Engineering with Computers (1999) 15: 155–170
© 1999 Springer-Verlag London Limited

An Integrated Approach for Computer Aided Design in Multibody System Dynamics*

A. Daberkow[1] and E.J. Kreuzer[2]

[1]Research Institute Frankfurt, Daimler-Benz AG, Frankfurt, Germany; [2]Ocean Engineering II and Structure Mechanics Section, Technical University Hamburg–Harburg, Hamburg, Germany

Abstract. *In this paper the integration of Solid Modeling CAD (Computer Aided Design) systems within a dynamic simulation environment is described. According to the different multibody formalisms and the computer codes, a general description and an organization of multibody system data are introduced to fulfill the needs of a modular simulation system based upon numerical and symbolical formalisms. With respect to different solid model construction methods, the multibody system components are generated by the extraction of an extended CAD database. A system-independent modeling kernel library is developed from the object-oriented construction of abstract data types and operations. The basic steps and the advantages in this new automated mechanical design chain are demonstrated by examples.*

Keywords. CAD; Data exchange; Dynamic simulation; Mass properties; Multibody system; Object-oriented design; Product data management; Solid modeling; Vehicle dynamics

1. Introduction

Nowadays, powerful computer programs are able to formulate the equations of motion and to simulate the mathematical model of even complex mechanical systems efficiently. To improve the transformation of the real system into a multibody system, the interaction of multibody formalisms with CAD systems has to be pursued urgently.

The modeling of a mechanical system by means of a multibody system is characterized by a composition of rigid bodies, interconnected by joints, springs, dampers and actuators (see Fig. 1). Force elements like springs, dampers and actuators acting in discrete attachment points result in applied forces and torques on the rigid bodies. Joints with different kinematic properties constrain the motion of the bodies of the system, determine the degree of freedom of the multibody system, and result in constraint forces and torques.

The main fields for industrial applications of such multibody systems are vehicle dynamics, machines and mechanism dynamics, robot dynamics and biomechanics. A classification of multibody formalisms may be obtained according to the principles of mechanics applied, the number of equations to determine the motion of the system, the system topology, or the way in which the system equations are generated, i.e. either in numerical or symbolical representation. A considerable number of computer codes have been developed for numerical equation generation, i.e. ADAMS [1] or DADS [2]. In the last decade, formalisms for the generation of symbolic equations of motion also became common. Computer programs like SD-Exact [3], NEWEUL [4,5] and RASNA [6], provide explicit analytical expressions for the system equations, including numerical values for the input parameters. A survey of the different formalisms and the computer codes

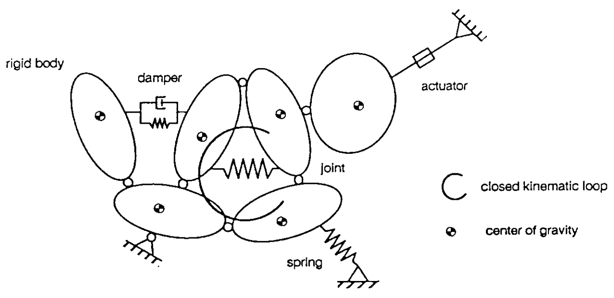


Fig. 1. Multibody system.

*This work was performed at the Institute B of Mechanics, University of Stuttgart, Stuttgart, Germany.

Correspondence and offprint requests to: Professor E. Kreuzer, Arbeitsbereich Meerestechnik II-Mechanik, Eissendorfer Straβe 42, D-21073 Hamburg, Germany. Email: kreuzer@tu-harburg.de

in multibody dynamics can be found in Schiehlen [7].

Today, CAD systems are crucial tools for designers in many areas of sciences, especially in mechanical design and engineering. In the early stages of development, CAD systems were merely used to assist in the two-dimensional drawing process. Nowadays, these systems are widely embedded in the industrial design and the construction process, while the wide application of three-dimensional Solid Modeling CAD systems is rare. Their use to support an analytically and topologically complete model, an interference detection, the tool path geometry or the calculation of surface and volume properties is then closely related to the geometric representation of solid models (see Mortenson [8] and Pahl [9]).

The design of a rigid body with a solid modeling CAD-system is characterized by a complete and exact data description of the three-dimensional object, either in constructive solid geometry or boundary representation. The geometric modeller PARASOLID [10] implemented in a considerable number of CAD systems maintains a boundary representation of solids. A survey of the modeling methods and the features of different CAD-systems is given in Puig-Pey and Brebbia [11] including newly developed rule-based, hybrid or object-oriented concepts.

Encouraged by the increasing applications of multibody formalisms and CAD systems, the coupling of solid modeling CAD systems with multibody simulation software became more and more attractive. For the numerical computer code ADAMS, a number of interfaces have been developed for the specific CAD systems (e.g. ARIES [12]). A further interface to the multibody formalism DADS is provided by the CAD-system Pro/ENGINEER [13]. All these systems have one thing in common: they support the calculation of mass properties from the solid model designed in the CAD system. With additional mechanical properties, such as the locations of applied forces and torques and of joint attachment points, the solid models represent the rigid bodies of the multibody system.

While further investigations are made to improve the automated exchange of design data between different vendor's CAD-systems, the STEP (STandard for the Exchange of Product model data) proposal [14] endeavors to evaluate a generalized design data model, including the coupling to analysis tools for finite element and kinematic analysis. At present, there is no STEP product model available for dynamic analysis. A first effort to develop a vendor-independent generalized data model for multibody systems, including preprocessing with CAD systems, was described by Otter et al. [15], and will be explained in this paper.

2. Background and Motivation

The combination of computer aided design and simulation of the behavior of multibody systems in an automated design and a concurrent engineering environment is desirable for various reasons. As multibody models become increasingly complex, the amount of engineering knowledge needed by a single designer has sometimes increased to an unmanageable level. This justifies a combined design with the dynamic analysis approach based upon advanced CAD-systems. In particular, a system dynamics investigation requires the basic parameters of mass, center of gravity and moments of inertia, without considering the geometry model, the modeling method and the solid model construction of the CAD system in use. A high degree of modularization demands an exchange of complete or single object data of a multibody system. Moreover, simulation results of a dynamic analysis have to be transferred to a CAD or graphics system without concern with, or restrictions on, the geometry model representation.

Furthermore, an efficient design process of mechanical parts requires a general interface to multibody computer codes (Fig. 2). This interface is claimed to serve as a compatible and comfortable post processor, concerning the different algorithms and implementations of multibody dynamics computer codes.

Moreover, geometric information about the attachment points of joints and force elements is expected to be evaluated in the design state. Commercially available multibody modeling software tools within CAD systems are mostly dedicated to one particular multibody dynamics computer code. Often, no options are supplied for a parametric multibody system description, or the modeling is restricted to either robot, mechanism or vehicle dynamics. Consequently, the formulation and motivation of a problem leads to the following methodology:

- Comprise the necessary data describing a multibody model for the different multibody programs.
- Examine the different geometry models of CAD systems for solids, and extract the relevant data for multibody systems.
- Define a geometry model for the representation of multibody elements in CAD-systems.

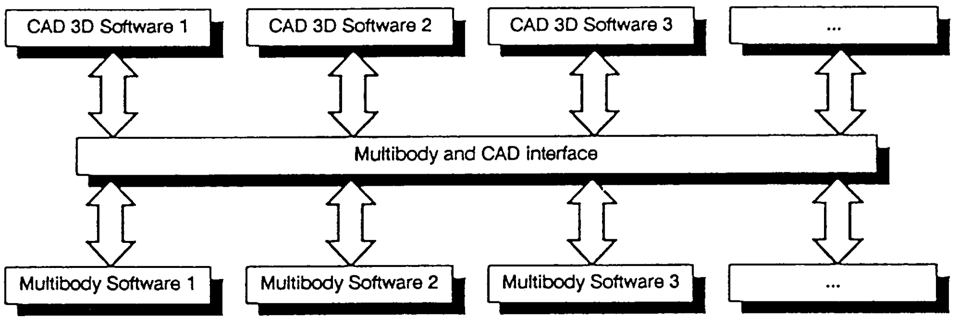


Fig. 2. Ideal automated model conversion in the product development process.

- From these objectives, design data types and operations and construct a software interface for a system-independent modeling of multibody systems.

3. Fundamentals of Multibody System Dynamics

The kinematics of general spatial multibody systems is described by means of reference frames for determining the rigid body location. On the basis of these body-fixed reference frames, the positions of joint and force elements are defined. The two basic concepts applied in multibody dynamics to derive the equations of motion are the Lagrangian approach, with so-called *Cartesian coordinates* in a numerical representation on the one hand, and the Newton–Euler formalism, with so-called *Lagrangian generalized coordinates* and the symbolical implementation on the other.

3.1. Cartesian Coordinate Approach

In the cartesian coordinate approach, a set of cartesian coordinates which is independent of the degree of freedom of the multibody system,

$$\boldsymbol{x}_i = [r_{ix} r_{iy} r_{iz} \phi_i \theta_i \psi_i] \tag{1}$$

is chosen for each body. In Eq. (1), the body's position is described by the 3×1 vector $\boldsymbol{r}_i$ and its orientation is specified by the 3×1 vector of the Euler angles. In technical systems, constraints restrict the relative position and orientation of bodies in a pair. A multibody system with nb bodies and f degrees of freedom has $q = 6 \cdot nb - f$ constraints. The kth body-fixed frame K_k on body i is represented by the vector to its origin ${}_i\boldsymbol{r}_{ik}$ and its rotation matrix $\boldsymbol{S}_{ik} = [{}_i\boldsymbol{e}_{k,1}, {}_i\boldsymbol{e}_{k,2}, {}_i\boldsymbol{e}_{k,3}]$ relative to the reference frame K_i of body i. (Fig. 3).

From the basic conditions for parallelism and orthogonality for each pair of frame unit vectors, a joint description library is derived, resulting in a $q \times 1$ constraint vector equation

$$\boldsymbol{\Phi}(\boldsymbol{r}_i, \boldsymbol{S}_i, \boldsymbol{s}_j, \boldsymbol{S}_j) = \boldsymbol{\Phi}(\boldsymbol{x}, t) \tag{2}$$

which describes the overall constraints of the multibody system with n_j joints, each with an $r \times 1$ constraint vector equation. The translational and angular velocities, as well as the accelerations of each body, are determined from Eq. (2) by differentiation.

The 6*nb* nonlinear equations of motion for spatial dynamics in the generalized Cartesian coordinates approach are derived from the D'Alemberts principle, to obtain

$$\bar{\bar{\boldsymbol{M}}}\ddot{\boldsymbol{x}} + \bar{\bar{\boldsymbol{k}}} + \boldsymbol{\Phi}_x^T \boldsymbol{\lambda} = \boldsymbol{f}^A \tag{3}$$

where

$$\bar{\bar{\boldsymbol{M}}} = \text{diag}\ (m_1 \boldsymbol{E}, ..., m_{nb} \boldsymbol{E}, \boldsymbol{I}, ..., \boldsymbol{I}_{nb}) \tag{4}$$

is the $6 \cdot nb \times 6 \cdot nb$ global inertia matrix with the mass m_i of body i, and $\boldsymbol{I}_i$ is the 3×3 is inertia matrix of the ith body defined with respect to the body-fixed reference frame located in the centre of gravity, $\boldsymbol{\lambda}$ is the $q \times 3$ vector of Lagrangian multipliers, $\boldsymbol{k}$ the $6 \cdot nb \times 1$ vector of centrifugal forces, and $\boldsymbol{f}^A$ denotes the $6.nb \times 1$ vector of the applied forces and torques, which in reality are complicated functions of $\boldsymbol{x}$ and further external signals. With Eqs (2) and (3), the Lagrangian equations of the first kind as a system of mixed second order differential-algebraic equations have to be solved.

3.2. Lagrangian Generalized Coordinate Approach

For the Lagrangian coordinate approach, a set of state variables depending on the f degrees of free-

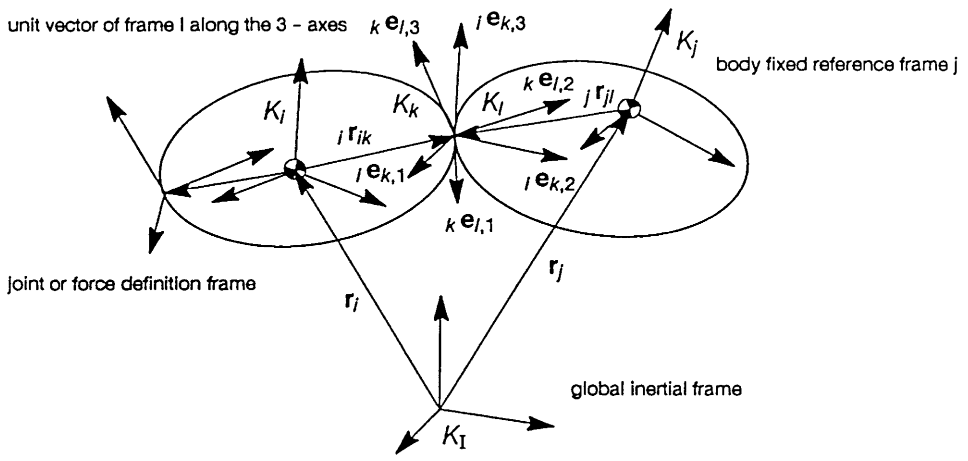


Fig. 3. Body fixed reference frames.

dom of the multibody system is determined from coordinates representing relative motions in the system. For a chain or tree structure topology in a multibody system (see Fig. 1), a $f \times 1$ vector of generalized coordinates $\boldsymbol{y}$ summarizing the relative joint coordinates between each pair of rigid bodies is introduced. Assuming that the relative motion of rigid bodies is described by the relative position and orientation of joint definition frames K_k and K_l, a position vector

$$_k\boldsymbol{r}_{kl} = {_k\boldsymbol{r}_{kl}}(\boldsymbol{y}) \tag{5}$$

and a rotation matrix

$$\boldsymbol{S}_{kl} = \boldsymbol{S}_{kl}(\boldsymbol{y}) = \boldsymbol{S}_{kl}(\alpha_{kl},\beta_{kl},\gamma_{kl}) \tag{6}$$

are defined with respect to the joint definition frame (cf. Fig. 3). The rotation matrix (6) is given by an angle representation of consecutive elementary rotations about consecutive joint definition frame axes. From the basic relations of position and rotation between joint definition frame axes, the location of the jth body is obtained recursively by

$$\boldsymbol{S}_j(\boldsymbol{y}) = \boldsymbol{S}_i(\boldsymbol{y})\boldsymbol{S}_{ik}\boldsymbol{S}_{kl}(\boldsymbol{y})\boldsymbol{S}_{jl}^T \tag{7}$$
$$\boldsymbol{r}_j(\boldsymbol{y}) = \boldsymbol{r}_i(\boldsymbol{y}) + \boldsymbol{S}_i(\boldsymbol{y})_i\boldsymbol{r}_{ik} + \boldsymbol{S}_i(\boldsymbol{y})\boldsymbol{S}_{ikk}\boldsymbol{r}_{kl} - \boldsymbol{S}_j(\boldsymbol{y})_j\boldsymbol{r}_{jl}$$

The translational and angular velocities $\boldsymbol{v}_j$ and $\boldsymbol{\omega}_j$, as well as the accelerations $\boldsymbol{a}_j$ and $\boldsymbol{\alpha}_j$ of each body, are again determined by differentiation.

A systematic choice of the vector $\boldsymbol{y}$ from the number of joint degrees of freedom needs further information in case of closed kinematic loops, (cf. Fig. 1), because the number of degrees of freedom is smaller than the number of relative joint coordinates. Several methods exist to select a proper set of independent generalized coordinates (e.g. see Wehage and Haug [16]); here the method proposed by Leister and Bestle [17] is used to choose independent variables in the vector $\boldsymbol{y}$. By cutting the closed kinematic loops of the multibody system, the independent generalized coordinates are always chosen as a linear combination of the relative joint coordinates. This linear combination and the independent generalized coordinates are specified automatically during the numerical simulation.

The nonlinear equations of motion for a multibody system are given by

$$\boldsymbol{M}\ddot{\boldsymbol{y}} + \boldsymbol{k} = \boldsymbol{q}^e \tag{8}$$

$$\boldsymbol{g}(\boldsymbol{y},\boldsymbol{h},t) = \boldsymbol{0} \tag{9}$$

where

$$\boldsymbol{M} = \sum_{i=1}^{nb}(\boldsymbol{J}_{Ti}^T m_i \boldsymbol{J}_{Ti} + \boldsymbol{J}_{Ri}^T \boldsymbol{I}_i \boldsymbol{J}_{Ri}) \tag{10}$$

is the $f \times f$ mass matrix with the mass m_i and the inertia matrix $\boldsymbol{I}_i$ of each body, $\boldsymbol{J}_{Ti}$ and $\boldsymbol{J}_{Ri}$ are the $f \times 3$ translational and rotational Jacobians resulting from the differentiation of the position vectors and the orientation matrices. The $f \times 1$ vector $\boldsymbol{k}$ denotes the vector of the centrifugal and coriolis forces, and the $f \times 1$ vector $\boldsymbol{q}^e$ the applied forces and torques. In case of closed loop systems, the Jacobians $\boldsymbol{J}_{Ti}$ and $\boldsymbol{J}_{Ri}$ are functions of a joint coordinate vector $\boldsymbol{h}$ and their partial derivatives. Here, besides the set of ordinary differential equations the implicit algebraic Eq. (9) has to be solved numerically for each given state $\boldsymbol{y}$ and $\boldsymbol{y}$. In case of tree structured systems, Eq. (9) vanishes and the vector of generalized coordinates $\boldsymbol{y}$ is equal to the vector $\boldsymbol{h}$ representing the joint degrees of freedom.

3.3. Conclusions for Basic Multibody Model Data

To devise a unified data model for both coordinate approaches, the multibody model data are classified on a conceptual level. For symbolic as well as numeric formalisms, a generalized classification relies upon the basic modeling elements of *frame, body, joint* and *force*.

To specify the joint and force definition frames, the position and orientation of each frame are determined with respect to the reference frame of the body, which has its origin in the mass center of the body. This position vector ${}_i\boldsymbol{r}_{ik}$ between the reference frame of body i and frame k is additionally supplied by symbolical variables for each vector coordinate. According to the three degrees of freedom of rotation, an angle representation is chosen to describe the frame orientation. By supplying additional rotation sequence information and symbolic variables for the rotation angles, a *full parametrization* is achieved for later parameter studies of joint and force element attachment points.

For each joint, the common joint definition frames define the connection between two bodies. With respect to the joint type, the directions of translation and the axes of rotation yield the characteristic relative joint position vector (5) and the joint matrix (6). Additionally, an initial offset of the joint coordinates is supplied. Figure 4 shows the *revolute* and *translational* joint definition frames of a joint library, as well as the joint position vector and the rotation matrix. From this information, the implicit constraint Eq. (2) of the Cartesian coordinate approach and an initial set of cartesian coordinates are determined. For tree-structured multibody systems and the Lagrangian coordinate approach, the explicit constraint formulation in the position vector $\boldsymbol{r}_j$ and the orientation matrix $\boldsymbol{S}_j$ in Eq. (7) yields the location of each body. For systems with closed loops, the offset joint coordinates serve as initial values for the method described by Leister and Bestle [17].

For each body, the mass and the inertia matrix are needed. The time variant representation of the inertia matrix $\boldsymbol{I}_i$ in Eq. (10) is obtained by

$$\boldsymbol{I}_i = \boldsymbol{S}_i \, {}_i\boldsymbol{I}_i \, \boldsymbol{S}_i^T \tag{11}$$

The constant elements of the inertia matrix ${}_i\boldsymbol{I}_i$ with respect to the body-fixed reference frame and the mass m_i are again supplied by a list of symbols in order to achieve a full parametrization.

The force definition frames for internal and external force elements serve to determine the actual lengths and the velocities of spring, damper and actuator elements. Symbolic variables for force actuators support parameter studies of different force characteristics. In the case of a general force law, the desired force characteristics have to be supplied or user-formulated as a function of the body location, the velocity and the acceleration. A classification of force and torque laws in the Lagrangian coordinate approach is given by Schiehlen [18].

4. A Short Survey on Geometric Modeling of Spatial Bodies in CAD Systems

With respect to the needs in multibody systems, the dominating constructive solid geometry and boundary representations are considered. Solid modeling fundamentals and construction methods are illustrated in detail by Mortenson [8] and Pahl [9].

4.1. Solid Body Construction and Geometry Models

The Boolean combination of two or more primitive objects to a new solid object is the main characteristic of a CSG (Fig. 5).

Here, the information about the transformation and Boolean operation, as well as the generation history of the primitives, is stored. For two-

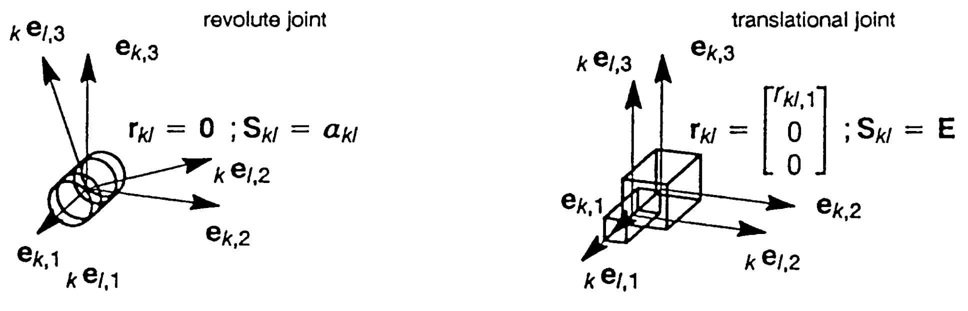


Fig. 4. Joint definition frames with position vector and rotation matrix.

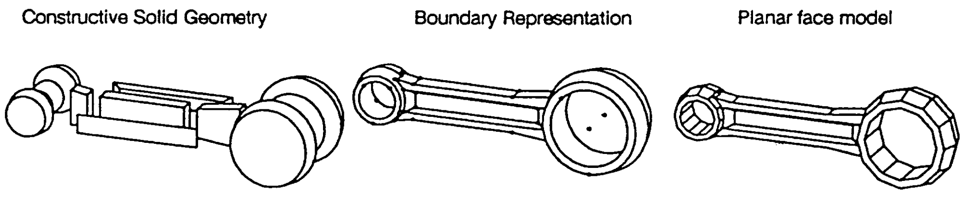


Fig. 5. CSG, B-Rep and Face geometry models of a solid.

dimensional projections of the CSG model, an equivalent wire or face model has to be derived from the binary tree of the primitives and their transformations. A B-rep model also allows the Boolean combination of primitive objects. Each primitive object and the actual modeling state are described by a complete spatial boundary, of which the topological validity may be checked by the application of the Euler operators to the enclosing faces, edges and vertices [8].

An access to these entities for local alterations and a derivation of equivalent wire or face model projections is easy. Faces, edges and vertices of the solid model of a connecting rod become evident from Fig. 5.

4.2. Calculation of Mass Properties

The mass property calculation methods differ according to the solid construction method. For the global properties of volume, surface area, moment of inertia and center of gravity, integral relations like

$$I = \int_{Solid} f^V \, dV \tag{12}$$

have to be evaluated (e.g. see Mortenson [8]) where $f^V = f^V(x, y, z)$ denotes a scalar property function. In particular, integral (12) can be used to calculate the centre of gravity $[x_c y_c z_c]^T$ for

$$f_x^V = \frac{x}{m}, \quad f_y^V = \frac{y}{m}, \quad f_z^V = \frac{z}{m} \tag{13}$$

Boundary representations allow the evaluation of integral (12) via surface integrals. From the Gauss theorem, it follows that

$$\int_{Solid} f^V \, dV = \int_{Solid} \operatorname{div} \boldsymbol{g}^V dV = \sum_{m=1}^{nf} \boldsymbol{g}^V \boldsymbol{n}_m \, dF_m \tag{14}$$

where $\boldsymbol{F}_m$ denotes the enclosing *m*th face of the solid with n_f faces and unit normal vectors $\boldsymbol{n}_m$. Figure 5 shows, as a special case of the B-rep, a *planar face model*, surface entities of which merely consist of planar faces.

Using the mass property calculation of PARASOLID implemented in a C programming language function `MASSPR`, one obtains from the input of one or more solid objects, each of which is supplied with a physical or a unit density attribute:

- `PERIPH`, the total surface of the objects,
- `VOLUME`, the total volume of the objects,
- `MASS`, the total mass of the objects,
- `COFG`, a 3×1 vector (array) describing the center of gravity of the objects with respect to the global CAD inertial frame, and
- `INERT`, a 3×3 intertia matrix (array) with respect to the center of gravity and the parallel axes to the global CAD 3D inertial frame.

It is obvious that these output data can be related directly to the input entities needed for the multibody modeling element *body*. From the center of gravity, the position vector $\boldsymbol{r}_i$ is determined automatically for an initial position of the body-fixed reference frame (Fig. 3). Choosing the axes of the reference frame parallel to the axes of the global CAD 3D inertial frame, the components of the inertia matrix ${}_i\boldsymbol{I}_i$ in Eq. (11) and the mass m_i are calculated from PARASOLID.

4.3. Further Geometry Models

Further geometry models which do not necessarily have volume properties exist. The planar face model shown in Fig. 5 also serves as a geometry model for high-speed 3D-visualization, and is implemented in graphic standards like PHIGS [19] or the SGI graphics language [20]. A single property of a solid can be derived from the face normal specifying the inner and outer parts of an object, while the coincidence of the vertices of adjoining faces is not guaranteed. The geometric modeling by *parametrized shapes* is appropriate for geometric objects, the shape of which is uniquely defined by a restricted number of parameters. Examples of parametrized shapes with an equivalent wire representation are

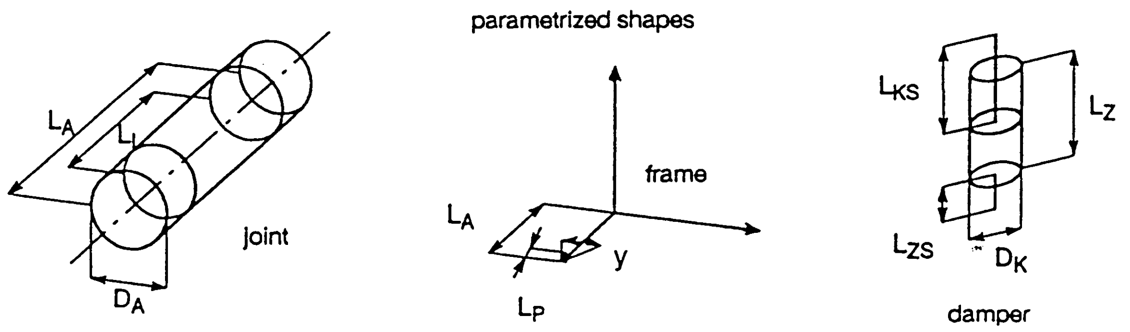


Fig. 6. Parametrized shape geometry model.

shown in Fig. 6. It is obvious that the parametrized shapes are well suited to serve as a geometry model for the multibody modeling elements *frame, joint* and *force*. Moreover, the relevant *results* of the mass property calculation modules of CAD systems are not dependent upon the geometry model of the CAD system.

5. Design of an Object-Oriented Data Model for Multibody Systems

From the proceeding sections, it follows that automated modeling and simulation requires a uniform description of multibody *and* CAD 3D modeling elements. Therefore, a data model has been defined for multibody systems to realize its organization in a hierarchical, object-oriented manner (see Otter et al. [15]. This basic data model represents nearly exclusively the method-independent and nonredundant data describing a multibody system.

By following object-oriented software techniques, the requirements of reliable and extendable programs are satisfied. While extreme differences exist in the definition and validation of the object-oriented approaches, the uniform idea is a software architecture focusing on the physical data and their relationship, rather than the program flow. Further principles of object-oriented design can be found in Meyer [21]. The multibody system data model means that *classes* are defined for the elements *frame, body, joint* and *force*, and additional *operations* are valid for these classes.

Especially for the proposed CAD integration, the data model is designed for the CAD 3D volume property calculation results, and includes the data for multibody formalisms in numerical and symbolical implementation. Figure 7 shows the general data communication flow of modules in the object-oriented multibody approach.

The modular design is of practical importance concerning the solid model preprocessing. While one or several bodies of a multibody system have been designed in a CAD system, other kinematic or kinetic data may be preprocessed by means of other user submodules. The strict separation of time invariant and time variant data allows the modeling and preprocessing of solid bodies with joint and force definition frame data from CAD systems, while time variant interactions are comprised in joint and force definition data.

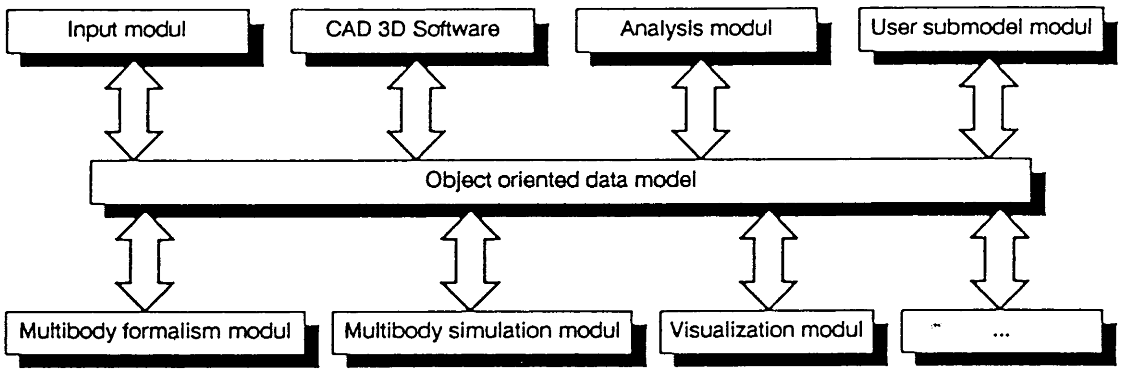


Fig. 7. Modules in an object-oriented multibody system data model.

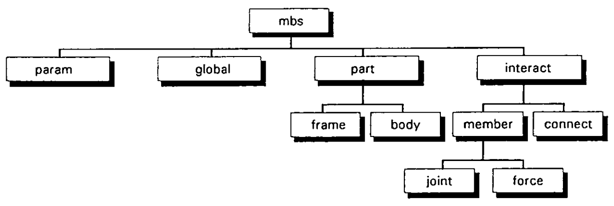
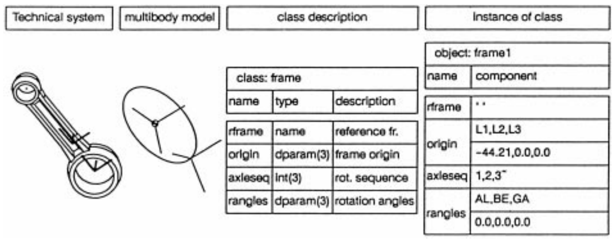


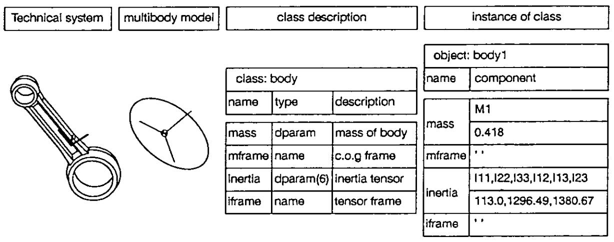
Fig. 8. Multibody modeling elements in a hierarchical structure.



class: frame		
name	type	description
rframe	name	reference fr.
origin	dparam(3)	frame origin
axleseq	int(3)	rot. sequence
rangles	dparam(3)	rotation angles

object: frame1	
name	component
rframe	' '
origin	L1,L2,L3
	-44.21,0.0,0.0
axleseq	1,2,3"
rangles	AL,BE,GA
	0.0,0.0,0.0

Fig. 9. Object of class *frame*.



class: body		
name	type	description
mass	dparam	mass of body
mframe	name	c.o.g frame
inertia	dparam(6)	inertia tensor
iframe	name	tensor frame

object: body1	
name	component
mass	M1
	0.418
mframe	' '
inertia	I11,I22,I33,I12,I13,I23
	113.0,1296.49,1380.67
iframe	' '

Fig. 10. Object of class *body*.

5.1. Multibody Classes and Operations

The key qualities of object-oriented software construction are the design and implementation of abstract data types and operations (see Meyer [21]). By the definition of *classes*, a representation scheme (*type*) of a quality of objects is determined. According to the characteristics of the class, each created *instance* (*object*) of a class is given by its *components*.

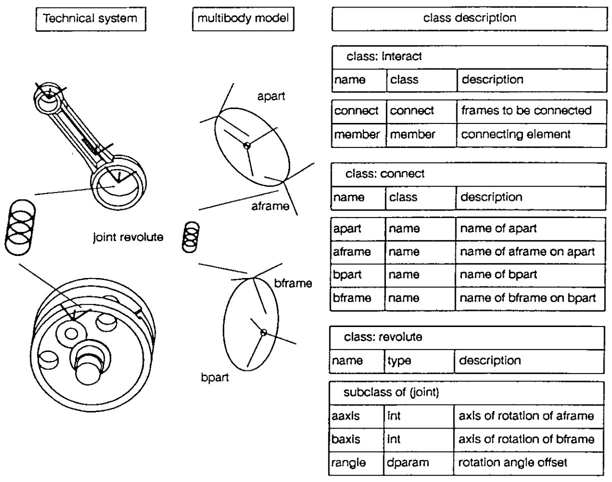
Valid *preconditions* are supplied as a fixed member of the class definition. From the physical multibody modeling elements, the classes *frame, body, joint, force, interact, global* and *param* are defined. These multibody modeling elements can be structured hierarchically (Fig. 8).

The objects of class *frame* belong to objects of class *part*; each *part* has an object of class *body* supplied with the equivalent mass properties. An object of class *mbs* serves as a superior node for the objects of classes *part* and *interact*. Figure 9 shows the components of an object of class *frame* which are determined from the mechanical and mathematical frame properties. Besides the name qualifier of the corresponding reference frame, the origin and orientation are described by a component *dparam*, comprising symbolic and numeric values. By the nonredundant specification of the rotation sequence vector, a variety of rotation descriptions, including Euler and Cardan angles, are supported. At least one object of class *frame* needs a definition with respect to the unique body-fixed reference frame.

A symbolical and numerical description is also included for the class *body*. Figure 10 shows that the components of the inertia matrix and mass are supplied by their numerical values, and a location of the center of gravity different from the body-fixed reference frame is taken into consideration by reference to an equivalent object of class *frame*.

Coupling elements of a multibody system are collected in a class *interact*. Interactions are valid between two objects of class *frame* on different objects of class *part*, either caused by an object of class *joint* or objects of class *force*. Again, the object components are designed according to the mechanical and mathematical joint and force properties. Different joint types are considered by equivalent subclasses, which *inherit* the properties of their superior class *joint* and *force*, respectively. Figure 11 shows the components of the classes *interact*, *connect* and *revolute*. A variety of rotation definitions are obvious from the *revolute* class. The rotation axes are chosen from the related objects of class *frame*, and an estimate for the initial joint orientation is provided by the component *rangle*.

Further important classes like *force* and a superior class *mbs* containing all objects of a multibody system are also defined. In the class *global*, the global properties of a multibody system are defined,

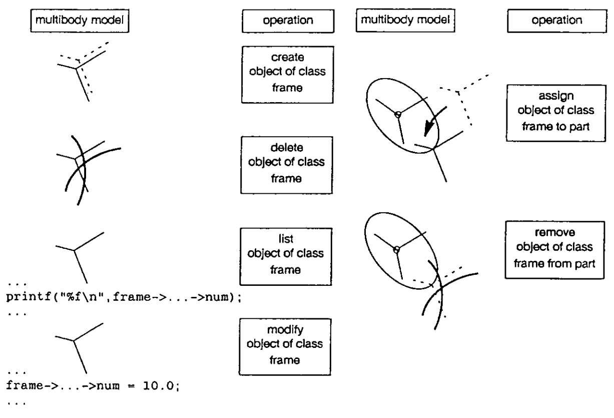


class: interact		
name	class	description
connect	connect	frames to be connected
member	member	connecting element

class: connect		
name	class	description
apart	name	name of apart
aframe	name	name of aframe on apart
bpart	name	name of bpart
bframe	name	name of bframe on bpart

class: revolute		
name	type	description
subclass of (joint)		
aaxis	int	axis of rotation of aframe
baxis	int	axis of rotation of bframe
rangle	dparam	rotation angle offset

Fig. 11. Components of class *interact*.


multibody model
operation
multibody model
operation
create object of class frame
delete object of class frame
list object of class frame
modify object of class frame
assign object of class frame to part
remove object of class frame from part

```
...
printf("%f\n",frame->...->num);
...
...
frame->...->num = 10.0;
...
```

Fig. 12. Object oriented operations for class *frame*.

like direction and magnitude of the gravity acceleration. In the class *param*, the symbolic variables of all multibody system objects are comprised [15].

Due to object-oriented software construction techniques, the composition of abstract data types in classes further demands a description of the *operations* valid on the process. Therefore, for all classes, the basic operations *create*, *delete*, *modify* and *list* are defined. Also, more complex operations take the relationships between objects of a multibody model into account (see Fig. 12), and therefore represent self-explaining modeling steps of a multibody modeling task.

During the assembly in a multibody modeling process, objects of class *frame* have to be assigned to or removed from an object of class *part* (Fig. 12). The equivalent operation is required to assign objects of class *part* or *interact* to the root object of class *mbs*. Further operations representing the assembling process are the assignment and the removal of objects of class *joint* and *force* to the associated object of class *member*. As a quality of the object-oriented approach, all these operations are also structured in a hierarchical manner.

Special operations are designed to calculate properties of an object. The operation *calculate location* determines the position vector ${}_i\boldsymbol{r}_{ik}$ and the orientation matrix $\boldsymbol{S}_{ik}$ of an object of class *frame* with respect to its reference frame. The result of this operation for an object of class *interact* is the location of the related aframe and bframe objects of class *frame* with respect to the global inertial frame. For an object of class *joint*, the specific relative position vector ${}_k\boldsymbol{r}_{kl}$ (5) and the orientation matrix $\boldsymbol{S}_{kl}$ (6) of the joint are calculated.

The object-oriented approach guarantees that the changes in class components and the addition of new joint or force classes affect only local modifications in the overall class and operation definitions. All further basic and extended operations for the multibody modeling process remain valid.

5.2. Graphic Description for Multibody Classes

To achieve a graphical representation of the multibody elements within the CAD system, further

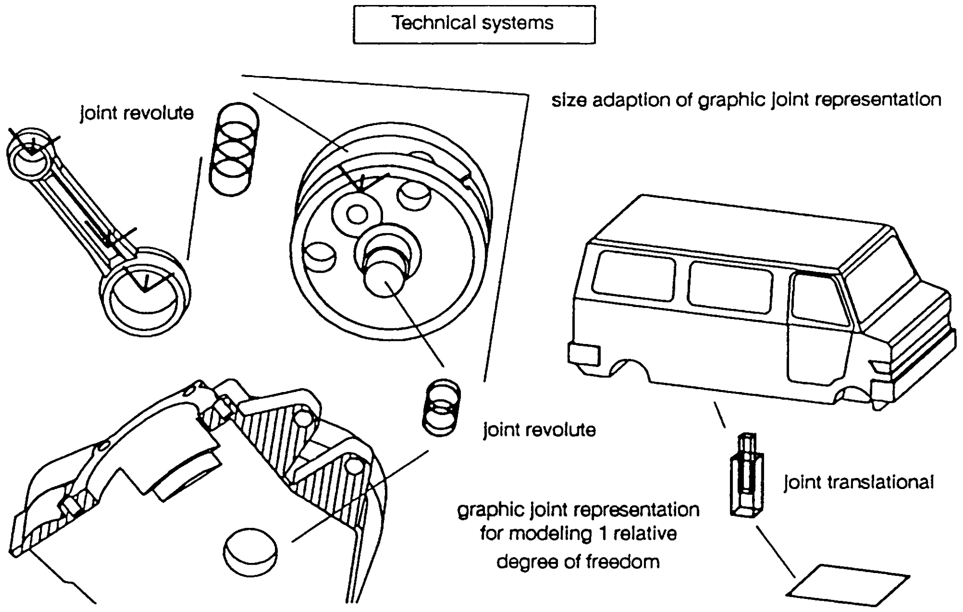

Technical systems
joint revolute
size adaption of graphic joint representation
joint revolute
joint translational
graphic joint representation for modeling 1 relative degree of freedom

Fig. 13. Graphic representation for the objects of class *joint*.

classes are required. The mechanical and mathematical properties of an object of class *frame* are determined entirely by its components. The information about the actual frame axis length, its color or its visibility depends upon the actual multibody size and modeling state. A geometry data model for multibody elements suitable for the machine, robot and vehicle is obtained by the following characteristics:

- Unique spatial representation of all multibody elements, their function and physical quantity.
- Arbitrary adaptation of the graphical representation to the actual shape and size of multibody elements.
- Abstract spatial representation of special multibody elements like joints to merely model the relative degree of freedom.

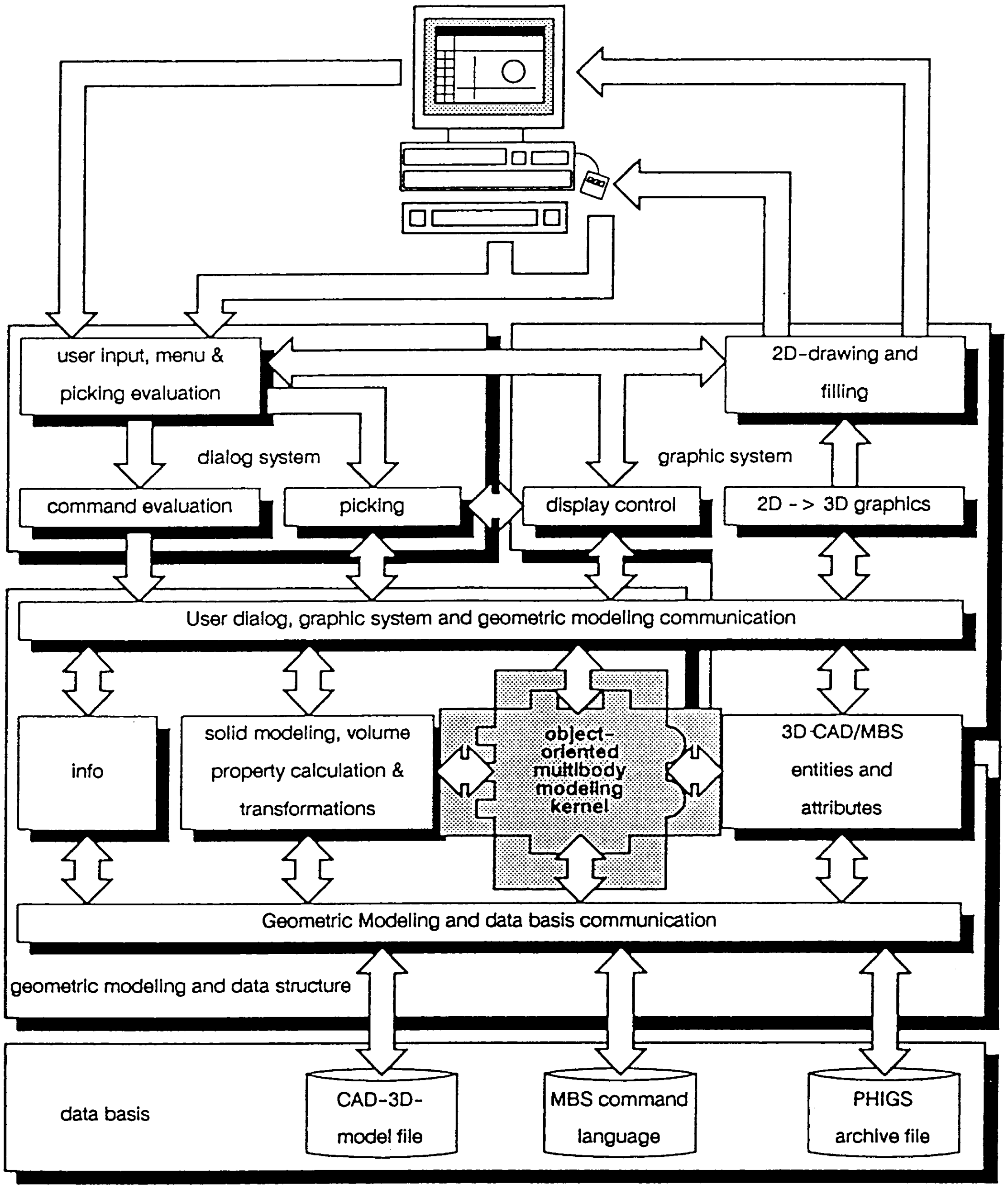

user input, menu & picking evaluation
2D-drawing and filling
dialog system
graphic system
command evaluation
picking
display control
2D -> 3D graphics
User dialog, graphic system and geometric modeling communication
info
solid modeling, volume property calculation & transformations
object-oriented multibody modeling kernel
3D-CAD/MBS entities and attributes
Geometric Modeling and data basis communication
geometric modeling and data structure
data basis
CAD-3D-model file
MBS command language
PHIGS archive file

Fig. 14. Integration of multibody modeling kernel library.

Figure 13 demonstrates these characteristics for a class *revolute* and *translational*.

From Figs 6 and 13, it becomes obvious that spatial parametrized shapes satisfy a graphic representation for objects of the classes *frame*, *joint* and *force*. The graphical representation of objects of class *body* is determined by the geometry model of the CAD 3D system. The definition of the classes *g3frame*, *g3joint*, *g3force* and the operations for the geometry data model are equivalent to the class design of the multibody data model.

A detailed description of the graphical representation, the classes including different initial locations for animation, and an object of class *g3global* comprising color, projection and viewpoint data is given in Daberkow [22].

5.3. Implementation of the DAMOS-C Multibody Modeling Kernel

In the present implementation, C is chosen as an intermediate language for its portability to practically integrate the object-oriented data model in a commercially available CAD 3D system.

Classes are implemented by the basic data types `int`, `char`, `float` and `double`. From these data types and *pointer references*, the data structures which serve as a named class declaration for the multibody modeling elements are constructed. For the classes *part*, *interact*, *frame* and *force*, whose objects may appear in an arbitrary number, the data structure *linked list* is applied to realize an insertion or deletion of the objects. According to Meyer [21], the implementation of object-oriented operations is performed by means of *routines*. Routines are mapped to the C data structures and functions, and yield a fundamental, high level software tool. The key classes and routines offer a system-independent modeling kernel library for multibody systems, supplied by interfaces to input and output, as well as for the graphical representation. This DAMOS-C (DAta model for MultibOdySystems implemented in C) library [22,23] contains routines to parametrize a multibody system, even in the state of modeling. Moreover, the consistence properties for multibody systems are checked. An assignment of an object of class *joint* is refused, for example, if the related objects are already connected by a joint. A deletion of an object of class *frame* is denied, for example, if this object serves as a reference frame in an object of class *interact* or for further frames. This open interface of the DAMOS-C library enables its integration in a commercial CAD 3D-system (Fig. 14).

The integration scheme shown in Fig. 14 reveals the interfaces to the different CAD 3D software modules. An extension of the CAD command language supplies additional commands which are necessary for the execution of the multibody modeling operations. To ensure the graphical display of the modeling elements, the *parametrized shapes* are modeled via the 3D wireframe entities of the CAD graphic subsystem.

The *reusability* of the multibody modeling kernel library as an important object-oriented quality has been proven in a second implementation [22], and will be shown in the following section. Based upon the graphics standard PHIGS, a 3D graphical modeling tool for multibody systems has been developed by integrating the DAMOS-C modeling kernel library. Here, the *planar face model* of the PHIGS graphic library serves as the geometry model for objects of class *body*. Objects of the classes *frame*, *joint* and *force* are displayed by the *parametrized shapes* geometry model and the PHIGS wireframe entities. Full access to the dialogue and the graphics system even allows the realization of high level interactive features, like an *undo* facility and fast immediate mode graphics. In the present implementation, the multibody model conversion from the extended CAD database is realized by a neutral multibody command language file (Fig. 14) to the database system RSYST [15]. The integrated RSYST multibody modules, like a symbolic Newton–Euler formalism, generate symbolic equations of motion, and automatically produce a problem-specific simulation program.

6. Examples and Applications

In this section, some modeling steps of mechanical systems are given to demonstrate the interaction between a user and a CAD 3D system. For each modeling step, the internal communication between the CAD 3D program modules and the DAMOS-C kernel library is explained.

6.1. Three Body Pendulum

Figure 15(a) shows the solid models designed in the CAD 3D system. In Fig. 15(b), these solid models are chosen by the user as objects of class *part*. Consequently, four objects of class *part* are created by the DAMOS-C kernel library. The mass and inertia properties of the related objects of class

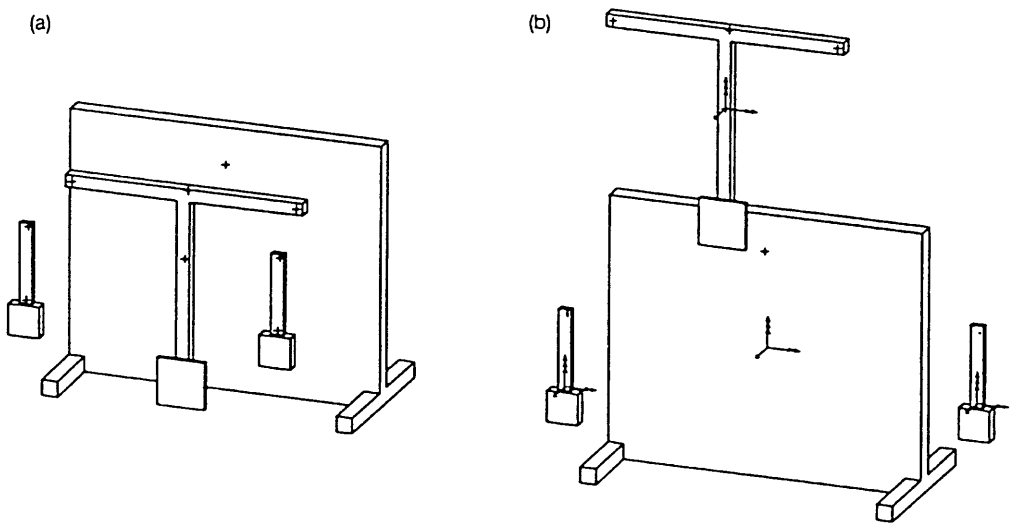


Fig. 15. Three body pendulum (a) solid models (b) solid models and objects of class *part*.

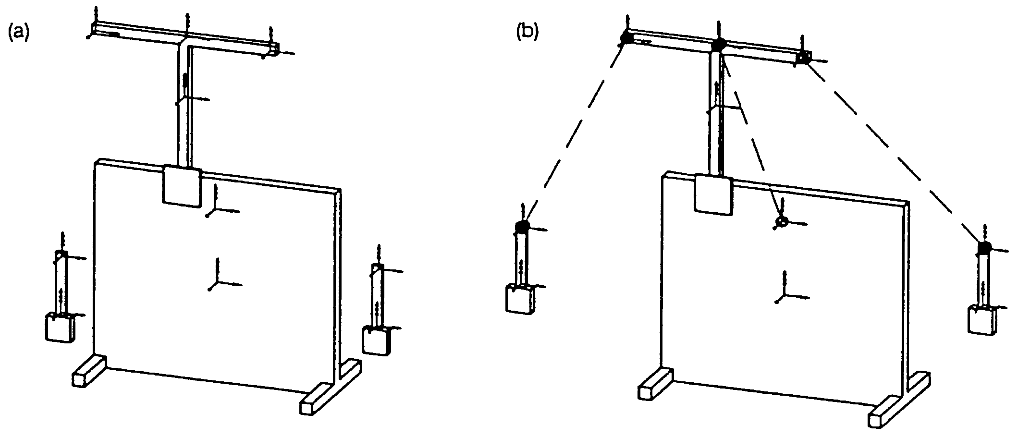


Fig. 16. Creation of joint definition frames (a) and revolute joints (b).

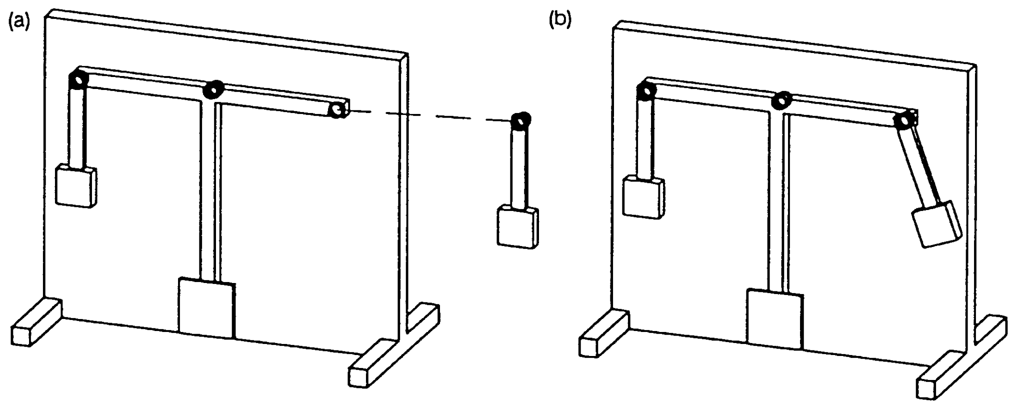


Fig. 17. Multibody system with arbitrary location (a) and assembly procedure (b).

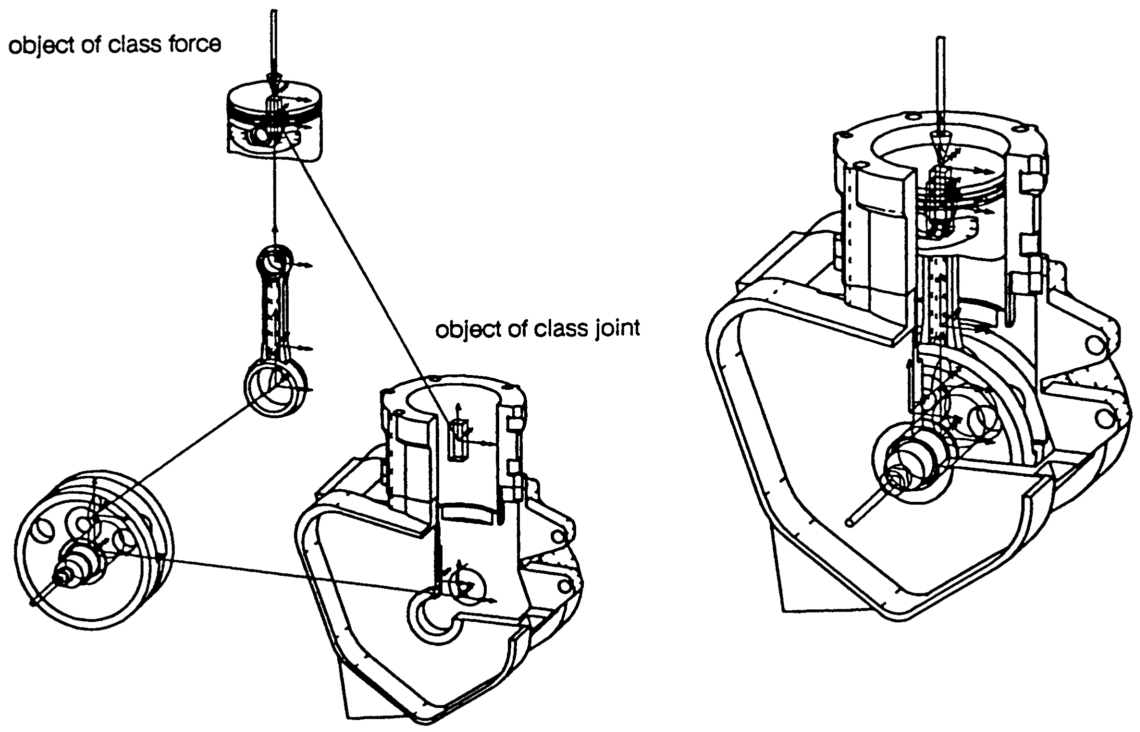


Fig. 18. Disassembled and assembled crank slider mechanism.

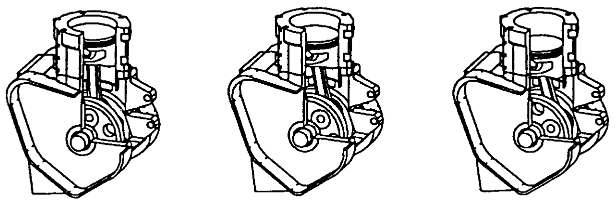


Fig. 19. Animation sequence of the crank slider mechanism.

body are calculated by PARASOLID and assigned to the *body* components. The *part* property of the solid models is visualized by the reference frames located in the centre of gravity. For each *frame*, an object of class *g3frame* is created with the default size and the color components, which can be adapted to the actual model size by the user. The initial location of the solid models is arbitrary in the CAD 3D model space.

The next modeling step consists of the creation of joint definition frames and objects of the class *revolute*. By default, the orientation of these frames is parallel to the specified reference frame. The position of the frames is defined by the CAD 3D *picking* user commands. Figure 16 shows these modeling steps, and the graphical representation of the objects.

According to Fig. 6 and Section 5.2, the objects of the class *revolute* are visualized by equivalent 3D circle entities in the *g3revolute* object, and the connection between the objects of class *part* is visualized by a connecting 3D line entity sized in the *g3interact* object between the aframe and bframe origins.

A further useful operation provided in the DAMOS-C kernel library is the assembly of arbitrary objects of class *part* by their connecting joints (Fig. 17). An initial multibody simulation condition is adjusted interactively by modifying the *rangle* component in the related object of class *joint*.

6.2. Crank Slider Mechanism

Figure 18 shows the disassembled and assembled parts of a crank slider mechanism. It becomes obvious that the graphical representation of objects of class *joint* can be adapted to the actual shaft diameter. Because of the wireframe model, *frame* and *joint* objects remain visible even if the CAD 3D model has been rendered with a hidden line elimination. An object of class *g3force* shows that an object of class *force* is included to model an applied piston gas force.

By the explicit constraint formulation of the Lagrangian generalized coordinate approach in the

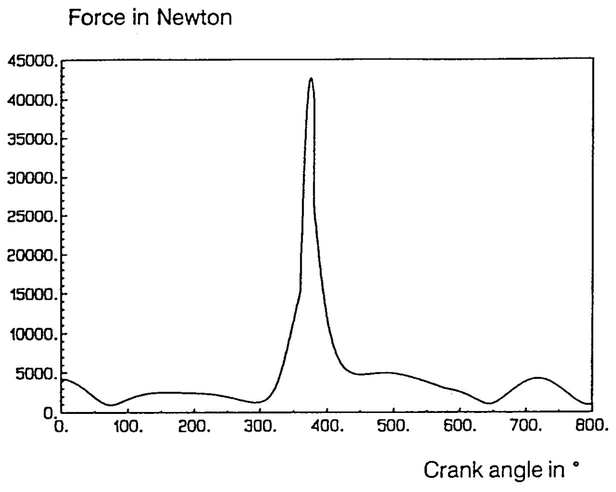


Fig. 20. Resultant crankshaft bearing constraint force.

Newton–Euler formalism, one gets the position vector $\boldsymbol{r}_j$ and the orientation matrix $\boldsymbol{S}_j$ of Eq. (7) for each simulation time step. Figure 19 shows an animated sequence of the simulation, which has been generated by re-reading the neutral multibody command language file with different initial conditions into the CAD 3D-system.

The multibody model conversion from CAD 3D by the neutral multibody command language generates the symbolic equations of motion with Lagrangian coordinates, and automatically produces a problem-specific simulation program. As a result of the simulation, a time plot of the resulting crankshaft bearing constraint force of the mechanism under the applied gas force is shown in Fig. 20.

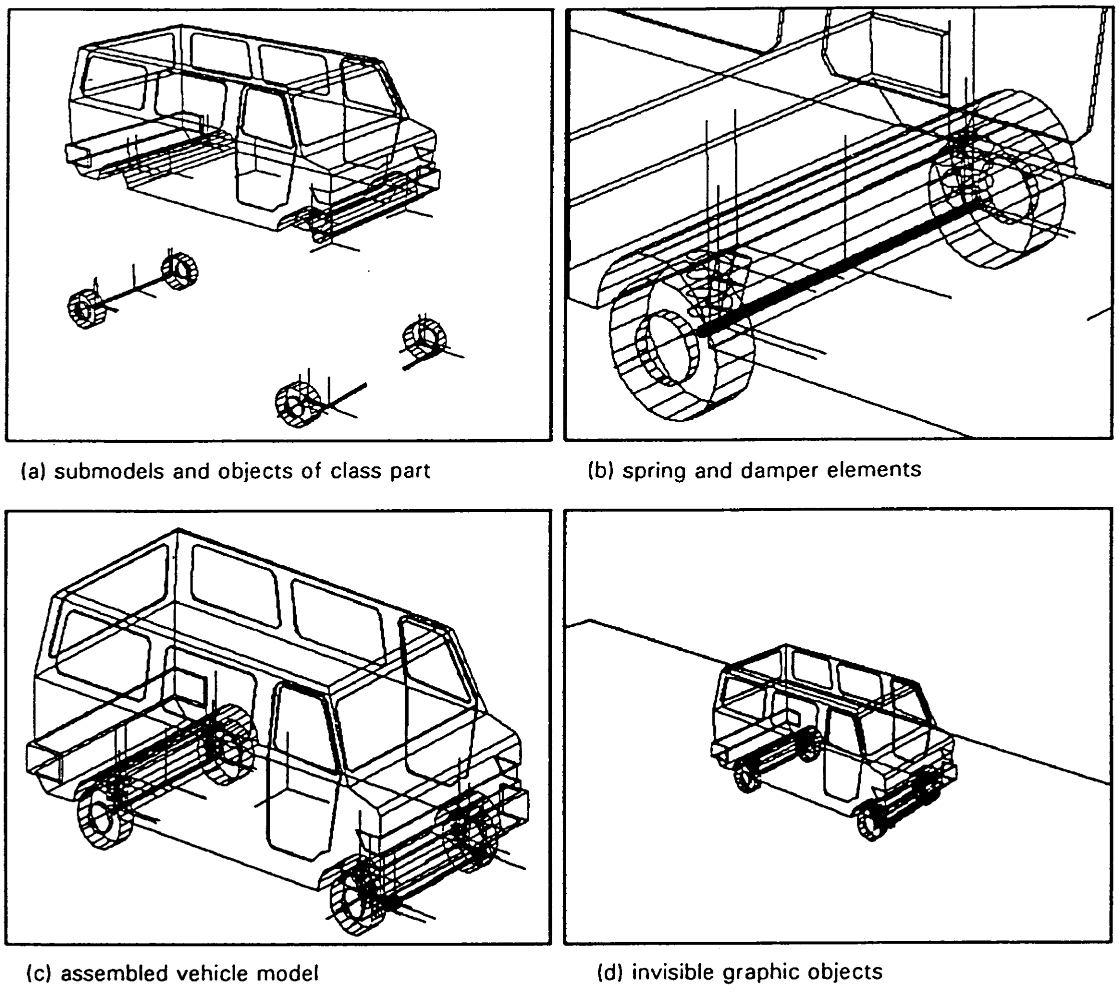

(a) submodels and objects of class part
(b) spring and damper elements
(c) assembled vehicle model
(d) invisible graphic objects

Fig. 21. Modeling steps (a)–(d) for vehicle dynamic analysis.

6.3. Vehicle Model

The system-independent kernel library DAMOS-C allows its integration in a completely different 3D graphic modeling system [22]. System-specific modifications are necessary for the PHIGS graphic system, and the dialogue system (see Fig. 14) Figure 21 shows the preceding modeling steps of a vehicle model for a subsequent dynamic analysis. To minimize the modeling expense, *submodels* of the rear and front axles consisting of an object of class *part* with an object of class *body* and arbitrary objects of class *frame* can be loaded into and assembled in the 3D graphic modeling system (see Fig. 21(a)). Force and damper elements connect the car body, front and rear axles (see Fig. 21(b)). The equivalent DAMOS-C objects of classes *force* and *damper* are created, and their actual size for the graphic representation is described in the *g3force* and *g3damper* object components. If a graphic representation of a multibody system becomes too complex (see Fig. 21(c)), single or complete classes of graphical objects can be made invisible by suppressing the graphic output (Fig. 21 (d)). Visibility information is stored in a separate component of each *g3* object.

7. Conclusions

In this paper, an integrated approach for computer aided design in multibody system dynamics is introduced. According to the different multibody formalisms and the CAD 3D geometry models, a new general data model for multibody systems, including the graphical description, is developed. This data model defines classes from the physical properties of multibody elements using object-oriented techniques, and even takes symbolic parameters for multibody systems into account. From the practical multibody modeling process, abstract operations are designed in an object-oriented manner. Moreover, unique spatial graphic representations for multibody elements are designed for mechanisms, robot and vehicle dynamics to support a preliminary CAD 3D modeling stage.

Object-oriented classes and operations are implemented in a system-independent, multibody modeling kernel library, DAMOS-C. The integration of this kernel library into a CAD 3D system and a further 3D modeling system demonstrates the advantages of this new approach in comparison with existing multibody and CAD system interfaces. For a CAD software system developer, the integration of the neutral kernel library opens his CAD system to a large variety of multibody computer codes. For a CAD system user, the additional self-explanatory multibody modeling methods which support a fast dynamic analysis in an early construction phase are available. For multibody system specialists, different CAD 3D systems offer identical multibody modeling methods to realize a fast preprocessing for the dynamic analysis. By providing fundamental and high level functions for a system-independent modeling of multibody systems, the goal of an integrated modular automated design and simulation tool is achieved.

References

1. Orleanda, N. (1973) Node-Analogous Sparsity-Oriented Methods for Simulation of Mechanical Systems, PhD dissertation, University of Michigan
2. Haug, E. J. (1989) Computer Aided Kinematics and Dynamics of Mechanical Systems, Allyn and Bacon, Boston
3. Rosenthal, D. E., Sherman, M. A. (1986) High performance multibody simulation via symbolic equation manipulation and Kane's method. Journal of Astronautical Sciences, 34, 223–239
4. Kreuzer, E. (1979) Symbolische Berechnung der Bewegungsgleichungen von Mehrkorpersystemen, Fortschritt-Berichte der VDI Zeitschriften, 11, 32, VDI, Dusseldorf
5. Kreuzer, E., Schiehlen, W. O. (1985), Computerized generation of symbolic equations of motion for spacecraft. Journal of Guidance, Control, and Dynamics, 8, 2, 284–287
6. Hollar, M. G., Rosenthal, D. E. (1991) Concurrent Design and Analysis of Mechanisms, Rasna Corp., San Jose, CA
7. Schiehlen, W. O. (ed.) (1990) Multibody System Handbook, Springer-Verlag, Berlin
8. Mortenson, M. E. (1985) Geometric Modelling, John Wiley, New York
9. Pahl, G. (1990) Konstruieren mit 3D-CAD Systemen: Grundlagen, Arbeitstechnik, Anwendungen, Springer-Verlag, Berlin
10. PARASOLID (1990) The PARASOLID Solid Modelling System Kernel Interface Reference Manual, Shape Data Ltd. Cambridge, UK
11. Puig-Pey, J., Brebbia, C. A. (1988) Computer Aided Design Handbook, Springer-Verlag, New York
12. ARIES (1991) ARIES Conceptstation Software Simulation Mechanism Reference, Aries Technology Inc., Lowell
13. Pro/ENGINEER (1992) Pro/ENGINEER-to-DADS Interface, Computer Aided Design Software Inc., Oakdale
14. Weber, H. R. (1988) CAD-Datenaustausch und -Datenverwaltung, Springer-Verlag, Berlin
15. Otter, M., Hocke M., Daberkow, A., Leister, G. (1993) An object oriented data model for multibody systems. Advanced Multibody System Dynamics – Simulation and Software Tools, 87–106, Kluwer Academic, Dordrecht

16. Wehage, R. A., Haug, E. J. (1982) Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. Journal of Mechanical Design, 104, 247–255
17. Leister, G., Bestle, D. (1992) Symbolic-numerical solution for multibody systems with closed loops. Vehicle System Dynamics, 21, 129–142
18. Schiehlen, W. O. (1986) Technische Dynamik, Teubner, Stuttgart
19. Rozells-Kuhnert, G. (ed.), Tinoco, K., Mosenus, D. (1990) Using PHIGS with FIGARO+, Template Software Division Inc., San Diego, CA
20. Silicon Graphics (1990) Graphics Library Programming Guide, Silicon Graphics Inc., Mountain View, CA
21. Meyer, B. (1990) Object-oriented Software Construction, Prentice Hall, New York
22. Daberkow, A. (1993) Zur CAD-gestutzten Modellierung von Mehrkorpersystemen, Fortschritt-Berichte der VDI Zeitschriften, 20, 80, VDI, Dusseldorf
23. Daberkow, A., Schiehlen, W. O. (1994) Concept, Development and Implementation of DAMOS-C: The Object Oriented Approach to Multibody Systems, Proceedings of 13th International Computers in Engineering Conference, San Diego, CA