereby, it ensures efficient development processes while simultaneously supporting the establishing of conventions for product development and market launch.

Among other factors, EDM consists of the common and complete management of all product and process-related data during product engineering and the visualization of this data in a manner suitable for the real business world. All these characteristics are based on the organizational and systematic support of engineering work beyond the product life cycle and the (departmental) borders of a company. Modern author systems (CAD-, CAM-, and CAE-systems) and the related simulation and visualization tools support the methods mentioned above at an IT-level, while EDM system solutions form the functional and administrative backbone.

Additional important features of EDM include the virtual product representation and its time-dependent data and configuration management concerning the maturity during the entire development process (e.g. maturity and status of data, DMU-configuration). This section examines EDM topics from a knowledge perspective. Specifically, the EDM workflow management and integration platform, including the related interdisciplinary data management, are adapted for modeling. These components provide the foundation for integrated applications in product development, which in turn lead to the development of design approaches for integrated data management in product development.

Besides the basic functional angle, aspects of engineering data management may also be viewed from a process-oriented and a system-specific view. The system-specific view mainly considers the functions, methods and systems used in EDM. In the present work, the process-oriented approaches of EDM are generally subordinated to the findings of PLM-oriented approaches.

The following sections provide background about the development of EDM and a definition of the basic concept. Furthermore, EDM data basis and its role in product development are described, as well as the structure of an EDM-system and its most important functions. Finally, some examples of integrated EDM applications in product development provide some insight into the potential for efficient use of EDM.

6.1 The Concept of Engineering Data Management (EDM)

EDM is a concept used for the interdepartmental integration of information and data flow, as well as business data flows and procedures, throughout the complete life cycle of products. In this context, the term engineering data comprises all data and documents created within product development and stored for further processing. Data are always stored and handled in relation to their respective products, so the terms product data or product data management (PDM) are sometimes used as well. Metafiles and different other kinds of documents consist of product and process data. EDM encompasses all of the functions that are necessary for editing and distributing product and process data and documents. In this context, a main task of EDM systems is to store, manage and provide data and documents that describe products and processes as subsets to the product life cycle [1].

6.1.1 The Y-CIM Model

The integrated controlling of business processes via information technology was first conceived in the early 1980s together with the beginning development of Computer

Integrated Manufacturing (CIM). At that time, the main focus lay on the integrated consideration and control of both logistic functions and research and development activities in the corporate context. The technical process chain focuses on the development and production processes of the actual products. The main task is to plan and implement the product portfolio, which comprises the construction and validation of products. Moreover, environments for the production of the products being developed have to be established. In particular, this means defining work schedules for employees and implementing programs for the control of production machinery. Both tasks are completed on the manufacturing level, where the work schedules for employees and control programs for machinery are listed.

According to Scheer [2], economic and technical process chains in production are the core of the Y-CIM model (see Fig. 6.1). The economic tasks are the planning and implementation of production orders. On the planning side, this includes the chronological planning of all orders, as well as determining the materials needed for production and scheduling their delivery. The production side includes the detailed planning of orders, the implementation of manufacturing orders and feedback on the progress status. The economic and technical process chains are independent from each other at the beginning. In general, this means that economic planning in the sense of order handling and technical planning for product development may be considered non-connected.

Based on the relationships between economic and technical process chain (as shown in Fig. 6.1), both chains may be considered as combined on the man-

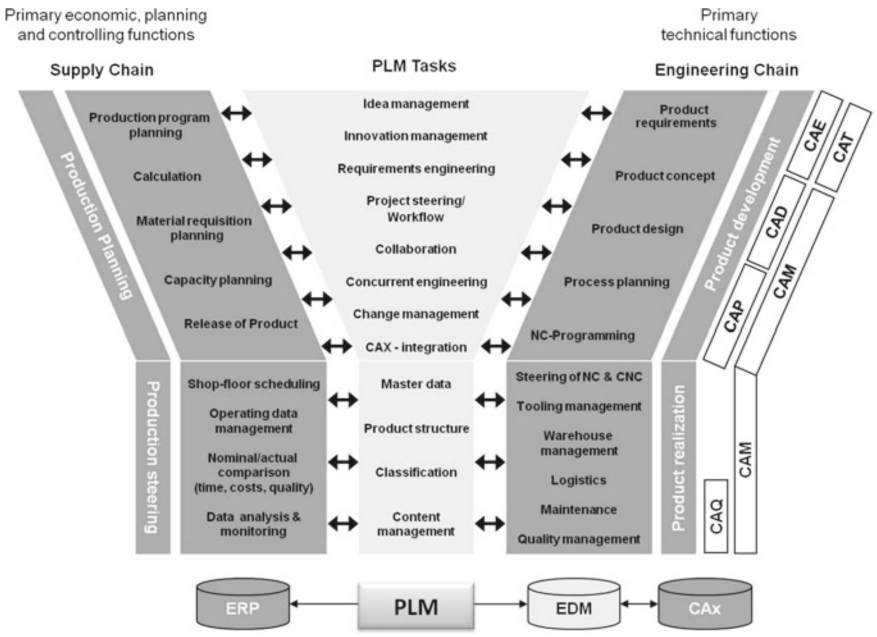


Fig. 6.1 Y-CIM model in industrial business

ufacturing level in the graphical representation. Assuming objectively logical and chronological flows, the issues form a Y. This shows how the chains are independent on the planning level but merge into each other on the production level.

Differentiation of Production Planning and Scheduling

The left side of the Y-CIM model shows the economically logical process chain in industrial corporations. Production and scheduling plans are established based upon sales forecasts and order extrapolations. Proceeding from the longer-term outline planning, the time range is reduced step-by-step, with the planning becoming increasingly detailed. The final task in industrial planning is the release of orders, which is the first point of contact to production and manufacturing. The right side of the Y-CIM model is predominant in issues of construction support, exploration of customer requirements and methods for product validation. In this context, marketing's primary task is to determine the basic properties the targeted customers expect for a future product, as well as shortcomings in current products (from the customer point of view) which must be remedied in the future.

The application of Computer-Aided Engineering (CAE) supports an integration of the aforementioned requirements into early product drafts. Using Computer-Aided Design (CAD), the product can be defined in terms of its actual composition, whereby the individual components of the product are defined and described. Beyond geometrical characteristics and weight definitions, this process includes performance requirements, material definition and tolerances for manufacturing. The specified product data are then implemented in Computer-Aided Planning (CAP) for manufacturing using relevant concepts and application systems. EDM harmonizes and controls data management for these application systems. Figure 6.1 shows the positioning of the deployed application systems, which are described in detail in Sect. 6.5.1. The Y-CIM model provides a generally valid representation of these topics and a clear differentiation and positioning of the EDM approach. Due to its primary technical function, EDM focuses on service structures, and especially on the engineering process. In this context, the engineering process refers to product planning, which is clearly different from product manufacturing.

6.1.2 PLM as a Foundation of EDM

Due to globalization, business concentration and outsourcing of both processes and complete business units, there is a growing need to improve the integration of core processes within companies and to optimize them by using modern technology. Therefore, the consistency of data originating from the product development process must be guaranteed throughout the subsequent phases of product life cycle. These phases include production and logistics, as well as customer processes such as sales, marketing, customer support and recycling [2]. The industry is constantly searching for new concepts of how to manage products in all business processes as efficiently as possible across their complete life cycles. This strategic concept, known as Product Life cycle Management (PLM), seeks to take into account the prevailing circumstances within both the sector and the specific company.

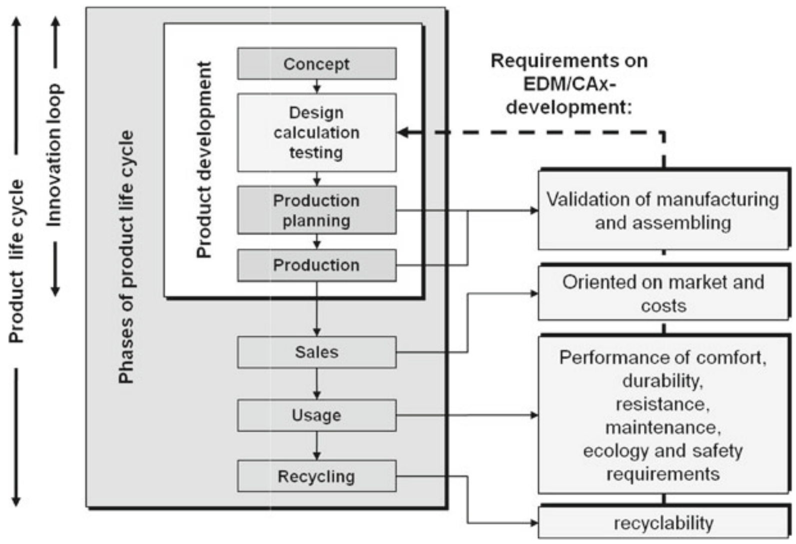

Product life cycle
Innovation loop
Phases of product life cycle
Product development
Concept
Design calculation testing
Production planning
Production
Sales
Usage
Recycling
Requirements on EDM/CAx-development:
Validation of manufacturing and assembling
Oriented on market and costs
Performance of comfort, durability, resistance, maintenance, ecology and safety requirements
recyclability

Fig. 6.2 Exemplary product life cycle in automotive industry [3]

A product's life cycle begins with product development and continues through manufacturing, sales, maintenance and finally withdrawal from the market. Since this long process involves numerous departments which have different information requirements, the goal of PLM is to optimize the design of processes. The product development process and the provision of full product information as required throughout a product's complete life cycle are of particular interest in this context. Figure 6.2 depicts a typical life cycle of a product in automotive industry. This figure also shows the requirements for EDM and CAx in the particular phases of the life cycle.

Today, the management of product data and engineering processes across the complete product life cycle is one of the most important management tasks in industry. Therefore, a significant expansion of PLM applications is expected in the near future, particularly in medium-sized businesses.

Development of PLM Technologies

Product life cycle management was first conceived around 25 years ago. At that time, so-called engineering databases offered IT support for administering information and data that originated from design processes. At first, the main focus was on data storage and the integration of systems. However, it soon became clear that this was more than just an IT task. The term product life cycle (PLM) caused quite some confusion because it carried different meanings when used in different contexts or fields. Today, the term PLM is not only defined imprecisely and in different ways, but the definitions are also constantly evolving. As shown above, PLM is a concept that is used in different ways by users and software providers.

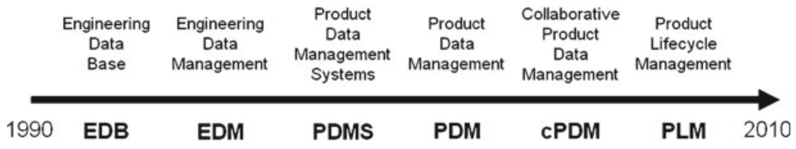

Engineering Data Base
Engineering Data Management
Product Data Management Systems
Product Data Management
Collaborative Product Data Management
Product Lifecycle Management
1990
EDB
EDM
PDMS
PDM
cPDM
PLM
2010

Fig. 6.3 Development of PLM-technologies

Figure 6.3 provides an overview of the development of PLM technologies in recent years. While industrial software developers consider their own systems (e.g. CAD, CAE, DMU, PDM, or ERP software) as PLM-oriented solutions, IT integrators do not see PLM as an IT system. Instead, they see it as an integration platform for engineering applications. On the other hand, strategic consultants define PLM as a vision, strategy or management concept. Regardless of the particular system, scientific institutions or market analysts view PLM as either an engineering/business approach or as a solution for the management of all product-related data and processes, depending on their specific perspectives.

6.1.3 Definition of Engineering Data Management (EDM)

There is no universal definition of EDM in the relevant literature. The following two definitions try to sum up the most common elements found in the various definitions of EDM:

- EDM offers an approach for integrating data and processes across the complete product life cycle and serves as a backbone for technical and administrative information processing by providing interfaces to CAD and PPC-systems, as well as to CAx-applications [4].
- EDM denotes the holistically structured and consistent administration of all data and processes that are generated, required and forwarded during the development of new products or the modification of existing products throughout the complete product life cycle [5].

Engineering Data Management System (EDMS)

EDMS act as integrative platforms that join required application systems (e.g. CAx-applications, Office programs, NC tools) via interfaces to one overall system.

Besides the data from the relevant systems, development processes and their connected processes are administrated and controlled. Data administration refers to the definition, creation, saving, controlling, control and spread of related business data. Data created during the product life cycle and administrated via EDMS include product configuration (e.g. spare parts lists), CAD models and drawings, random types of electronic and non-electronic documents, as well as project and workflow plans.

In accordance with the definition, an EDMS defines, controls and checks the process of product development or creation and also administers the dataflow. This is accomplished via the definition of processes and the related states and state changes, access rights and other activities along the process chain. Project management and scheduling can also be integrated into EDMS. EDMS can be used in individual areas or business units or on a company-wide basis. Thus, they are not restricted to the classical engineering area in the manufacturing industry, but rather can be considered for general information administration and process control requirements. For this reason, the ability to carry out the integrated design of product and processes offered by EDMS lies within the central management of master data, which enables continuous, widespread access to product data that is always consistent. An additional potential lies within the active control, creation, change and distribution of data, which enable systematic process management on the working, team and management levels [1]. Today, the remaining weaknesses of EDMS lie in the areas of standardized data exchange, integration of external applications and the parallel editing of documents. These problems mainly stem from the heterogeneity of the CAx-systems involved. One further problem is the redundancy of data between EDMS, PPC and ERP, which significantly handicaps product and process design in the implementation phase [6].

Concepts Related to EDM

The confusion which already arose when defining PLM is further increased by the implied system orientation of EDM and the resulting relation to PDM. For this reason, the concept of engineering data management is used as a synonym for a number of other concepts that share the same core targets, including:

- Product data management (PDM)
- Engineering database (EDB)
- Technical information system (TIS)

Some other synonyms and related concepts are:

- Document management (DM)
- Product information management (PIM)
- Technical (Team) data management (TDM)
- Technical information management (TIM)

In the United States, engineering data management (EDM) is commonly referred to product data management (PDM). This book adopts the following simplified definition of engineering data management as a discipline of data management within the engineering process that follows the concept of PLM using EDM/PDM systems.

6.2 EDM in Virtual Product Development

Virtual product development can boost the innovative power of OEMs and suppliers in the automotive industry. Expert knowledge is prepared for several parties with different viewpoints in a targeted manner using a primarily digitalized reproduction of company-wide accessible process and product data. The task is not only to manage and represent the development process up to the time of manufacturing, but rather to depict the complete product life cycle. Thus, companies can incorporate various external partners and suppliers in their local working environments. As a result, innovation cycles can be significantly shortened, and costs can be significantly reduced.

Crucial factors that contribute to the dramatic increase in the complexity of development projects and products for automotive suppliers include shorter modeling cycles, reduction of product development time, relocation and outsourcing of development tasks, growth of variants and the trend towards IT systems. These crucial factors will shape the requirements for EDM in the future.

6.2.1 Process Orientation in Product Development

Since the beginning of the 1990s, several approaches to the introduction of business process orientation in companies have been developed and applied. They provide the basis for process orientation in different disciplines. The most important approaches in this context are Business Reengineering and Business Engineering. Business Reengineering (BR) is a concept for restructuring organizations that involves a radical redesign of business processes.

Hammer/Champy [8] and Davenport [9], the two most well-known proponents of Business Reengineering (BR), have expressed their support for the idea of a quick transformation of an organization into a business-process-oriented enterprise. The core processes of the enterprise must be reorganized and remodeled according to target quantities, such as costs, time, quality and an increased customer orientation.

According to Österle [7], business engineering (BE) follows a top-down principle. In this principle, the business strategy takes precedence over the business process when establishing the connection to the information system. However, BE extends the procedure to take common business conditions into account. According to BE, these common business conditions are that a project can be positioned on any of the three possible levels, as long as the other two levels are involved. Figure 6.4 shows the integration of business strategies, processes and information systems within the different disciplines of a company. As the figure shows, it is necessary to consider strategy, process and system for every subject area, even if they take place on different business levels.

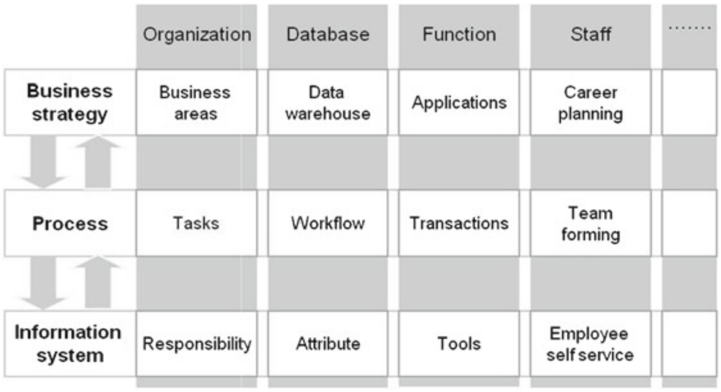


Fig. 6.4 Dimensions of business engineering [7]

The business strategy determines the general conditions for the enterprise, which provide the basis for decisions that must be made concerning business segments, IT structures, etc. Tasks are defined and teams are formed on the business-process level. These definitions act as a frame for the next level of detail, which is the level of information systems. The modeled business processes are implemented in information systems applications. Both the business-process level and the information systems level provide the basis for EDM, which seeks to connect the strategy with operative applications.

6.2.2 EDM as Integrated Management Approach

As stated above, when considered as an integrated management approach, EDM can have a crucial influence on a company's success. In the future, the successful introduction of EDM solutions will therefore be a very important component of business strategy. For this purpose, the systematic development and implementation of a long-term EDM strategy as part of the overall business strategy can be helpful. The strategic introduction of EDM is specific to the company. To accomplish this task, there are numerous methods and experiences from successful EDM users available today. While best-practice experiences can prevent mistakes and incorrect decisions, they cannot provide competitive advantages. Hence, a company should review existing experiences critically and use them selectively, while still daring to develop their own new, innovative concepts. Managers of companies that want to survive and be successful in the future must also recognize the importance of strategic management and devote time and attention to it, despite the enormous operative pressure.

The introduction of an efficient engineering data management system is indispensable for the establishment and successful operation of these very complex, distributed and knowledge-based engineering processes with the relevant employees and supporting IT systems. Recently, product life cycle management has become accepted as an integrated approach for engineering management across the complete life cycle of a product. As mentioned above, PLM comprises concepts, method and tools for the management of product-specific data and engineering processes, as well as for the integration of operative methods and application systems (e.g. CAD, CAE, CAM) in cooperative, globally distributed product life cycles, including manufacturers, customers, suppliers and partners. Although the concept of EDM includes tasks and interests of the management, the top management level of companies rarely feels involved. The concept also includes data, which again implies the management of computer systems, software applications or system administration of a specified area.

Thus, EDM is a management issue. Without EDM and the reasonably extensive deployment of modern technologies, it is impossible to provide the necessary organizational prerequisites for the development of innovative products at reasonable prices and in the short time that is available. In this context, EDM plays a key role because without the correct use of innovative technologies, there will be no innovative processes, and without innovative processes, the production of innovative products is unlikely. And a company without innovative products has little chance of being competitive, let alone becoming a market leader.

So many things depend on the way managers face these tasks. Company management cannot leave decisions about product data management to the individual business units. There is a wide variety of reasons why individual business units do not share the overall business view on this issue. Although EDM mainly concerns product development, it is relevant in nearly all areas of the company in synergy with PLM. As explained above, data and information are created and required within the business process. The representation of these processes, with all of the relevant functions, data, information and IT systems, forms the ideal basis for EDM strategies. Moreover, it forms an ideal platform for the maintenance and control of the implemented EDM processes. Users of IT systems are not the only ones who profit from this approach; it also enables a streamlined IT infrastructure with the resulting streamlined data structures. The generic representation of EDM processes can provide an overview of the interaction between EDM and the business processes, with a particular focus on the process map. The process map can be used to classify the business processes as either core or support processes and to visualize their position within the framework of EDM strategies. The development of sub-processes with different levels of detail make it possible to address all of the processes within the particular segments that comprise a product's life cycle. In the context of EDM, this means the product development process in particular.

6.2.3 The Product Development Process

The main result of the Product Development Process (PDP) is the intellectual product, which is the product description including all associated documents, descriptions, specifications, digital models and draft documents of all associated equipment (e.g. tools, machines, facilities). Additional processes include product manufacturing, the result of which is the physical product that is created via manufacturing and assembling, as well as processes involved in purchasing and the provision of the necessary operative resources (e.g. facilities, equipment, staff, financial resources). PDP comprises requirements management, portfolio and product planning, product design, and the actual product development, as well as the construction, analysis, calculation, simulation and testing. This process section yields a complete description of the (for the time being) virtual product. Within the framework of product definition, engineers are also involved in supply chain management (SCM), the design or acquisition of equipment, the planning of manufacturing and assembling processes (process planning), and the generation of technical documentation. An integrated EDM concept should represent all of the derivable requirements for technical product documentation.

Through development processes, engineers plan, design, define, verify and document products, as well as their production, implementation, use and disposal. Most engineering activities take place at the beginning of a product life cycle, which means before implementation and production. Various methods (e.g. design methods) and tools (e.g. CAD, CAE, CAM applications) support operative engineering processes and tasks. Engineering tasks that are continuously supported by EDM are often called *Digital Engineering* or *Virtual Engineering*. In the near future, changes in the industrial environment will drive the following important changes in engineering:

- Increased engineering cooperation across locations and disciplines
- Augmented engineering of innovative, multidisciplinary products, including embedded computational components
- Increased application of product-related services and comprehensive customer solutions
- Increased customer integration in early phases of product life cycle
- Increased importance of engineering in the product use phase and at the end of the product life (e.g. due to new regulations regarding product return and recycling)
- Increased protection of copyright (due to theft of ideas and product piracy)
- Increased attention towards new laws and regulations (e.g. product liability)
- Increased efficiency of engineering tasks (due to increased competition)

6.2.4 EDM Support in Virtual Product Development

In the context of EDM, virtual product development is understood as the most comprehensive description possible of a virtual product. Although the goal is to minimize the number of physical prototypes required, the products being developed are still real, which should influence all relevant product-development-related decisions.

All results of improvement processes should also contribute to a less expensive and faster production. For example, digital packaging, geometry checks and digital assembly evaluations reveal design problems (e.g. intersections of parts) long before the actual vehicle assembly. Building body prototypes for crash-tests is far more expensive than attempting to get the results via computer simulation of crash scenarios.

In the area of product life cycle management, the goal is to manage all data (e.g. CAD models) and types of work emerging from product development as completely as possible and to have the corresponding threads run together in one place. In former times, it was necessary to purchase many individual components. Today, a supplier is expected to deliver complete systems, such as the entire interior design or the complete lighting equipment. In order to create and conduct virtual product development systematically, a general process should be defined in combination with the specific processes concerned. The procedure should be designed such that the important, general functional units of engineering are queued in a logical order, which serves as a reference sequence. This reference sequence represents the foundation of EDM approaches.

In relation to the product, the EDM approach assembles all relevant product data in an EDM structure. This is the basis for the configuration of the specific product, the creation of virtual product models, and thus for an early validation (e.g. for manufacturing, factory planning and service). During each individual phase, the currently relevant product data are created and saved in relation to either specific phases (e.g. the concept, structure and production phases) or to a specific model and variant. While product definition occurs top-down, product validation occurs bottom-up within complete vehicle tests. Thus, several model and phase-specific product structures co-exist. Currently, it is not possible to align them to one common, life cycle-oriented product structure. The development of such a product structure is one of the primary challenges in virtual product development. Immersive visualization supports decision-making in complex problem situations (e.g. in styling review, construction testing, simulation data evaluation, assembly/disassembly research, tool certification and factory design).

EDM also supports simultaneous engineering with a particular target: to reduce the total sequence time by enabling parallel processing of procedural steps from conception to production scheduling that have traditionally been executed one after another. This target can be reached by means of a central 3D CAD data model, which supports all communication between the parties involved in development. Changed workflows enable design engineers to get feedback from manufacturing earlier, so they can incorporate this feedback into their work. Thus, post processing both during and after the design procedure can be reduced. The disadvantage of an early integration of different departments by simultaneous engineering is that development cycles are often carried out with insecure data status (see Sect. 2.2.5).

EDM supports another approach for efficient product development called frontloading. Front-loading involves the shifting of partial processes and resources that are crucial to results to early phases of development with the target of reducing development time. Providing high-quality product data at the beginning of development can help to identify mistakes early and to avoid costs for changes. The task of EDM is to establish the required framework to support this shift of resources and to aggregate knowledge and data related to a product as much as possible, in order to allow the product to go into production as quickly as possible.

Engineering workflow is the detailed connection of all activities related to the creation, description and alteration of a product, including processes across the complete life cycle of a product. These processes include operating sequences in the planning, design and adoption of the product, as well as the related workflows for change and technical release. The engineering workflow supports the most detailed level of processes.

Collaboration involves cooperation and the exchange of information between departments or business units inside and outside of the company along the product development process. The cooperation of different companies with a common goal creates networks of suppliers, customers and development partners that are focused on adding value. A reorientation around organizational structures and processes that are conducive to mutual cooperation should be important for all participants.

EDM can help provide the prerequisites for the realization of this ambitious vision, including:

- Cooperative, virtual and integrated product creation processes
- Early and close integration of engineering partners
- Increasing the number of experts who can contribute to a project
- Common cost and time management
- High transparency of decision making processes
- Synchronization of product networks
- Flexible and efficient system integration and system interfaces
- Efficient data management concepts

6.2.5 EDM Process Integration

EDM plays a key role in the redesign of products. The roles of the people involved are being gradually redefined, and processes are being reshaped. This is where the body of a project is created, to which the employees from the project team contribute steadily. In former times, it was a given that concept and design, construction and testing, tool and mould design, production planning and manufacturing were all established within one company. Today, this is only true in exceptional cases. Within the network of different companies and project partners, each individual participant has a specific understanding of its role, which is derived from the aforementioned tasks. For example, one company might develop innovative concepts and manage a project that leads to in-house serial production. Another company might deliver components or modules according to exactly defined requirements. Yet a third company might be responsible for calculations, simulation and perhaps even the design of different types of test beds. The distribution of tasks and responsibilities can vary greatly, depending on the industry, the product, and the size of a company [10]. Extensive international and intercontinental networks of engineering offices, development partners and suppliers have led to dramatic developments in data management systems in recent years. Within this network, EDM has become the essential engineering backbone.

Tasks of EDM Process Integration

Every sequence of operations that uses resources to transform input into results can be considered a process. Organizations have to identify numerous linked and mutually dependent processes in order to contribute meaningfully to the introduction of an EDM system solution. This systematic identification and handling of different processes within an organization and the mutual influences of these processes is called a process-integrated approach. Methods are available to support process integration. The need for both IT and methodological support are closely linked to the global trend towards an increased complexity of products and processes. In terms of processes, it is also necessary to establish cross-company and multicultural collaborations with linked work sequences, so that mutually influencing components can be developed simultaneously in different locations. In terms of products, the demands for flexibility and versatility have led to a greater number of models and options [11].

In order to enable processes oriented EDM, a system-comprehensive methodological level should be established within a company, which involves many organizational and cultural changes. The primary task of this level is the translation of strategic company targets into process requirements. Based on these requirements, appropriate working methods can be defined in the form of suitable process models (set processes). In turn, these set processes and customer requirements provide the basis for the development of concepts for IT support. The IT support concepts are then communicated to IT system developers, who then add additional system requirement details. The process integration level accompanies the development, testing, introduction and use of system solutions in the operative processes, which are implemented according to set processes. The process integration level must also track operative results in order to compare them to the defined company targets.

6.3 EDM Database

6.3.1 The Role of Development Data

Product data and development data originate and evolve during development. An efficient and secure electronic administration is one of the central prerequisites for successful engineering. These data do not become irrelevant upon product release and manufacturing, which is another important reason for the application of EDM. One might ask why the role of product development data has evolved over the years. The reason is that today complete, generally comprehensible, three-dimensional product descriptions can be used for numerous tasks that are not necessarily restricted to engineering. Furthermore, such product descriptions are the prerequisite for additional process optimization in other core areas, especially in manufacturing and assembly. This enhanced role of EDM is also apparent in the growing use of 3D CAD systems.

EDM is not only an instrument for the improved organization and efficiency of development processes, but also a hub that allows product data to be used as a knowledge resource throughout the company. EDM provides a crucial interface between engineers, designers and experts from various other disciplines involved in the product life cycle. Thus, EDM is also a crucial ingredient for the cohesion and effective functioning of interdisciplinary project teams, which are common in progressive companies [10].

Company-Specific Data

Company-specific data is of importance to a company and to the performance of its business processes. These data are derived from an organizational knowledge base and then documented in order to enable the transfer of knowledge to other authorities of know-how. System elements of the organizational knowledge base (i.e. employees) have to examine data by information processes to categorize such data as company-specific. Ongoing changes in business contexts also make it necessary to consider options for ensuring that company-specific data are always up-to-date.

Intelligent Document Management

Manufacturing companies also possess a world of documents beyond the article and beyond parts master data and product-relevant data and files. This additional data world is growing and becoming more and more complex and confusing. In terms of a company’s competitiveness and the organization of both internal processes and communication with partners and customers, the importance of this world of documents cannot be specifically quantified. For a long time, it has only been possible to estimate its importance. General document management differs in many ways from the specific conditions of engineering. The engineer usually works with particular authoring systems for design and simulation, and the results of these procedures play an important role, either directly or indirectly, for the product description.

The overflow of digital documents that are related to neither one particular article number nor to one particular project has reached a point where companies are forced to take action. Firstly, all relevant information should be organized, which includes more than just product data. Secondly, document storage must be simple and secure enough to allow employees to access and manipulate data without problems. Thirdly, data administration must enable quick locating and accessing of documents without knowledge of the document's actual content, origin or authors [10].

EDM supports engineers in storing and managing the emerging documents in a structured way. For example, engineers know very exactly which assembly unit a part belongs to during the design process. The working structure applied by engineers is generally the product's structure. Therefore, employees of the engineering business unit generally know how and where to search for relevant information, and their exploration is generally successful when using EDM. This expert unit is the main user of EDM in companies today. However, there is a huge amount of data generated which are not gathered by the engineering department but play an important role. For engineers, different processes beyond the application of their specialized software (e.g. table calculations, scheduling, presentation design) are minor sequences as far as *data management* is concerned. If they save the results of these sequences in individually created directories on local machines or a decentralized data storage unit, product development is not influenced directly. Indirectly, however, the lack of organization and a proper overview can lead to data inconsistencies. This problem was the original trigger for the creation of document management systems that support responsible parties by structuring, administrating and searching specific documents. Although the range of applications that contribute to the data such a program must manage is generally far wider than the equivalent range for EDM systems, the task is still very similar.

6.3.2 EDM Documents

Many documents emerge during product development and must be managed carefully. With the growing use of IT in companies, the number and types of documents are growing as well. Within a company, documents must always be managed, regardless of whether or not they are linked to product data, such as photos, presentations, letters, emails, animations, digital mock-ups (DMU), videos, drawings and protocols, to name only a few important representatives. Even totally independent from 2D and 3D CAD (for which PDM was originally developed), there is an enormous need to manage the different types of data that document any working step, procedure or contract. Until recently, such data were mainly created manually and stored in physical folders. Often, they were created using programs for text editing or outmoded methods on a computer. In the end, the data were printed on paper, signed and stored as documents, without even recording this somewhere or registering the storage location [10].

6.3.3 CAD Data in EDM

When introducing a database-oriented administration system, a primary focus should be placed on the administration of design data. Such database systems offer far more possibilities that should be considered when selecting a system without forgetting about future add-ons. CAD systems provide functionalities, which include main features for the management of CAD data. This represents a potential for the linkage with functionalities of EDM. As an example, a central access to design data and maintenance of a central assembly structure can be established. Users need to be made aware of the general complexity of the system in a logical step-by-step manner, in order to ensure acceptance of the system. Appropriate interfaces and integrations offer bi-directional matching of data between EDM and CAD systems. Thus, users can work in their usual CAD environments and enter all of the necessary input there. Functions beyond the scope of a CAD system (e.g. freezing or releasing a data record) are triggered directly in the EDM system. Securing this process also eliminates the need for frequent search for latest data and the integration of development sites and suppliers spread across the world.

6.3.4 Digital Mock-Up (DMU)

The concept of digital mock-up (DMU) includes more than just the *digital representation of a vehicle*. DMUprovides the foundation for a constructive development because individual results from different business units are always incorporated and can be evaluated holistically. The DMU is a digital vehicle dummy that can be used for various investigations, simulations and evaluations (e.g. collisions, functional minimum offsets, geometric properties, legal requirements and requirements of assembly and service) (see Sect. 2.2.2).

The source for this representation is the data management system for geometrical data, which enables structured storage of CAD data and attributes for a complete vehicle. In most cases, it does not rely on the product data management system to store the product-describing data. This structure makes it possible to define different variations of vehicles and to manage them accordingly. However, the number of variations to be represented also increases the complexity which the system must handle. The digital mock-up is thus the foundation for an interdepartmental *communication platform* and *information hub*. EDM systems enable the creation of complete vehicle structures based on meta information. This means that components do not have to be provided as CAD models, but can still be integrated in the product structure as components of the superordinate part list. Integrated EDM systems enable the automated provision of viewing-data formats and integrated viewing mechanisms of predefined virtual product configurations without implementing separate software.

Virtual products in vehicle development processes consist of many variants and combination options. Using variant management in the EDM system, particular assemblies can be filtered from a *150 %-parts list* after the definition of component affiliation. These concretely defined variants of a vehicle serve as the basis for the evaluation of package, geometric and functional characteristics or as a basis for subsequently performed simulation processes.

6.3.5 The Virtual Product

With the performance ability of modern information and communication systems, a vision of a consistent digital product creation process has emerged. This process could shift product development to the virtual world as far as sensible. Thus, there is a concept of a virtual product development that can be implemented in several steps of integration. However, the integration of additional product information categories in virtual product models clearly increases both the degree and the complexity of *virtuality*. As a consequence, the virtual product (i.e. its computer representation) may be viewed as the result of the simulation of development phases of a technical product and thus as a central information carrier of a completely computer-based, virtual product development.

Functional units must be validated within the optimization of the product development process in PLM and the linked EDM system based on virtual models. Therefore, the virtual product includes all computer-based and integrated models of a product throughout the entire product life cycle using virtual reality technologies as an environment. The virtual model makes it possible to consider different groups of participants (e.g. developers, suppliers, manufacturers and customers) equally. In addition, the model makes it possible to handle the virtual product and its specifications up through service and recycling in an exclusively virtual manner and to evaluate properties and functions in an environment that is close to reality. Such a virtual product is also referred to as a virtual or digital prototype. The concept of virtual mock-up (VMU) is also used, although this term denotes the expansion of DMU to a virtual prototype. Beyond the geometric validation provided by DMU, a VMU is intended to validate the technical aspects of both the functionality and production of a product with reduced or no use of physical prototypes.

The creation of such a prototype is the primary strategic target of modern IT-supported product development processes for a common communication and decision making because it could make it possible to examine the future product's quality in all its facets in advance. Furthermore, products could be defined in relation to customer demands, and the feasibility of their manufacture could be tested in advance. To date, specific characteristics of the product are altered in the workstations via modeling and verification functionalities (i.e. virtually). Thus, the virtual product is the computer-based, realistic representation of a technical product with all functions required within the product life cycle, which implies a near-realistic simulation model that includes all product information and functions, as well as all the properties of the later real product.

Digital Prototypes (DPT)

The application of digital prototypes (DPT) enables the examination of all parts of a new product in terms of their interplay and functionality at an early stage of development. For example, endurance runs are conducted to test the reliability or breaking strength of certain elements [10].

The increased sophistication of 3D modeling and its gradual evolution into a common tool of automotive product development have had a significant impact on the testing phase. The more sophisticated 3D modeling has become, the more the testing phase has shifted away from hardware prototyping to digital prototyping as a virtual product. Testing and calculation methods have also been digitalized in recent years. Besides the Finite Element Method (FEM) for calculating component stresses, a variety of programs are available today that allow development teams to simulate nearly all kinds of product functionality on the digital product. It is easy to see the great potential of these simulations. For example, some automotive manufacturers are already simulating the noise in different driving situations. However, there is one shortcoming in modern product development: the data management of these disciplines is not yet integrated continuously throughout the complete product development process.

6.3.6 Data Security

Data security, one of the main issues with respect to managing development data, is a multifaceted topic:

- Firstly, the central purpose of EDM is to collect and save all data created in the course of product development. The data are not only stored, but also made continuously available. However, this is not always the case with hardcopy drawings and working schedules. Therefore, without electronic data management, it can never be guaranteed that minor changes in a drawing or part model will not be lost.
- Secondly, EDM-administered data is secure against unauthorized use or change because PDM uses access rights to reliably control data access.
- Thirdly, EDM data offers high information security because the system always knows the current status and version of any document. It also knows if it is still up-to-date or out-dated, the author of the data, and the system they were created with.

In addition, this information security is a crucial factor for increased process security because only correct data and files can lead to the successful achievement of project goals. Finally, it is important to mention another aspect of data security that deserves special attention because it is not automatically present when using EDM - securing data against theft. Since this aspect of security can be quite complicated, a complete treatment of this topic is beyond the scope of this book.

6.4 Engineering Data Management System (EDMS)

It is crucial to take a closer look at some concepts from the system-technical perspective before investigating the functions and applications of EDMS in detail. In essence, information systems technologies that are part of the common programmer toolkit provide the foundation of EDM functions.

6.4.1 Product Data Management System (PDMS)

In essence, and for the purpose of the present work, the concept of engineering data management systems (EDMS) is based on a classic PDM system. Thus, the functional description of the system is executed in the manner of a PDMS. All comprehensive discussions in this work are performed in the context of the definition of EDM found in Sect. 6.1.3, in which PDMS is defined as a main component of EDM. The designation EDM is used for the general system-technical relation in the further discussions. The term PDM is only used when the PDM system is meant exclusively and explicitly.

Studies have shown that the search for relevant information can take up between 30 and 70 % of an engineer's project work [12]. Therefore, if engineers could work with up-to-date data from the very beginning, this would generally lead to higher product quality because wrong decisions caused by obsolete or missing data could be avoided. This requires the supply of not only product-specific data, but also indirectly product-related information (e.g. the names of experts who might contribute to the solution of sub-issues, even if they are not directly involved in the development process).

PDMS has a long history that started some decades ago. These systems grew out of a desire to manage CAD drawings. The growing need for CAD drawings increased the need for data management tools that would be able to handle different types of digital documents created by different programs and systems. Such tools were called engineering database (EDB) or technical information system (TIS); other designations (primarily used in the US) were engineering document management system (EDMS), product information management (PIM) and product data management system (PDMS) [11]. These systems and programs were the main components of the central administration and organization of technical data and information in the CIM concepts of the 1990s. The functionality of these systems developed from the initial approaches of pure document administration to the administration of all development data of product development, and then further to the management of product data and schedules of all complete product development processes and product life cycles. Even though the term product data management system (PDMS) has prevailed, there is still a great variety of concepts.

Building a PDMS

A PDM system offers users numerous application-related functions, which can organize, administer and distribute all of their electronic data and documents in a product and project-related manner either within their own company or among several companies in the case of cross-company cooperation. Anderl [13] states that the main functions are those for object or element administration, privilege and access administration and process and data administration. In addition, cross-application functions enable the administration and customizing of PDM systems and thus guarantee the check-in and check-out of data stored in the system database and the archiving on system volumes.

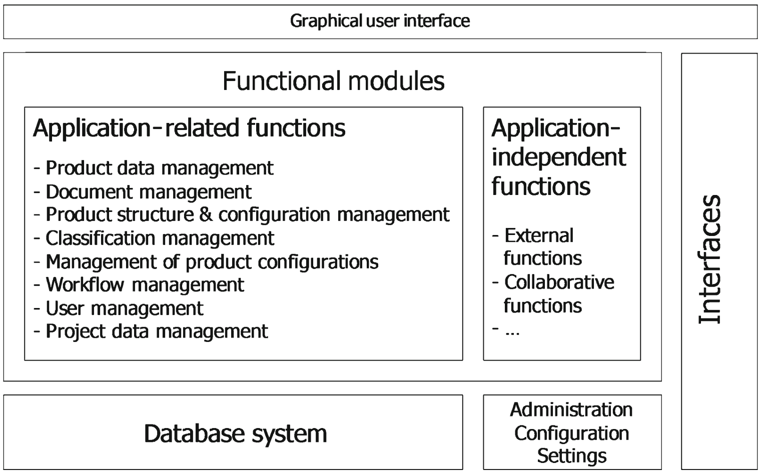

Graphical user interface
Functional modules
Application-related functions
- Product data management
- Document management
- Product structure & configuration management
- Classification management
- Management of product configurations
- Workflow management
- User management
- Project data management
Application-independent functions
- External functions
- Collaborative functions
- ...
Interfaces
Database system
Administration
Configuration
Settings

Fig. 6.5 Exemplary PDM system architecture [14]

Figure 6.5 provides a basic overview of the principle design of PDM systems as suggested by Anderl. At the core, there are the two functional modules, which are divided into cross-application and application-related. These are built upon a data foundation that contains a database and a database management system (DBMS). User interfaces and system interfaces enable communication between human beings and machines. According to Bullinger [4], PDM is the key technology for targeted knowledge management in product development. In this approach, product information should not only be made available to people who are involved in product development, but also to those outside of the actual product development process (e.g. suppliers or customers). This makes it possible to reach the desired mutual corporate targets.

Although PDMS were developed as a kind of product knowledge management tool for the central long-term archiving of product data, their archiving function ends when the data reading and writing system ends, be it a technical end or a general

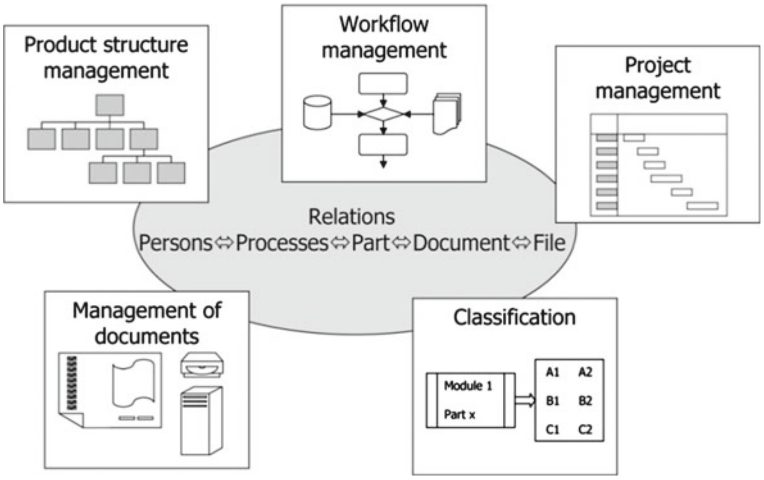


Fig. 6.6 Main functionalities of PDM, according to [15]

loss of data. Another issue is, that there are currently no integrated concepts for the general storage of particularly important product documentation files that should be filtered from the overwhelming mass of PDM information. Such files should be saved, archived and maintained in a separate part of the system so that they can be manually isolated from other data and intentionally transferred to another system.

Main Functionalities of PDM

The main functionalities of PDM can be divided into five areas.These areas are connected via relationships between the acting persons, the predefined processes and the product components (parts and assembling), which are described in documents and files. Figure 6.6 shows the five main functionality groups of PDM. In general, product-oriented functions include data and processes related to the generation and description of the product itself, whereas process-oriented functions include all tasks related to the development and life cycle.

Product Structure Management

The management of product structure and variant data supports the organization of components and modules. Especially in the case of complex products, the assembly of different versions requires specific methods for recording and administrating the dependencies and interactions of built-in elements. Modifications of individual components or sub-assemblies during the development phase or revisions in the product life cycle are documented and evaluated. An exact knowledge of the product structure ensures the generation of current bills of material during the entire development phase.

Workflow Management

Global companies act globally. Engineering in Austria, production in China, and the company leadership in the USA—configurations like these are common in the automotive industry. Globalization entails opportunities and risks—and in the case of automobile manufacturers, a significant challenge for data management. Pre-defined standards and rules in PDM specify who is allowed to change a component and how this modification has to be approved. Workflow management includes the control and distribution of documents and the integration of the customer and supplier into different processes. During development processes, the management of CAE data and the change management (including problem identification, modification measures and release management) are important. Applications include the control of documents (modification, verification), the integration of customer and supplier, the distribution of documents, CAD data management, DMU/CAx workflow and change management.

Document Management

PDM controls the access rights of product data and ensures the use of current information. Modern PDM systems have direct interfaces to software applications (design software, cost-calculation programs or manufacturing-related software) and are directly accessible for modification or data updates. Document management covers the provision of data and documents and data archiving, as well as document control and distribution. Applications include the provision of data and documents, data archiving, document control and document distribution.

Classification

Standardization of components and the use of standard parts are important factors for the reduction of product costs. Classification of product data across the entire product range supports the detection and application of (existing) parts and modules in the development of new models.

Project Management

The management of product data and the storage of information related to components and structure support project-management-related tasks. In this way, a powerful PDM system is able to simplify calculations related to cost and time for both current project and the advance planning of future projects.

PDM systems are based on client/server-based software architecture. They are characterized by a modular configuration of system components, which are organized by relational or object-oriented database management systems. PDM systems are equipped with several data interfaces and user-oriented modules to enable effective data transfer. Figure 6.5 shows the main modules of an exemplary PDM system architecture.

6.4.2 Application-Related Functions of EDMS

Based on a PDMS, an EDMS can easily be defined with some additional functions. These functions primarily consist of improved application relation, increased integration of authoring systems, application of simulation and viewing tools, and workflow management across projects, processes and data. Based on the design of PDMS (see Fig. 6.5), the particular functional entities of an EDMS can be described. From the wide variety of application-related functions of EDMS, master data management, document management, release of changes and the DMU process are described in detail below as representative functions for automotive product development.

The cross-application functions of EDM are:

- Change management
- Workflow management
- Process management
- Project management
- Communication
- Research, visualization
- User administration
- Data protection, data security/storage
- Archiving

Master Data Management

Master data management (or product data management) comprises the administration of all product-describing master data (e.g. documents, stock lists, parts lists, product structures), as well as specifications and classifying properties. In addition to data expansion during the development process (e.g. from CAD start-up model to the final design model that is released), revisions and configurations must also be managed [2].

Master data management requires clear definitions of data objects and their data fields to guarantee consistent, non-contradictory and harmonized master data throughout the complete product life cycle. Master data management also defines maintenance processes for the creation and alteration of data and specifies the organizational entities responsible for these processes. In this context, the best practice is to maintain data where they are created in the technical department and then combine them with a central point of master data maintenance (technical product documentation) that is also responsible for the process as a whole. This overall process is understood as the continuous control and optimization of data and may even cover the maintenance of crucial data content (e.g. article number determination).

Finally, master data management also includes the definition of a master IT architecture with data flow and data distribution to all systems that need master data for post-processing. For this purpose, the clear definition of which systems will maintain and change which master data objects has proven successful. These systems are the single starting point for data distribution. Additional generic PDM processes that mainly happen in the context of manufacturing and logistics (e.g. enterprise content management, asset life cycle management, quality management) are beyond the scope of the present book.

Document Management System (DMS)

Document management (DM) is designed to help with such tasks as scanning, creating, administering, forwarding, storing, archiving, opening and searching documents, with the primary goal being to increase productivity by shortening the document processing time and by providing the necessary information immediately. Document management systems (DMS) do not only support office processes but also optimize them via the definition and support of dynamic sequences of procedures and the integration of static information objects.

Currently, many businesses contain inefficient work procedures and redundant parallel work. Time-intensive document search, long processing times, multiple storing, lack of transparency of systems and procedures, differences in the applied data formats and the incompatibility of partial systems inhibit integrated, customer-oriented and efficient workflows. In addition, the number of documents has exploded in recent years and continues to grow. This situation makes it necessary to use computer-supported document management systems. Effective and efficient office work implies company-specific technology management and efficient information management that support business processes. Information management determines the flow of information in production and administration. A growing number of companies are recognizing that improved information management is a important factor for competitiveness. Specifically, information management includes the logical and complete administration and editing of all documents that a company produces or requires.

Imaging is the conversion of paper documents into digital representations. Imaging data allows only a rough estimation of complete system functionalities. Imaging systems allow users to create an electronic copy of a paper document via a scanner and to save the digital representation in a computer, which eliminates the media disconnect between the computer world and the paper world. However, it yields a non-editable digital representation of a page (i.e. a representation that cannot be interpreted by a computer).

The market for DMS has developed enormously in recent years. Companies that introduce such systems expect cost reduction and improvements in productivity and quality. A growing number of small and medium-sized businesses that use electronic document management emphasize that these expectations are more than just the marketing strategies of the system providers. DMS supports the storage of drawings in technical offices, the storage of receipts in banks, the archiving and locating of a variety of textual and pictorial material in magazine publishing houses and press agencies, as well as text and image flow and applications in insurance agencies and public services. Just as PDMS was becoming accepted as a suitable tool for the administration of development-specific data (i.e. design data), other software came on the market for the administration of general documents.

While PDM soon focused on a clearer and more reliable representation of complex 3D design models and the automated creation of parts lists, DMS focused on revising and linking random documents. No matter whether companies adopted DMS or PDM first, eventually they began to wonder whether it would help optimize procedures if only one database was used. This means one database to collect all documents in synergy with document data and to administer them accordingly. The ability to provide such functionality is another important criterion for the selection of a PDM system.

6.4.3 EDMS Architecture

Integration platform features direct the coupling of EDMS and other software applications, whereby the exchange and synchronization of attributes and parameters or structures occur on a meta-data level via an integrated data model. Usually, this integration involves no exchange of native data, and even when such an exchange occurs, the relevant data are *checked out* to provide some applications. Therefore, EDMS maintains the status of data master for all integrated applications. (Section 6.5.1 provides a detailed description of CAx-integration, one important application of the functional module integration platform in vehicle development.)

Putting together all of the functional modules partially described above, it is now possible to provide an overview and functional design of the modules in the form of an EDMS architecture diagram.

Figure 6.7 shows the specific functions of a comprehensive engineering data management system in a rough alignment with the superordinated components following the design of a PDMS:

- Application integration platform
- System integration
- Project control environment
- Engineering database
- Interfaces, data exchange and data preparation
- Access and administration

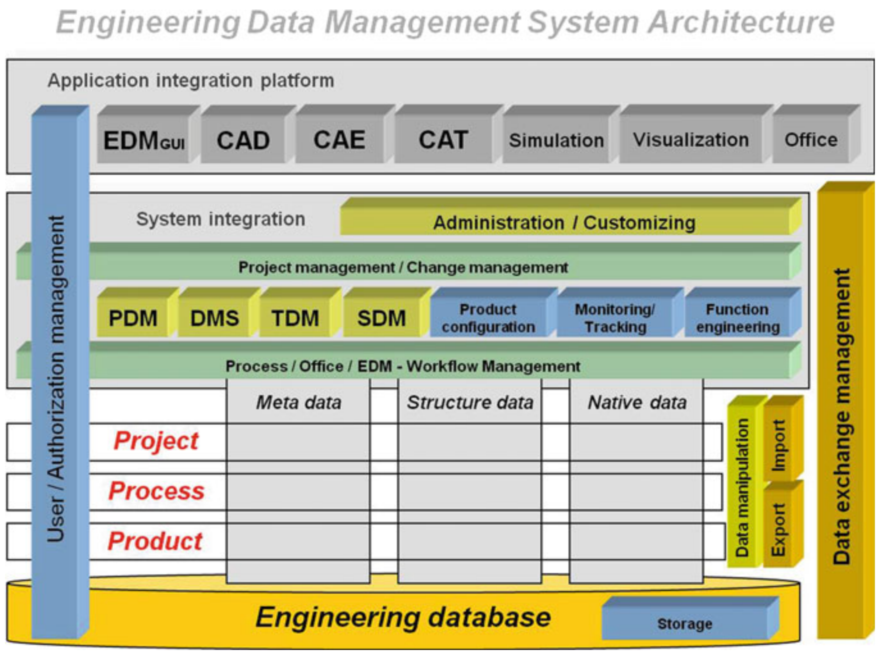


Fig. 6.7 EDMS architecture

Built upon a data foundation that is now obviously extended with project and process data, the figure once again shows the main modules, application, control and integration platform of an EDMS. The integration platform contains the aforementioned CAx-applications. The control module contains the process, project and change management functions on the one hand, and the other operative workflow management on the other hand. An over-arching user/access rights management system controls who can use or access the data.

6.4.4 EDMS Interfaces

At the beginning of computer-supported product development, very few interfaces existed. Systems with software that was completely separate from other applications were considered ready to use. Only special hardware was able to run certain software, and data output was performed via screens, printers or a plotter and saved on data tapes and (compared to today's standards) huge discs. Although these circumstances are long gone and past, isolated islands of many different products from numerous providers on different hardware have developed in many companies over the last years. As companies continue to optimize their processes, their systems and software become more and more specific, and the importance of the links between these elements increases.

In terms of product data management, the goal of interfaces is not only to enable data exchange, but also to ensure that data is always up-to-date and synchronized, even beyond system borders. The goal is to implement the majority of these functions in the background (i.e. with no user effort). The degree to which such systems function as integrative components ultimately determines the overall quality of a particular system technological solution. This function is precisely what is expected from EDM in its complete form - the integration of all systems and applications involved in the product development process. It is necessary to define how the master data of different systems align with each other. This definition ensures that data is mapped correctly to the defined fields whenever a data record is created, changed or deleted, thereby guaranteeing that both systems will remain accurate and up-to-date. When data structures differ between different systems, automatic synchronization is required.

Integration has a significant advantage if all parties involved are using PDMS. In such cases, geometry data can be displayed via a viewer that is usually not saved on the ERP side. All of this is possible and has already been implemented several times. Unfortunately, due to old conflicts resulting from a failure to use interfaces as bridges, one aspect generally does not receive the necessary attention—the synchronization of data (either automatically or manually) and the question of who triggers synchronization and which information is transferred from where to where.

EDMS Interfaces for Data Exchange

This section covers the support of a controlled, documented and traceable method for exchanging data between the internal clients and external customers or suppliers of a company:

- Internal clients/suppliers: design, CAE, production planning, purchasing
- External customers/suppliers: OEM, component/system development suppliers

The functions of such import/export interfaces are:

- Data transformation, mapping and conversion
- I/O logics und methods
- I/O automation
- I/O quality, conformity and plausibility checks
- Protocolling of data transfer processes

6.5 Computer-Supported Engineering in the Context of EDM

The strong growth of IT-supported development is ongoing. Replacing drawing tables by computers and software was the first step, which is being repeated today with the implementation of parameter-oriented working software in a similar way. CAx systems are more than just an electronic replacement of previously existing methods.

They are also an extensive functional extension to fulfill the increased requirements of product development. Simultaneous engineering has increased the amount of information and data present in the development process and thus further increased the complexity of tasks. Flexible yet standardized methods are required to maintain the overview and to provide possibilities for continuous monitoring that can be implemented from the organizational level down to the CAx detail level.

6.5.1 How CAx Changes Product Development

Although PDM can manage such tasks as the administration of released design data and relevant parts lists and their preparation for manufacturing, logistics, calculation of costs and product management, this is not yet EDM. The core of EDM is the management of the complete product development process starting with the initial request and then continuing through concept and detailing, serial development and validation, mould and tool casting, and all the way to the prototype and examinations, including the complete development history with revisions change requests and workflows. However, EDM has additional possible applications in the context of the structured management of three-dimensional product data in the CAx environment (e.g. CAD, CAM, CAE) [10].

Software applications are represented by different groups of authoring systems, such as:

- Computer-Aided Design (CAD) refers to computer-supported engineering and is used as an umbrella term for 2D and/or 3D applications.
- Computer-Aided Engineering (CAE) refers to computer-supported engineering procedures and encompasses all methods of computer support for working procedures in the technical context. This includes calculations, analyses and simulations, such as strength calculations or flow simulation.
- Computer-Aided Process Planning (CAP or CAPP) refers to computer-supported work scheduling, which generates data for component manufacturing and assembly based on conventional or CAD data.
- Computer-Aided Manufacturing (CAM) refers to computer-supported manufacturing, which features the direct control of manufacturing facilities and supports transport and storing systems.
- Computer-Aided Testing (CAT) refers to the computer-supported and automated implementation and the interpretation of hardware tests [16].

CAD, CAE, CAT and CAM have a significant influence on the process of product creation. Figure 6.1 shows their positions in the Y-CIM model. None of these fields can be considered an isolated field, and engineers must have at least a basic knowledge of all of them. If such knowledge is present, CAx can clearly accelerate the production process. Since all fields require expertise, they are tackled by multiple experts. This requires communication (see Fig. 6.8) between them, which also depends on a common terminology.

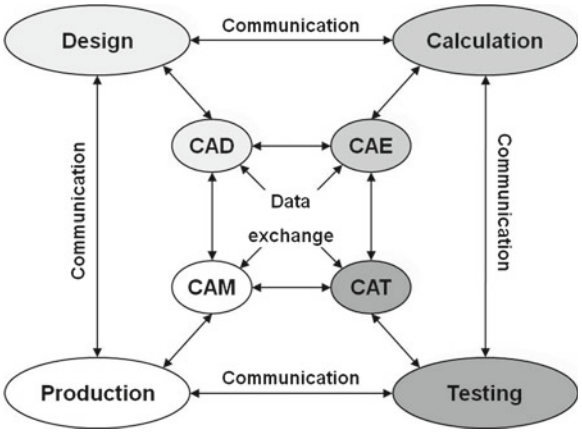


Fig. 6.8 Relationship between CAD, CAE, CAT and CAM

The even greater challenge is the exchange of data between the fields. First, there must be a suitable exchange format. Second, the requirements for the individual fields differ. This makes data exchange harder and inhibits cooperation to some extent. Within the complete process of a simulation, more than 50 % of the time is used for data research and management [17]. Continuous data management between parts list, CAD and simulation and the integration of simulation data (properties, results etc.) into an EDMS can reduce this percentage.

6.5.2 CAD Integration

In most companies, the most important authoring system in product development is certainly CAD software. This is where models are created, detailed and subdivided into components and assembly groups depending on their intended functions. The efficiency and duration of engineering and the degree of creativity, which is the engineer's core competence, depend on the degree of integration of the EDM software.

CAD software is particularly important when multiple CAD tools are used (i.e. in the case of multi-CAD installation). One frequent practice is to equip each of these programs with its own team data management system (TDMS). This is frequently the only way that specific software features can be used. Furthermore, it is only possible to represent results in the models and to make changes if access to these special features is granted. More often, it is nearly impossible for these special features to handle models, mock-ups or drawings from other applications because they are completely integrated data management systems that might even be based on the same data format as the one provided by the CAD package.

This highlights one of the primary challenges faced by EDM as a tool for integration—the need to exploit the full potential of a heterogenic CAD landscape without losses and to provide a central management of development data. Another aspect of EDM integration is that the design process is normally equipped with a CAD system, either alone or in connection with proprietary management software that is intended to last for several years. But what happens if the system must be altered? What if, for whatever reason, a different CAD system needs to be installed? How can design data from previous projects be handled in such cases? How can they be maintained and further used in the new system, if necessary? This can lead to enormous costs for companies who lack a systematic, electronic storage system for all of the data that can handle output and, ideally, the conversion of the relevant data. In this case, it cannot be guaranteed that none of the 'old data' will be lost in the conversion.

Integration basically means that PDM functionality is offered within a CAD system. Design engineers do not have to leave the application, but rather can create, save, load or delete CAD data in PDM via user interfaces. For these engineers, integration mainly means an extension of application menus, which includes functions that the CAD system alone cannot provide [10]. The integration of CAD and EDM also means that design engineers do not depend on a single CAD system with consistent classification features and EDM functionalities. In multi-CAD environments, this advantage should not be underestimated. On the other hand, it means that EDM accesses the data of connected CAD software directly to represent them subsequently via particular viewers.

One critical point for most EDM implementations is the question of how well and how completely existing development data (e.g. data created by the customer) can be converted and imported as a basic design status to the EDMS. When implementing an EDM, the project team faces thousands of CAD objects, all stored on decentralized, individual CAD work stations, which must be imported into the EDMS. In addition, EDM and CAD are rarely used in parallel. For this reason, high-quality EDM software should always offer tools for importing existing data. Beyond this initial import, it is important to plan and organize data change management with customers and suppliers and to implement it in EDMS. For this purpose, virtual data milestones are established where specific data management activities take place. These milestones are called virtual prototypes (VPT) or digital prototypes (DPT) for the virtual product. To this end, so-called data synchronization points (DSP) are set up for operative data management. All these control and operative activities are represented as project planning in EDM workflows.

6.5.3 CAD Implementation

In order to reduce the development time for a product and to guarantee the availability of product data for all affected departments, most manufacturing industries need their CAD systems to be directly integrated into their business processes and EDM systems.

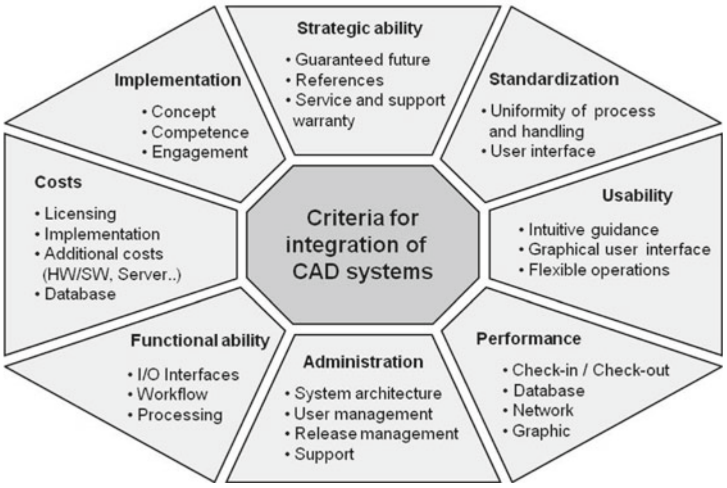


Fig. 6.9 Criteria for the integration of CAD-systems

According to Scheer [2], it is necessary to consider some important criteria for the integration of CAD systems into a PLM environment, as shown in Fig. 6.9. Thus, EDM should be planned in advance, and project-specific functional requirements must be coordinated between the engineering teams and the EDM development. Usually, EDM functionality is implemented in a development project following an EDM stage plan.

6.5.4 Virtual Computer-Generated 3D Product Design Models

In current engineering product development, steadily growing virtualization tasks are causing an increase in the emphasis being placed on 3D product design models. These models mainly consist of:

- the 3D CAD model, which is the geometric foundation of the product and thus also the foundation of any modern virtual product development;
- the 3D DMU model for digital mock-up, which is exported from 3D CAD models as a digital dummy for different simulation purposes;
- the 3D CAE model, which is specially prepared for calculation and simulation;
- the 3D VIEW model, which is a simplified 3D CAD model for viewer solutions;

- the 3D VR model, which is created during product creation by converting 3D CAD models based on the focus of the VR system in use and is edited in terms of properties and behavior; and
- the 3D STORAGE model, which is a format that is derived from a 3D CAD model and is independent from any particular authoring system or 3D viewing software. It is used for office applications or for purposes of reproducibility in long-term archiving (e.g. 3D PDF format).

In this context, a 3D CAD model mainly represents the geometric properties of a technical product, its product design and its structure. CAD models are composed of different model-building geometric basic objects: points, lines, surfaces or volume models. Among these options, the volume model is the most prominent due to its current performance, particularly with respect to its re-usability in product design in parallel and subsequent partial product creation processes.

The creation of 3D CAD models is supported by so-called technical modeling because geometric modeling alone does not provide an adequate product description. In technical modeling, technical properties that do not only influence geometry (e.g. material, tolerances and surface treatment) are assigned to the model. In addition, relationships between geometric elements and functional information (e.g. features that influence geometry) are assigned. The features are particular aggregations of design characteristics in terms of their properties, values, relationships and constraints. Chapter 4 includes a detailed introduction and description of modeling techniques in 3D CAD.

One difficulty lies in importing 3D model product data into the EDM system to integrate them in engineering data management. It is also necessary to create specific data models for the administration and control of features and to integrate these models into EDM. Consequently, the use of different CAD authoring systems causes severe problems for EDM integration. This leads to software and hardware problems, as different software providers follow different market strategies, continuously adapt their interfaces, and use differing CAD methods and training programs for design engineers.

The 3D DMU model represents a digital model of a CAD-based product that contains the component geometry and assembly group structure. Geometric design primarily evolves via tessellation from a 3D CAD model (derivative approach), but there is also a native approach which uses original digital masters of CAD models (cf. Sect. 2.2.2). The DMU serves as an information and communication tool for model analysis, which is primarily used for validating and optimizing a product with regard to its spatial and functional design. DMU systems offer numerous functions for this purpose, such as a visualization of the virtual product for the evaluation of engineering spaces and collisions via kinematic simulation. The aim of DMU is to provide consistent availability of multiple views of the design, the expected functional range of the future product, and further technological relations. One important application of DMU is the joining of heterogeneous models from different CAD systems, after which the results can be re-imported to the CAD model, if necessary. The DMU geometry reference is usually represented in the team data management (TDM) of an EDM. Thus, the unambiguous DMU specifications are product configuration, product view and model state for a defined virtual milestone.

Viewing models are generated to provide all users with access to a product's visualization (regardless of their particular authoring systems) using popular viewing tools. Commercially available viewing tools are inexpensive, easy to use and may be used in a wider application range via WEB technologies, even across different company. Each VR model relates to one 3D design model and is then extended by the representation of the relevant design model's behavior in the virtual world. VR models are often created from surface and volume-oriented 3D CAD systems, which do not depend on the interfaces of the particular VR systems used [18]. The VR software in VR systems can also be used to represent the movements and reactions of a VR model. VR is presented to the viewer in real time. The real-time presentation of the core of VR (i.e. connected 3-dimensional computer graphics), immersive design and the need for interaction lead to high requirements for VR models and their virtual operational environment. In particular, sensory fidelity must be taken into account to prevent uncomfortable side effects for the viewer both during and after the virtual experience.

All of these different *3D design models* must first be connected and represented in EDMS. However, the greater challenges are the integration of these different CAD representations with their respective product structure views, as well as the distribution of data to highly differing applications and the integration of specific authoring systems (CAD, CAE, DMU, VR, viewer, office software) into EDMS.

6.6 Integrated EDM Applications in Product Development

The following sections examine some concrete application scenarios in detail, focusing on EDM support of CAD and CAE applications and their integration in the product development process.

6.6.1 Functional Dimensioning and Optimization in Early Design Phase

The role of CAE in the early design phase is continuously changing. Parametric geometry models are able to support the re-calculation of models from former projects to provide basic data for initial calculation or simulations. These early geometry model often feature simplified geometries, which lead to a reduced level of detail in CAE processes. Currently, vehicle concept decisions must be validated during drafting and development. As continuous CAD descriptions are not available in this early phase, CAE methods and procedures for a rapid creation of parametric geometry models are directly interlinked. Only then, the CAE engineer can contribute to new developments by evaluating crucial influences on the required functionalities.

Evaluation steps are always oriented towards the state of development. The ability to adapt reactions during the design phase based on the relevant depth of information is essential. Even if no detailed information is available in the early phase, rough parametric models should be provided that are able to define space, topology, styling and package requirements. Topology-geometry models are also evaluated in terms of their suitability to fulfill functional requirements throughout development.

In the end, these models are meant to provide *interlinked vehicle design*, meaning that they should provide a foundation for the concept team, which consists of specialists for package, calculation, pre-development and design. Topology and geometry models should be able to adapt quickly to new package or styling information. Moreover, they have to support the optimization of geometrical topologies and address issues of standardization, commonalities, vehicle families and exchangeability. In addition, topology geometries should ideally offer a data interchange interface. This common data exchange platform will serve for:

- Re-using implicitly parametric vehicle models
- Creating, structuring and distributing knowledge
- Using knowledge from current and past development projects
- Improving of data and data consistency

A topology geometry model is highly valuable. The concept engineer must be able to provide statements on crash behavior, stiffness changes and geometric compatibility with existing 2D/3D package requirements. In this way, the topology geometry model supports the engineer in deriving not only an analysis model but also a design model (including design variables, design space, target function and constraints) and in directly verifying and supporting the optimization of the topology. For this purpose, a topology geometry model enables the automatic generation of a robust, high-quality grid for finite element computations. In addition, it derives the necessary joinings (e.g. bonds, seam welds, spot welds) and the FE conditions from parameters. In this context, it is important that the degree of detail depends on the state of information during the design process. Thus, there is always a compromise between the desired precision of results and analysis efforts.

6.6.2 Consistency of Simulation Data in Optimized Design Processes

The desirability of having continuously usable data throughout the design process is obvious. However, it is important to acknowledge some conflicting targets, such as those present in functional validation and geometric design requirements. The used design tools should always follow and support the actual current states of information. Parametric models must be designed in such a way that the design engineer can implement parametric changes that were not previously planned at any time. In order to fulfill these requirements, references in parametric design systems are changed and defined for interpretation (e.g. for topology optimization). To enable these features, the design system should automatically create and interpret new geometric configurations and the resulting parameterization. The demand for a parameterized concept model that can be used continuously up to the point of serial production is neither realistic nor productive.

EDM plays a essential role here because EDMS can import parameter data from authoring systems through direct interaction and thus cover most of the issues of product development. On the other hand, EDMS can also act as the parameter database itself, so non-interlinked systems (e.g. ERP and PPC) can be supplied with parameter data. EDMS is also used for configuration management in this context and can therefore be understood as a *knowledge database* as well. High data integrity at this level also increases the degree of automation of data management activities and consequently increases the trust in the reliability, availability, currentness, and quality of data.

6.6.3 Interdisciplinary Consistency of Simulation Data

The effort required for the management and administration of product information is growing significantly as the volume and complexity of data continue to grow. Company-wide system integration solutions in technical departments are currently limited to design data due to the relatively high degree of overlap of information from different CAD systems. As the complexity of product structure definition grows, completely new requirements are emerging within and between different engineering disciplines because the level of shared information is very low (e.g. design and simulation data). Interdisciplinary structures of this kind enhance the quality of representation of product development history and configuration and thus represent an initial step towards interdisciplinary integration. However, the requirements are considerably higher for extending this approach for the common use of application data from other disciplines. For this purpose, the information objects within product structure must inevitably be adapted and standardized in such a way that they can be processed and interpreted by different systems and tools from multiple disciplines. The premise for this is the structuring and modularizing of the processes that deal with these information objects. For example, one requirement is to structure operational processes such that they have input and output information, which establishes a foundation for shared information objects.

Configuration management, the core competence of EDMS, also offers a general solution. The difficulty here lies mainly in defining such a structuring standard that takes into account the requirements of all disciplines and in institutionalizing this standard in the company. In addition, the respective experts must create and maintain the different structures in EDMS. Finally, the issue of change management should also be addressed on every structural level.

6.6.4 Integration of Design and Simulation

Development times are getting shorter in order to react as quickly as possible to trends and customer demands. New, up-to-date development methods must be established to keep pace with this trend and prevent a decrease of product quality. It is very expensive to remedy product defects if they are detected too late. Thus, late detection results in considerably higher development costs and, in the worst case, even to a delayed market launch. Design evaluation has traditionally occurred at the end of the design phase. Since different types of simulation (e.g. mechanical, thermal, kinematic and acoustic simulations, as well as flow simulation and molding simulation) are often implemented without CAD-integrated simulation solutions, data transmission and conversion is cumbersome, and specialist staff must be employed. Automotive manufacturers need more integrated simulation at earlier stages of the development process in order to reduce costs (especially costly physical testing) and to optimize the product quality and marketability.

Since there is a potential for high savings at the beginning of component and module development, it is clear that engineers should use simulations in the early design phase. Highly automated simulation solutions that are integrated into design environments at an earlier stage eliminate time-consuming tasks. The difficulty for the engineer in using CAx tools is also reduced when the application is integrated into the relevant CAD environment. Close collaboration between design engineers and CAE engineers can improve early product optimization as well. Using the same CAx system saves precious time by avoiding data conversion and redundant data keeping.

Synergies between design and simulation software within one EDM environment clearly facilitate the use of simulations. In this way, simulation can be completely integrated into the development tool, and transparent connections to design geometry can be established. Thus, engineers can implement required changes at an early stage and reduce the duration of simulation cycles. This leads to several advantages:

- No geometry interface → no problems of geometry transmission and no effort for geometry recovery
- Associative and generative connection to geometry simulation → simulation is always up-to-date
- Access to parametric information from design (e.g. wall thickness, positions and center of gravity definitions)
- Result evaluation according to pre-defined rules via existing standard functions or CAD applications
- Direct manual or automated change of geometry and updating of results → quick evaluation of variants
- Use of analysis templates → easy to replace geometry and reuse FE-definitions
- One single application environment → identical user handling and menu structure
- Controllable CAD/CAE data management (e.g. subscribing, change management, analysis of potentials)

6.6.5 CAD/CAE Data Management

Automotive manufacturers make increasingly more use of CAE methods to calculate vehicle properties. These properties affect the body and chassis, as well as safety components, aspects of comfort and power train. The desire for early optimization and functional approval of a vehicle (i.e. before a hardware prototype is even built) is the main reason why CAE methods have become more popular in recent decades. The goal is to use a virtual prototype for calculations as early as possible, in order to find faults and to implement design-related changes in the CAD model [16].

In some configurations, especially in large companies, EDM systems control process sequences and administer product data, and the integration of product structure and parts lists is already highly sophisticated within the design departments. In contrast, CAE systems deliver their results in the context of virtual product validation via simulation and calculation with a delay. As a frequent consequence, the development state of the product does not match the current calculation results. To improve the system, the defined target must be to integrate CAE systems into the development process in a synchronous manner and to manage the CAE system data in an EDM system.

For this reason, subsequently performed simulations and the emerging results must be managed in the EDM system with reference to design data. To guarantee consistency, data storage must be harmonized between simulation tools and the EDM system. In this context, the relevant data to be exchanged are:

- 3D geometry as 3D CAD models or 3D CAE models
- Product structures (DMU, assembly sequences, FEM, NVH or crash simulation models)
- Mass properties (mass, center of gravity, moment of inertia)
- Material data (material, thickness of metal sheet, density)
- Joining systems (e.g. welding points, rivets)
- Technology data

Figure 6.10 shows an EDMS architecture schema with CAx and xDM integration. As basically described in the schema of EDMS architecture in Fig. 6.7, the individual functional modules again establish a common engineering database.

The process-oriented workflow control is situated in the upper area of project and process management. The integration platform in between is a bit more complex because different disciplines with different applications and therefore different data formats are integrated, and integrated configuration management is required for the definition of digital prototypes (DPT). Additionally, different xDM systems result in a non-interlinked database, which creates an additional challenge for operative data exchange. The target is to manage all data relevant for CAD/CAE within one EDM. In the near future, the variety of functions offered by different software applications, as well as the high amount of different software providers will preclude a holistic integration of EDMS. However, an initial successful step would be to at least use EDM functions consistently for workflow management, change management, central

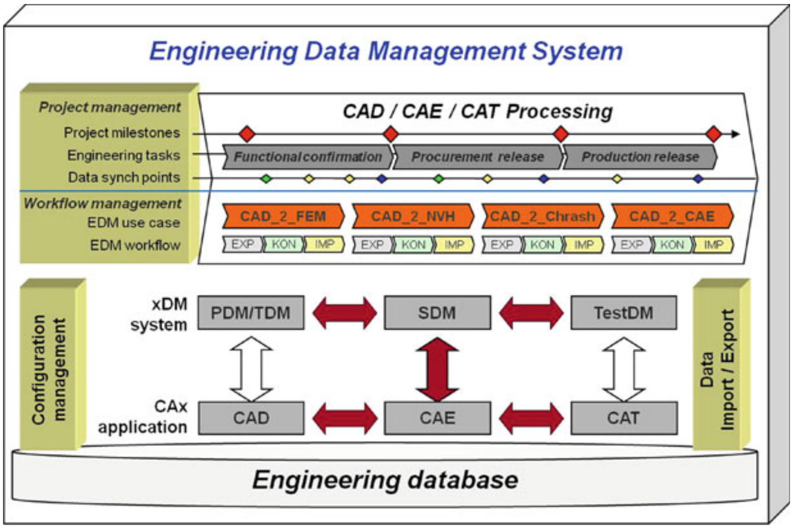


Fig. 6.10 EDMS architecture with CAx and xDM integration

data storage, data preparation and data import/export in all CAx disciplines. Thus, for the time being, the advantages provided by existing links in data management systems and software applications need to be integrated into the EDM approach as a whole. In terms of CAD application, this is accomplished by classical product data management systems and the increasingly important team data management (TDM) systems. In the case of CAE, the integration can be achieved by different types of simulation data management (SDM) systems.

References

1. Abramovici, M., Bickelmann, S.: Engineering Daten Management (EDM) Systeme: Anforderungen, Stand der Technik und Nutzenpotentiale, CIM-Management (1993)
2. Scheer, A.W., Boczanski, M., Muth, M., Schmitz, W.G., Segelbacher, U.: Prozessorientiertes Poduct Lifecycle Management. Springer, Berlin (2005)
3. Grabowski, H., Schnack, E.: Berechnung und Simulation in der Konstruktion. Lecture script at TU Karlruhe (2009)
4. Bullinger, H.: EDM: vom Datenmanagement zum Wissensmanagement. In: Chrysler, D. (ed.) Proceedings EDM Forum. Stuttgart, Germany (1999)
5. Eigner, M., Stelzer, R.: Product Lifecycle Management. Springer, Berlin (2009)
6. Spur, G., Krause, F.: Das virtuelle Produkt - Management der CAD-Technik. Hanser, München (1997)
7. Öterle, H.: Business Engineering - Prozeß- und Systementwicklung. Springer, Berlin (1995)
8. Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. HarperCollins, New York (1993)
9. Davenport, T.H.: Process Innovation: Reengineering Work Through Information Technology. Harvard Business Review Press, Bosten (1993)

10. Sendler, U., Wawer, V.: CAD und PDM- Prozessoptimierung durch Integration. Hanser, München (2011)
11. Eigner, M., Hiller, C., Schindewolf, S., Schmich, M.: Engineering Database - Strategische Komponenten in CIM-Konzepten. Hanser, München (1991)
12. Doblies, M.: Globales Produktdatenmanagement zur Verbesserung der Produktentwicklung. Phd Thesis, TU Berlin (1998)
13. Anderl, R.: Produktdatentechnologie B - Produktdatenmanagement. Lecture Script at Technische Universität Darmstadt, Germany (2003)
14. Hirz, M.: CAx in Automotive and Engine Technology. Lecture script at Graz University of Technology, Austria (2011)
15. Goltz, M., Müller, D.: PDM/PLM - Verwalten von Produktdaten ohne Grenzen. IMW Institutsmitteilung **27**, 57–67 (2002)
16. Meywerk, M.: CAE-Methoden in der Fahrzeugtechnik. Springer, Berlin (2007)
17. Anderl, R.: Produktdatenmanagement in der Simulation und Berechnung. Produktdaten J. (2002)
18. Berliner, K.: wissenschaftliches Forum für Produktentwicklung e.V.: Technology monitoring 1/01 - Virtual Reality. Technical Report, Paderborn, Germany (2001)

Chapter 7
Knowledge Management in Product Development

Generating and managing knowledge about a certain product is crucial for automotive product development and should therefore be assigned a high priority. This chapter describes product knowledge as a basis for investigation, as well as its development along the product life cycle. Thus, knowledge takes center stage as a kind of *customer* of engineering data management. The discussion investigates the fundamentals of knowledge, knowledge management and knowledge transfer, as well as the related most important basic models and approaches. Although no new definitions and concepts are presented here, this chapter is intended as a summary of current scientific findings in the area of knowledge management, which provides background for further analysis.

7.1 Product Knowledge

Although there are many slightly different definitions of *product knowledge,* essentially it involves:

- Knowledge and information contained in the product (geometries, functions, hardware)
- Information about the product—explicitly documented knowledge as information
- Knowledge about the product—knowledge in the heads of product developers

The following analysis addresses the development, meaning and life cycle of product knowledge, as well as definitions of the *product knowledge product* and product knowledge management.

7.1.1 Development of Product Knowledge

The contemporary approach to data processing is different because data are not only computed and stored, but rather augmented with additionally linked information that is provided in a user-friendly way and helps with efficient interpretation. Whereas product data is merely a string of characters without any indication of usage, product information also provides knowledge about sequences, facts or events regarding certain products. Since product information has a certain reference to the problem itself, it is goal oriented and has its own semantics, from which the related product knowledge is drawn. The consistent application of product knowledge within a discipline leads to mastery of the associated routine and finally becomes product knowledge competence. In North's [1] concept of *stairs of knowledge*, the ability to compete is the final step at the highest level.

Product knowledge emerges from various origins. Just as fundamental human knowledge is learned through upbringing and all manner of education, product knowledge also develops naturally in every human being who has an interest in a particular product. In terms of organizational product knowledge, this means that methods of product data management and knowledge management must be used in order to build up a collective knowledge base, which establishes and preserves the acquired product knowledge. In this context, product knowledge represents the entire body of information surrounding a product. This knowledge is created through product development and product advancements and is further extended to a range of variants and versions. Ultimately, knowledge aggregates along the entire product life cycle. The combination of basic knowledge, professional expertise and knowledge from experience, combined with knowledge about processes and the life cycle, leads to the product knowledge, which is also influenced by socioeconomic factors.

Engineers are specialists. The quality and level of innovation of a company's products depend on the special knowledge and expertise of its engineers, as well as their inventiveness and creativity. Ultimately, customer satisfaction and economic success depend also closely upon these aspects. While engineering knowledge is first and foremost in the heads of engineers, it is also increasingly being stored in the computer data generated by engineers. Such data (especially the increase of geometrical design elements, assemblies, modules and products) can be key factors for integrating different engineering disciplines into the overall processes of product development. This data is also an important factor for the continuous guidance of core processes within a company.

A company has to gain possession of the special knowledge of engineers, which has often been seen as private property. Instead of maintaining *personal ownership* of expert knowledge, engineers offer it up for general use, which transforms it into a company resource. Experience has shown that the engineers are only willing to do this if they can count on improvements in work conditions and when they have or possess the skills and resources required to make information more generally available. For many engineers, this implies the ability to reuse design models or variant investigations, as well as the ability to trace all changes of product development and product documentation. Since EDM provides these functions, it makes product development more efficient. One prerequisite is that data management must be understood as not only a depository, but also a tool for steering processes. Admittedly, EDM also involves changes in both the individual work of engineers and the form and content of collaboration between the disciplines involved. Individual engineers will have to abide by certain rules and will also have to perform additional data management that, at first glance, will not be inherently beneficial for their particular range of activities. Nevertheless, engineers must fully embrace all of the aforementioned principles in order to achieve a successful process optimization.

7.1.2 Life Cycle of Product Knowledge

Product knowledge as a class of information passes through nearly the same characteristic life cycle as the product itself. This means that it evolves and eventually becomes obsolete; it is acquired and then forgotten. While this could be seen as a harmless and natural process, if the product knowledge is an essential company asset, it can also be fatal. Based on the introduction of the knowledge transfer in product development in Sect. 7.3, this section examines the product life cycle chain for product knowledge, which consists of:

- *Product motivation*—provides the impetus to examine the product more closely
- *Product investigation*—the collection of all suitable, findable, explicit data about a product, as well as any related information and knowledge
- *Interpretation of information*—the structuring and evaluation of collected facts
- *Awareness of knowledge*—the storage of new product knowledge
- *Publication of knowledge*—the dissemination of new product knowledge
- *Application of knowledge*—the practical use of product knowledge
- *Selection of knowledge*—the preliminary stage of product knowledge conservation
- *Loss of knowledge*—the preliminary stage of product knowledge oblivion

The long-term preservation of product knowledge, which is similar to product preservation and long-term knowledge preservation, can be seen as the establishing of the active product knowledge pool.

7.1.3 Defining Product Knowledge

Nowadays, product knowledge (especially specific product knowledge) is one of the most important factors of production and economic success for organizations around the world. Product knowledge provides the basis for powerful innovations because it collects the know-what, know-how and know-why for all particular products across their development states. Certain aspects, such as a product's design (including colors and materials), always capture the spirit of the time and society's current taste. Product development lines also reflect the technical capabilities of society during a certain period in history and provide information about preferred technical solutions and choices in engineering designs.

Product knowledge can be enhanced by providing early expertise. This can be done by using simulations and computations to generate and evaluate knowledge, as well as to provide other relevant information. In addition, product knowledge can be expanded via feedback from product life cycles, evaluation methods, optimization methods and organizational decision support. Today, it is possible to conserve knowledge in knowledge documents and knowledge products. However, simply storing such special knowledge in PDM systems, document management systems or certain electronic memory media is not sufficient. Instead, there is a growing demand to be more knowledgeable about the stored data's content, rather than simply knowing where and how it is stored. Furthermore, it is necessary to plan for the possible breakdown of the hardware system or a software-related data loss. For this reason, continuous maintenance of the digital inventory is essential, and data is stored in the conservative, traditional manner (i.e. put down on paper) to cover the risk of data loss. Weaknesses and aberrations in particular products can be easily tracked and shown over the different stages of development, which can be beneficial for further product development.

7.1.4 Product Knowledge *Products*

Product knowledge *products* are the concrete manifestation of fact knowledge (*know-what*), process knowledge (*know-how*) and explanation knowledge (*know-why*). More precisely, they represent explicit, impersonal knowledge about products in their versions and variants. Such representations also include knowledge about different kinds of products and product classes. According to Probst [2] these products are also part of *product knowledge conservation* if they can be reused and are valid for a longer period.

7.1.5 Product Knowledge Management

Within the product development process, product knowledge includes technical and economic information about the necessary methods, facts and rules about the expected product life cycle, and the product assembly. This knowledge can be universal, sector-specific, organization-specific or product-specific. This knowledge data should be collected and preserved, and all created documents and applications of product development should be stored centrally and archived conveniently, at the very least. Although current PDM and document management systems can store and archive data effectively, the state-of-the-art technology is often unable to implement content linkage or to store knowledge data such that it can be easily located and published. Thus, EDM systems must be altered in the coming years to make them more similar to so-called product-knowledge-management systems. This can be accomplished by integrating knowledge management activities into these original EDM systems. This alone is one approach for archiving data on product knowledge over a long term.

7.2 Fundamentals of Knowledge Management

7.2.1 Knowledge and Knowledge Management

Knowledge management is a management tool which can be used to design and control processes and can also be applied to support the general development of an organization. The task of knowledge management is to create boundary conditions for a better cross-linkage of knowledge senders and receivers, with the ultimate goal of improving the generation and application of knowledge. Wohinz has suggested that knowledge management can be interpreted as the management of knowledge systems [3]. Thus, successful knowledge management means a purposeful investment in people, organization and technology.

Defining Knowledge

In order to focus specifically on knowledge, it is important to delineate the intended scope of the term. Researchers from disciplines such as philosophy, psychology and sociology have used the term knowledge, and so many people have written about the topic that there is no clear, generally accepted description of knowledge. This paper draws on the ideas of Probst, who defines knowledge as "*The whole set of experience and all skills that individuals use to solve problems.*" [2]

Differentiating Data, Information and Knowledge

Many users have sought to define knowledge, and their definitions are normally influenced by their own particular challenges and preconceptions about the topic. The following list of terms highlights some of the main criteria for differentiating between different terms (e.g. characters, data, information and knowledge), as well as the correlation between different hierarchy levels among these terms and conditions [4].

Data can consist of symbols or sequences of characters (e.g. letters, digits, special characters) which are combined meaningfully and are in line with a code or syntax. However, raw data is not interpreted, and thus it is possible to draw conclusions based solely on data [5].

Information, on the other hand, is created when data is evaluated and placed into context. By adding an economic perspective or interpretation, raw data is converted into meaningful information that can be used to guide operations and decisions. Knowledge, in turn, develops when information is further connected in a goal-oriented way and classified within the context of experience. The dependence on context might give rise to the problem that the same information can lead to different forms of knowledge about the same circumstances if it is placed in different experience-based context.

North's Stairs of Knowledge

North's concept of *stairs of knowledge* helps to illustrate the important definitions related to the terms data, information and knowledge [1]. Knowledge only becomes valuable to a company if such knowledge (knowledge—what) is turned into mastery (knowledge—how). For this reason, knowledge has to connect to action. However, since mastery only leads to concrete actions if a certain drive exists, a relevant motivation or desire is indispensable. Actions shows results that can be used to measure how a person, a group of people or an organization (e.g. enterprise, university institute) uses knowledge to solve a problem in terms of a certain defined target. This ability, which is also called competence, becomes well defined at the moment when knowledge is applied.

Core competences, which are seen as extraordinarily important for organizations, are a collection of skills and technologies which include specific knowledge and cannot be easily copied by competitors. They allow the organization to satisfy the customer and can be transferred to new products and product areas [6]. Because they are unique, core competences ultimately help determine the competitiveness of an enterprise, as knowledge in the form of competences becomes a strategic resource that provides the foundation for developing competitive advantages.

7.2.2 Basic Elements of the Knowledge Base

Structuring of Knowledge

The relevant literature covers the ways knowledge can be structured and how different forms of knowledge can be distinguished. In the present book, knowledge is structured into knowledge psychology, expression and knowledge carrier [7], see Fig. 7.1.

Structuring in Terms of the Knowledge Carrier

This form of structuring shows that knowledge is not only an individual possession but is also important for an organization:

- *Individual knowledge* is the knowledge of each person. It is context-free, and its availability depends on the knowledge carrier.
- *Collective knowledge* is only important within a particular environment or organization (team, enterprise, etc.). It is knowledge from different people that can be combined to result in target-oriented actions and is owned by all members of a collective.

Structuring Based on the Ability to Express

This classification shows whether or not the knowledge carrier is aware of the knowledge and therefore whether or not it can be expressed:

- *Explicit knowledge* is knowledge that is conscious and can be expressed. The knowledge carrier is aware of it and can talk about it.
- *Implicit knowledge* is knowledge of which the carrier is not consciously aware. It either cannot be captured or can only be captured with a significant effort (e.g. by using special questioning or analysis techniques).

Structuring Based on Knowledge Psychology

Knowledge psychology distinguishes between declarative knowledge and procedural knowledge:

- *Declarative knowledge* relates to all facts (e.g. operations, issues) and is constituted as *knowing what*.
- *Procedural knowledge* is an awareness of the way cognitive processes are carried out and is also referred to as process knowledge or *Knowing how*.

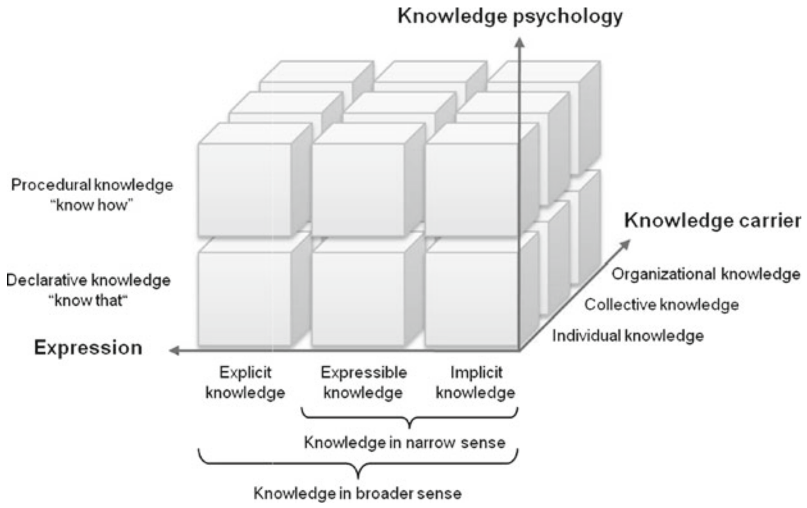


Fig. 7.1 Structure of knowledge

Basic Forms of Knowledge

There are three basic forms of knowledge:

- Fact knowledge (*know-what*) describes knowledge about specific facts and events.
- Process knowledge (*know-how*) encompasses the knowledge required to execute development processes.
- Explanation knowledge (*know-why*) includes knowledge about cause-effect relationships and the coherence of the causal chain.

The form of knowledge also states whether it is explicit or tacit. In addition, the knowledge dimension includes encoded knowledge, which can include knowledge in figures and textual descriptions, data and formulas in books, documents, computer files, databases, software or videos. The material forms of this knowledge are called knowledge products, which are explicit and independent from individuals. If a knowledge product can be stored and is valid for a longer period, it is also known as a *knowledge preserve*. Knowledge integrated in databases can only be fully utilized by targeted human actions [2].

Implicit and Explicit Knowledge

Knowledge can exist in different *aggregate states*. The differentiation between explicit and implicit knowledge can be traced back to Polyani [8], who argued that all people know more than they can articulate. He called the part of knowledge that cannot be expressed *tacit*, which is also known as implicit knowledge. The ability to communicate and the means of doing so can be used to differentiate between these two forms of knowledge.

Implicit Knowledge (Tacit Knowledge)

Implicit knowledge is strongly based on experience and personal values, and the individual is often unaware of it. It is also difficult to observe, articulate and formalize. It is the part of knowledge which an individual either cannot express or can only express with a great effort. It is the kind of knowledge that is nearly impossible to be expressed completely comprehensibly. Tacit knowledge, therefore, is strongly tied to the individual person.

Explicit Knowledge

In contrast, explicit knowledge is schematic. Since it can be encoded with languages, figures, characters, symbols, drawings, etc., it is also easier to communicate. Implicit and explicit knowledge can be transformed into each other. Implicit knowledge must be codified in order to translate it into explicit knowledge. Once knowledge has been codified, it can be accessed and understood by other people, and it can also be organized, used and shared with others.

Individual, Collective and Organizational Knowledge

Individuals are the central carriers of organizational knowledge because they have the ability to transfer data into knowledge and are able to use that knowledge for the company's benefit. However, it is not sufficient to view the knowledge base only from the individual's perspective. In fact, many processes contain elements of collective knowledge and provide the basis for effective organizational action [2]. Organizational knowledge is another kind of knowledge that contains both individual knowledge and the collective knowledge shared by groups. This organizational knowledge is more than the sum of knowledge of individuals, and it provides an orientation and a frame for the actions of individuals in an organization. The combination of individual and collective knowledge forms the organizational knowledge base.

Organizational knowledge is created by sharing knowledge between individuals within the organization or possibly also including individuals outside of the organization in order to take actions that contribute to the fulfillment of a stated goal. Thus, the organizational knowledge base represents an extension of the collective knowledge and offers a certain potential. It includes individual knowledge which is not accessible to an organization, as well as combinations of individual knowledge which did not previously exist. A company's raw, stored data does not count as part of the organizational knowledge base.

7.2.3 Knowledge Management in Industrial Management

In the present book, knowledge management does not mean managing raw knowledge, but rather the management of knowledge systems. Knowledge management involves creating a guiding concept for the target-oriented design, steering and development of an organization. Knowledge management must establish boundaries in order to create and link individual bodies of knowledge. Although the approaches to *knowledge management* are as different as the definitions of the term knowledge, generally speaking, two ways can be used:

- *People-oriented approach:* The focus is on the employee as *knowledge carrier*.
- *Technology-oriented approach:* The focus is on the supporting information and communication technologies (i.e. a greater focus on data and information management).

The management of knowledge in industrial management can be seen as a basic prerequisite for target-oriented action, which helps create and maintain a competitive edge. However, knowledge management goes beyond corporate borders. The newest trends are trying to create an efficient interface to customers, which can transfer and exploit all customer-related knowledge (*customer knowledge management*). Basically, this represents a combination of customer relationship management with knowledge management.

7.2.4 Basic Model of Knowledge Management

For experts in applied sciences, it is often difficult to grasp the importance of the field of knowledge management. Therefore, it is important to elaborate the concepts of knowledge management in an easily comprehensible manner, and a knowledge-oriented organization design should be used to present the goals in a transparent manner.

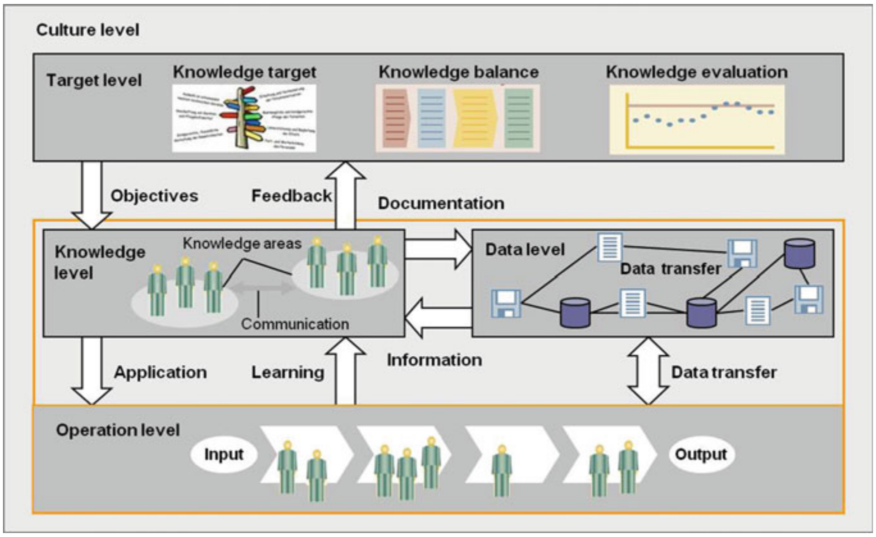


Fig. 7.2 The basic model of knowledge management

Figure 7.2 shows the basic model of knowledge management as a possible approach for the knowledge-oriented investigation of an organization. It is divided into five levels—the knowledge, data, operational, target and cultural levels. These levels identify the aspects that are important when dealing with organizational knowledge and define the organizational boundaries that can be established to optimize the development, transfer and use of knowledge [9].

Levels of the Basic Model

The traditional differentiation of knowledge and data/information leads to the distinction between a knowledge level and a data level. The knowledge level includes the personal element of the organizational knowledge base. On the knowledge level, employees play the central role, and establishing links and communication between employees is also important. In contrast, all of the organization's data and information can be found on the data level, which includes the explicit, collective knowledge, as well as the related technological infrastructure for storing, organizing and distributing knowledge.

However, the real value is added in different business processes on the operational level of an organization, where the knowledge is implemented in certain actions. The gathering and interpretation of operative results and their resulting adaptation for further actions can bring an adjustment of individual and organizational memory, which is also known as learning. The target level is above the three levels of acting, knowledge and data. It defines the targets and requirements of knowledge management. Here, specifications from organizational strategies are broken down to the level of individual business processes. The strategic knowledge management and the controlling of knowledge are also found on this level. The knowledge targets relate to the design of important knowledge activities, but also to the creation of necessary boundary conditions.

All of the knowledge activities described in the first four levels exist within the context of the organization's culture level. The culture of an organization has a important influence on how knowledge is handled. In this context, it is crucial that a tolerant, open culture of learning be established so that employees can communicate openly with each other and be willing to share their knowledge. Generally speaking, tools, methods or software solutions do not really help if they are not accepted by the people involved, and the organizational culture has a strong influence on this acceptance.

Cross-Linking of Levels

Direct interactions between the target level, the knowledge level, the data level and the operational level are particularly important. The culture level of the knowledge model lies in the background and determines the establishment of cultural boundary conditions for activities on these other levels. Demands from the target level are directed to the operational level and the knowledge level, which requires specific knowledge targets. Targets on the operational level are generally aimed at a business process and normally include knowledge-related goals. With regard to the execution and effectiveness of the measures undertaken, controlling should provide regular feedback to the target level, which makes it possible to adjust targets or to derive further actions to achieve the relevant objectives.

The knowledge and operational levels are linked due to the application of knowledge on the operational level and the action-based learning process. Once again, action-based learning creates new knowledge at the knowledge level itself. Finally, the knowledge and data levels are directly linked through the documentation and information processes. Analysis transforms data into knowledge, which is then transformed into knowledge and made explicit. The operational and data levels are in turn linked via data transfer, which provides important data for the execution of processes to the data level, where the data itself is stored and processed.

7.2.5 System Orientation in Knowledge Management

Knowledge management can now be explored in terms of defining a knowledge system. A system consists of individual elements which are correlated to each other and combined to form a unit. The connections can be material flow, information flow, relative positioning, functional dependence, etc. Each individual element can also be seen as a system itself. The arrangement of elements and their connections are called the structure of a system [10].

Although the purpose of the system does not need to be pre-defined, it will influence the interactions within the system. The knowledge system can therefore be seen as a socio-technical system in which people and technical facilities interact with each other. The purpose of such a system can be described with the modules of knowledge management.

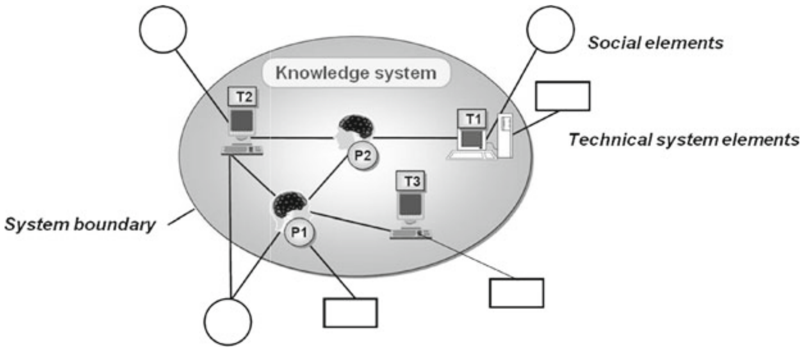


Fig. 7.3 Individuals and technical facilities as elements of a knowledge system

Transfer relationships in a knowledge system are closely related to system elements. As previously described, a socio-technical system consists of social elements (e.g. humans) and technology system elements (e.g. computers). Although the social elements can be assigned to both the operations and knowledge levels, only the knowledge level is relevant for this discussion of transfer relationships. The elements of the technical system are assigned to the data level and represent the data of a company. Figure 7.3 illustrates the knowledge system, which contains people (P1, P2) and technical facilities (T1, T2, T3) that are connected to each other. The figure also shows that connections can exist with elements beyond the borders of knowledge systems.

The knowledge system is therefore a specific expression of an industrial operation system and has a correlation with the system of value creation. As knowledge carriers, people form a social subsystem and are seen as elements of the value chain on the operational level [3]. Organizational and technical facilities (tools) are the elements of the technical support subsystem. It is common to distinguish between three levels of knowledge management (cf. Rehäuser/Krcmar) [4]:

- The knowledge use level—the basis for knowledge requirements and provided knowledge
- The knowledge holders level—people who are responsible for meeting knowledge needs
- The knowledge infrastructure and information processing level

Knowledge management can therefore be interpreted as the management of knowledge systems.

7.3 Knowledge Transfer in Product Development

Knowledge transfer is a key factor for the efficient design of product development in the automotive industry. The complexity of the automobile product itself leads to a very complex product knowledge that needs to be maintained and exchanged via transfers between all of the parties involved. A variety of specific and complex development, optimization and simulation processes depend strongly upon each other. Additionally, they are accompanied by a pronounced change management. One significant challenge is the manufacturer/supplier relationship across multiple organizational levels and units. Here, the related knowledge transfer must be managed with an eye towards protecting know-how, which is seen as an important competitive advantage. In addition, the communication between employees of a project team and the support of systematic and structured data management play significant roles in the efficient transfer of knowledge. A detailed analysis of the knowledge processes is necessary in order to identify and properly interpret the factors that influence knowledge transfer. The following sections describe the basic concepts of knowledge transfer.

7.3.1 Definition of Knowledge Transfer

In the context of knowledge management, the contact between two individuals is important, and the question of whether or not knowledge is *transported* is particularly relevant.

Hartlieb [11] defines knowledge transfer as follows:

“*The knowledge demand is satisfied by the transfer of existing knowledge. A knowledge transfer has taken place if the knowledge receiver has a similar understanding of the knowledge content as the knowledge sender.*”

The knowledge transfer is an important object in the investigations of knowledge-oriented engineering data management.

Knowledge Transfer Versus Knowledge Induction

Sammer [12] describes the term *knowledge induction* as follows:

“*Knowledge of an individual becomes data through documentation—this data and only this—is transferred and induces possibly similar knowledge in the knowledge receiver.*”

In this book, the term *knowledge transfer* is used to express a concept that goes beyond this definition of *knowledge induction*. First, a distinction is made between direct and indirect transfer of knowledge. Within indirect knowledge transfer, a further differentiation can be made between knowledge transfer through telecommunication and knowledge transfer through documentation and information, whereby only the transfer of knowledge through the documentation and information processes fits within the definition of the term knowledge induction.

7.3.2 *Transfer and Transformation Processes in the Knowledge System*

From the comparison of data and knowledge, the following transfer and transformation processes are derived, which are essential for further analysis and the design of knowledge transfer processes.

Figure 7.4 shows a matrix of possible data and knowledge transfer relationships based on the work of Nonaka/Takeuchi [13].

Knowledge Transfer Process

This relationship deals with the direct transfer of knowledge between two individuals (knowledge transfer without data management).

Documentation Process

Sammer [12] defines documentation as the transformation of knowledge to data. Documentation can be understood as the process of knowledge externalization by which data is created in the form of numbers, languages, texts and pictures, which is designed to transfer knowledge as accurately as possible to other individuals.

Information Process

The information process involves the transformation of data to knowledge. Arbitrary data triggers the individual processing of information, by which the individual's cognitive system provides context and meaning to the raw data.

Data Transfer Process

This involves a direct transfer of data (i.e. data to data), which can be performed via different media (e.g. printed documents, electronic media) and is mostly supported by modern information and communication technologies.

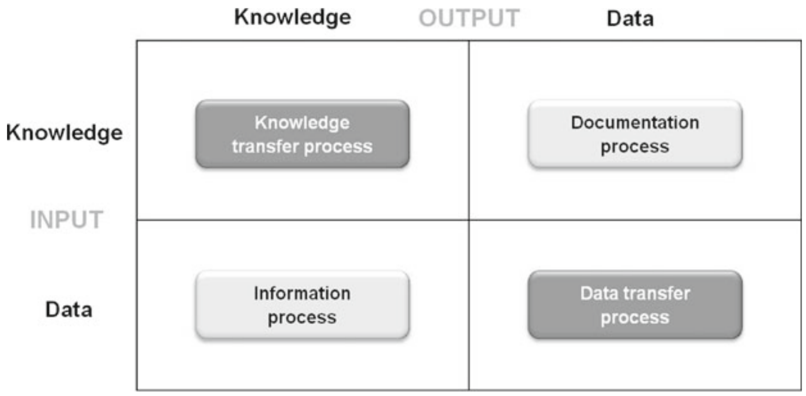


Fig. 7.4 Transfer matrix for data and knowledge, adapted from Nonaka/Takeuchi

7.3.3 Direct Versus Indirect Knowledge Transfer

It is important to note that shared knowledge (or collective knowledge) can be created through either direct knowledge transfer (i.e. the upper left box in Fig. 7.4) or through indirect knowledge, which involves the information and documentation processes mentioned above (and normally the data transfer process as well). In the latter, it makes no difference if the information is made available in hard-copy form (i.e. printed documents) or in electronic form (e.g. PC memory).

7.3.4 Direct Knowledge Transfer

Direct knowledge transfer describes the transfer of knowledge via communication between two individuals in a social system. It can be understood as the linking of two human entities, each of which can be further divided into motor and sensory sub-systems. For instance, face-to-face communication is a typical prerequisite for the occurrence of a direct transfer of knowledge. The transferred knowledge is encoded by the sending person into signals, which are then picked up by the receiver. The receiver can only generate knowledge from signals by decoding them. Decoding, which refers to the personal interpretation of the signals, is also influenced by personal context knowledge. In order to bring the recipient up to the sender's level of knowledge, both parties must have a common contextual knowledge (common language, common understanding of the terms, etc.). Of course, this is also related to the field of knowledge, and in the end, no transfer can be achieved if the two parties do not share a common contextual knowledge.

7.3.5 Indirect Knowledge Transfer

As already mentioned, the indirect transfer of knowledge between two individuals is performed via the technical subsystem, which means that a link is established between the social and technical system elements, which helps support the knowledge transfer in a socio-technical system. Thus, the knowledge system is divided into a technical subsystem and a social subsystem. In the technical subsystem, a person performs data-technical actions (e.g. speech, exerting mechanical force to operate a keyboard), and data (e.g. in the form of numbers, language, texts or images) are created. This data then becomes externalized knowledge, which can be stored and/or transferred in the technical subsystem. Subsequently, these data can be coded into signals, which are spatially and temporally decoupled. Those signals can be gathered by a person, further classified with personal context, and can thereby create knowledge.

Through telecommunications, knowledge can also be transferred via the technical subsystem, which is totally decoupled from any location. For knowledge to be successfully transferred, recipients must be able to generate knowledge from the information contained in recorded signals by classifying it within the context of their individual knowledge. For knowledge transfer through the documentation and information processes, the effectiveness and efficiency will depend on both how well the receiver can adapt to the context knowledge of the sender, and how well the sender takes this aspect into account when creating the documentation. It has to be noted that, due to errors in the signal conversion and the limited range of signals, the quality of the knowledge transfer may be less than that which occurs via a face-to-face conversation.

Furthermore, this chapter does not go into details of the application of telecommunications, but focuses on the importance of knowledge transfer via documentation and information processes, because they are more important for the knowledge-oriented EDM. The indirect transfer of knowledge can be divided into three sub-processes:

- Documentation process
- Information process
- Data transfer process

The signal interface between the technical and social subsystems in information management is referred to as the user interface. A transfer of knowledge via documentation and information transmission requires data that is available on one of the many forms of storage media [14].

The Documentation Process

The knowledge of an individual always provides the background for documentation. Individuals try to create data and documents by encoding knowledge in a way that will make it accessible to others. This can only succeed if the information is encoded into signals. Content has to be coded in order to transfer it to a technical subsystem and to store it. The technical subsystem can be used both to store data and to create signals. In all cases, there must be a human who acts on the technical subsystem in order to create a signal code or to store data. For example, this could involve the use of a keyboard or speech recognition programs. The main human tasks for the creation and storage of data are:

- The human must provide signals for the technical system in a form that is workable.
- The human must interact with the technical subsystem.

The context knowledge of receivers should be taken into account when documenting or coding the knowledge. This means that the author has to think about the minimum knowledge context that humans must possess in order to successfully complete the knowledge transfer process by creating knowledge themselves. Thus, the effectiveness and efficiency of the knowledge transfer depends on how well the knowledge sender knows the context knowledge of the receivers and how well the sender can adapt the documentation to this context.

The Information Process

Data formed through coding, signal transfer and storage can generate knowledge at a later time through the information process. The information process is the process by which a receiver generates knowledge by acquiring and decoding a signal. This process is affected by the individual context knowledge of the recipient and also on different physical signals (e.g. sound waves from an audio tape, light waves when reading) which can be absorbed by human sensors. The receiver must possess the necessary context knowledge for a satisfactory knowledge transfer to occur, and the quality of knowledge transfer increases if more of the receiver's human sensors are stimulated.

The Process of Documentation and Information on the Data-Level

A temporal decoupling can be achieved when data is permanently stored on a storage medium. To be more precise, it is the temporal decoupling of the documentation process (sender) and the information process (receiver). Since data is created based on human knowledge, the documentation process and the subsequent information process can also be seen as a kind of knowledge transfer. Figure 8.4 in Sect. 8.1.5 shows the signal flow of the entire procedure. Knowledge carrier A encodes knowledge into data and stores it in the technical subsystem. Knowledge carrier B can then access those signals time independently and can further create knowledge by the information process, during which knowledge carrier B processes and interprets the knowledge within his/her own knowledge context.

7.3.6 The Definition of Knowledge Logistics

The definition of knowledge logistics is based on the work of Hartlieb [11], who uses an analogy to material logistics:

"*Operative knowledge logistics uses appropriate interventions to ensure that all available knowledge required for the particular operations is provided. It must be temporally and locally available, accessible and provided in an appropriate form.*" (author's translation)

Engineering data management can be linked with the logistic process because the functions that comprise logistics can also be applied to EDM. Thus, knowledge logistics can be seen as the reference framework necessary for the development of knowledge-oriented approaches in engineering data management. When examined in more detail, knowledge logistics can be understood as the management of knowledge demands, knowledge supply and knowledge transfer, thereby connecting these three domains. Entities responsible for certain tasks, knowledge carriers or management can create and express knowledge demand. The knowledge supply consists of the actual organizational knowledge base. Knowledge demand can be fulfilled by the transfer/transformation processes and should be executed by the engineering data management. Therefore, it is necessary to define the tasks of EDM in relation to the definition of knowledge management.

It is important to differentiate between two distinct yet interrelated forms of knowledge management: the management of existing knowledge and the management of knowledge enhancements. Knowledge-oriented EDM can be seen as the part of knowledge management that focuses on the management of existing knowledge in an organization (i.e. the emphasis is on managing the existing knowledge base via data management). Although knowledge development (i.e. managing the enhancement of the knowledge base) is not the primary object of investigation here, it should be mentioned that the implementation and application of EDM also has an effect on the prerequisites and development of the knowledge base.

7.4 Process Orientation in Knowledge Management

The motivation for process orientation derives from the challenge of implementing knowledge management directly into the added-value chain. Negative effects often arise in an organization due to the separation of various management activities, operational activities and associated specializations. For example, individual departments may not be operating at their best, or they may have implemented unnecessary or incomprehensible interfaces. Since such adverse effects occur particularly in functional and organizational structures, companies are trying more and more to overcome these problems by integrating perspectives [9].

One related approach is process orientation, which focuses on the relevant value-adding operations of an organization, and which considers various processes and tries to optimize them across the departments. A process owner is given the responsibility for an entire process, thereby making it possible to regard the process as a unit and to optimize it cross-functionally. Thus, interface problems can be minimized, and a consistent customer focus can be implemented along the entire value chain. A process is a set of activities that are geared towards a specific goal and are established to convert input (e.g. production factors, information science) into a defined result. Each process is initiated by a start event and ends with the attainment of one or more final states. Here, functions and their sequence of targets are determined on the one hand by the demands of internal and external customers and stakeholders, and on the other hand by overall organizational goals.

The term *business process* is applied to any process that contributes either directly or indirectly to add value for an organization. The traditional starting point of process orientation is derived from the value chain model, which is based on an organizational analysis of value-adding activities. Process management is based on process orientation and implements this orientation in all divisions of an organization. Process management encompasses the analysis, planning, controlling, steering and optimization of processes.

Processes take place on the operational level, which includes the business process model of an organization and its systematic management. For a process-oriented approach, it is useful to distinguish between three categories of knowledge:

- *Knowledge for the process* is required for the optimal performance of business processes.
- *Knowledge from the process* is knowledge that is created during value-adding activities, which can include knowledge and experience. The goal is to develop this collectively in an organization.
- Beyond the first two categories, which are generally assigned to knowledge management, there is also knowledge about the process itself. This process knowledge is associated with process management.

The connection between process management and knowledge management can primarily be regarded from two points of view, the knowledge-oriented process management and the process-oriented knowledge management.

7.4.1 Knowledge-Oriented Process Management

In the first case, knowledge management can be seen as providing support for process management. This assumes that knowledge is generated in processes and then also reapplied. Furthermore, the goal is to optimize the integration of knowledge management and process management, which should ultimately lead to a *knowledge-based* process management. In such a system, a process is managed with a particular emphasis on the resource knowledge.

Regarding knowledge-based process management, the following points are important:

- It deals with the optimal and efficient design of processes, with a focus on knowledge management methods.
- The goal is to optimize processes in a way that they support the best possible learning and that new knowledge can be created as a result of the findings and operations within processes.
- There is a focus on effective knowledge-relevant process design.
- Knowledge-oriented process management is operated by the *process owner*.
- The findings of knowledge-based process design should be increasingly reflected in the standardized and documented processes of the company.

7.4.2 Process-Oriented Knowledge Management

If knowledge management is viewed as a process itself, the process orientation is introduced to knowledge management. According to the typology approach of Oesterle [15], process-oriented knowledge management can be seen as a supporting process. Probst [2] suggests that an initial approach is provided by the components of knowledge management. These components can be viewed as sub-processes and are addressed in knowledge management. Regarding process-oriented knowledge management, the following points are significant:

- The intention is to apply methods of knowledge management to fixed and largely unchangeable processes to create the expected value.
- It acts directly on the operation itself and on the people acting within the process.
- The aim is to improve the efficiency of the process (process efficiency).
- Process-oriented knowledge management is controlled by a *process manager*.
- Specific, knowledge-related adaptations and characteristics of the process sequence can be defined in work instructions.

It is advantageous to integrate the perspective of knowledge in a process-oriented structure of the organization and to place the business processes at the center of knowledge management. In this way, knowledge can be more easily generated because knowledge about the process and knowledge from the process are directly related to applications. Therefore, knowledge can be used to support business processes in an action-oriented way. In addition, the process orientation provides approaches for an easier measurement of the costs and benefits of knowledge activities and knowledge-intensive business processes.

7.4.3 *The Knowledge Process in Interaction with the Added-Value Processes*

The previous discussion of process-oriented knowledge management emphasized the knowledge-oriented modeling of business processes. The general process orientation in knowledge management is related to the business process level. In the operative level, the process orientation can also be related to the added-value process. The following section takes a closer look at the process orientation of knowledge processes, with a particular focus on knowledge operations that conceptualize the knowledge process as a knowledge transfer between two added-valued processes. Thus, the fundamental considerations may also include the concept of added-value processes, as defined by Remus [16]. In order to minimize the effort of mapping knowledge processes, it is useful and appropriate to select processes that interact extensively for knowledge analysis.

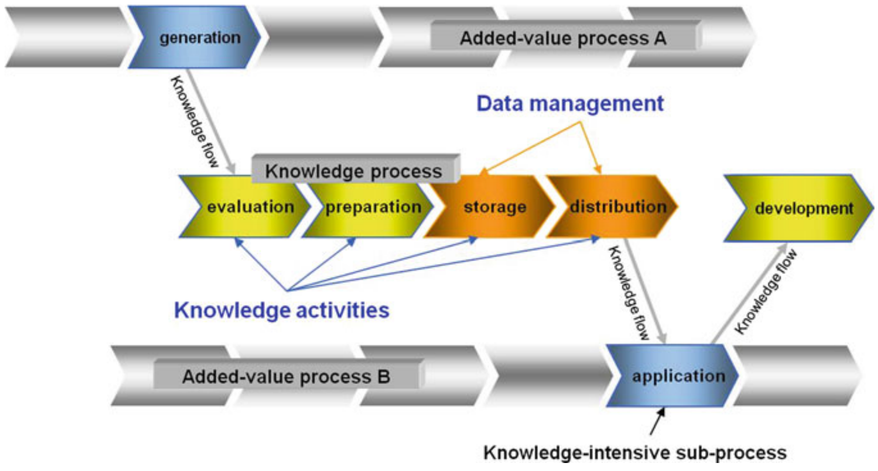


Fig. 7.5 The knowledge process in interaction with the added-value processes

Figure 7.5 presents a knowledge process between added-value processes with typical knowledge activities. As suggested above, support from data management is required for a continuous knowledge process behind specific knowledge operations. The question now arises of how knowledge processes between added-value processes can be implemented, supported and designed, and also if particular knowledge operations have to be linked with additional processes or actions.

References

1. North, K.: Wissensorientierte Unternehmensführung. Wiesbaden, Germany (2002)
2. Probst, G., Raub, S., Romhardt, K.: Wissen managen - Wie Unternehmen ihre wertvollste Ressource optimal nutzen. Wiesbaden (2006)
3. Wohinz, J.W.: Industrielles Management - Das Grazer Modell. Wien, Graz (2003)
4. Rehäuser, J., Krcmar, H.: Wissensmanagement, Chap. Wissensmanagement im Unternehmen. Berlin, New York (1996)
5. Wohinz, J.W., Oberschmid, H.: Wissensmanagement. Induscript, TU Graz (2007)
6. Hammel, G., Prahalad, C.: Competing for the Future. Harvard Business School Press, Boston (1994)
7. Scheuble, S.: Wissen und Wissenssurrogate - Eine Theorie der Unternehmung. Deutscher Universitäts, Wiesbaden (1998)
8. Polyani, M.: The Tacit Dimension. Routledge and Kegal Paul, London (1966)
9. Wissensmanagement Forum: Das praxishandbuch wissensmanagement - integratives wissensmanagement (2007)
10. Haberfellner, R., et. al.: Systems Engineering - Methodik und Praxis. 11th edn. Verlag Industrielle Organisation, Zürich (2002)
11. Hartlieb, E.: Zur Rolle der Wissenslogistik im betrieblichen Wissensmanagement. Phd Thesis, Graz University of Technology, Austria (2010)
12. Sammer, M.: Wissensinduktion in Organisationen. Phd Thesis, Montanuniversität Leoben, Austria (1999)
13. Nonaka, I., Takeuchi, H.: Die Organisation des Wissens: Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen. Frankfurt/Main, New York (1997)
14. Willfort, R.: Innovationsdienstleistungen im wissensorientierten Management von Innovationsprozessen. Phd Thesis, Graz University of Technology, Austria (2000)
15. Öterle, H.: Business Engineering - Prozeß- und Systementwicklung. Springer, Berlin (1995)
16. Remus, U.: Prozessorientiertes Wissensmanagement - Konzepte und Modellierung. Springer, Berlin (2002)

Chapter 8
Knowledge-Based Engineering Data Management

Knowledge-oriented engineering data management (knowledge-oriented EDM) seeks to use process-oriented knowledge management to identify the knowledge-intensive activities of the value-added process. These activities are then interlinked by using knowledge-based process management to develop knowledge processes. The relationship between two processes is what provides knowledge management with a process orientation. In engineering data management, the primary goal of knowledge orientation is to answer the following questions:

- How is it possible to capture and classify the underlying database of product knowledge and then make it accessible?
- What knowledge demand can be fulfilled by systematic EDM?
- How can the EDM support the knowledge process through data management and workflows?
- How can different knowledge demands be transferred between knowledge holders and task carriers via EDM?

By answering these questions, the knowledge factor can improve the quality and efficiency of engineering data management. The goals are to integrate the findings obtained into the value-added process, to generate knowledge from processes, and then to use this knowledge. This should lead to the establishment of a continuous improvement process (CIP) that (ideally) occurs in all areas of an enterprise and is supported by EDM.

Some important elements of knowledge management can be implemented immediately by using integrated CAD and EDMS. One of the most important targets of this technique is to re-use parts and entire products, including even the re-use of product structures that have already been proven effective. The know-how of the people involved exists precisely in these products. The ethos of knowledge management is not to dispose of something after using it once, but rather to maintain it and organize it wisely, so that it can be accessed again for similar projects. EDM alone does not represent knowledge management. Even if an organization has not addressed the questions of the parameterization and application of knowledge management, this says nothing about the need for EDM or its potential benefits.

8.1 Basic Models and Approaches of Knowledge-Oriented EDM

8.1.1 System-Oriented Reference Frame of Knowledge-Oriented EDM

For further considerations of the integration of knowledge management in engineering data management, it is necessary to establish a uniform definition that places it into the context of the basic model of knowledge management. The approach for knowledge-oriented EDM can now be applied to the system-oriented frame of reference. Thus, tools for data searching and data submission are assigned to the IT facilities within the knowledge system, while software applications, their functionality and data management systems are assigned to the technical facilities of the value-added system. For this reason, it is possible to apply and depict engineering data management on the general approach of the system-oriented frame of reference. As an example, the application of CAD template models within a development process requires a specific system configuration (e.g. pre-settings, standards). These guidelines should be provided and administrated by the EDM in the context of the entire IT-architecture.

8.1.2 The Knowledge Process as Connection Between Business Process and Support Process of EDM

One of the key factors for successful knowledge management in organizations is the integration of knowledge management operations into business processes. This knowledge is often seen as particularly valuable because it increases the effectiveness and efficiency of the organization in its core business processes. In addition to business processes, which are seen as the primary starting point for the development of knowledge management, other types of processes are also taken into account, particularly the processes of continuous maintenance of the knowledge base, the knowledge management processes and the overall knowledge (transfer) processes [1].

EDM supports processes of continuous maintenance, including editorial activities, innovations of internal service providers, the compilation and preparation of documents, and the examination, correction and update of information. Meta processes of knowledge management or of knowledge management processes are knowledge management projects or operations that develop resources for the knowledge work (e.g. the design and implementation of knowledge management applications, the education of qualified knowledge workers or the consultation of organizational responsibilities within an enterprise). Knowledge processes transfer and develop knowledge beyond business processes and business cases because they also coordinate the handling of business processes and continuous maintenance activities. EDM is therefore like a continuous support process, which supplies business processes with the necessary product and process data.

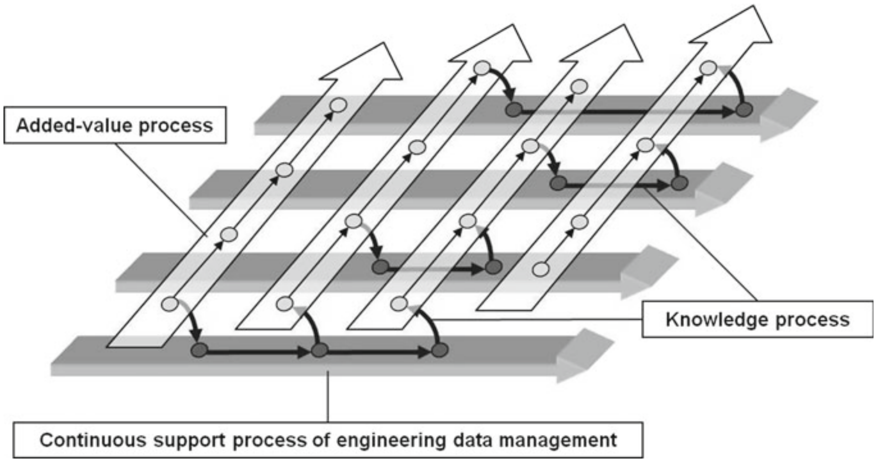


Fig. 8.1 The knowledge process as a link between the business process and the continuous support process of EDM

Knowledge management in business processes (i.e. its primary target) is unthinkable without relying on secondary support and transfer processes. Therefore, the second key factor in process-oriented knowledge management is to adjust and link different process types with each other. Figure 8.1 shows that the knowledge process provides the link between business processes and the continuous support process of EDM. Knowledge-intensive processes are characterized by flexible and non-predictable knowledge requirements. They produce different and (at the time of modeling) only partly foreseeable results and are characterized by strong knowledge transfers between different individuals and business cases. Process-oriented integration brings the fields of knowledge management and engineering data management together on the knowledge, operational, and data levels.

8.1.3 Integrated Approach to Added-Value Processes

The defined reference frame shows clearly that the levels of knowledge and data observation should not be considered separately. Instead, it is necessary to connect and integrate them on the operational level into the value creation process. Up to this point, the focus has been on integrating knowledge management in business process modeling. The focus will now shift to the issue of combining disciplines in knowledge management and data management. The questions are: what is the role of knowledge management processes for operational data management, and what opportunities does a knowledge-oriented view of data management afford?

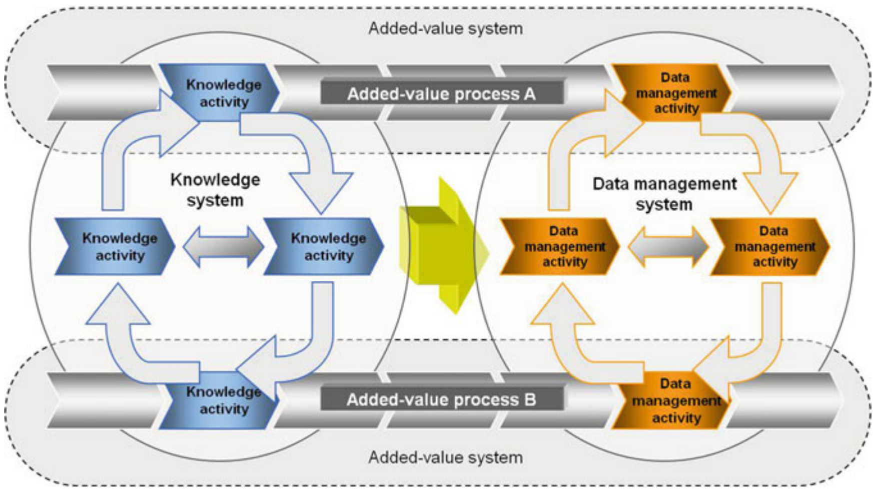


Fig. 8.2 Integrated perspective of knowledge and data management operations between added-value processes

Figure 8.2 shows the link between knowledge processes and added-value processes, including a further enhancement with the data level and its characteristic properties. It can be argued that, theoretically, a data management operation has to be considered for every knowledge operation in the knowledge process. In this approach, it is crucial that the project management requirements must drive the operational data management. In the end, the individual data management operations are triggered by the requirements of knowledge processes. The need for a knowledge-oriented data management is justified by the challenge of identifying the operational requirements of data management and further satisfying them with appropriate methods. To achieve this, the relevant data management operation must be defined for each knowledge operation, and the knowledge process must then be incorporated into a systematic data management.

8.1.4 Model for the Integration of Knowledge Processes and Data Management

Data management is the set of methodological, conceptual, organizational and technical measures and procedures used to deal with data. The aim is to exploit the full potential of data in business processes and to ensure the optimal use of the data in ongoing operations. As discussed in Sect. 7.3.5, knowledge is transferred in EDM via the documentation and information processes, as well as by the corresponding data transfer. This requires a complex data management in order to achieve a continuous knowledge process, which means that processes should be designed in the EDM environment with respect to data management.

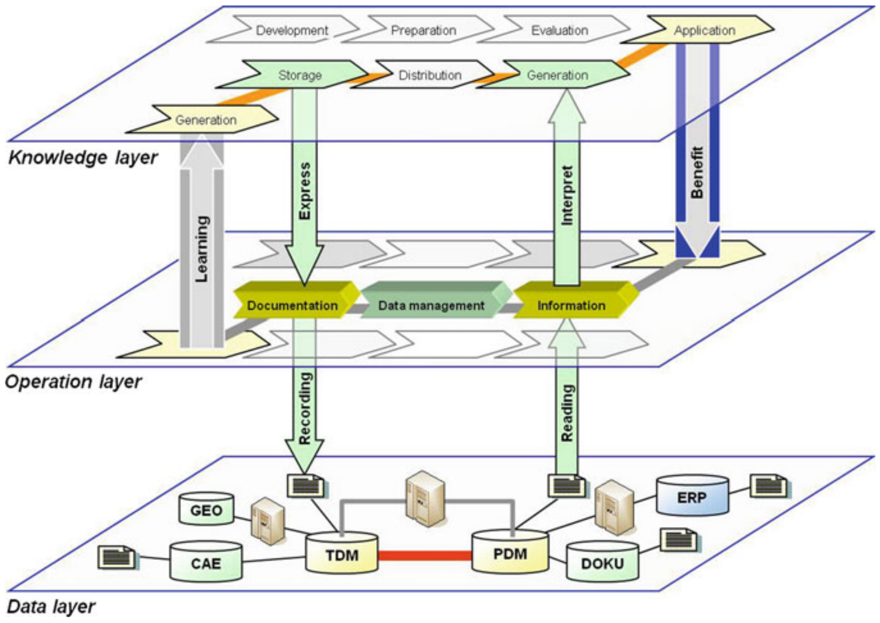


Fig. 8.3 Model for the integration of knowledge processes and data management through the operational level

Figure 8.3 shows how this relation can be modeled across the process-oriented operational level. The interaction takes place between the knowledge and data levels of an organization, whereby one level is receiving and the other is providing knowledge respectively data. Business-process-oriented knowledge management must assign equal importance to the employees and the technical facilities. Thus, the efficient integration of the knowledge and data levels into processes makes it possible to supply each subsequently developed business process with knowledge and data from other business processes that have already been established in the organization.

A new approach has been consciously employed in Fig. 8.3 to emphasize the operational level and its *documentation* and *information* processes. Since these two processes provide the most important links between the knowledge and data levels, they are highlighted here. A theoretical contemplation and study of data management activities leads to the conclusion that knowledge processes play an important role for the activation and the sequence of such data management activities.

As Fig. 8.3 shows, knowledge processes can be represented as an integrated interface layer between the added-value processes and data management. In this configuration, knowledge processes perform tasks which have not been previously performed and build the basis for knowledge-based data management.

8.1.5 From the Knowledge Transfer Model to the Knowledge-Oriented Engineering Data Management

Projecting the previously established models and approaches onto Hartlieb's knowledge transfer model shows that this knowledge transfer model can be extended by incorporating the frame of reference of knowledge-oriented EDM, as shown in Fig. 8.4.

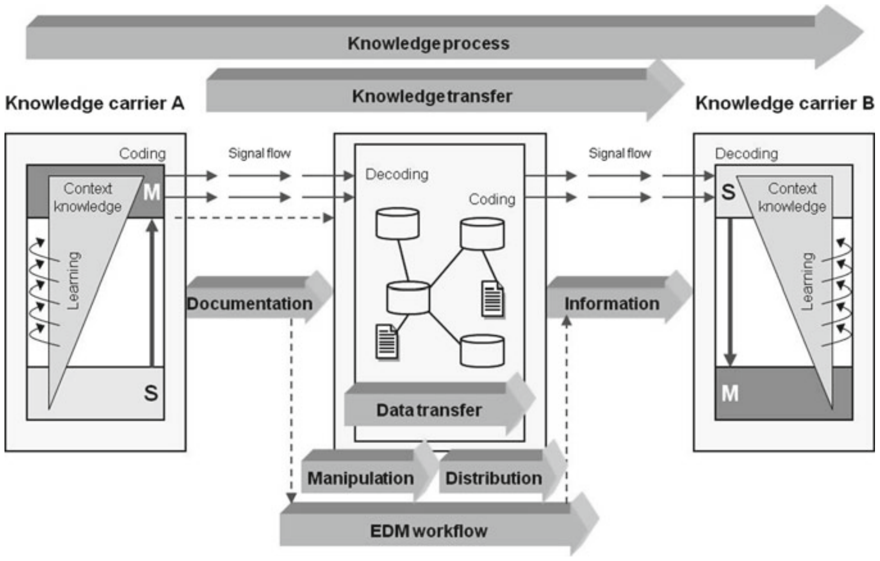


Fig. 8.4 Knowledge transfer model of knowledge-oriented engineering data management

This model contains a more detailed description of data transfer, which includes the prerequisite that data management requirements should always be placed in the context of current knowledge processes. The knowledge process between knowledge carrier A as transmitter of knowledge (Motorium M) and knowledge carrier B as receiver of knowledge (Sensorium S) is the underlying consideration for this approach. In terms of process orientation, this knowledge transfer commonly takes place between two added-value processes or two activities within a added-value process. Therefore, the portion of knowledge which cannot be transferred through communication within a shared context will be transferred via a data-management-supported indirect knowledge transfer.

This figure also shows that the success of knowledge transfer depends on the continuity of the knowledge process. In the context of EDM, this means that the support by data management and its functions is not sufficient, and that the *documentation* and *information* processes must also be supported. EDM must also receive, interpret and translate the signals from sender and receiver that enter into the knowledge process in order to support an ongoing process flow. Finally, the figure also shows that a workflow accompanies the knowledge transfer, which is a system-supported way to ensure that the knowledge process is not interrupted. In order to compensate for the different requirements for the interpretation of encoded knowledge between sender and receiver, the EDM must perform data processing for both the receiver and the user. However, since the workflow only supports the process of knowledge transfer, EDM must also provide a mechanism for data distribution so that the data reaches its place of usage, as well as a mechanism for the physical data transfer. Based on the knowledge-intensive processes, relevant knowledge processes can now be defined and analyzed according to this particular model. This makes it possible to develop EDM under the consideration of the aforementioned aspects, which can be seen as knowledge-oriented engineering data management.

8.1.6 Model for the Reconstruction of the Knowledge Base and Database

In order to prepare the knowledge transfer model of the knowledge-based engineering data management in a logical manner, it is necessary to conduct a thorough analysis of the knowledge base and database. This is crucial for the quality of the derived EDM measures and activities. Based on the analytical structure of knowledge-oriented EDM (which is discussed in the relevant literature [2]), this structure can now be schematically summarized as a model for the reconstruction of knowledge/data requirements and the data services offered.

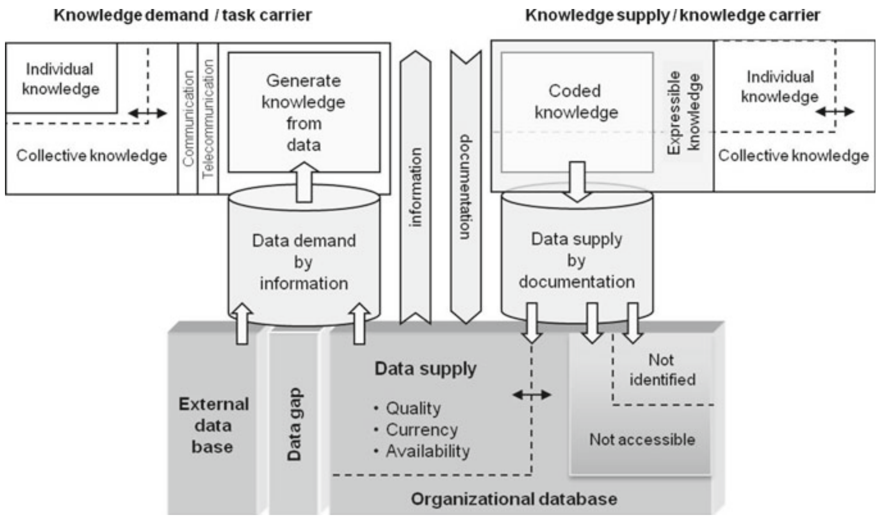


Fig. 8.5 Model for the reconstruction of knowledge- and database

Figure 8.5 shows a model for analyzing and reconstructing knowledge demands and knowledge supply using existing in-house and external data potential. The knowledge/data transfer takes place between the task carrier on the left side and knowledge carrier on the right side. The availability of organizational and external databases form the basis, while the processes of documentation, by which the knowledge carrier tries to offer his knowledge in form of data to the task carrier, act between. The task carrier tries to satisfy the data demand via the process of information and the decoding of data by the recipient creates new knowledge. This model is also applied in Sect. 9.1.4, which demonstrates the need to conduct a systematic analysis of the established structure in order to build up knowledge or provide data.

8.2 Requirements for the IT Support of Process-Oriented Knowledge Management in EDM

The application of technical subsystems in knowledge management is justified by the potential simplification of procedures. In the field of knowledge management, this simplification refers primarily to the optimization of the processes of information, documentation and data transfer. The documentation process can be seen as the basis for subsequent underlying processes. Documentation must be receiver-oriented, which means that it must take into account the knowledge that the receiver has to apply for a successfully data interpretation. EDM has the task of supporting the documentation process via appropriate methods and applications.

The technical subsystem can support the documentation process by managing and administering data. In this context, data management can be understood as the storage, archiving/backup, structuring and classification of data, as well as data distribution and retrieval. For the documentation process (i.e. the transformation of information/knowledge into data), the technical subsystem can thereby provide assistance by performing tasks such as the search for information (search engines), translating data (translation engines) and linking data.

One of the key design parameters in knowledge management is the interface between man and machine. This aspect, which is often referred to as the *user interface*, ultimately influences the way that information processes can be triggered. In practical applications, the factor '*ease of use*' defines the extent to which the user interfaces meet user requirements. One of the most common mistakes is that interface designs ignore or fail to fulfill user demands. The process of *data transfer* interlinks particular system elements of the technical subsystem. Here again, this process may be hindered by incompatibilities between particular elements. In practical application, technical barriers (e.g. system failures, lack of networks, using too many different systems and application products) place significant limitations on the theoretical knowledge transfer process.

8.2.1 Modeling Approach for the Technical Subsystem

In order to design the interface between man and machine (or between the operational level of business processes and systems) in accordance with the needs and practices of EDM, the fundamental approaches to modeling business information systems (IS) must be examined firstly. The four most common approaches are the following:

- *Functional decomposition* is based on the decomposition of IS functions into multi-level sub-functions when defining interfaces.
- The *data flow approach* attempts to define IS as a set of data flows which are transformed by actions. Data storage is made available for the temporal buffering of data streams.
- *Data modeling* focuses on describing the structure of the database of the IS. This structure consists of data object types with associated attributes. The individual data object types are connected via defined relationships.
- The *business-process-oriented approach* marks the transition from a primarily static and structured view of IS to a dynamic and behavioral perspective.

In order to describe the modeling scope of the different approaches, a distinction is made between the static view of the IS function (function view), the data structures (data view), the communication channels (interaction view) and the dynamic view of operations (task view). A function is understood as a task which is performed with the assistance of computers.

Defining the Task Structure

As the connecting element between process and system, a task is defined as the goal of task-related human actions. Figure 8.6 shows a task structure as the connecting element between process and system in the area of EDM.

To indicate the possible degrees of freedom in the specification and implementation phases of a task, the terms external and internal view of a task, task carrier and process are used here with following meanings.

The external view of a task defines what should be achieved and when. It influences the effectiveness of a process. Here, it is important to identify the right tasks, to define them and finally to perform them. The top left-hand area of Fig. 8.6 shows how the right tasks can be derived. Initially, it is important to define manageable business processes based on corporate objectives. These processes are extended deadlines, organizational factors and responsibilities and thus value-defined, value-adding support processes can be derived. Relevant, mission-critical processes can then be more detailed and arranged in a task structure, which makes it possible to describe tasks more exactly. At the same time, a role structure is created with the purpose of describing functions for staff. The definition of the external view is supported by the enterprise's handbook of quality management, which defines standards and job instructions for operating processes.

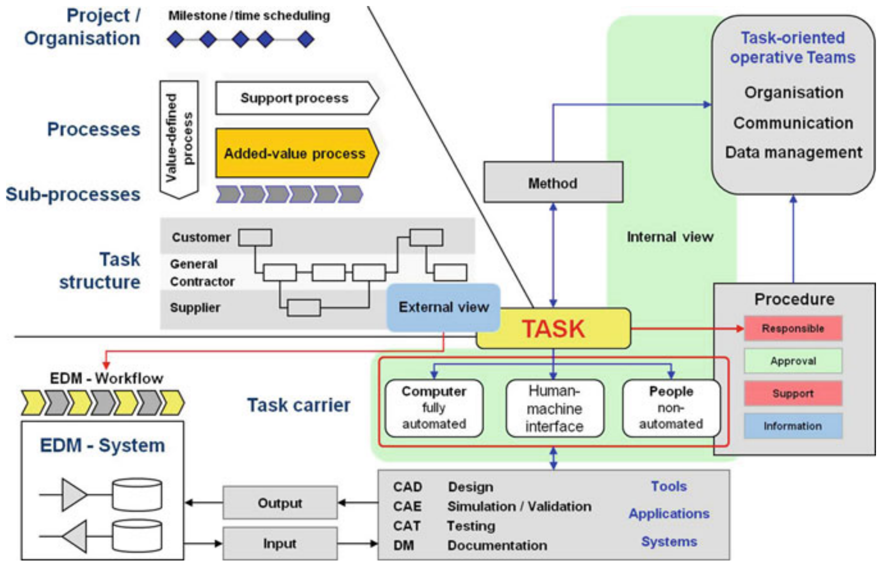


Fig. 8.6 Task structure as the connecting element between process and system in EDM

The internal view of a task specifies the efficiency of this task, which is basically about doing the task in the correct way. It defines the problem-solving method of the task and refers to how the task will be completed (the task carrier) i.e. whether the task is done by humans or machines. How should the task be executed to reach the task target? The internal view therefore defines the method, tools and systems which are most suitable for the fulfillment of a task. This includes the assignment of responsibility and defines what an employee must achieve, what task is involved and how long it should take.

The method - *how to do assignment solutions* - has to be defined for each particular task and simultaneously forms the basis of the definition of employee skills. It is also the basis for the determination of requirements for system, software solution and application. The human being as the task carrier represents the most important connection to merge the internal and external views and carries out the task in a self-contained way. The task-oriented formation of groups includes contexts of teams, organizations and roles. The assignment of tasks and responsibilities and the competences and qualifications of employees can be derived from such a correlation in order to facilitate the fulfillment of a task. Finally, the human being uses systems for the execution of tasks to a certain extent. The procedure describes the operational implementation of the task - *who can execute the solution procedure and with what?* - This can be done by:

- People (non-automated task)
- Computers (fully automated task)
- Man-machine systems (partly automated tasks)

Depending on the scope of the tasks and the complexity and repetition level, system supporting methods can be defined and then integrated into the EDM environment. This system support can be established through application processing (e.g. CAD, CAE, CAT) or in a cross-functional way through EDM workflows or across systems interfaces and networks of the EDM system. The economic efficiency of a process is characterized by the definition of the external view (effectiveness) and the internal view (efficiency) of a task and also the qualification of task carriers (skills, responsibility) and the degree of system support for the procedure. In the EDM environment, the external view of tasks basically defines and controls the EDM workflow, while the internal view affects applications and functions with its methods.

8.2.2 The Database of Knowledge-Oriented EDM

The database consists of components of the technical subsystem that are able to process data, as well as documents and metadata of an organization, regardless of their origin. The organizational memory is supported by generating and recording data. This is especially relevant in the documentation of business-relevant data, which allows other knowledge carriers to re-integrate knowledge into the information process at any time. The automotive industry must deal with an extraordinary variety of data. The essential forms of data are:

- Documents (e.g. measurement reports, specification sheets, standards, protocols, testing regulations, rules, product descriptions, modification documents)
- Metadata (e.g. computation data, mass, gravity center, technical data)
- Geometries (e.g. design models, textures, designs, styling surfaces, facades)
- Process data (e.g. design, simulation, planning, manufacturing, production)
- Data structure (e.g. configurations, process-oriented views of interest)
- Project management and process management data (e.g. milestones, schedules, costs)

8.2.3 EDM Workflow Support of Knowledge-Intensive Processes

Engineering data management systems (EDMS) that are currently available on the market are often either insufficient or poorly suited for supporting knowledge-intensive processes. In order to represent process-oriented knowledge, it is necessary to integrate so-called contexts into the workflow module of EDMS. Using contexts integrates the workflow editors into the workflow process and therefore gives them access to process-related knowledge that is not included in the process model, such as requirements, processes handling experience, reasons for decisions, knowledge about customers or competitors, and information on time limits. The context, which contains a whole set of documents that represent process-related knowledge, is tied to a specific workflow object of the process structure (process, action, application, template of documents and business case) or the organizational structure (organizational unit, role, editor). This context association to a workflow object determines when a document is relevant for the context of a processing operation. The developers can thus find additional knowledge within the workflow execution via these different types of contexts. This knowledge can be necessary for a particular operation or can be relevant in a certain business case.

8.2.4 Management, Transfer and Steering of Knowledge-Oriented EDM

A major goal of knowledge-oriented EDM is to support the task carrier's ability to act as effectively as possible at the relevant time and in the relevant place. Thus, task carriers must use the knowledge offered through data in the optimal manner in order to carry out development tasks and hence meet customer requirements. Here, the cross-linking of knowledge demands and knowledge offers is very important and takes place at the operational, knowledge, and data levels, whereby the demand generation, the place of demand and the time of demand have to be taken into account as well. The individual levels also have to be considered more closely during the transfer. A transfer can only be initiated if the existing knowledge and data is transparent and available. In terms of the availability of data in a very complex development process data are continuously generated anew. The essential points are:

- Data generation
- Data preparation
- Data allocation
- Data evaluation
- Data control and management

Data transfers between technical subsystems (e.g. simulation programs and databases) are executed on the data level. The transfer of relevant data (e.g. protocols, product data, structures, geometries) guarantees the ability to act at the receiving point of an engineering task. Before providing further analysis about design in knowledge-oriented engineering data management, some background information is necessary. Here, the exclusive focus is on cases in which the knowledge carrier generates the necessary knowledge from data that has been sent through a completed data transfer.

8.3 Knowledgeware in Product Development

Exponential growth in the area of computer-based development is still continuing. Replacing the drafting board by computers and its related software was a first step which is nowadays being repeated due to parametric design and simulation systems. State-of-the-art CAD systems not only replace earlier design methods, but also provide a strong functional enhancement that can help to cope with the growing demands of product development. The use of knowledgeware requires more organizational effort than the pure installation of CAD software and the introduction of users to its functional range. Beyond an extensive knowledge of the product being developed and its context, design engineers must also possess IT know-how. Only this additional know-how will enable them to create software systems that will offer advantages in subsequent processes.

However, even modern CAx systems cannot generate new design models with just the push of a button. Human beings remain the critical factor and must have the support of adequate tools to be able to cope with the complexity and information flood that characterize simultaneous engineering. One additional important step is the standardization of development tasks, which increases the service capacity of parameterization. The extension of known CAD functionalities not only offers potential improvements in collaboration between product development and production engineering, but also in the area of knowledge integration. The terms *knowledge-based design* and *knowledge-based engineering* (KBE) (see Chap. 5, pp. 309) are used when organizational knowledge is used in product development. This knowledge, which must be suitable for the product development process, is used in a way that does not rely on individuals. One approach to support the enhancement and integration of product knowledge during product development is the method of parametric-associative design. This approach enables an integration of intelligent tools into a CAD environment, which then make it possible to distribute and support knowledge within an organization. Since available data has to be captured and mapped as knowledge in CAD models, it is possible to transfer crucial parameters.

8.3.1 The Parametric-Associative Approach

The parametric-associative approach is becoming more common in vehicle development. It supports network thinking, development and the design of vehicle assemblies within different process stages. Chapter 4 includes an introduction of the design-related aspects of parametric-associative development and gives insights into functionalities of parametric-associative CAD. In the years before the introduction of parametric-associative design, vehicle development was divided into sequential development phases. However, the parametric-associative approach enables the use of computational monitoring to make processes transparent and simultaneous in different development phases. The associative parametric approach allows *knowledge* in CAx/EDM-integrated applications to be stored and managed. Processes, specialized departments and suppliers that were formerly kept separate can now move closer together and interact during different stages of development using new networking strategies. Parametric-associative CAD models store knowledge about the vehicle development process and make it reusable for new vehicle projects.

The advantage of this new transparent vehicle development is the continuous control over different phases and the resulting time and quality optimization. These qualities are particularly important in the contemporary automotive industry, where the increasing complexity of vehicle models and the concomitant increase in information content demand improved efficiency. Minor changes in a process section, which previously would have lead to uncontrolled effects on individual components, are now optimized and controlled within the CAD environment. The traditional development method, in which CAD systems offered isolated, static models, is being replaced by flexible parametric models. This new approach creates relationships between individual components and assemblies so that changes of vehicle components that are dynamically linked can be automatically altered through several stages of development [3].

In the past two decades, the growing number of vehicle projects and the related requirements have led to a certain segregation of the parties involved in development. Coordinators responsible for projects and processes are mostly separated from the design development, which is mainly handled by development partners. Due to this, OEMs have lost much engineering knowledge that would be crucial for the continued systematic progress of associative parametric design processes. Extensive efforts by all parties involved in the process are needed in order to create a continuous parametric-associative process chain from the layout phase to manufacturing. The combination of design with computation and production supply planning is opening up new paths for optimization.

8.3.2 The Fundamentals of Parametric-Associative Design

In parametric-associative design, geometry-related information is not only stored as the product shape but also as cross-links between geometry elements. The shape of a model will be automatically adjusted if the variables that define the geometry model (i.e. parameters) are changed or replaced, but the design goals will also be taken into account. This highlights the importance of the ability to reuse CAD models for similar design applications (cf. Chap. 4, pp. 241 and Chap. 5, pp. 309). Concerning CAD models, a huge variety of control parameters are available. In standard models, it is sufficient to visualize important parameters explicitly in tree structures. However, geometry families can also be joined in design tables and can therefore be altered by combining explicit parameters. Applying formulas in CAD programs or tables can further extend the range of control parameters. In the case of very complex models, control parameters can also be derived from programmed macros and codes. In addition, design tables and macros can be used to exchange parameters with other CAE programs, as well as to optimize parameters.

Parametric-associative design presents a significant opportunity to collect and refine company and employee know-how and to make this asset accessible for future developments. All design engineers must be challenged to link their personal knowledge with other CAE processes and to enhance it using parametric-associative design. The following elements are essential for this method:

- *Parameters* can be both numerical and geometric elements (e.g. lines, curves or surfaces).
- *Implicit parameters* are (hidden) parameters (e.g. defined in 2D sketches of 3D models or drawings).
- *Explicit parameters* are visualized in the CAD model structure tree parametric and control the geometry elements.
- *Design tables* are able to define parameter families for the control of geometry in parts and assemblies.
- *Checks* (e.g. warnings via signals and messages) alert the user of any non-compliance in the parameters defined.
- *Checks with reactions* trigger branched design steps if, for example, defined parameters are changed during the update process.
- Any values, such as from mechanics (momentum, acceleration) or geometry (volume, trigonometric functions), can be represented as parameters and applied in formulas or rules.
- Complex relationships and design variants can be controlled by macros (e.g. programmed in Visual Basic).

8.3.3 Knowledge Management and Product Configuration

Knowledge management and product configuration are treated in terms of their characteristics within EDM. *Design engineers*' and *simulation engineers*' knowledge is stored as computer-generated data. The goal is to use specific programs to process data such that certain tasks in creative engineering can be automated as well. In the past, large enterprises developed some approaches that tried to achieve this goal in scientific projects, and a few software companies even tried to create programs with such abstract goal formulations in order to create approaches which could handle applications from different fields. The ulterior motive was to create standard software which could be used in different areas of expertise [4].

However, with regard to the complexity of engineering, it can be stated that the development of such software solutions is a serious challenge. Although computer programs, which can make many things much easier, have become indispensable tools (especially in the area of engineering), computer programs are not able to replace the creativity of engineers. The approach developed in recent years, called knowledge management, is more realistic. This term encompasses the improved use of available know-how. Parameters, templates and models can be applied to store and document the experience of engineers, which is thereby made reproducible. One tangible application of this principle is the parametric-associative concept vehicle, which is described in more detail in Sect. 9.3.

References

1. Hoffmann, M., Goessmann, T., Misch, A.: Unsichtbar oder Vergessen-Wie man “verborgenen Wissensprozessen” auf die Schliche kommt. In: Professionelles Wissensmanagement-WM2001 (2001)
2. Hartlieb, E.: Zur Rolle der Wissenslogistik im betrieblichen Wissensmanagement. Ph.D. thesis, Graz University of Technology (2010)
3. Tecklenburg, G. (ed.): Die digitale Produktentwicklung. Expert (2008)
4. Sendler, U., Wawer, V.: CAD und PDM-Prozessoptimierung durch Integration. Hanser, München (2011)

Chapter 9
Advanced Applications of CAD/EDM in the Automotive Industry

The following investigations examine the product development process in the automotive industry. Subjects relating to current problems were specifically selected from the EDM environment of product development.

9.1 Applications for Knowledge-Based EDM

Now that knowledge processes have been identified as a relevant component of data management operations, and the general positioning of knowledge and data management has been clarified, it is important to analyze in detail how these two things interact. In terms of key knowledge-intensive operations, the focus is on the level below the top level of the sub-processes of the business process model. Here, knowledge-intensive operations have to be executed, which requires the application of knowledge processes. Within the highly IT-driven research and development area, knowledge-intensive activities are very often coupled with intensive data management operations. Although the system landscape herein varies, product data management plays a central role in automotive development.

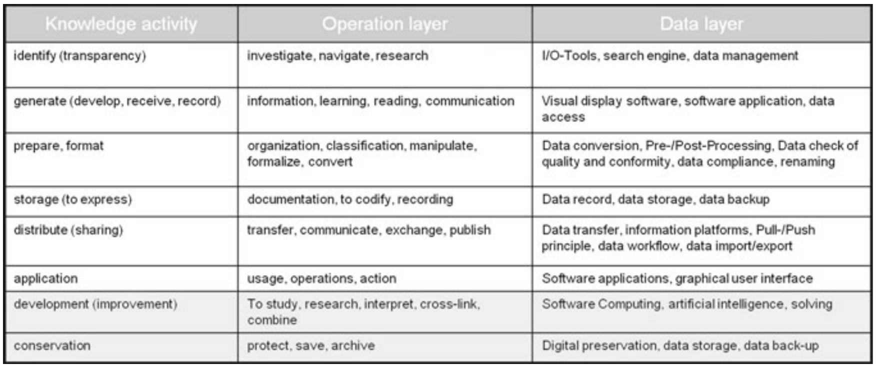
If the product development process demands a certain result at a certain time (milestone), and if the desired results require the execution of knowledge-intense operations, then the requirements for this milestone also influence the requirements placed on the realm of data. These requirements are combined in a *data-synchronization point*. Data management activities can be performed if the requirements of the synchronization point are fulfilled at a certain time. In this case, the knowledge process can be completed, and the partial result of the knowledge-intensive operations can be fed back into the process. The determination of these synchronization points and the chronological order of data management operations and knowledge operations depend on the particular applications. Section 9.2.3 describes an application that uses the concept of *data roadmap*.

9.1.1 Relevant Knowledge Operations in EDM

Knowledge operations determine the way that *knowledge* is handled at each step of the knowledge process. However, it is necessary to define the knowledge operations that are used to analyze various knowledge processes in terms of specific implementation scenarios.

Figure 9.1 provides a summary of a case study that shows the EDM-relevant knowledge operations and their derivation on the operational and data levels. To understand the relationship between knowledge processes and data management, it is important to specify the following knowledge operations, which are necessary for reporting:

- Knowledge identification
- Knowledge generation
- Knowledge storage
- Knowledge preparation
- Knowledge distribution
- Knowledge application



Knowledge activity	Operation layer	Data layer
identify (transparency)	investigate, navigate, research	I/O-Tools, search engine, data management
generate (develop, receive, record)	information, learning, reading, communication	Visual display software, software application, data access
prepare, format	organization, classification, manipulate, formalize, convert	Data conversion, Pre-/Post-Processing, Data check of quality and conformity, data compliance, renaming
storage (to express)	documentation, to codify, recording	Data record, data storage, data backup
distribute (sharing)	transfer, communicate, exchange, publish	Data transfer, information platforms, Pull-/Push principle, data workflow, data import/export
application	usage, operations, action	Software applications, graphical user interface
development (improvement)	To study, research, interpret, cross-link, combine	Software Computing, artificial intelligence, solving
conservation	protect, save, archive	Digital preservation, data storage, data back-up

Fig. 9.1 Derivation of knowledge operations at the action and data levels

After the knowledge base has been implemented and the knowledge processes have been modeled, knowledge operations can be used to derive the required operations down to the data level.

9.1.2 Factors that Influence Knowledge Transfer Via Data Transfer at the Operational Level

Value-adding processes that are based on a defined working procedure generally take place on the operational level. If the operations are mainly intended to support knowledge management (which is the case in product development), the investigation can be limited to two relevant processes: the information process and the documentation process. Thus, both documentation and information processing can be seen as special types of operation, in which a human being influences a technical subsystem with the goal of encoding knowledge and making it accessible to other people. This process generates data and adds additional value. When implementing EDM, the process of data transfer is particularly important for the quality of the knowledge transfer.

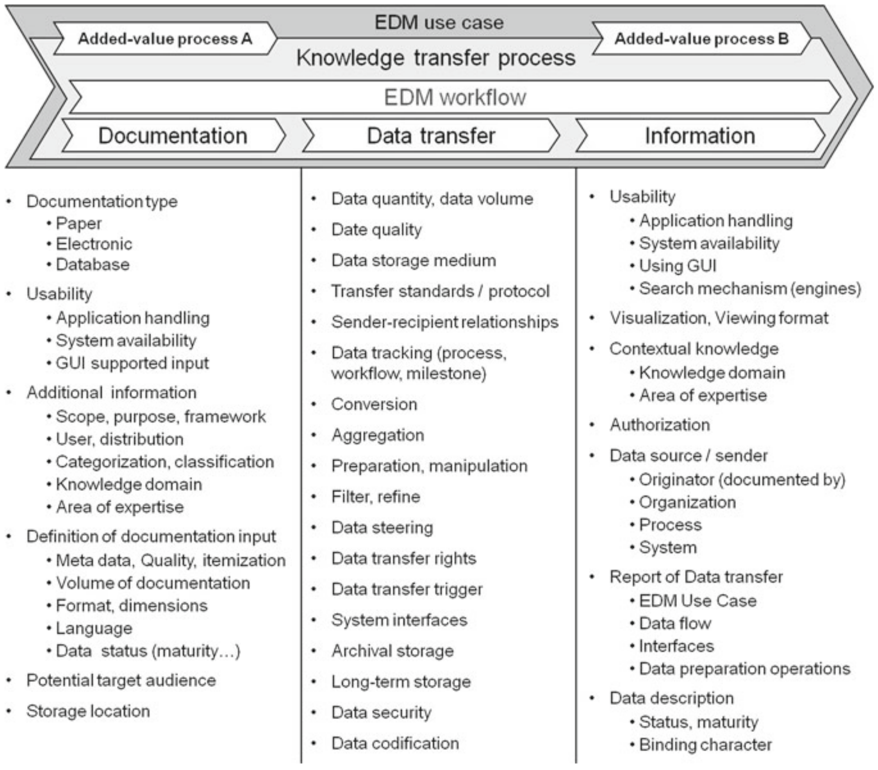


Fig. 9.2 Factors that influence the data transfer process

Figure 9.2 provides an overview of the possible interventions in the process of data transfer between value-added processes A and B. This overview is the result of an analysis of numerous EDM use cases in the course of a case study. In data transfer, the following data management operations can be executed:

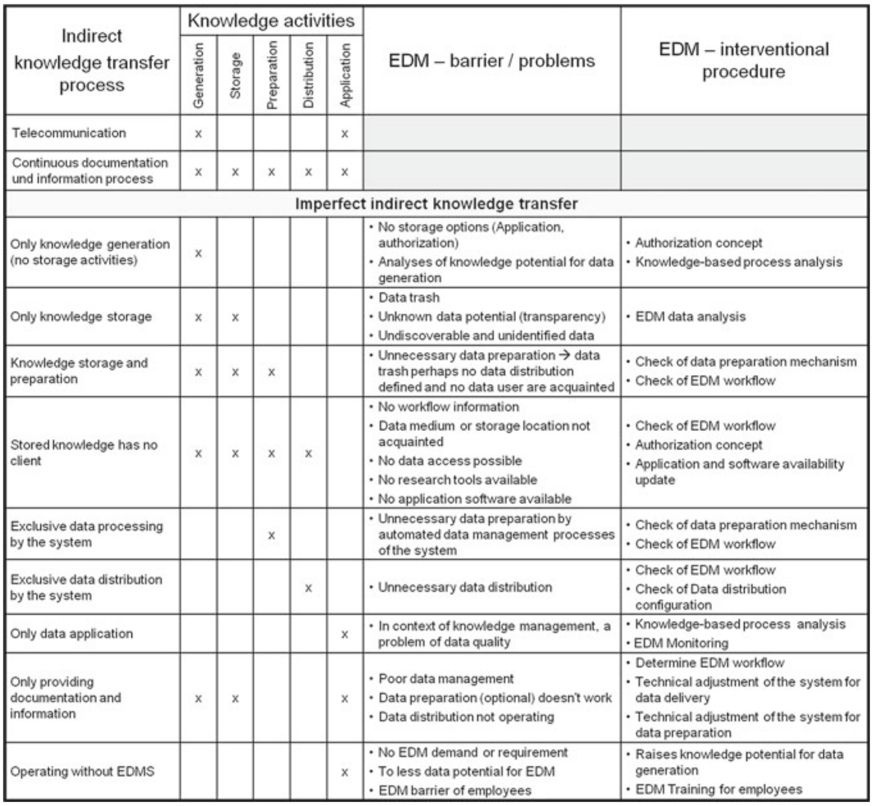
- Data enhancement
- Data filtering
- Data preparation
- Data conversion
- Data manipulation

Since each individual knowledge operation can potentially affect the quality of the knowledge transfer in the relevant overall knowledge transfer process, there are a number of barriers that may disrupt the knowledge flow.

9.1.3 Data Management Barriers in Indirect Knowledge Transfer

Knowledge processes based on information from the business process analysis are used to analyze the actual knowledge transfer. Thereby, barriers occur in the data management of the EDM use case.

Figure 9.3 shows different characteristic combinations of knowledge operations and their influence on knowledge transfer. It also shows the related EDM problems and intervention measures. The *knowledge activities* column shows when knowledge processes are not working. Based on this pattern, classifiable data management operations can be derived, which are described in the *EDM interventional procedure* column.



Indirect knowledge transfer process	Knowledge activities					EDM – barrier / problems	EDM – interventional procedure
	Generation	Storage	Preparation	Distribution	Application		
Telecommunication	x				x		
Continuous documentation und information process	x	x	x	x	x		
Imperfect indirect knowledge transfer							
Only knowledge generation (no storage activities)	x					• No storage options (Application, authorization) • Analyses of knowledge potential for data generation	• Authorization concept • Knowledge-based process analysis
Only knowledge storage	x	x				• Data trash • Unknown data potential (transparency) • Undiscoverable and unidentified data	• EDM data analysis
Knowledge storage and preparation	x	x	x			• Unnecessary data preparation → data trash perhaps no data distribution defined and no data user are acquainted	• Check of data preparation mechanism • Check of EDM workflow
Stored knowledge has no client	x	x	x	x		• No workflow information • Data medium or storage location not acquainted • No data access possible • No research tools available • No application software available	• Check of EDM workflow • Authorization concept • Application and software availability update
Exclusive data processing by the system			x			• Unnecessary data preparation by automated data management processes of the system	• Check of data preparation mechanism • Check of EDM workflow
Exclusive data distribution by the system				x		• Unnecessary data distribution	• Check of EDM workflow • Check of Data distribution configuration
Only data application					x	• In context of knowledge management, a problem of data quality	• Knowledge-based process analysis • EDM Monitoring
Only providing documentation and information	x	x			x	• Poor data management • Data preparation (optional) doesn't work • Data distribution not operating	• Determine EDM workflow • Technical adjustment of the system for data delivery • Technical adjustment of the system for data preparation
Operating without EDMS					x	• No EDM demand or requirement • To less data potential for EDM • EDM barrier of employees	• Raises knowledge potential for data generation • EDM Training for employees

Fig. 9.3 EDM barriers in the knowledge transfer process

9.1.4 Reference Process for the Knowledge-Oriented Development of EDM Use Cases

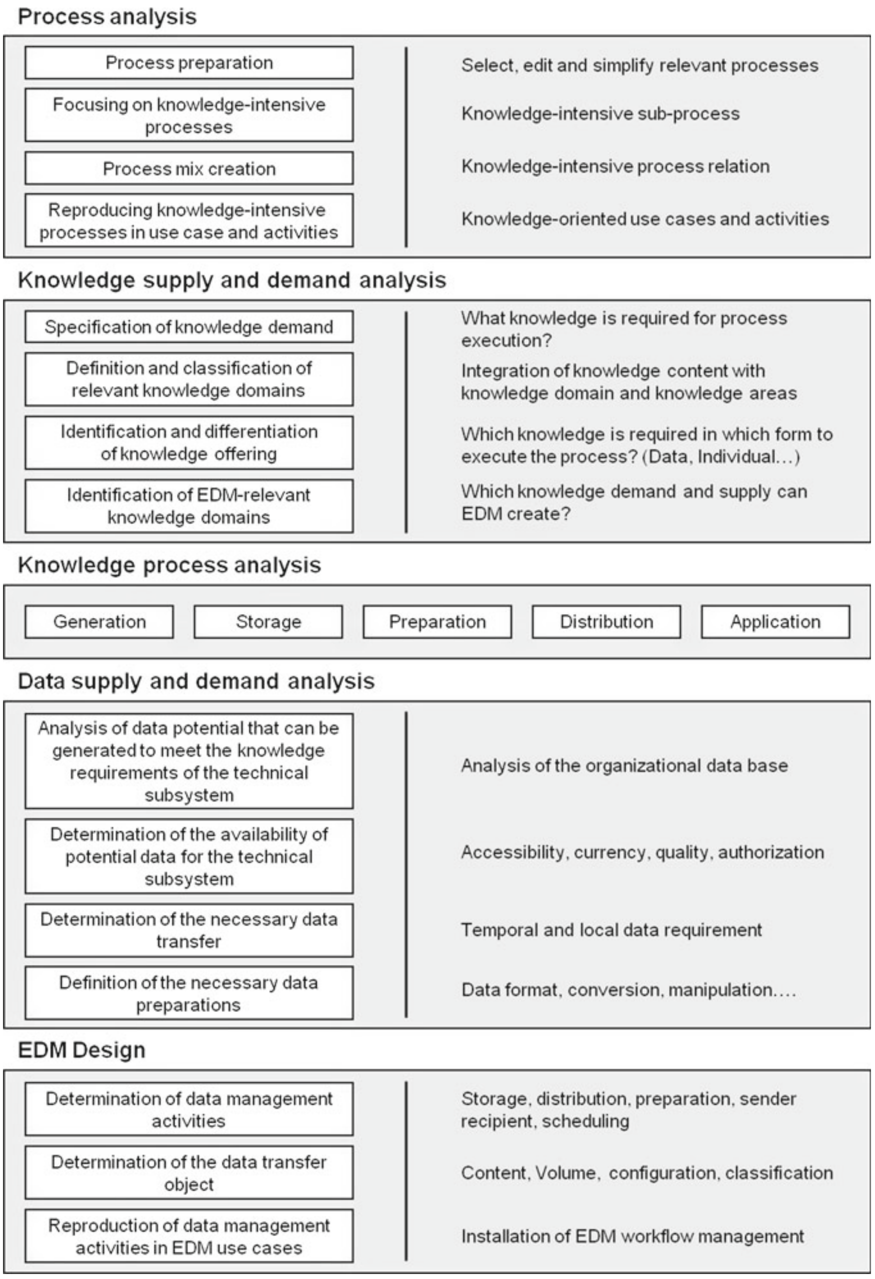

Process analysis
Process preparation
Focusing on knowledge-intensive processes
Process mix creation
Reproducing knowledge-intensive processes in use case and activities
Select, edit and simplify relevant processes
Knowledge-intensive sub-process
Knowledge-intensive process relation
Knowledge-oriented use cases and activities
Knowledge supply and demand analysis
Specification of knowledge demand
Definition and classification of relevant knowledge domains
Identification and differentiation of knowledge offering
Identification of EDM-relevant knowledge domains
What knowledge is required for process execution?
Integration of knowledge content with knowledge domain and knowledge areas
Which knowledge is required in which form to execute the process? (Data, Individual…)
Which knowledge demand and supply can EDM create?
Knowledge process analysis
Generation
Storage
Preparation
Distribution
Application
Data supply and demand analysis
Analysis of data potential that can be generated to meet the knowledge requirements of the technical subsystem
Determination of the availability of potential data for the technical subsystem
Determination of the necessary data transfer
Definition of the necessary data preparations
Analysis of the organizational data base
Accessibility, currency, quality, authorization
Temporal and local data requirement
Data format, conversion, manipulation….
EDM Design
Determination of data management activities
Determination of the data transfer object
Reproduction of data management activities in EDM use cases
Storage, distribution, preparation, sender recipient, scheduling
Content, Volume, configuration, classification
Installation of EDM workflow management

Fig. 9.4 Reference process for the knowledge-oriented development of EDM use cases

As a central investigation object, this use case shows the correlation between knowledge processes and data management in a reference model. Thus, the planning phase for data management can be integrated into the knowledge-oriented concept.

The use cases now generate requirements as they are normally generated in the general planning phase, where process methods set the requirements for the data management. With the positioning of use cases and the representation of the knowledge processes that occur along these use cases, it is possible to refer to this as a knowledge-based planning phase in data management (see Fig. 9.4). When use cases are applied to generate requirements, the knowledge-oriented use case analysis also provides the requirements for the development of EDM methods and advanced EDM features. By providing knowledge about the relationships between business process model, knowledge processes and the operational data management, one step is made towards an EDM method development. This development is executed on the basis of knowledge-oriented investigations and can be seen as the final step of a multi-stage concept for integrating knowledge management into the product development process.

The previous section described the close connection between knowledge processes and data management, which provides the starting point for some purposes. Not only can it be used for mapping the current processes, but it can also contribute to the development of efficient methods for engineering data management. If EDM is seen as the central knowledge base for a variety of knowledge-intensive actions in the development process, then the knowledge-oriented view of data management must be a good starting point for method development. This holistic approach, which also connects to the business process model, establishes a solid foundation and makes this style of developing operations and methods a promising tool.

9.2 Integrated CAD Data Management in Automotive Engineering

This section deals with CAD data management, which is an essential topic in the area of engineering data management. The aim of the case study in this section was to develop a concept for integrated CAD data management. The concept is process-oriented and seeks to establish requirements more clearly by using the *knowledge point of view* by restructuring the CAD knowledge base. In the end, this increases the quality of CAD data management. The project investigates the integration of CAD data management with DMU and VMU in the product development process and basically deals with CAD data management, CAD model description, CAD data quality, CAD workflow management, CAD data monitoring and the control of CAD data in the development process. Furthermore, a more detailed representation of the CAD process is provided before, with a focus on the documentation of product data. Data which had previously been mapped in drawings is now transferred to the digital products. The aim is to design a master plan for integrated data management in CAD applications. All of the following design approaches for CAD data management are based on the procedure for process analysis and the reconstruction of the knowledge base that is presented in Sect. 8.1.6.

9.2.1 Challenges Related to the Topic

Companies usually establish CAD data management based on certain rules known as CAD standards. These standards do not sufficiently take into account the way in which demands placed on the CAD data management change during the product development. There is usually a rough plan for data management activities at project milestones, which makes it necessary to plan the product development process in terms of data management as well. Specific project requirements for data management or specific data management between two processes are not usually taken into account. The analysis of data management operations in terms of knowledge is even less incorporated. The following systematic procedure is used below to demonstrate an efficient and effective method of EDM design:

- Process analysis by application of a process matrix
- Reconstruction of the knowledge and databases
- Transformation of knowledge operations into data management operations
- Derivation of appropriate design approaches for CAD and EDM

The process-oriented data management must be starting with the begin of the development process. Only if this process is described in detail and all interfaces are Defined, the CAD data requirements for individual process steps can be defined.

9.2.2 Concept of Integrated CAD Data Management

The results from these case studies can be combined to form the concept of integrated CAD data management. EDMS supports basically all of these requirements via standard function modules and is thus the ideal integration platform. Figure 9.5 shows the functional modules:

- CAD scheduling (project orientation, data roadmap)
- CAD geometry reference
- CAD data quality, progress and maturity
- CAD workflow (process orientation)
- CAD data monitoring

These functional modules are integrated in CAD data management and are embedded between the project milestones and schedule data.

The following sections discuss the individual functional modules of integrated CAD data management in more detail.

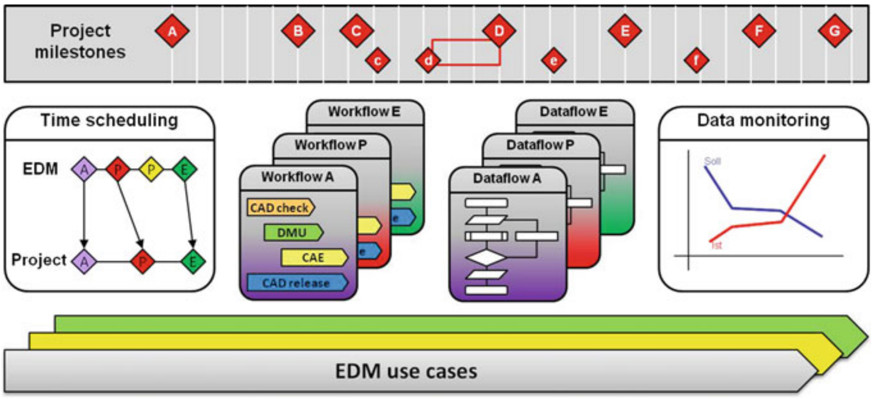


Fig. 9.5 Concept of integrated CAD data management

9.2.3 *CAD Scheduling*

As a basis for CAD data management planning, it is necessary to align key milestones in the project and to retrieve the resulting demands for CAD data management. These demands provide the first reference points for the planning of data management operations and the necessary adjustments to EDM methods and systems. Project-specific needs always require adjustments to the milestones. For example, some milestones may be omitted, others might be added, or milestone events and lines may be altered or shifted. Possible reasons for such changes include adapting to predefined quality gates. The time schedule also provides the basis for the creation of the data roadmap.

Concept of Data Roadmap

The data roadmap concept is a management tool for CAD data management. It is divided into four levels and is temporally linked to the project schedule.

Figure 9.6 schematically shows such a data roadmap. On the first level, a link with the project time schedule is established, in which the major project milestones and additional required quality gates are defined in terms of CAD data management. On the second level, the development of the virtual product is mapped, which can be done by using the complete body of CAD data. This takes place at a designated time and corresponds to the requirements of the geometry reference. On the third level, the documentation of the necessary EDM milestones, which required for data management, is created. This step is performed more precisely, since it will control the data management operations between different specialist processes. Finally, on the fourth level, the EDM use cases are mapped, from which the specific data management activities can be derived.

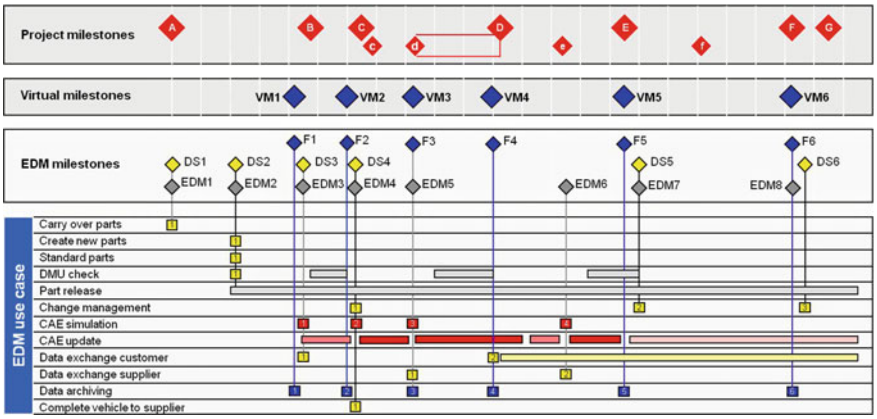


Fig. 9.6 The concept data roadmap in the product development process

The data roadmap contains not only the dates of data management, but also references to the respective descriptions of data content (e.g. quality, configuration, size). The data roadmap also shows the contents which are defined by the geometry-maturity reference. When producing this graphic illustration, the challenge is to show the temporal parallelism of processes in a manageable way without losing information. The arrows (i.e. connections) principally show that data flows from a data source to a data sink, while no statement is made about the granularity of contents. The following example will clarify the issue of granularity. The simulation of both material strength and multi-body simulation (MBS) requires information for validation. This information includes data from the product geometry and its structure. In order to perform its dynamics simulation (e.g. an analysis of a suspension), MBS only needs connection point coordinates of the components involved. The exact geometrical shapes of the semi axel, wheel carrier, stabilizers, etc. are not relevant in this case. The outputs of MBS are spatial curves, which define the impact of involved components on the driving behavior. The only feedback to the design process are potential changes of coordinates or envelope curves, which represent the positioning of particular components. In contrast to this, the strength calculations and their derivations require consistent and detailed geometries, which are then processed in the appropriate systems and enriched with additional information. This example shows that MBS is a simulation discipline that performs several and meaningful computations with reduced, specific information (e.g. in the area of axle analysis or vehicle dynamic simulations). It terms of the data roadmap, it is important to provide reduced data for specific simulation processes.

During the progress of development, the data granularity increases. Within virtual development, the virtual product models are able to describe the real-life models with a high accuracy, but it is impossible to achieve a 100 % formulation of all characteristics. For this reason, it makes sense to concentrate on the description of main characteristics (e.g. geometry, structure, material and connecting technology), which can be introduced as product data objects in the data roadmap.

Expected Benefits of a Data Roadmap

- Raising employee awareness: If vehicle project teams, led by a data manager, embrace the data roadmap concept and understand it's benefit, this tool can become a living procedure.
- Reduction of relatively meaningless validation operations: Partial simulations are included in the schedules of some projects that seem to make no sense at that particular time, which sometimes means simulations must be repeated at a later date.
- Less problem solving resulting from missing data: If a team agrees that a validation is not possible at the moment because data is missing, then it automatically reduces the number of unsolved problem points. Clashes within a project are reduced, and employees can concentrate on important things again.

9.2.4 A Concept of Geometry Reference

CAD models are created in the CAD author system, and data is provided via EDMS using an integration platform. The methods of data storage and data transfer depend on the depth of the interface between CAD and EDMS. Although the transfer of particular models is usually less problematic, more significant barriers are often revealed when it comes to the exchange of structural information. The goal is to provide an explicit geometry reference that contains the geometry states necessary for the functional supply and production-related supply of specialist teams.

Figure 9.7 shows the reference geometry approach, which can be described in three dimensions:

- The *product configuration* essentially defines the composition of the vehicle product set. Here, drivetrain variants or feature variations are mapped, for example.
- The *product structure* shows different views of the virtual product (e.g. DMU, CAE, assembly).
- The product development dimension maps the various degrees of maturity along the product development process. These maturity degrees are temporally controlled by milestones or generations.

The figure also shows that the application of reference geometry once again requires preparatory and operational data management operations in the project. This is mainly evident in design, simulation and the operational data management, with its various applications.

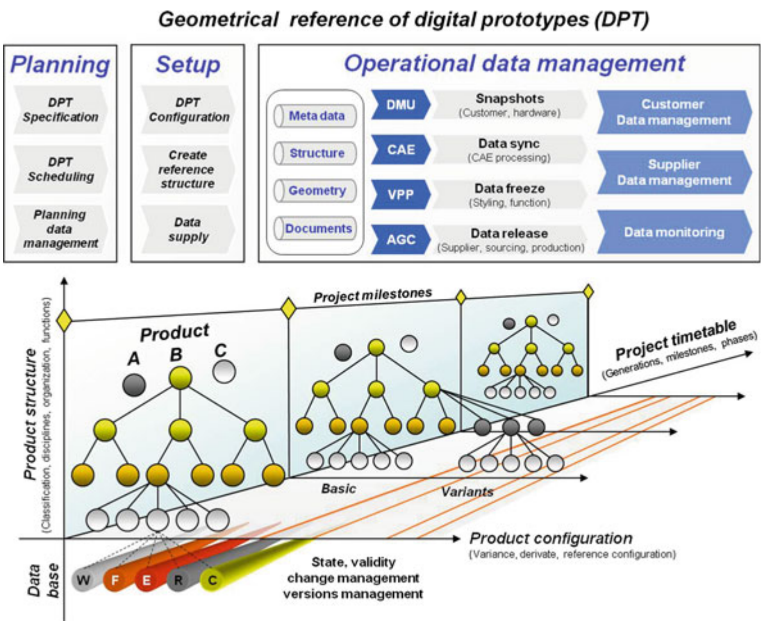

Geometrical reference of digital prototypes (DPT)
Planning
DPT Specification
DPT Scheduling
Planning data management
Setup
DPT Configuration
Create reference structure
Data supply
Operational data management
Meta data
Structure
Geometry
Documents
DMU
CAE
VPP
AGC
Snapshots (Customer, hardware)
Data sync (CAE processing)
Data freeze (Styling, function)
Data release (Supplier, sourcing, production)
Customer Data management
Supplier Data management
Data monitoring
Product structure (Classification, disciplines, organization, functions)
Project milestones
Product
A B C
Project timetable (Generations, milestones, phases)
Basic
Variants
Product configuration (Variance, derivate, reference configuration)
Data base
W F E R C
State, validity change management versions management

Fig. 9.7 Concept of geometry reference for a digital product

9.2.5 CAD Data Quality, Progress and Maturity

During the development process of a mechanical product, which can begin with a base body and continues until all the necessary design elements have been included, the CAD model becomes more and more detailed. This is recorded in the defined maturity level model. The data roadmap also specifies which information is needed by particular departments (e.g. DMU department). Ideally, these data bundles represent all of the data which required as part of a product life cycle in each functional unit.

However, where the data must come from, what quality level it must have, and when it must be made available is not defined at that point. In order to supply data for a particular geometry reference concerning the data roadmap under consideration of quality aspects, the product is described with a three-dimensional maturity level model. These CAD documents are also provided for certain applications on the basis of this model. This describes the development status of a particular model and all design specifications, such as maturity, weld points or tolerances. If the conditions necessary for all maturity levels are defined, then every model can be assigned to a particular maturity level.

The Three-Dimensional Maturity Level Model for CAD Documents

Figure 9.8 shows an approach of a three-dimensional maturity level model for CAD documents in the EDM system, which can be divided into the following segments:

- EDM maturity - usability maturity of the geometry
- EDM quality - what requirements are there for the CAD geometry, and which ones are fulfilled?
- EDM status - what obligations can the process hand over to the CAD data?

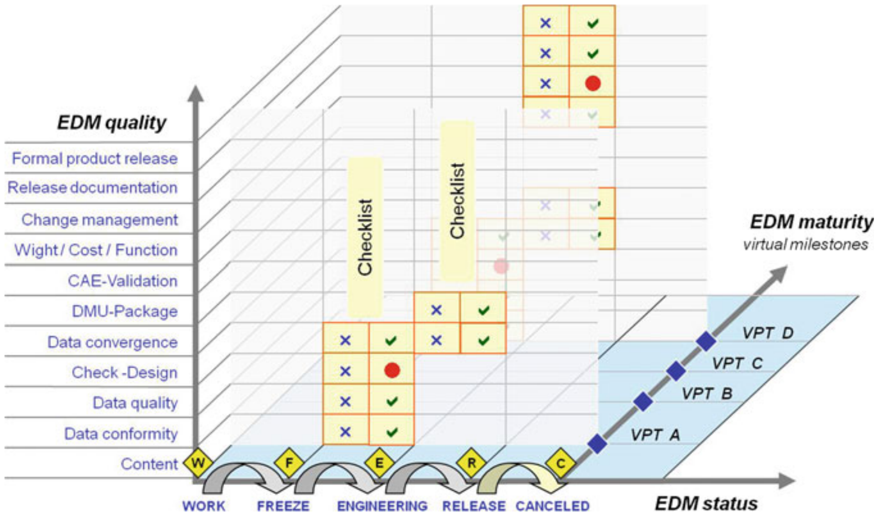


Fig. 9.8 The three-dimensional maturity level model for CAD documents

The maturity model, which depends on the defined geometry reference in the data roadmap, influences the generic EDM workflow. In this context, the CAD data quality has a particular meaning that can be defined by the following aspects:

- Quantitative quality: In terms of data, this indicates the degree of fulfillment (up to 100 %) in relation to the geometry reference.
- Quality of conformity/administrative quality: This value corresponds to the system-technical and/or organizational requirements and says nothing about the quality of content.
- Content quality: This measure indicates the consistency of selected data in relation to the application, or if the results of a previous test were correct.
- Quality chronology: When considering the individual kinds of quality, it is useful to maintain a logical sequence. For example, it does not make much sense to validate the scope (quantitative quality) if the stability of the period of consideration has not been previously investigated (quality of availability).

9.2.6 Generic EDM Workflow for CAD Data Management

Workflows have to be established in order to deal with the logistical requirements of the data roadmap. These workflows have to be adjusted during the project start-up phase and have to be adapted to the particular project requirements. The workflows should be derived from the PDP so they will reflect interactions between departments. They have to be linked to data registers for each milestone.

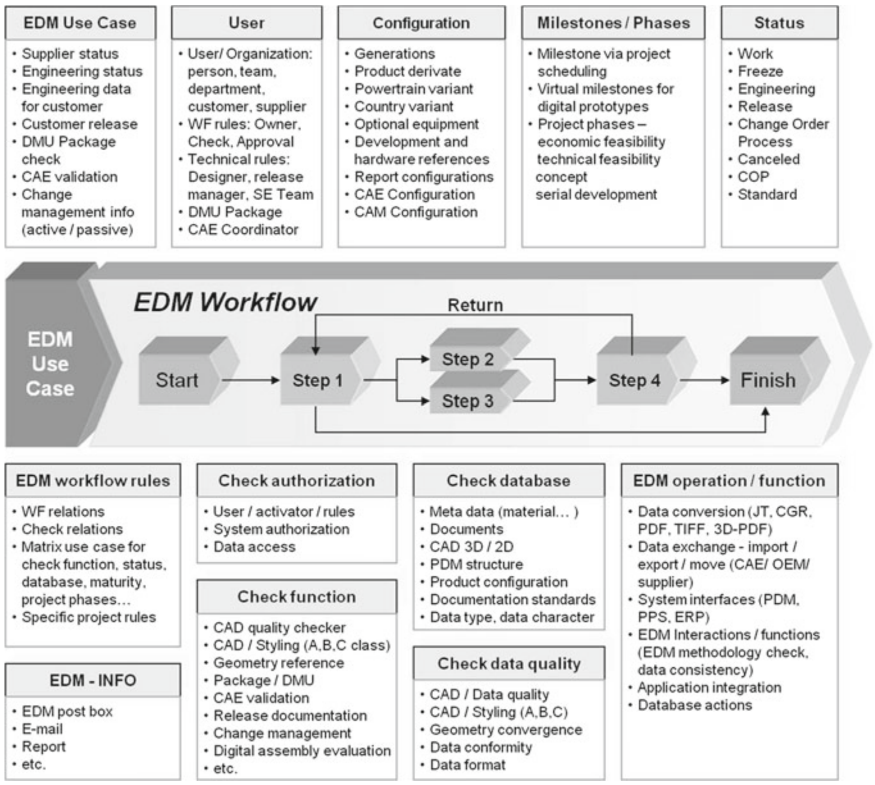


Fig. 9.9 Generic EDM workflow for CAD documents

Figure 9.9 shows the generic structure of an EDM workflow to control the CAD data management system and its influencing factors. The central element is based on a use case of the EDM with its workflow process steps shown. The workflow has a defined start, which is provided with a trigger. The workflow consists of several steps with branches and returns and is processed in a system-controlled manner. Within the workflow, the process-relevant aspects of CAD data management are requested, tested, and modified. In addition, further EDM activities and information flows for product or process data are derived. The workflow also takes the pre-definitions of the *data roadmap* into consideration.

9.2.7 Data Monitoring

The data content is monitored at least for every milestone. It would seem to be practical to appoint a data manager, who has the complete overview and reports to the project management.

Data Management Operations

- **System:** Basically, there has to be a system environment available so that data can flow and can be exchanged. Without this, data monitoring is meaningless.
- **Quantity:** The complexity has to be monitored and guided during the time of investigation.
- **Operative:** The operative activities have to be depicted, whereby the question is which data has to be delivered by which providers at which times.
- **Time:** The system's cycle time is investigated to deals with computational power.
- **Workflow:** Includes classification number, workflow-status, number of applications etc.

To document and report the progress of a development process, it is necessary to define quantifiable classification numbers. These numbers should be easy to derive and to project. Typical criteria that are accessed within CAD monitoring include:

- Data currency
- Data availability
- Data quality
- Data consistency
- Level of integration (use, configuration)

Although CAD data monitoring is not a standard functional module of EDMS, it would make sense to implement CAD monitoring in existing EDM environments due to the broadly available database.

9.3 A Parametric-Associative Concept Model for Initial Vehicle Development

In automotive development, the main characteristics of a new car model are determined in the initial phases of product generation. Since the conceptual definition has to consider various, partially conflicting boundary conditions and requirements, the resulting vehicle concept always represents the outcome of intensive optimization processes. This multidisciplinary optimization requires a high flexibility from the tools, methods and processes applied. In the present approach, the integration of CAD and CAE disciplines results in an easy-to-handle tool that helps support and maintain the entire conceptual full-vehicle process. This starts with initial ideas, sketches and specifications, continues with the 3D CAD representation of boundary conditions, the formation of preliminary and ultimately final vehicle concept geometries, and finally provides functional layout procedures for weight estimation, propulsion layout and driving dynamics behavior. The following sections include an introduction of the working fields involved in conceptual full-vehicle development and describe the methods of resolution developed for the integrated approach. The integration of different engineering disciplines into one comprehensive software model provides significant potential for the process-oriented combination of knowledge management and engineering data management. The method of application is focused on the specific corresponding processes. This requires the implementation of complex data models and data management strategies, while maintaining a focus on product knowledge throughout the entire development cycle.

9.3.1 Requirements for Automotive Concept Phases

Automotive full-vehicle development processes start with the definition of product specifications, an initial functional layout and the general vehicle package configuration. An initial full-vehicle layout is generated through a combination of initial styling proposals and technology concepts. In particular, the challenge for developing innovative technologies is to create a highly flexible 3D CAD model that can help deal with numerous package variants and can enable an efficient optimization. This optimization must take into account legislative guidelines, styling and technical functionalities, and must help to address the issues of drivetrain configurations, energy storage systems, vehicle driving characteristics and much more (see Sect. 1.2 for a detailed description of the entire automotive development cycle). Integrated development strategies, including parametric-associative geometry creation, interlinked with simulation and computation procedures, have to be applied to fulfill the requirements of multidisciplinary work packages in early development phases.

Modern development processes generate knowledge about various product characteristics in early process stages, which helps to reduce the engineering effort and risk in subsequent (cost-intensive) sequences. This requires powerful and user-friendly methods and strategies for conceptual automotive development, which help to meet the various, highly integrated demands during these initial engineering procedures. Figure 9.10 shows an example of the workflow of working areas in an automotive concept phase.

Requirements for automotive concept phases:

- Quick identification of relevant facts for design and packaging
- Ergonomic viewpoints (e.g. passenger seating position, accessibility of control elements, entrance areas, car-boot characteristics, efficiencies of mirrors)
- Simple implementation of design data (exterior and interior surface models)
- Implementation of existing DMU components (e.g. engine and drivetrain, interior and exterior components, human models)

- Easy classification of various legislative influences (e.g. active and passive crash safety, visual requirements, lighting, space requirements for passengers and luggage)
- Weight management, axle load distribution
- Integration of safety technologies
- Functional layout of propulsion and energy storage technology
- Chassis and suspension pre-dimensioning
- Discussions with customers, development partners and designers
- Comparison of different competing/benchmark products based on simple data sources (photos, drawings and simplified 3D data)

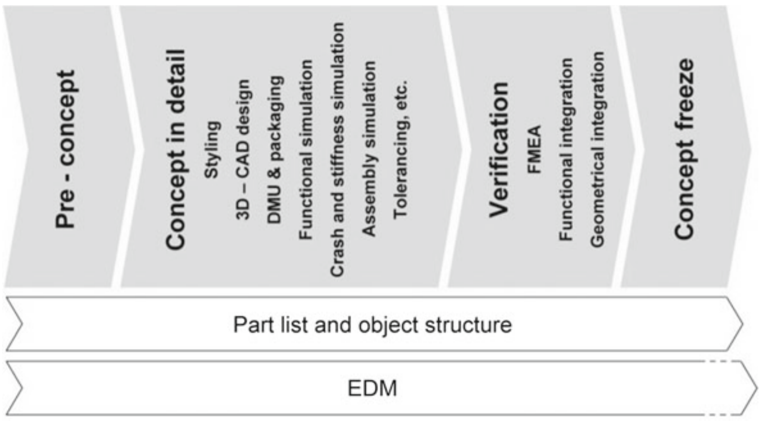


Fig. 9.10 Working areas in sample automotive concept phase

During the initial design and layout phase of a new car, an optimal interaction between the styling process, technical engineering operations, economic pre-calculation and customer-related inputs represents an important factor for an efficient concept generation. The concept phase of automotive development has to consider numerous factors that influence the definition of a full-vehicle concept model.

In virtual development, this vehicle model is displayed in different ways to account for the product structure list, the conceptual vehicle cost structure, weight and mass lists, finite element meshes, styling models, and of course a 3D CAD model structure. All of these representations of a concept vehicle serve for particular fields of development and are generated and maintained in different departments. In most development processes, these subareas are treated more or less separately, and the data transfer between the disciplines is focused on the tasks in each section. This procedure leads to an opaque full-vehicle development process, which has to be monitored carefully and with a significant organizational effort.

Figure 9.11 shows the most important factors that influence an integrated 3D CAD model in the early development phase. In the present approach, the 3D CAD vehicle geometry is placed in the middle of different working areas, thereby linking the disciplines and serving as a data collector and representation model for the entire development progress. At the beginning, initial styling studies are implemented into the geometry model to enable an adjustment of the targeted geometry data (e.g. vehicle dimensions) with the styling proposals. This initial styling information can be integrated in the form of simple 2D sketches, drawings or initial scan data of clay models. In later phases, computer-aided exterior and interior styling surfaces are imported into the 3D CAD model to facilitate detailed studies of the vehicle styling in terms of legislation-based boundaries (crash and pedestrian safety regulations), ergonomic tasks and dimensional viewpoints.

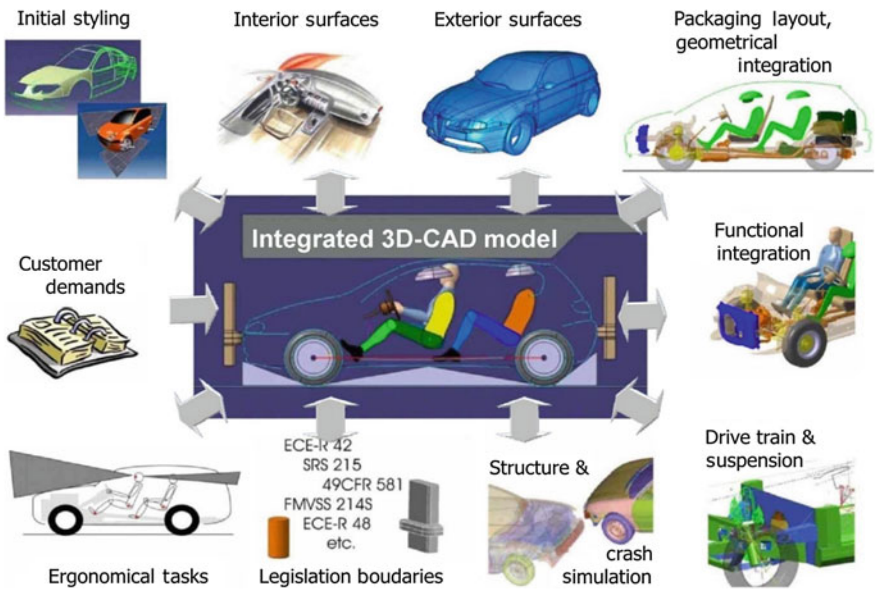


Fig. 9.11 Factors that influence an integrated 3D CAD model in the early development phase [1]

Applying a complete parametric CAD-model already in the initial development phase enables a significantly enhanced integration of concept model and serial development status. One challenge is the efficient representation of the comprehensive functionalities of full-vehicle models, including the required flexibility for variation and optimization loops. The parametric-associative structure of the present vehicle model requires the consideration of all possible modifications to achieve a stable vehicle model. One important factor thereby is the integration of EDM strategies to exploit the full potential of the integrated approach.

One main procedure in conceptual development deals with the packaging layout and geometrical integration. These processes address the geometrical arrangement of vehicle components, as well as the definition of the passenger space and luggage compartment. The geometrical investigations are performed within the boundaries of targeted comfort and space requirements, vehicle dimension definitions and technical module space requirements. In this phase, the virtual concept vehicle architecture is defined, which often includes a combination of new geometries and existing components from predecessor models (e.g. drivetrain, chassis, fuel tank, air-conditioning system). In addition, the packaging requirements of alternative drive train concepts have to be considered. The influence of gas tanks, electric motors and batteries, hybrid engine configurations and other technologies on conventional and future vehicle arrangements is investigated.

The basis for the packaging layout and geometrical integration is a 3D CAD model structure, which contains all geometry data as well as simultaneous interfaces to several disciplines. DMU processes (e.g. component positioning, clash and distance analysis) and sectioning functionalities support the concept phase and the pre-development phase significantly. In state-of-the-art full-vehicle development, the geometrical integration starts with the definition of passenger requirements. The important elements here are the entrance area, seating position, head and elbow clearance and the accessibility of control units. The car type has a significant influence on the seating position. There is a notable difference between the development of a sports car and that of an SUV. In addition, the target markets influence passenger space requirements. A car for the Indian market has to meet different room and luggage targets than a car for the European market, for example. All of these factors have to be considered from the early development phase on and define the vehicle characteristics in general. Figure 9.12 shows a dimension concept of a modern car. A large number of dimensions are visible, which show the interaction of technical components and ergonomic viewpoints. The main vehicle dimensions are prescribed in several standards. The most important standard for European car manufacturer is the GCIE standard (Global Car Manufacturers Information Exchange Group), which defines a long list of ergonomic and technical car dimensions [2].

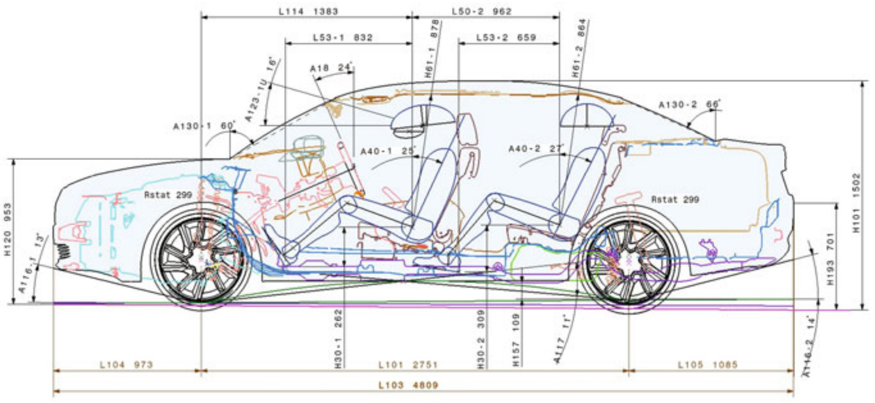


Fig. 9.12 Full-vehicle layout of a modern car with about 30 selected ergonomic and body dimensions

The module structure and crash simulation deals with the definition of an initial vehicle body arrangement. In this development phase, the vehicle body structure is generated with consideration given to load and durability requirements, as well as crash and stiffness demands. The body layout has to consider the applied frame configurations, materials and connecting technologies. Several factors (i.e. the vehicle category, targeted vehicle mass, production quantity, cost influences and platform strategies) influence the choice between conventional steel-based concepts, aluminum space frames or combined structures. During the development process, the body-structure model is generated in 3D CAD with ever increasing accuracy. In different steps of maturity, the geometry data are transferred into a finite element calculation program to enable an evaluation under predefined load conditions. The results of the calculation directly drive a subsequent geometry modification.

The implementation of parametric-controlled interfaces to connect the 3D CAD model with the generation of meshes and other relevant data is important. This phase is characterized by high flexibility demands on the geometry model. The degree of detail, which is relatively low at the beginning, increases during the development process. Weld spot and connection technology definition as a function of applied materials and technologies are defined. At the end of conceptual engineering processes, a verified body structure provides the basis for series development. Figure 9.13 shows an example of an early car body structure, ready for simulation.

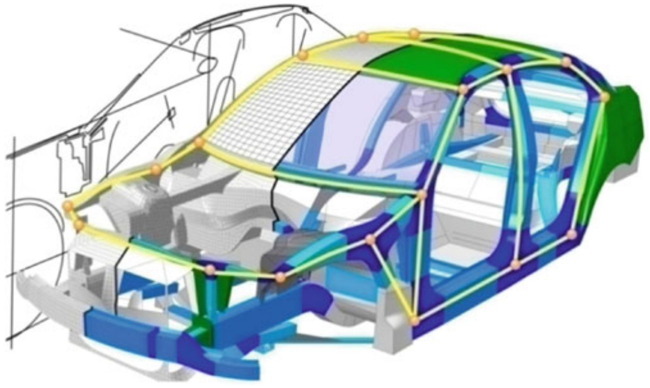


Fig. 9.13 Conceptual vehicle body structure [3]

Drivetrain components have to be inserted to find space in the vehicle environment. In the early concept phase, different power train technologies are evaluated in terms of the targeted driving performance and fuel consumption. The ability of selected engines to fulfill the requirements of the new car concept has to be approved. In the case of hybrid concepts, the configuration of internal combustion engines, electric motors, batteries and control units has to be developed. The outline figures of engine and transmission components are delivered as DMU models to enable initial placement studies. In most cases, these components are derived from other models or from specific development departments. Besides the geometrical aspects of the drive train unit, the functional aspects of future exhaust emission legislation, fuel consumption and performance targets in the implementation of hybrid concepts and alternative engines have to be considered. This application is mainly done concurrently in the responsible engine development department, supported by linked simulation and optimization processes.

In addition, the initial layout of vehicle components is supported by the functional integration of several modules. Whereas the drive unit is developed in specific departments and delivered as a closed unit, the integration of several components into the vehicle setup is a part of the full-vehicle development process. An efficient data transfer between the participating engineering departments is an important factor of success during the entire conceptual development. All these factors have to be considered during the conceptual layout of a new car, whereby the legislation varies between different markets.

The ergonomic layout of a car is closely related to these safety concerns. In the concept phase, the targeted vehicle class provides the principle layout of the seating position in each row, the number of passengers and the arrangement of luggage space. Besides these principal decisions, brand-specific boundaries influence the ergonomic layout. The seating position and the passenger space are important physical factors. All of these ergonomic elements influence the layout of the door entrance area, the opening angle, the handling of the trunk cover and other factors.

Specific customer demands are related to the vehicle class or the specific brand and can be influenced by market peculiarities. In general, cars developed for global markets have to consider a wide variety of legislations and regulations (e.g. crash and safety, lighting, exhaust emissions, performances and weight, assurance grading), while cars built for a specific market can be developed by focusing on the requirements of the target market.

9.3.2 Integrated Approach to Virtual Concept Development

The present approach includes the implementation of an integrated vehicle model to improve conceptual automotive development. All of the processes in the concept and pre-development phases are linked to the full-vehicle model in order to connect the different fields of development (e.g. geometry, legislation, functional aspects). This integrated full-vehicle model covers geometry data, functional data and different interfaces for data transfer to connected processes. A powerful database contains a list of all of the information relevant for the control of geometry models and for other simulation and calculation procedures. The product visualization is performed by an integrated, parameterized 3D CAD full-vehicle model, which is controlled by the main database. The 3D CAD geometry model serves as a basis for several geometry-related investigations and also as a demonstration unit for the entire concept model. A bi-directional connection to the database enables a parametric geometry control, as well as a tracking procedure and data archiving. This virtual concept model is connected to several supplementary tools and procedures via data interfaces. Thus, the geometry model can be understood as a display of the overall vehicle concept model.

Overview of Past and Current Development in the Area of Conceptual Full-Vehicle Design Using Computer-Aided Methods

Due to the importance of conceptual vehicle layout, this topic has been intensively explored by automotive manufacturers, universities and research institutes. Since the beginning of the 1980s, virtual development methods have emerged in the automotive industry and have also been applied in conceptual vehicle design. The following section includes a selection of related research work and publications. In addition, the integrated approach presented in the current chapter is delimited in relation to the general state of the art.

Hänschke published initial findings for the conceptual representation of car models in 1986 [5]. In a co-operation between the Technische Universität Berlin and a German car manufacturer, the CAD-based tool AURORA (Automobiltechnisches, anwenderorientiertes Entwurfssystem zur Optimierung der rechnergestützten Auslegung) was developed, which enabled the creation of simplified geometrical elements for the representation of vehicle outer contour and inner geometries for packaging-relevant investigations. In this way, it was possible to generate a variable vehicle model within a CAD environment to represent the space requirements of components as well as of general, legislation-based boundary conditions. As a further development of AURORA, Heinke and Deter introduced a parametric system for the representation of conceptual vehicle geometries in the 1990s [6, 7]. The geometrical representation was based on wireframe elements, which were composed using a skeleton model. Once created for a specific car, the model enabled parametric modifications of the geometries, which supported variant studies and optimization loops. By exporting points and coordinates, parallel simulation procedures (MBS, FEM) were supplied. These initial findings for the parametric creation and representation of a simplified full-vehicle model served as a basis for several further approaches and software solutions in the ensuing years, e.g. [8].

Bulheller established an integrated product data model, which contained the product geometry, technological information (e.g. materials, tolerances) and information about production planning in one structure in 1994 [9]. This integrated product model served as a basis for product representation and as a data source for different development processes, thereby enabling simultaneous engineering approaches, including a linking of CAD and CAM.

Drawing on the concepts of Heinke [6] and Deter [7], Rasenack developed a method for the parametric control of vehicle packaging geometries [10]. The method included the creation of simplified, functional geometries of engine and drivetrain packaging, as well as the implementation of wheel envelope for initial investigations in conceptual full-vehicle development. Besides the pure geometrical representation, he developed initial methods for the support of geometry-based packaging optimization loops.

Parametric design methods significantly enhanced the possibilities for conceptual vehicle geometry creation. Besides the geometrical representation, the design model serves as a basis for simulation procedures, such as FEM calculation. Focusing on the requirements for conceptual layout of a vehicle structure in terms of strength, stiffness and NVH characteristics, Zimmer has been engaged since the middle 1980s in the development of a FEM-based platform for the representation and simulation of vehicle body structures [8]. Unlike conventional 3D CAD-model-based approaches, Zimmer's method is based on the parametric generation of simplified geometry models, which are created to be used directly for subsequent meshing procedures for the supply of FEM simulation.

The implementation of parametric 3D CAD systems into full-vehicle development processes at the beginning of this century enabled the creation of new integrated methods in vehicle development. Several research projects were carried out between 2000 and 2005, which explored the development of enhanced design methods for automotive concept phases to tap the full potential of parametric-associative design. The following three doctoral theses represent a selection of the comprehensive research work in this area. In 2001, Gessner investigated the possibilities for creating geometrical features which represent predefined models for packaging studies [11]. The introduction of parametric geometries for the representation of packaging-relevant factors, such as the view areas of passenger, head clearance, etc., supported the creation of new vehicle models, especially in early development phases. Next, Forsen combined the possibilities of parametric design and associative methods for the efficient creation of body in white components in automotive development in his 2003 doctoral thesis [12]. Using examples from automotive applications, he introduced and evaluated different possibilities of parametric-associative design, including systematic approaches for the improvement of data quality and knowledge-based reuse of template models. In addition, Forsen pointed out the importance of hierarchical structuring of parametric CAD models and of dependencies in complex models. At the same time, Böhme investigated the classification of parameters and the data model structure for the efficient combination of package-relevant components. Furthermore, he introduced methods for the definition of knot geometries within a CAD model for the supply of CAD-external simulation and variant studies [3].

In the middle of the last decade, a comprehensive approach for the support of conceptual vehicle layout was developed by a commercial software supplier in a joint venture with several research institutes and German automotive manufacturers. The vehicle layout tool CAVA (CATIA V5 automotive extensions vehicle architecture) is embedded in a standard 3D CAD software package [13] as an integrated CAA (component application architecture) module [14]. The tool supports the general layout development of a new car by providing CAD surfaces as boundary conditions of vehicle main dimensions, geometrical representation of selected legislations, and passenger ergonomics basics, as well as vision wiper and mirror layout. To provide up-to-date information, the software is revised continuously with the current laws and trends.

In recent years, several automotive manufacturers have developed 3D CAD-based tools and methods to support their conceptual vehicle development, following front-loading and simultaneous engineering approaches. These solutions are specifically configured for the requirements in each company, and they often include specific features and functionalities. As an example of manufacturer-specific methods, Tesch introduced an approach for the parametric derivation of new vehicle concepts based on an existing model structure (i.e. from former models) at BMW [15]. The approach is based on a simplified, parametric vehicle model, which mainly consists of the underbody structure. In this structure, several components from existing models are implemented (e.g. drivetrain, suspension). This conceptual configuration serves for derivate studies and packaging optimization.

Nikol represents a second example of manufacturer-related conceptual design methods [16]. In his publication, he introduced a conceptual full-vehicle design method, which was developed by Audi. In this approach, a new model architecture is always based on an existing design of a former car model. This provides a high level of geometrical detail, including the ability to utilize knowledge from the serial design of former models. However, this method restricts the degree of freedom in terms of creating completely new vehicle types and architectures.

Since 2002, this author has been involved in the development of a method for the integrated consideration of different influencing factors in conceptual automotive development. The project started with the parametric creation of simplified geometries within a 3D CAD environment to enable an automated representation of vehicle main dimensions. In the course of continuous expansion and improvement, several additional modules and functionalities have been implemented, so that the current version covers a broad field of applications for the early definition of car concepts. As an additional field of investigation, the functional vehicle layout has become an important aspect of vehicle layout procedures. The development of methods that would enable the connection of a geometrical vehicle representation and its boundary conditions, and the calculation and simulation procedures which are required for the estimation of driving characteristics, fuel consumption and others was a significant challenge. The current version of the tool is used by both an international automotive engineering and component supplier and a German car manufacturer. The close co-operation with these companies provides an effective project environment for the further development of tools and methods which will support the comprehensive working fields in conceptual automotive development.

Architecture of the Integrated Approach

In the present approach, the virtual car model of the first step consists of database-controlled axel and wheel dimensions, wheel base, car outer dimensions and configurations of passenger seat points (SgRP). The implementation of additional components expands the virtual car model through the positioning and evaluation of both externally controlled components (e.g. variable crash barriers) and non-parameterized components (e.g. simplified 3D models of drivetrain, chassis or interior modules). Functional aspects are incorporated using additional modules, which enable early calculation and simulation processes. In this way, the integrated geometry model is linked with a mass calculation module, the calculation of the center of gravity and an estimation procedure for the required propulsion performance (by considering targeted acceleration and vehicle speed values). In addition, the model can be used to estimate fuel consumption (or energy consumption in case of an electric drive line) in standardized driving cycles of a given or assumed drivetrain configuration. An implemented single-track model of the vehicle supports the assessment and evaluation of driving dynamic characteristics.

The parameters that influence the early development phase predefine the structure of an integrated concept vehicle, which forms the core of a supporting tool for all concept phases of automotive development projects. This concept tool consists of a database, which includes a logical order of data modules, and a linked 3D CAD model comprising seven main sections (Fig. 9.14). An externally controlled 3D CAD-based development strategy simplifies the conceptual design and packaging process by using an external data pool, which controls the parameterized model. The external data collector includes a key collection of relevant dimensions, positions and/or ergonomic viewpoints. The data sheets also permit modifications and variant studies of the virtual car by project partners who are not primarily specialized in the applied 3D CAD design software, thereby providing direct access to the relevant information on vehicle geometry.

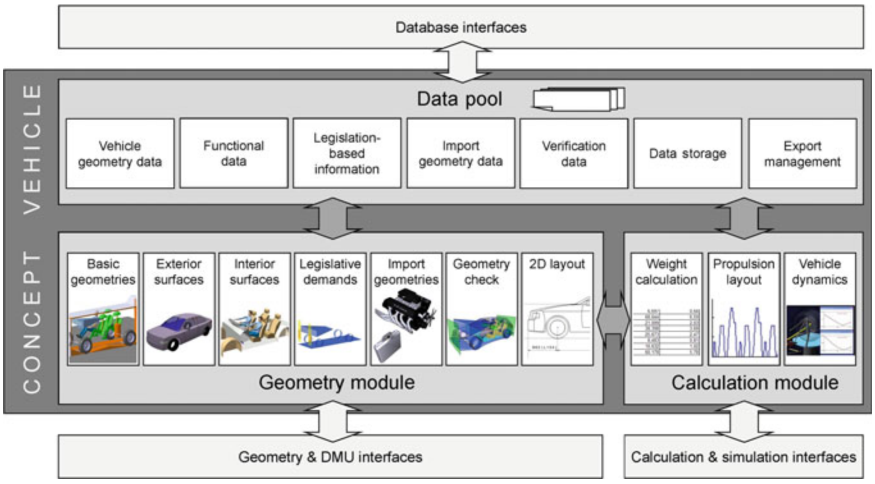


Fig. 9.14 Principle architecture of the concept tool

The data pool architecture consists of seven main sections, in which all main geometries are defined in accordance with the GCIE standardization. Data for functional investigations are handled in a specific module, which also includes mathematical calculations and relations. The legal-based tasks include crash and safety, visual regulations, lighting equipment, ergonomic data and other governmentally regulated areas. Data from standardized import geometries control the dimensioning and positioning of additional simplified geometry models, such as drivetrain components, suspension, luggage elements, and interior models. The verification data sheet enables a definition of control mechanism, different check geometries and the generation of predefined 2D sections. A data storage system supports the creation of variant studies, the handling of benchmark data and release archiving throughout the entire development process. An efficient data exchange with the 3D CAD system and different simulation programs is performed with the help of an export management system.

The virtual vehicle model in the geometry module integrates several sections, which combine elements to form a logical unit with cross-module dependencies and parameters controlled by the external data pool for a centralized input and editing. The modular construction and open architecture provide the required flexibility and allow for the implementation of additional components. The parametric 3D CAD model includes the visualization of the following components:

- Basic vehicle geometries, which consider car dimensions in relation to the GCIE regulation
- Exterior surfaces (simplified, engineering-based surfaces considering the basic layout, legislation requirements and packaging-relevant facts)
- Interior surfaces (simplified, engineering-based surfaces considering ergonomic viewpoints, packaging-relevant facts, legislation boundaries (e.g. view, mirrors))
- Check geometries for the evaluation and verification of the virtual concept vehicle
- Import geometries to enable an implementation of predefined automotive components (e.g. engine, drivetrain, suspension)
- 2D sectioning (possibility of automated sectioning for release definitions and a basis for technical discussions both in DMU tasks and in the field of detailed engineering)

A separate calculation module is connected with the data pool and with the geometry module. This calculation module consists of different functional-oriented sections, which support full-vehicle-related layout during conceptual development. Thus, the integrated and highly flexible formulas and simulation models support early weight estimation, different power train layout procedures and primary vehicle dynamics calculations. A bi-directional information flow from the data pool to the parameters of the virtual car model and from the CAD system back to the data pool efficiently supports geometrical conceptual working steps. An advanced parameter structure organizes the input of relevant data to each component of the tool. Besides the information transfer from the input and data organizing tool to the 3D CAD system, a reverse data exchange from the 3D CAD system to the data pool guarantees an efficient workflow during the entire development processes.

The user-friendly disentanglement of the data pool and the actual geometrical environment offers the advantage of a clearly arranged overview of all decisive characteristics that are necessary for the model description. Furthermore, the use of spreadsheets opens up the possibility of adopting standardized parameters directly from automotive tables. Taking this into account, the preparation of the collected

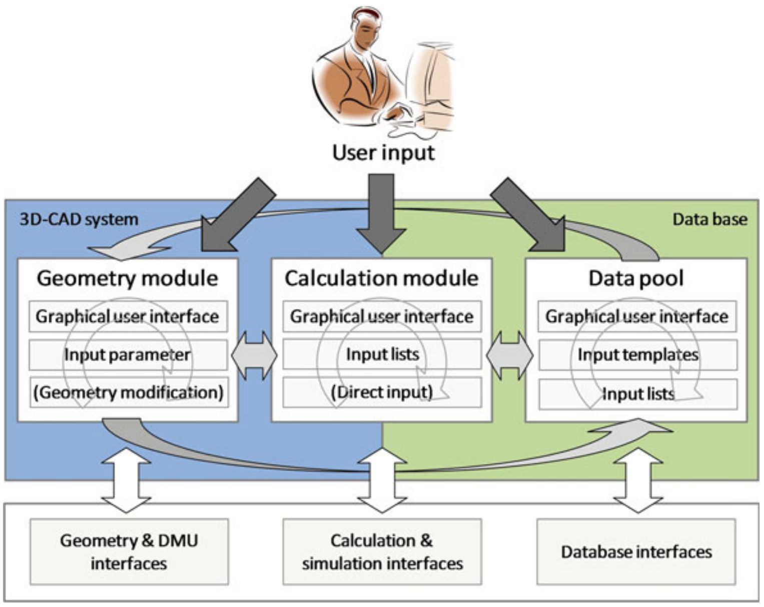


Fig. 9.15 User operation and data flow [1]

model input (e.g. databases, established standards or customer-specific information) takes place in the form of interconnected tables. The input templates include information concerning the positioning coordinate system, wheel dimensions, seat orientation of the passengers in the third row, load conditions, approximated spring constants, axle load distribution and position of the accelerator pedal points based on the GCIE standard. The parameter values contained in these tables are accessible for explicit editing. Concurrently, the inserted values are monitored with regard to their plausibility by the help of predefined borders. Figure 9.15 illustrates data flow and parameter control between the user, the data pool, the geometry module and the calculation module.

9.3.3 Data Pool Structure and Parameterization Strategy

The parameterization of the entire concept tool is based on a list of exactly defined parameters, which include all of the information required for the control and operation of the different working fields in the data pool, the geometry module and the calculation module. The complete list of parameters is managed in the data pool (or the database), whereby each parameter has its counterpart in the corresponding sections of the geometry and the calculation module. The information flow between the modules and the data pool is bi-directional. This enables user-friendly modifications either by changing parameters in the data pool or by changing them in the working area by modifying geometry values or values of calculation procedures. One key feature of the parameter-based system is the ability of users to adjust the parameterization, which provides essential flexibility in the concept phase. Thus, various associations between different parameters can be defined and modified during different steps of the development process. In this way, users can add new parameters or configure formulas or relations with respect to current project requirements. For example, engineers can decide whether the wheelbase is a value that will be entered or one that will be calculated based on the vehicle wheel coordinates.

According to the method of object-oriented parameterization, each parameter of the system is equipped with a list of parameter attributes [17]. This method expands the information content based on the different applications of parameters. In this way, it is possible to integrate a universal parameterization for different operations in the concept tool, such as geometry creation, geometry structure organization, check operations, or functional vehicle computations.

The integrated data pool configuration manages the complete parameter structure of all enclosed modules and functionalities, including the template structure, a user interface and a tracking system. The data pool itself serves as a centralized parameter platform and supplies the geometry model as well as the calculation and simulation processes with the necessary information. This structure ensures a logical parameterization architecture in different applications while simultaneously preventing data redundancy. Besides the main operational tasks, the data pool is equipped with a user-configurable interface module, which supports the definition of different data exchange formats to supply multiple applications and a quick adaption to new requirements.

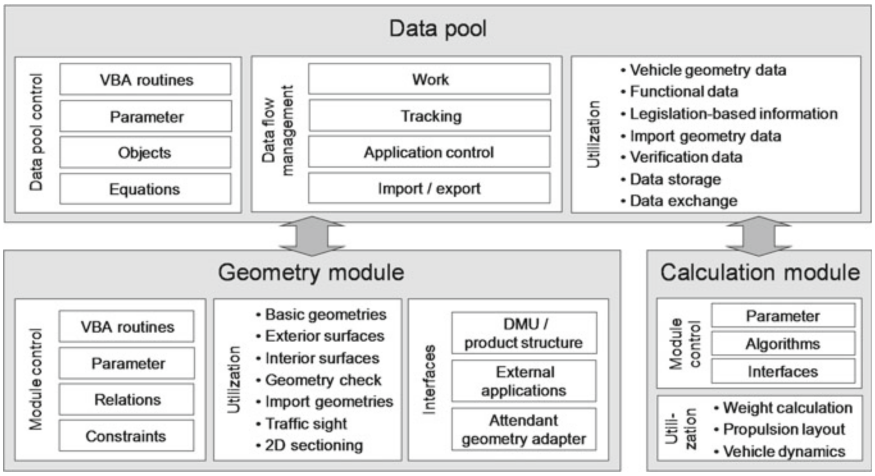


Fig. 9.16 Parameterization strategy of the integrated concept tool

Figure 9.16 displays the parameterization structure of the data pool, the geometry module and the calculation module. The data pool control system is based on several VBA routines (visual basic for applications), which facilitate automated procedures to control and perform the different functionalities and operations. At the same time, graphical user interfaces support an efficient and easy handling. The data management of the entire concept tool is based on the integrated parameter structure. Besides parameters, specific objects include information of geometry components (e.g. envelope of geometries), calculation components (e.g. diagrams or maps) or parameter interrelations (e.g. check parameters, reference points). In addition to managing parameters and objects, the data pool organizes equations from the variable parameterization in the geometry module. For this purpose, linear equations from the geometry module are controlled in a way that different parameters of a linear equation can be variably defined as input or as output values.

The data flow management combines information from different sources and provides accessibility to the data. The *work* area serves as an input platform in the data pool and always holds the current data status. The *tracking* area consists of two modules. An efficient save management module makes it possible to save processes of the work status during the development process, while the load management module supports the reloading of the desired development status. Internal calculation procedures, such as the computation of geometrical and functional parameters (e.g. lengths, center of gravity, flow resistance areas), are handled by the application control module. The import/export area manages data exchange processes with external applications. This includes importing supplementary data of geometry components, as well as data for calculation procedures. In addition, the export system permits an efficient transfer of parameter configurations for use in external programs. As mentioned before, the data pool holds all of the data of the concept tool that is utilized in both the geometry and the calculation modules. Vehicle geometry data, legislation-based information and import geometry data are transferred to the geometry module to control the conceptual vehicle geometry model. Functional data are handled for different calculation procedures. Several check operations include verification data of predefined concept check procedures. In addition, the data pool includes operations of data storage (tracking) and data exchange.

The parameterization of the geometry module consists of the module control, the module utilization and an interface structure. The entire data flow and the processes are controlled by integrated VBA routines, which also support data input and user handling. The parameter structure in the geometry module is completely integrated in the parametric-associative geometry definition structure. This means that the geometry definition is based on predefined parameters, corresponding associative geometry configurations and geometrical constraints. Relations are used to express connections between parameters in the form of geometrical functions or of logical configurations in equations, which makes it possible to apply variable parameter input strategies.

The configuration of different associative geometry sections enables the definition of an integrated conceptual vehicle model in the applied CAD software package. This includes basic geometries, exterior and interior surfaces, and a traffic sight module. The incorporation of import geometries into the product structure enables a consideration of components delivered in the concept process. As a separate feature, the geometry check module supports an evaluation of different variants throughout the course of conceptual development.

The geometry module is equipped with interfaces for an enhancement of the concept development and for the integration of external geometries and applications. The DMU/product structure interface supports the integration of product structures into the virtual concept vehicle model. This includes a structuring of the virtual vehicle model into sub-modules and components based on a predefined bill of material or a neutral product structure list. This segmentation of the product into several layers and modules supports an efficient integration of components and modules into a digital mock-up model. In cooperation with the different modules of the concept vehicle, the DMU integration supports a detailed consideration of provided geometries, including their specification characteristics and their space requirements. External applications, such as modules for the investigation of suspension systems (including kinematics) or the integration of parametric geometry models from libraries, are imported and positioned in the vehicle mock-up via a specific interface. Besides the DMU structure and the integration of external applications, an attendant geometry adapter enables the import of additional, subsequent geometry elements, which are able to support the design process. These can be data from existing vehicle models, benchmark data, or the integration of components and modules for pre-studies.

The calculation module consists of two main areas. The module control section organizes the data exchange between the data pool and subsequent external applications via data interfaces and manages different algorithms and calculation procedures. In the second module, different utilizations of functional vehicle layout are organized, such as weight estimation, power train layout procedure and conceptual vehicle dynamics simulation.

9.3.4 Geometry Creation in Conceptual Vehicle Development

A conceptual vehicle model, which includes all of the geometry elements for the description of a virtual concept vehicle, is generated in a 3D CAD software package. These include wireframe elements (points, lines and curves), surfaces and solids. The geometry elements are completely parameterized, whereby two types of geometry-based parameters occur. In the concept vehicle, there are a total of about 500 driving parameters which define the geometry dimensioning, and there are secondary parameters, which result from geometry-related operations. The driving parameters are controlled by an embedded parameter table, which is connected with the CAD external data pool.

The geometry module is composed of seven main sections, which are interlinked by a number of formulas and dependencies. These mathematical relations are also parameter driven and are displayed in a separate relations table. Besides the linkage to the data pool, the geometry module is equipped with an interface to the calculation module and interfaces to external applications. A connection to the calculation module handles all geometry-based information for the functional vehicle layout, such as principal vehicle dimensions (e.g. suspension characteristics), the centers of gravity of components, and areas for the calculation of air resistance. External interfaces work with program-independent geometry data formats and form a linkage to styling or calculation software packages.

Figure 9.17 shows an example of a vehicle concept that results from all of the sections displayed in the geometry module. A combination of the seven modules forms an integrated 3D CAD conceptual vehicle, which includes the vehicle basis geometries, technical-related exterior and interior surfaces, check surfaces for legislative and ergonomic boundaries and imported silhouette data. A geometry-based evaluation of the concept can be performed by visual checks and by automated verification cycles in the geometry check module. The following sections include a short description of the seven sections of the geometry module and provide an overview of the interlinking functionalities and applications.

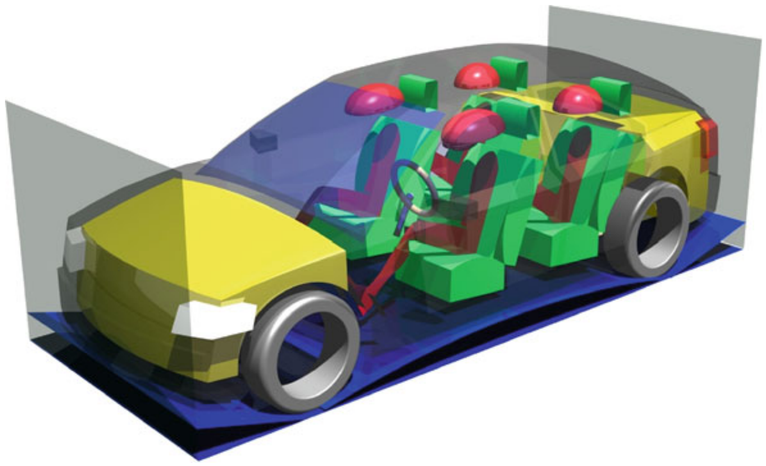


Fig. 9.17 Conceptual vehicle layout displayed in the geometry module

The general vehicle dimensions are displayed in a basis geometry section, which includes the boundary surfaces of vehicle main dimensions. In this way, a number of standardized interior and exterior car dimensions are displayed as simplified geometrical elements (e.g. the vehicle length, width and height, the wheelbase, front and rear ramp angles, maximum opening dimensions of doors and closures).

In total, the basis geometry section includes 45 dimensions and geometry boundaries. The boundary surfaces do not display the geometry of the car itself, but rather represent check gauges for the geometrical vehicle concept configuration. This method enables a vehicle setup definition at an early conceptual phase without information about the car styling or other secondary parameters. The basis geometry section supports the entire development process, beginning from the first dimensioning, maintaining the definition of technical, legislative and ergonomic-related configurations, and finally serving as check geometry for the confirmation of the vehicle concept.

The confirmation procedure is performed in a separate check unit, which is linked with the basis geometry section. For this purpose, the basis geometry section is

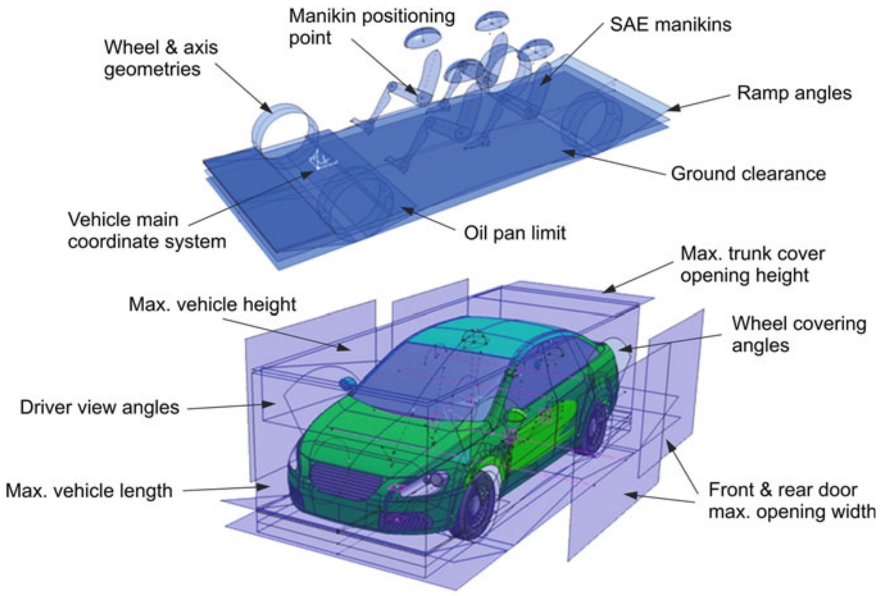


Fig. 9.18 Selection of boundary surfaces and dimensions in the basis geometry section

enhanced with functionalities from a commercially available software package, which covers legislation-based information [18]. The close integration of the off-the-shelf software package in the applied 3D CAD system [13] enables a direct embedding in the geometry module of the virtual concept vehicle. This close integration facilitates the import of both the object definition and the parameterization strategy into the overall software architecture. Figure 9.18 shows a selection of geometries and dimensions displayed in the basis geometry section.

All of the dimensions in the basis geometry section follow the guidelines of SAE standards and specifications corresponding to the GCIE guidelines. The exterior geometry section represents simplified outer surfaces of the virtual concept vehicle. These surfaces are technical-based and highly flexible to enable the geometrical description of different car types. The exterior geometry section does not generate any kind of styling surfaces, but rather a complete technical representation of the outer car silhouette. The applied wireframe and surface elements are parametric-associatively interlinked and integrated into the logical parameter management of the geometry module as a template model. The manipulation of the exterior section is performed by the CAD -external database or by data input via graphical user interfaces (GUI) in the CAD environment. Figure 9.19 shows a sample selection of available vehicle templates of car exterior geometries.

In case the predefined template geometry cannot fulfill the requirements of a new vehicle concept, an additional freestyle unit is integrated. This unit enables the definition of vehicle geometries not covered by common car setups. In this way, the generation of completely new, future-oriented concepts for individual traffic

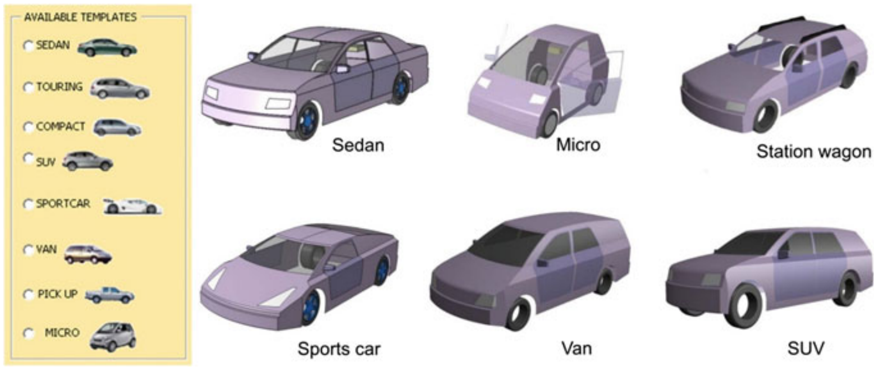


Fig. 9.19 Available vehicle templates and a selection of sample car exterior geometries

is supported without limitations from existing categories and integrated into the functionalities of the virtual concept vehicle. The freestyle unit enables a high degree of geometrical freedom (Fig. 9.20).

The concept vehicle interior geometries are represented in a separate section, which includes simplified surfaces of seats, dashboard, steering wheel, gearshift, pedals, cabin and trunk inner surfaces, as well as a number of boundary geometries for ergonomic investigations (e.g. knee and elbow clearance, entrance area, head clearance). The logical configuration of the interior geometry section is the same as in the exterior geometry section. It contains a template-like characteristic, with the option of parameter-controlled geometry adjustments.

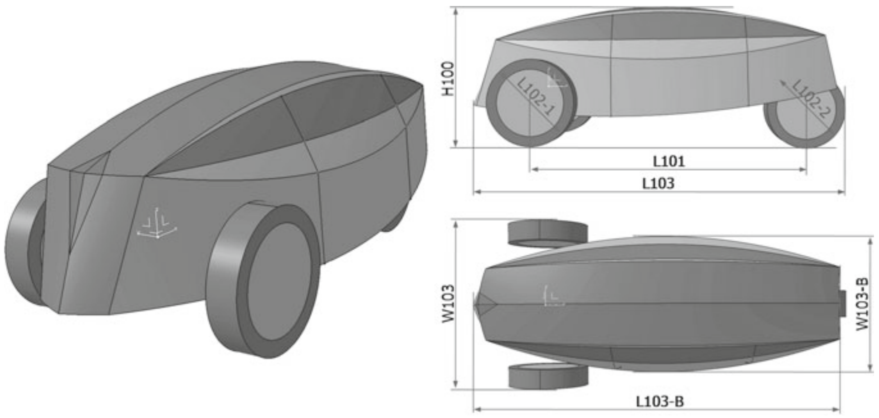


Fig. 9.20 Example of freestyle exterior surfaces with a selection of main dimensions in accordance with the SAE standard J1100

Since the interior geometry section is linked to the basis geometry section and the exterior geometry section, modifications made in any section are incorpo-

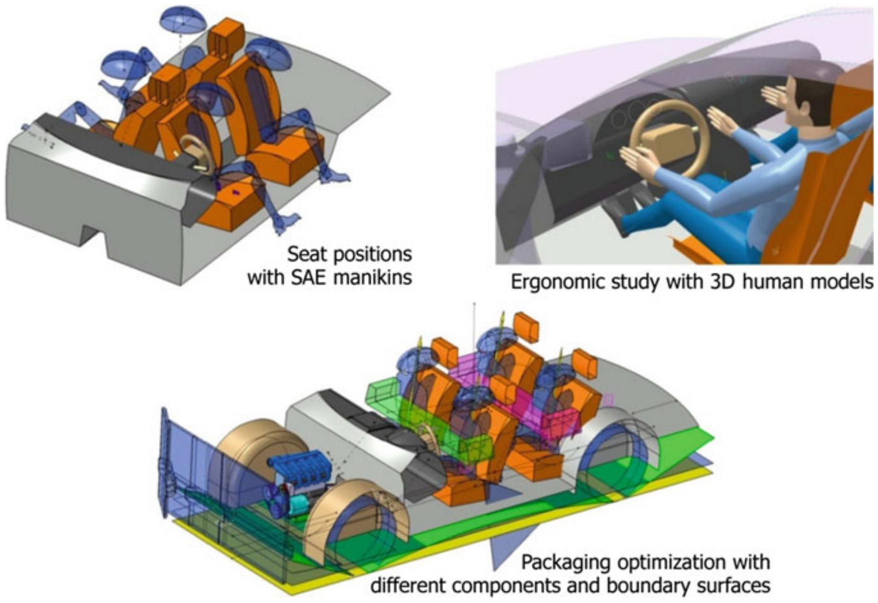

Seat positions with SAE manikins
Ergonomic study with 3D human models
Packaging optimization with different components and boundary surfaces

Fig. 9.21 Interior geometry section with application examples

rated into the entire virtual concept vehicle. The interlinked parameter management ensures that each parameter in the database has an exactly defined corresponding geometry parameter in the targeted section of the CAD model, which influences dependent parameters in other sections. Depending on the functionalities of the applied human models, anthropologic and ergonomic studies can be carried out. Vehicle interior characteristics, such as seating positions, accessibility of control units, comfort dimensions and entrance areas, can be optimized.

Figure 9.21 shows examples of interior-surface-based studies in the early concept phase. Besides the representation of the simplified inner surfaces of a car, the integration of human models and boundary surfaces enables detailed ergonomic studies, packing optimization with different vehicle components and the evaluation of legislative regulations.

The general geometry module architecture is oriented to the general vehicle dimensions, boundary surfaces and check geometries of a virtual concept vehicle. The integration of additional component geometries, which are required for DMU and packaging-relevant investigations, is performed in an import geometry section. This section offers two different kinds of functionalities. Variable parametric-associative templates of selected components include a fuel tank dummy, gas tanks, battery units, electric motor dummies and a collection of common suspension systems. These templates are directly integrated into the parameterization concept of the geometry module and thus connected with the data pool. These include the outer contours of common internal combustion engines between 1.6 and 6 liter displacement and typical transmissions, as well as a selection of standardized components.

In addition, an interface for the integration of external DMU geometries supports the implementation of packaging-relevant technical standard components or carry-over parts into the virtual concept vehicle (e.g. engines, transmissions, engine cooling systems, chassis components, seat models, air conditioning systems, window lifters, loudspeakers, suitcases). The parts from external sources are positioned by a parameter-controlled positioning axis system, which enables data-based control and save management.

Import geometries from external sources are implemented as non-variable DMU elements that describe the outer contours of components. The import geometry section offers a selection of implemented dummies for engine, transmission and suitcases, which can be selected in a database and positioned in the vehicle model by a parametric reference axis system. Additional parts for conceptual studies are implemented via a geometry interface. In this method, existing components from other cars can be taken into consideration during the concept phase and can serve for ergonomic studies, packaging and clash analysis, evaluation of styling concepts in terms of space requirements and other functions.

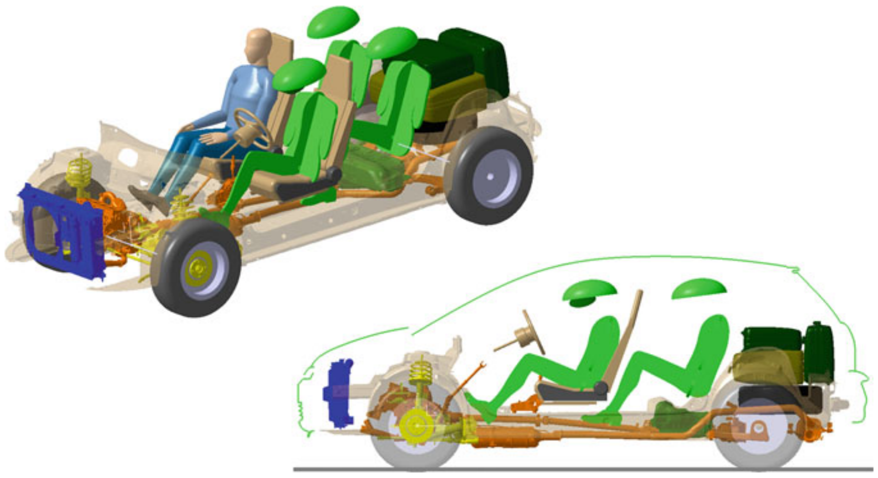


Fig. 9.22 Packaging study with import geometries

Figure 9.22 shows an example of the results of a packaging study with simplified geometries of the virtual concept vehicle and supplementary CAD components from a predecessor model. In the case of the displayed study, the car being developed had to adopt components from former models. Therefore, a representative engine, drivetrain component, suspension parts, steering system and the under carriage were loaded into the conceptual vehicle layout to evaluate the geometrical possibilities of the new model.

Geometry Checks

The geometry check section enables a validation of the virtual concept vehicle based on a list of predefined quality parameters, which define the general vehicle dimensions in accordance with the GCIE and SAE standards, legislative requirements, ergonomic characteristics and packaging-relevant factors (e.g. component collisions, passenger space requirements, the layout of luggage space). The check procedure is performed on demand during the development process or in the course of specified milestones. Thus, it can be applied to support optimization cycles, as well as for verification and technical release procedures.

The geometry check section contains boundary surfaces, which are defined in the basis geometry section, the exterior section and the interior section, as well as check vectors and check dimensions, which enable additional tests. A direct parametric dependency of the check program on geometry driving sections ensures the currency of the checking process and the integration of check results into the parameter and data management system of the virtual vehicle model. The check procedure includes automated verifications of clash or overlapping geometries and user-oriented visual inspections. The results of automated verifications are saved in a check results list by the program, while the manual inspections are performed with a list of prescribed check tasks and are recorded by the user. Figure 9.23 illustrates a selection of check surfaces and their application on a vehicle model.

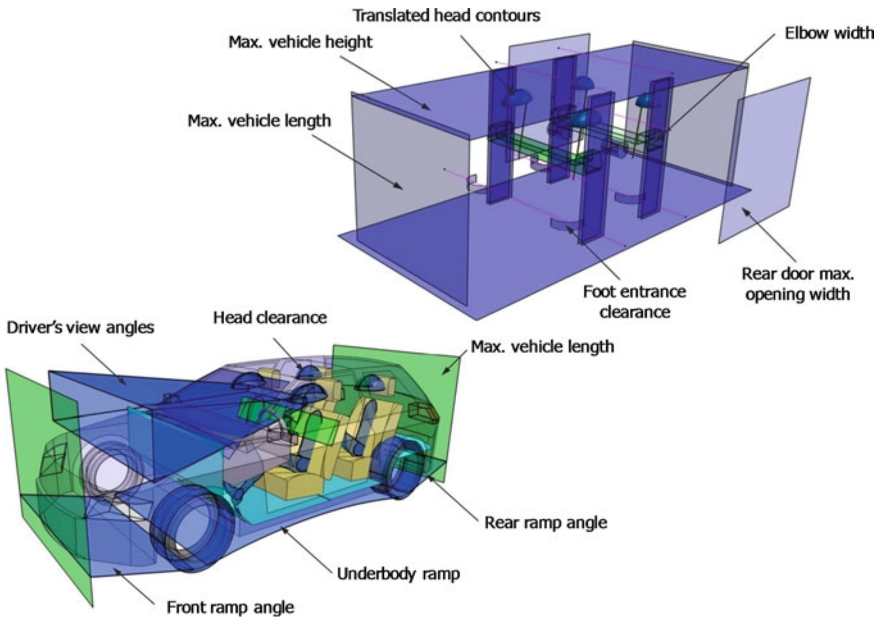


Fig. 9.23 Selection of check surfaces and their application on a vehicle model

Once the technical solutions have been approved by the check functionalities, the development status can be saved, and the geometrical and functional parameters can be imported into the data management system. In a later step, when 3D CAD styling data are available, the technical exterior surface is replaced by styling surfaces, which also have to fulfill the technical and legislative requirements. These styling surfaces are analyzed with the geometry check section to visualize the conformance with given requirements or, in the case of disconformities, to highlight and localize divergences.

The engineering-styling convergence always represents an important task in automotive development. In many cases, the two departments differ in their interpretations of product requirements and in their determinations of solutions. If the styling solution passes the check procedure, the vehicle concept status is approved and can be saved. In the event that the geometry check discovers discrepancies between the styling surfaces and the concept specifications, the vehicle styling must be reconsidered with respect to the technical necessities.

An integrated 2D sectioning functionality enables an automated parametric derivation of 2D drawings from the virtual concept vehicle. These drawings include predefined views and sections and describe the dimensional concept of the car. Geometry dimensions, which are defined in views or sections, are generated automatically in accordance with the GCIE standards and are linked to the 3D CAD model. Modifications of the model in 3D working space affect the dimensional drawings directly to keep them updated during the development process. The dimensional concept displays the complete vehicle, including annotations and text tables. All sections, views and dimensioning are embedded in the global parameterization management system of the virtual concept vehicle, including a connection to the data pool.

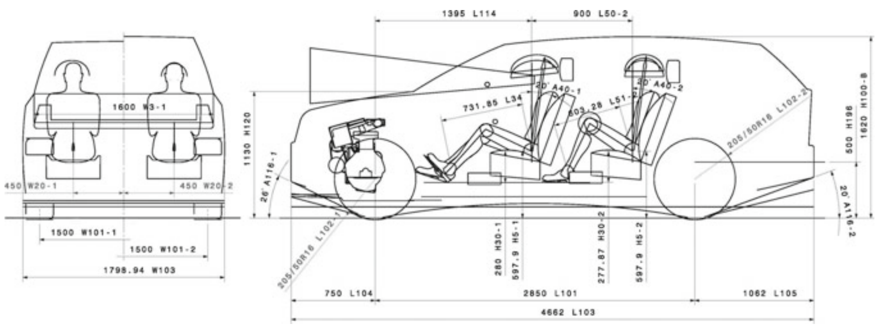


Fig. 9.24 Front view and side view of a dimensional vehicle layout

The relevant and critical geometrical dimensions provide an overview of key characteristics related to the vehicle size, packaging solutions and ergonomic viewpoints. The dimensional concept is a part of the full-vehicle description during the conceptual development and serves for data tracking and release archiving. Figure 9.24 shows the front view and the side view of a dimensional vehicle layout in the early concept phase with selected vehicle components and dimensions.

External Interfaces

The geometry module is embedded in a commercial 3D CAD software package, which offers different possibilities for geometry data exchange. The open structure enables an easy-to-handle implementation of external vehicle mock-up data for the integration of components, product structures or full-vehicle geometries. The integration is performed in a separate DMU area and managed by the GUI-supported data organization strategy of the virtual concept vehicle. In addition to vehicle DMUs for geometry-based investigations, a flexible CAE interface enables the connection of structural optimization software, which is used for conceptual pre-calculation of vehicle body, NVH, stiffness and crash performance [3, 19]. The interface works bi-directionally and exports the necessary geometry data from the virtual concept vehicle to the simulation software. In the second direction of data flow, body structure models, including simplified beams, carries and joint elements, are imported into the virtual concept vehicle and serve as geometrical components for packaging studies. A general geometry export function supports the data transport via neutral data interfaces. In this way, selected geometries of the virtual concept vehicle are saved as neutral data formats and can be imported into several common design or simulation software packages. These neutral data formats consider the geometrical characteristics, including dimensions and shape, but they do not include any parameter or history data of the concept model. Thus, different CAE applications can be integrated into the virtual vehicle development process, such as noise-vibration and harshness simulation, finite element calculations, multi-body simulations, specific crash-related calculations and others.

Besides simulation, different processes in the field of product planning and technical calculation can be supported with the supply of concept geometry data from an initial phase on. These include project and product planning, cost calculation, early production engineering tasks and the estimation of technical and economic feasibility studies. An efficient data transfer to external engineering and component suppliers enables a direct integration into the development process. Vehicle styling is an important process during the conceptual development phases. In the present approach, a close interaction between the engineering-based technical development and the styling department is supported by an integrated styling export function, which enables the definition and transfer of (technical) boundary surfaces from the virtual concept vehicle into the applied styling software packages. For this purpose, the surfaces to be exported are selected in a dialog box and converted into a neutral data format. This neutral geometry is imported into the applied styling software and serves as input data for the development of the vehicle exterior and interior styling surfaces. At the same time, the exported surfaces are saved in the parametric-associative virtual concept vehicle to ensure a clear assignment during the development progress. All related parameters are saved in the data pool's tracking system for archiving purposes. The exported boundary surfaces for the styling creation consist of curves and surfaces from the basis geometry section, the exterior section and the interior section and represent the virtual concept vehicle at the selected development status.

9.3.5 Processes and Applications in the Project Flow

The initial phase of automotive development includes the following core processes:

- Geometry generation
- Geometrical integration
- Functional integration

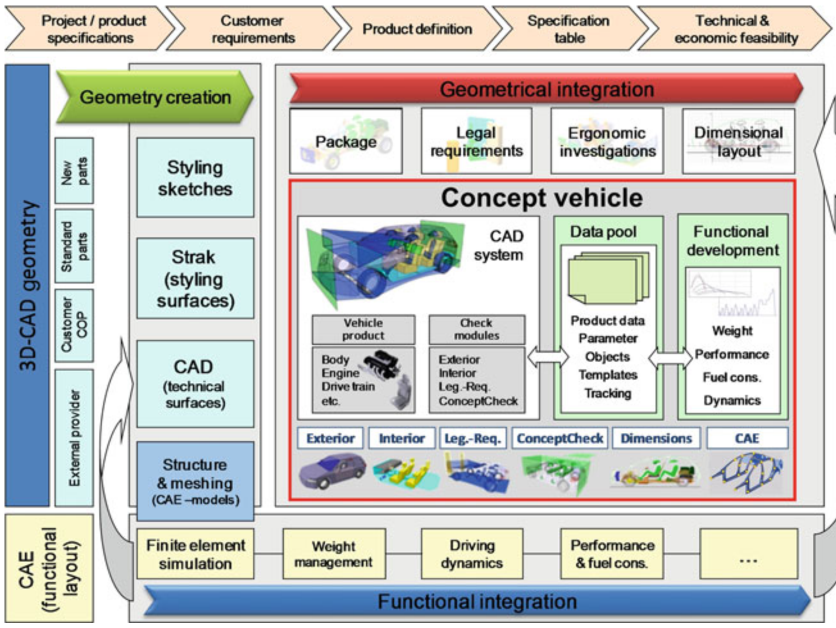


Fig. 9.25 Modules and functionalities of the virtual concept vehicle in the process flow [20]

Figure 9.25 shows an overview of the relevant disciplines and verification methods in relation to the core processes. The process targets include different product-describing aspects, which are displayed as superordinated requirements. All relations between geometry creation and functional development are linked with processes of geometrical integration in order to support verification, as well as functional integration and evaluation. This enables a simultaneous optimization of the vehicle architecture in the form of a centralized working area. The intersection of multiple disciplines in a centralized virtual concept vehicle enables a concurrent consideration of often conflicting viewpoints. This is precisely the main application of the present approach of an integrated concept model, which is defined and controlled by the vehicle architect. The process of geometry generation is driven by the definition or description disciplines of styling and design, which deliver styling surfaces and data for carry-over parts and standard components that are already available in the early concept phase.

In the concept phase, high-quality CAD data of new components that can be used as technical surfaces are rare. Most of the geometries are defined as rough estimations for package investigations only. During processes of geometrical integration, the application passes through diverse sub-tasks of packaging and layout, virtual ergonomic investigations, verification of legal requirements, the generation of a dimensional concept and the definition of a mass package. The functional verification includes a transfer of CAE-relevant information from the geometry model as input data or boundary conditions for simulation processes to enable a consolidation of workflows which were strictly separated previously. The parametric-associative generation of comprehensive geometry data for a virtual concept vehicle under the consideration of legal requirements, functional aspects and customer-related influences enables an accurate characterization of a new vehicle concept from the initial process phase on. In the course of the development project, a rough vehicle package is refined stepwise by the implementation of several components and modules.

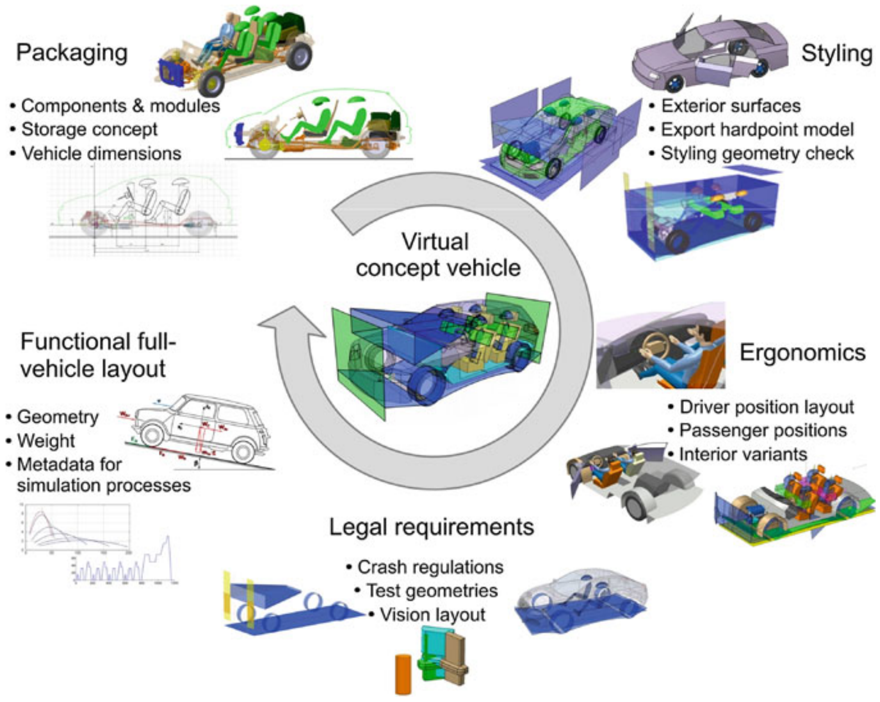


Fig. 9.26 Interdisciplinary consideration of the workflow in an automotive concept phase using the integrated approach [1]

Figure 9.26 provides an overview of the application based on the workflow of typical project engineering procedures. Starting from first vehicle data, an initial conceptual 3D CAD model is defined. Depending on the requirements of each individual development step or verification process, different modules of the virtual concept vehicle are used. Figure 9.26 shows an interdisciplinary consideration in which all tasks from the conceptual development cycle are performed within the virtual vehicle model. Since data transfers to external applications are also triggered and monitored by the virtual model, the general data flow is managed by one centralized system. The results of all working procedures are saved and maintained in an integrated database structure. The centralized data management concept avoids inconsistent information flow and supports the execution of variant studies and optimizing cycles significantly.

A conceptual automotive development process can start with the definition of packaging-relevant vehicle data, which correspond with the requirements in the specification table. This process includes the representation of vehicle main dimensions through appropriate parametric boundary surfaces and geometrical elements in the 3D CAD model. An additional implementation of predefined components and human models enables the definition of an initial full-vehicle packaging concept. Sources for predefined components or modules can include existing vehicle models (e.g. predecessor models), geometry data pools or simplified 3D CAD models. The integration of envelope geometries from external sources and simplified predefined components, as well as the representation of boundary surfaces for the vehicle dimensions in one 3D CAD model lead to an early and efficient setup of a conceptual full-vehicle model. This concept model provides the basis for further investigation and optimization cycles, which deal with the interactions of components, vehicle dimensions and space requirements of passengers, as well as an early estimation of the storage concept.

During the initial phase of development, the limited availability of information about the vehicle styling makes it difficult to consider the future outline shape. To avoid problems in later steps, two-dimensional styling sketches can be implemented directly into the 3D CAD vehicle model. This procedure allows for an early assessment of the technology-styling convergences and makes it possible to carry out modification or optimization cycles from the beginning. The implementation of the styling process itself into the conceptual full-vehicle development is performed by deriving relevant geometries from the virtual concept vehicle. These surfaces are summarized in a hard point model, which represents the specification geometries of vehicle main dimensions and the space requirements for passenger, luggage or components, as well as legal instructions. An import into the applied styling software is accomplished via neutral geometry data interfaces.

One important work package in early vehicle development is the definition of ergonomic solutions for driver and passengers. The generation of an interior module, which comprises seats, dashboard, control elements and cabin-limiting surfaces, is directly connected to the positions of human models as driver and passengers. Derived from the seating positions and the space requirements, a variable configuration of the 3D CAD interior model supports the initial definition and optimization cycles under the consideration of specific ergonomic necessities.

The legal requirements for full-vehicle development include instructions for passive safety, geometrical prescriptions for lighting equipment and relevant standards for driver vision and passenger safety. A representation of each regulation as a geometric check surface enables a direct implementation into the data structure of the integrated virtual vehicle model and thus a consideration throughout the entire development cycle.

Besides different geometry-based applications, the integrated vehicle model includes a module for functional layout and evaluation for full-vehicle development. The consolidation of geometry-related tasks and functional aspects enables an interdisciplinary treatment of the entire concept phase, which supports the consideration of a broad variety of requirements and characteristics. The estimation of vehicle weight and the position of the vehicle center of gravity is a fundamental challenge in functional concept development. Linking the database and the geometry module facilitates an easy-to-handle estimation of both the total vehicle mass and the vehicle center of gravity independent from the project development status. Driving performance and fuel (i.e. energy) consumption are estimated based on geometry data, mass data and imported information about engine and drivetrain configurations, under the consideration of specific driving resistances at different load conditions. The computation of representative numbers for driving behavior includes vehicle speed, acceleration and climbing potential and enables an early assessment of different power train technologies in terms of their suitability for the relevant vehicle concept. Besides standard drive units, electric motors and hybrid propulsion configurations are also considered. Finally, lateral driving and vehicle handling characteristics are computed by using an integrated single-track model, which combines information from the geometry module, the vehicle mass and center of gravity estimation, the drive line layout and the data pool. The simulation of driving tendencies (under-steering, neutral and over-steering) under the consideration of different axle load conditions enables an early assessment and optimization of influencing parameters to achieve a well-balanced chassis and vehicle layout.

Modern 3D CAD software packages enable the implementation of 2D studies, sketches or drawings into the 3D model. This possibility brings the engineering-based construction and DMU development close to the styling process. Based on an open architecture, 2D studies can be implemented into the virtual car model to perform different checks related to ergonomic viewpoints (passengers), packaging boundaries (drivetrain or chassis components) or legal-based influences (e.g. safety and crash regulations). Advanced design software provides an additional ability to generate 3D surfaces from 2D sketches, which allows for a direct conversion of 2D studies in the automotive 3D CAD model. In the case of supplied 3D hardware (style studies, clay models or detailed scale models), scan or measurement data can be directly imported into the CAD software to serve as a basis for subsequent surface generation processes.

Figure 9.27 displays an exemplary work flow in the integrated 3D CAD model during an automotive concept phase. The initial steps start with the definition of concept modules. A logical arrangement of these basic modules results in an integrated 3D CAD model, which includes the main dimensions and check geometries of the most important legislative-influenced components. The adaptation and adoption of additional import geometries leads to a vehicle packaging concept and then to a dimensional layout. If no design surfaces are available in an early concept phase, simplified parameterized exterior surfaces can be used to visualize the outer geome-

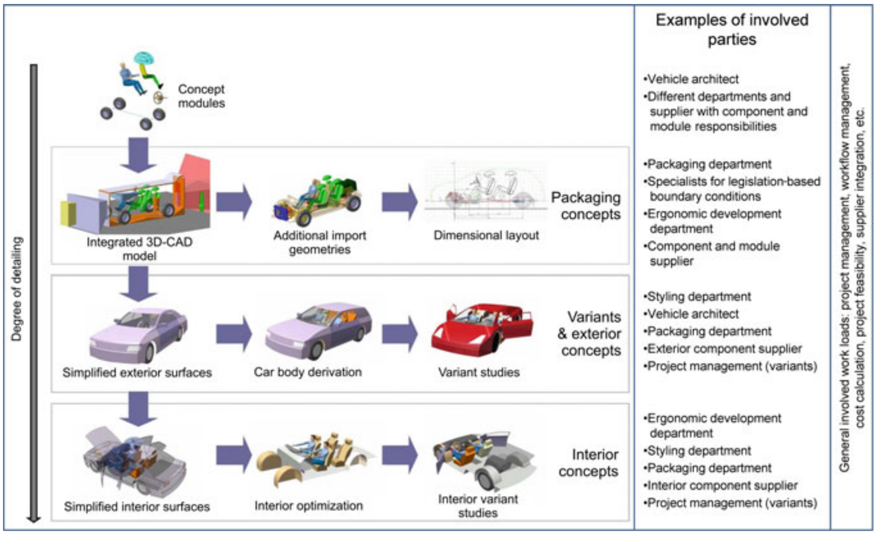

Degree of detailing
Concept modules
Integrated 3D-CAD model
Additional import geometries
Dimensional layout
Packaging concepts
Simplified exterior surfaces
Car body derivation
Variant studies
Variants & exterior concepts
Simplified interior surfaces
Interior optimization
Interior variant studies
Interior concepts
Examples of involved parties
•Vehicle architect
•Different departments and supplier with component and module responsibilities
•Packaging department
•Specialists for legislation-based boundary conditions
•Ergonomic development department
•Component and module supplier
•Styling department
•Vehicle architect
•Packaging department
•Exterior component supplier
•Project management (variants)
•Ergonomic development department
•Styling department
•Packaging department
•Interior component supplier
•Project management (variants)
General involved work loads: project management, workflow management, cost calculation, project feasibility, supplier integration, etc.

Fig. 9.27 Geometry-focused workflow example of the integrated 3D CAD model [1]

tries of the concept vehicle. These simplified exterior surfaces can be replaced with the outer surfaces in later steps or can support the visualization of car body derivation or variant studies. Similarly, parametrically controlled, simplified interior surfaces visualize car seats, dashboard and other components in a very early concept phase, thereby enabling initial ergonomic studies and optimization processes. Due to the variable structure, the applied methods of data organization and geometrical parameter control make it possible to switch between the modules at any time for an optimized evaluation and modification of the virtual concept vehicle model.

9.3.6 Product Knowledge in Integrated Virtual Concept Development

A stepwise generation of the vehicle concept in combination with a continuously increasing share of verified data leads to an increase in product maturity throughout the conceptual development process. The product maturity itself is defined by the fulfillment of clearly prescribed milestones, which enables a reporting and observation procedure based on the achievement of objectives. Every application step in the workflow is accomplished by a coupled reaction of the 3D CAD model (or calculation model) and the database structure, which leads to a continuous increase of knowledge in all areas involved. This is made possible by an increasing level of detail in geometry creation and growing information content in both the calculation module and the tracking system of the database. Compared to earlier steps, the execution of improvements and variant studies or the implementation of new aspects into the virtual model increases and strengthens the project status and thus the knowledge status of the product.

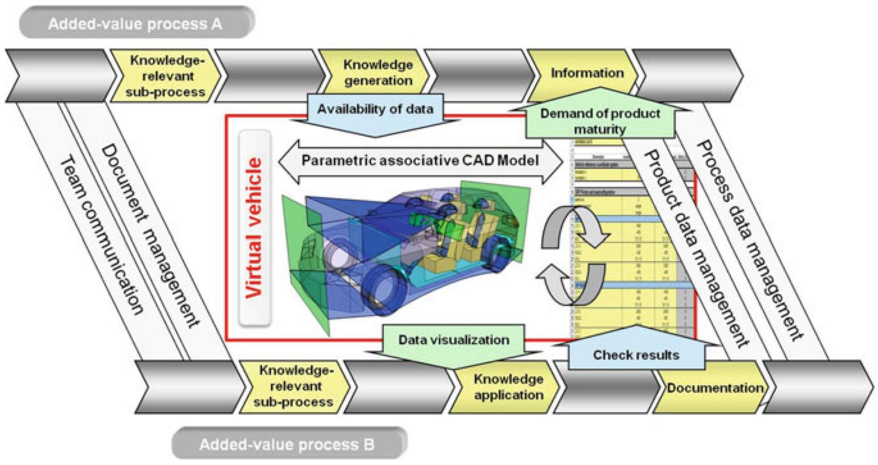


Fig. 9.28 The virtual concept vehicle embedded in an automotive development process [21]

Figure 9.28 shows an example of two automotive development processes connected by an embedded virtual concept vehicle. Besides the purely technical-oriented procedures for the conceptual development of a new vehicle, the integrated architecture of the virtual development tool supports the communication between the two parties involved and the required knowledge transfer. This is achieved by an easy-to-handle product representation via 3D CAD geometry data, as well as the universal parameterization strategy and a data-based parameter management.

The actions performed in sub-process (A) generate data for the definition of a virtual concept vehicle, which is able to visualize the information of (A) in a parametric-associative 3D CAD model. A parallel sub-process (B) has access to the CAD model and is able to perform different operations with relation to the vehicle model. In this way, the execution of sub-process (B) is accomplished with an added value in terms of a product interpretation and evaluation. In an inverse consideration, the sub-process (B) increases product knowledge by providing access to virtual model data, such as during the documentation of verification or checking procedures. The subsequent activities in sub-process (A) can read this information from the database and use it as a basis for further development steps.

Ongoing data flow and data enrichment cycles during the development process increase the information density in the virtual concept vehicle and consequently the knowledge about the product. A combination of an integrated database structure and a fully parametric-associative geometry definition enables a comprehensive product description based on object-oriented parameters, which are transferred into different associated processes. An efficient interpretation of product information in external applications (e.g. finite element calculations) is achieved by exported parameters in combination with parallel available geometry data from the 3D CAD model. In the inverse process direction, an increasing level of knowledge is facilitated by an accumulation of information in the database. Newly revised or implemented parameter values are transferred from a sub-process (B) via the database structure into the sub-process (A). For example, the results of check procedures in (B), which reveal possible failure in the fulfillment of legal predictions, are used for geometry modifications, which are performed in (A). The extended consideration of numerous working fields in conceptual development processes in combination with a detailed parameterization of a virtual vehicle model itself and its influencing boundary conditions enables the generation of a knowledge-base-oriented tool for the stepwise creation of automotive concept studies, all within a virtual environment.

9.3.7 Integration of the Virtual Concept Vehicle into the Knowledge-Based EDM

After having investigated the application-oriented view of the *virtual concept vehicle*, it is now interesting to discuss the integration of the method of associative parametric geometry creation into the model of knowledge EDM. This requires a process-oriented view of tools, which must then be integrated into the model of knowledge processes between value-added processes (see Sect. 7.4.3).

Figure 9.28 shows the virtual concept vehicle embedded between two specialized product development processes and the knowledge process running in between. This process is supported by the knowledge activities *documentation* and *informing* and enables access to the data layer by providing appropriate system interaction. Thus, this represents an indirect transfer of knowledge. Knowledge is transferred into the project area, which involves coupling CAD modules with the database. The link to previous applications also leads to increasing knowledge in the project environment of the concept development.

On the data level, the transferred data is upgraded through the associative parametric design method because users can interpret the data in an application-specific way and can therefore use the data in a certain context in the subsequent processes. In other words, knowledge transfer becomes more efficient due to the additional visualization of the geometry of the product/concept data itself, as well as the visual representation of the geometric relationships, which is helpful for the functional assessment of the vehicle concept. In the opposite direction, the flow of knowledge and data also generates added value as geometric changes can be made to the concept using parameters that are either drawn directly from the database or prepared via another method.

The product maturity increases continuously due to the knowledge gained via the step by step build up of the vehicle concept and the continuous application of secured data. The progression of the concept development process essentially determines the product maturity. To be more precise, data reliability and the level of data protection, together with the established product knowledge, leads to product maturity. If product and process data are continuously communicated and documented throughout the development process, then the product model represents not only a tool for documentation, but also a knowledge database. Such a knowledge database is required to support development because administration data is highly contextual, and a knowledge database would make it possible to reproduce the data throughout the development process.

Figure 9.28 shows that the following aspects are important in the interaction of multiple processes:

- Team communication
- Document management
- Product data management
- Process data management

Product knowledge can only be managed if all of these factors are combined. Next, all of these considerations must be united and systematically integrated into an EDM environment. To this end, the associative parametric modeling, which can be seen as the core method of virtual concept vehicles, is integrated as an additional module, and an EDM system takes over the functions of data and process management as its core modules.

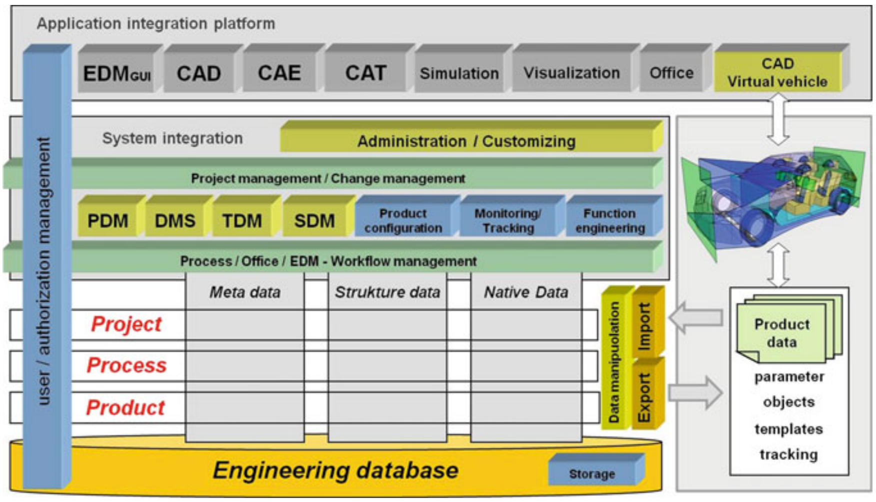


Fig. 9.29 Engineering data management system—architecture with integrated *virtual concept vehicle module*

Figure 9.29 shows the integration of the *virtual concept vehicle* module into the architecture of an EDM system. On the one hand, it forms an interaction interface for CAD applications in the integration platform, and on the other hand, it provides a two-way interface to the database of the EDM. This interface does not only allow the input/output of necessary CAD model parameters but can also provide additional data that the CAD model can use to inform the user. As a result, the original functions of the virtual concept vehicle are divided into associative parametric modeling and the core functions of an EDM system. The number of non-value-adding routines and side-line jobs are thereby reduced, and existing system interfaces are avoided, which enhances the entire virtual process chain and makes it possible to focus on the actual core tasks of the concept work.

In the automotive industry, concept tools are already being used to evaluate vehicle concepts in terms of geometric features and geometric specifications in the early stages of the innovation process. This is even possible with limited data availability and can lead to a better concept validation. The concept tool supports the principle of frontloading by providing a higher informative value in the earlier developmental stage than was previously possible. This reduces the number of changing loops required in the concept phase and thus shortens the development time.

9.4 Analysis and Design Process of the Operating EDM

The procedure for process analysis (Sect. 9.1.4) and the activities involved in the design of the operating EDM can be displayed together in a combined procedure model.

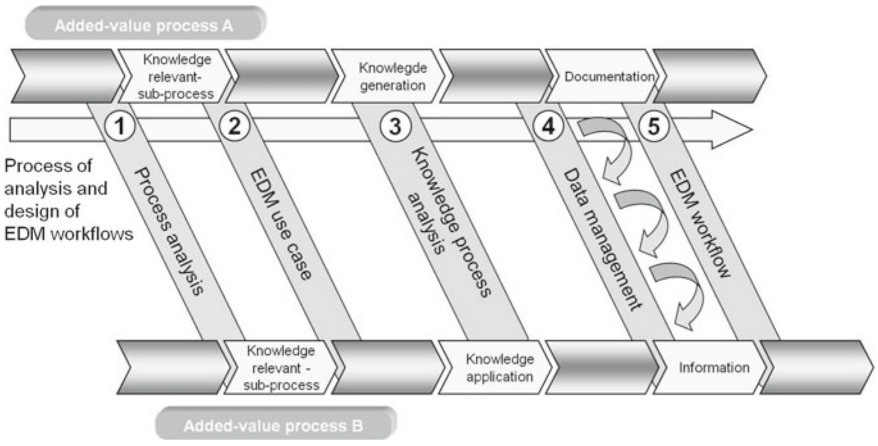


Fig. 9.30 The analysis and design process of the operating EDM

Figure 9.30 shows the five process steps that were addressed in the previous chapters, from the process-oriented task to the application in the EDM system:

- Process analysis links the knowledge processes thereby creating knowledge-intensive processes, which serve as additional objects of analysis.
- To form this inter-process link, an EDM use case is defined in order to capture all necessary EDM conditions or demands.

- With this frame of reference, the knowledge base or database is reconstructed and the knowledge process is therefore analyzed.
- The individual knowledge activities are then allocated to the relevant information management operations and referenced to the available data basis.

The procedure model can therefore be applied generally for a knowledge-oriented data management analysis, although it has been demonstrated here specifically for the implementation of an EDM workflow.

9.4.1 Integrated Consideration of Design Measures

An integrated knowledge-based engineering data management can be created by combining the knowledge-based strategic and operational design measures.

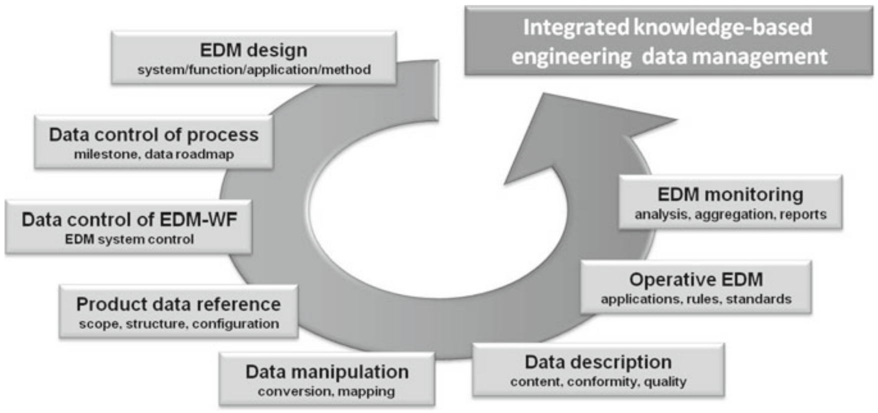

Integrated knowledge-based engineering data management
EDM design
system/function/application/method
Data control of process
milestone, data roadmap
Data control of EDM-WF
EDM system control
Product data reference
scope, structure, configuration
Data manipulation
conversion, mapping
Data description
content, conformity, quality
Operative EDM
applications, rules, standards
EDM monitoring
analysis, aggregation, reports

Fig. 9.31 Model of integrated knowledge-oriented engineering data management

Figure 9.31 shows the main starting points used in an integrated approach, which are also used for a systematic and continuous EDM development.

References

1. Hirz, M.: An approach of multi disciplinary collaboration in conceptual automotive development. Int. J. Collaborative Enterp. **2**(1), 39–56 (2011)
2. GCIE: Global Cars Manufacturers Information Exchange Group, Model Year (2008)
3. Nelsen, M.: FCM - Fast Concept Modeller. Product Presentation (2009)
4. International Organization for Standardization: Road vehicles - Functional safety. ISO 26262 (2009)
5. Hänschke, A., Kramer, F., Kondzilla, R., Wollert, W.: Das rechnergestützte Entwicklungssystem für Fahrzeuge AURORA. VDI-Berichte **613** (1986)

6. Heinke, O., Kondziella, R., Appel, H.: Variable Konzeptentwicklung mit einem Fahrzeugentwurfsystem. VDI-Berichte **968** (1992)
7. Deter, T., Oertel, C.: Einsatz eines Entwurfssystems beim Entwicklungsprozeß des Automobils. ATZ Automobiltechnische Zeitschrift **97** (1995)
8. Zimmer, H., Hövelmann, A., Frodl, B., Hänle, U., et al.: Entwurfstool zur Generierung parametrischer, virtueller Prototypen im Fahrzeugbau. VDI-Berichte **1559** (2000)
9. Bulheller, K.: Das Produktmodell als Kommunikationsbasis im Entwicklungsverbund. VDI-Berichte **1148** (1994)
10. Rasenack, W.: Parametervariation als Hilfsmittel bei der Entwicklung eines Parameter-Package, vol. D 83. Offset-Druckerei Gerhard Weinert, Berlin (1998)
11. Gessner, K.: Package-Feature für die Kommunikation in der frühen Phase der Automobilentwicklung. Fraunhofer IPK/IRB (2001)
12. Forsen, J.: Ein systemtechnischer Ansatz zur methodischen parametrisch-assoziativen Konstruktion am Beispiel von Karosseriebauteilen. Shaker, Aachen (2003)
13. Dassault Systems: CATIA V6. Date of access: 2009–11-10. http://www.3ds.com/products/catia
14. Potthoff, J.: CAVA: Fahrzeugauslegung mit CATIA V5 unter Berücksichtigung gesetzlicher Vorgaben und Richtlinien. Die digitale Produktentwicklung **1148** (2008)
15. Tesch, F.: Strukturvariabilität: Ableitung von Derivaten in Produktfamilien mit CATIA V5 in frühen Phasen. Die digitale Produktentwicklung **1148** (2008)
16. Nikol, B.: Definition von Auslegungsmethoden durch Standardisierung von V5-Tools in der Audi-Konzeptentwicklung. Die digitale Produktentwicklung **II 1148** (2010)
17. Meyer, B.: Objektorientierte Softwareprogrammierung. Hanser, München (1990)
18. Transcat: CAVA - CATIA V5 Automotive Extensions Vehicle Architecture. date of access: 2008–11-14. http://www.transcat-plm.com
19. SFE - SOLUTIONS FOR EXCELLENCE, Gesellschaft für Strukturanalyse in Forschung und Entwicklung mbH/SFE CONCEPT: date of access: 2010–07-01. http://www.sfe-berlin.de
20. Dietrich, W., Hirz, M., Rossbacher, P.: Integration von geometrischen und funktionalen Aspekten in die parametrisch assoziative Modellgestaltung in der konzeptionellen Automobilentwicklung. In: 3. Grazer Symposium Virtuelles Fahrzeug, Graz (2010)
21. Dietrich, W.: Zur prozessorientierten Integration von Wissensmanagement in das Engineering Data Management. Phd Thesis, Graz University of Technology, Graz (2010)

Curriculum Vitae of the Authors

Mario Hirz has been awarded an M.S. degree in mechanical engineering and economics, a Ph.D. in mechanical engineering, and a venia docendi in the area of virtual product development. He is a regular lecturer at the Graz University of Technology and a frequent guest lecturer at universities and automotive manufacturers throughout Europe and Asia. As head of the research area for Virtual Product Development at the Institute of Automotive Engineering, he is responsible for different international engine and vehicle R&D projects. His research topics comprise design methods, knowledge-based engineering and efficient development processes. Dr. Hirz has published more than 120 works and has received several national and international awards for his scientific contributions.

Wilhelm Dietrich has been awarded an M.S. degree and a Ph.D. in mechanical engineering and economics at Graz University of Technology. His research activities and scientific contributions are focused on knowledge-based engineering data management. Since 2000, he has been employed at MAGNA STEYR Engineering AG & Co KG and is competent in the development of CAD and EDM methodology and systems. He was responsible for several areas of virtual product development and was project manager of a number of EDM R&D projects. As head of the vehicle architecture and function department, Dr. Dietrich is currently responsible for vehicle concepts, package layout, ergonomic and complete vehicle functions.

Anton Gfrerrer received the M.S. degree in mathematics and descriptive geometry from the University of Graz, Graz, Austria, in 1989 and the Ph.D. degree from Graz University of Technology (TU Graz) in 1992. He is currently an Associate Professor with the Institute for Geometry, TU Graz, and also lectures at the University of Leoben. His research fields are geometry, CAD, kinematics and robotics.

Johann Lang received his M.S. degree in mathematics and descriptive geometry at Graz University in 1977 and his Ph.D. degree at Graz University of Technology (TU Graz) in 1979. He is currently an Associate Professor with the Institute for Geometry, TU Graz. His research fields are geometry and kinematics.

Index

A
Adapter model, 301, 304
Added-value process, 391, 395, 397
Affine
  combination, 227
  transformation, 54
Algebraic surface, 158
Algorithm
  Aitken, 120, 210
  Cox-de boor, 194
  de casteljau
    for Bézier curves, 92
    for Bézier surfaces, 184
    for tensor product Bézier volumes, 234
    triangular case, 230
Analysis
  curvature, 276
Approximating surface, 220
Approximation
  bivariate, 216
  parameter correction, 141, 224
  univariate, 137
  with cubic B-splines, 143
Arc length, 76
Assembly design, 289, 313, 318
Assembly structure, 245, 259, 274, 294
Automated routines, 309, 311
Axis system, 252, 272, 300

B
Barycentric coordinates, 226
Basis solid, 277, 278
Bending energy, 130
Bernstein polynomials, 87
  bivariate, 228
  on a triangular domain, 228
  on an arbitrary interval, 91, 125, 184
  properties, 87
Bézier
  curve
    connecting, 98
    continuation, 95
    definition, 88
    degree elevation, 98
    modeling, 96
    properties, 90
    rational, 112
    splitting, 94
  surface, 182
    connecting, 189
    continuation, 187
    degree elevation, 186
    modeling, 188
    rational, 200
    triangular, 230
  volume, 233
    continuation, 234
    modeling, 235
Bézout's theorem, 82
Binormal vector, 80
Bivariate
  approximation, 216
  interpolation, 206, 208
    Hermite, 210
    Lagrange, 208
Blending function, 181
Boolean operation, 254, 279–281, 285
Boundary element, 256

B-spline
  basis functions, 101
  curve
    closed, 108
    definition, 102
    endpoint interpolation, 106
    local control, 103
    properties, 103
    rational, 115
    uniform, 103
  surface, 193
    closed, 195
    corner interpolation, 194
    local control, 197
    properties, 194
    rational, 201
Bumper system, 328–330
Business
  engineering, 338
  process, 334, 362, 381, 391, 394
  reengineering, 338

C
CAD, 21, 241, 359
  data management, 414, 421
  EDM integration, 347, 360
  implementation, 361
  integrated concept model, 425, 431, 446
CAE, 34, 359
CAE data management, 368
CAM, 41, 359
CAP, 359
CAPP, 359
Car body, 13, 15, 17, 250
CAT, 359
CAx, 359
Centripetal parameterization, 136
Chassis, 2, 6, 17
Chebyshev space, 117
Chordal parameterization, 136
Circle, 73, 175
  osculating, 78, 79
Circular arc, 113
C^k-continuity, 75, 149
C^k-curve, 73
C^k-surface, 147
Class
  of differentiability, 73, 147
Coincidence condition, 296
Collaboration, 47, 343
Complementary conic arcs, 114
Component positioning, 294, 296, 300, 304
Computer-aided design, 359
Computer-aided engineering, 334, 359
Computer-aided manufacturing, 359
Computer integrated manufacturing, 333, 350
Concept phase, 14, 25, 32, 327, 423
Concept vehicle, 424, 426, 432, 433, 437, 439–448, 450–454
Cone, 166
  quadratic, 162
Configuration management, 366, 368
Conic, 84, 113
Conic section, 84
Constraint, 39, 46, 245, 294
Continuation quality, 242, 267, 276, 282
Control grid, 233
Control net, 181
Control polygon, 86
Convex hull property, 90, 184, 230
Coons patch, 206
Coordinate system, 256, 296, 298–300
  Cartesian, 52
  cylindrical, 53
  main, 57, 296, 298–301
  right-handed, 57
  spherical, 53
Coupling function, 261, 262
Cox-de boor algorithm, 104, 194
Curvature, 77, 81
  of a curve, 77
  Gaussian, 155, 276
  mean, 155
  normal, 154
  of a planar curve, 77
  of a surface, 152
  principal, 155
  radius of, 78
  surface curve, 153
Curvature analysis, 276
Curve, 70, 242, 256, 276, 285
  2nd order, 81, 83
  arc length parameter, 76
  binormal vector, 80
  curvature, 77, 81
  equation, 73
  freeform, 86
  inflection point, 78, 79
  movable in itself, 174
  osculating circle, 78, 79
  osculating plane, 78, 80
  parameterization, 70
  planar, 72

curvature, 77
planar algebraic, 81
polynomial, 83
principal normal vector, 80
rational, 83
tangent vector, 80
torsion, 77, 81
transcendent, 81
Cylinder, 161, 166
Cylinder head, 41, 42, 48, 253, 284, 306

D
Data
classification, 353
demand, 399
exchange, 31, 44, 45, 252, 358
flow, 401
maturity, 419
monitoring, 422
progress, 419
quality, 419
roadmap, 416
scheduling, 416
security, 349
structure, 340, 434
supply, 399
synchronization, 409
transfer, 385, 400, 410, 412
Data management, 44
activities, 397, 404
activity, 395, 410, 412
barrier, 412
factors, 410
integrated CAD, 414, 425, 431, 434, 446
knowledge-oriented, 396
master, 354
systems, 30
Database, 344, 399, 403, 434
de Boor
net, 193
polygon, 102
de Casteljau algorithm
for Bézier curves, 92
for Bézier surfaces, 184
for tensor product Bézier volumes, 234
triangular case, 230
Degree elevation
curve, 98
surface, 186
Design
freeze, 20
knowledge-based, 309, 310, 312
non-parametric, 245
parametric-associative, 254, 259, 298, 405, 406, 422
process, 11
Developable surface, 167
Digital
engineering, 341
mock-up, 15, 22, 35, 285, 286, 290, 290, 347, 353, 423
prototype, 348, 349, 361
Dimension concept, 426
Direct modeling, 246
Divisor
greatest common, 67
of a polynomial, 67
Documentation process, 384, 387, 397, 398
Document management, 345, 350, 353, 355, 374
Door window motion, 235
Draft angle, 254, 269, 272, 321
Drivetrain, 13, 28, 36, 318, 423
Driving dynamics, 19, 423

E
EDM
application integration, 356, 364
application-related function, 351, 354
architecture, 356
CAD integration, 347, 360
configuration management, 332, 337
documents, 346
integrated management, 339
interfaces, 357
knowledge-oriented, 388, 393, 398, 399, 403, 404, 413, 452, 454
process integration, 343
process-oriented, 400
support process, 394
system, 337, 350, 453
use case, 413, 454
workflow, 332, 403, 421, 454
Ellipse, 85
Ellipsoid, 163
Elliptical point, 156
Elliptic paraboloid, 163
Engineering
database, 350, 356, 368
data management, 331, 336
data management system, 336, 350

Equation
  of a planar curve, 73
  of a surface, 157
Euclidean motion, 60
Exhaust gas emissions, 19
Exterior geometry, 439, 440
External parameter control, 315
External solver, 324, 325
Extrapolation, 262, 266, 268, 275
Extruded surface, 261

F
Feature-based modeling, 256, 282, 283
Fillet operation, 267
Fill surface, 206, 264, 265
Freeform
  curve, 86
    rational, 110
  surface, 181
Frenet frame, 79
Frenet-Serret formulae, 80
Frontloading, 14, 23, 49
Frozen geometry, 302
Function
  blending, 181
Functional layout, 423, 424, 449
Fundamental form
  first, 157
  second, 157
Fundamental theorem of algebra, 68

G
Gaussian curvature, 155, 276
GC^k-continuity, 75, 149
GCIE standard, 426, 432, 434, 444
Geometry
  check, 436, 438, 443, 444
  reference, 301, 303, 418
  representation, 362
Greatest common divisor, 67
Guide curve, 262–264

H
Helical
  displacement, 59
  gear, 174
  surface, 173, 175, 235, 237, 238
    circular, 173
    ruled, 173
Helix, 64, 71, 175
Hermite
  interpolation, 124, 210
  patch, 212
  polynomial, 126, 211
Homologation, 13, 20
Hyperbola, 85
Hyperbolic paraboloid, 163, 165, 204
Hyperbolic point, 156
Hyperboloid, 163

I
Inflection point, 78, 79
Information process, 385, 397, 398
Interior geometry, 440, 441
Interpolating cubic B-spline surface, 214
Interpolation
  bivariate, 206
  cubic spline, 127
  Hermite, 124, 210
  Lagrange, 119, 208
  univariate, 116
  with cubic B-splines, 130, 214
  with cubic segments, 122
Isometry, 54
  even, 58
  odd, 58

J
Join function, 266

K
Knot
  multiple, 106, 194
Knot vector, 99, 101, 102, 115, 119, 127, 137, 143, 193, 202, 206, 207, 210, 214
  centripetal, 136
  chordal, 136
  non-uniform, 103, 202
  uniform, 103, 135, 194, 202
  with multiple knots, 106
Knowledge, 375
  activity, 412
  base, 345, 379, 415
  carrier, 376, 398, 400
  collective, 376, 379
  database, 44, 49
  declarative, 377
  definition, 375

explicit, 377, 378
individual, 376
implicit, 377, 378
induction, 384
logistics, 388
management, 309
operation, 410
organizational, 379
preserve, 378
procedural, 377
process, 391
stairs of, 372, 376
structuring, 376
system, 379, 382
tacit, 378
Knowledge-based
data management, 399, 400, 452
design, 309, 310, 312, 405
engineering, 405
engineering data management, 455
process analysis, 412
process management, 390
Knowledge management, 375, 379, 423
basic model, 380
process orientation, 389, 390
system orientation, 382
Knowledge transfer, 383
barrier, 412
direct, 386
indirect, 386, 412
influences, 410
process, 384
Knowledgeware, 404

L
Lagrange interpolation, 119, 208
Lagrange polynomials, 119
Library based design, 320, 321, 323
Line, 70, 175, 256, 285, 294, 298
coordinates, 62
geometry, 62
reflection in a, 57
Linkage strategies, 314
Local control
B-spline curve, 103
B-spline surface, 197

M
Macro routines, 252
Master model, 33, 317, 326, 328–330
Matrix
orthogonal, 54
tridiagonal, 130, 134
Mean curvature, 155
Meridian plane, 170, 173
Modeling
Bézier surface, 188
feature-based, 282, 283
Model range, 1, 2
Model structure, 246, 247, 249, 250, 252, 301
Mold design, 280, 284
Monomial basis, 67
Motion
Euclidean, 60
in itself, 174
Multi-model link, 247, 289
Multi-section surface, 279

N
Net
rectangular, 181
triangular, 229
Normal curvature, 154
Normal vector
principal, 80
NURBS, 115
NURBS surface, 202

O
Offset condition, 296
Operating process, 401
Orientation, 58
Osculating
circle, 78, 79
plane, 78, 80
Overhauser subspline, 124

P
Parabola, 85
Parabolic point, 156
Paraboloid
elliptic, 163
hyperbolic, 163, 165
Parameter
domain, 70, 145
line, 147, 233
surface, 233
transformation
of a curve, 72
of a surface, 146
Parameterization, 134, 310, 312, 320, 434, 435
admissible, 72, 145
centripetal, 136

chordal, 136
of a curve, 70, 72
of a surface, 144, 145
smooth, 73
uniform, 135
Parameter line, 227
Parametric-associative
concept model, 422
design, 254, 259, 298, 405, 406, 422
Parametric modeling, 32, 246
Parent-child relation, 247
Partition of unity, 86
Pattern function, 268
PCA, 216
Planar algebraic curve
order, 81
Plane
equation, 159, 257
fitting, 216
osculating, 78, 80
parametric representation, 145, 226
reflection in a, 55
tangent, 148
Plücker coordinates, 62
Point, 52, 246, 256–258, 278, 285, 294, 296, 298, 299, 303
elliptical, 156
hyperbolic, 156
parabolic, 156
reflection in a, 58
umbilic, 155
Point cloud, 242, 256
Polygon
control, 86
de Boor, 102
Polynomial, 66
basis, 67
Bernstein, 87
bivariate, 68
degree, 66
function, 69
Hermite, 126, 211
Lagrange, 119
multivariate, 68
root of a, 68
space, 66
trivariate, 68
univariate, 66
zero of a, 68
Polynomials
product, 67
Positioning
component, 300
Position vector, 53
Principal component analysis, 216
Principal curvature, 155
Principal normal vector, 80
Process
added-value, 391, 395, 397
knowledge, 378
knowledge-intensive, 395, 403
of documentation, 397, 398
of information, 397, 398
Process management, 338, 394
knowledge-oriented, 390
Process planning, 341
Product
configuration, 366, 407, 418
data, 19, 31, 44, 353
data management, 10, 21, 25, 30, 43, 248, 252, 290, 299, 332, 351, 352
data management system, 350
development process, 331, 341, 416, 428, 446, 450
life cycle management, 335
structure, 352, 418
Product knowledge, 371, 373
concept development, 450
life cycle, 373
management, 351, 374
Production confirmation, 20
Production planning and control, 336, 337
Profile curve, 261, 263, 264, 272
Project
management, 353
milestones, 416, 418
Propulsion concept, 16, 19
Prototype freeze, 20

Q
Quadric, 160
classification of, 161
of revolution, 171, 177

R
Rational
B-spline curve, 115
B-spline surface, 201
Bézier curve, 112
circular arc, 113
of degree 2, 113
Bézier surface, 201
curve
degree, 83
freeform curve
properties, 111

function, 69
surface, 159
tensor product surface, 199
Real-time simulation, 39
Reflection
axial, 57
in a line, 57, 58
in a plane, 55
in a point, 58
Regression
plane of, 216
Revolution
continuous, 61
surface of, 170, 201
Robust approximation, 140
Root of a polynomial, 68
Rotation, 57
Ruled surface, 165

S
Sand core, 253–255, 281
Scaling operation, 268
Scattered points, 216, 220
Screw
displacement, 59
line, 64, 71
motion, 61, 64, 236
parameter, 61
Seating position, 15, 16
Series development, 7, 13, 25, 249
Sheet metal part, 250, 270, 283, 285
Side window, 235
Simulation data management, 365, 368, 369
Simultaneous engineering, 36, 48, 342
Skeleton model, 259, 260, 285
Sketch, 242, 252, 256, 257, 277
Skin surface, 278
Solid design, 276, 277, 286
Spatial transformation, 54
Sphere, 158
Spline
curve, 99
function, 99
surface, 192
Split operation, 266, 286
Startup model, 249–251, 296
Steel mold, 253, 254
Styling fillet, 268
Styling freeze, 19
Subspline
function, 99
overhauser, 124
surface, 192
Supply chain management, 341
Surface, 242
2nd order, 160
algebraic, 158
B-spline, 193
Bézier, 182
triangular, 230
curvature, 152
curve, 146
developable, 167
equation, 157
fitting, 220
helical, 173, 175, 235, 237, 238, 261
circular, 173
ruled, 173
intersection, 176
movable in itself, 174, 235
multi-section, 261–263, 278
offset, 286, 287
of revolution, 168, 170, 258, 261
parameterization, 145
rational, 159
ruled, 165
tensor product, 181
transcendent, 159
translational, 261, 263, 264, 272, 289
Surface design, 260, 277, 278, 285
Suspension, 6, 15, 36, 39, 291
Swept surface, 263

T
Tangent
plane, 148
to a curve, 74
Tangent vector, 80
Target definition, 13
Target specifications, 17, 18, 27
Task
carrier, 400
structure, 401
Technical subsystem, 400
Template library, 321
Template model, 309, 312, 317, 329, 330, 430
Tensor product
surface, 181
rational, 198
volume, 232
Tessellated geometry, 36, 41, 45
Torsion, 77, 81
space curve, 77
Torus, 172
Trajectory
of a motion, 61

Transformation, 241, 268, 287
  affine, 54
Translation, 56
Translational surface, 261, 263, 264, 272, 289
Triangular Bézier patch, 230
Triangular patch, 225
Trim operation, 266, 267, 272, 274
Tubular helical surface, 174

U
Umbilic point, 155
Unfolding, 269
Uniform parameterization, 135
Univariate
  approximation, 137–139
  interpolation, 116, 119

V
Vandermonde matrix, 118
Variation diminishing property, 90
Vector function
  derivative, 73
Vector space, 66
Vehicle concept, 1, 10, 16, 424
Vehicle package, 423, 447
Virtual
  concept vehicle, 428, 446, 450
  engineering, 341
  environment, 21, 29, 241, 286
  mock-up, 33, 348
  product, 30, 348
  product development, 29, 338, 341, 428, 446, 450
  prototype, 361, 368, 418
  reality, 39, 364
Volume, 242

W
Workflow management, 421
Working space, 241, 256, 285, 289, 298

Y
Y-CIM model, 332

Z
Zero of a polynomial, 68