Reza N. Jazar · Liming Dai *Editors*

Nonlinear Approaches in Engineering Applications

Automotive Applications of Engineering Problems

Springer

Nonlinear Approaches in Engineering Applications

Reza N. Jazar • Liming Dai
Editors

Nonlinear Approaches in Engineering Applications

Automotive Applications of Engineering Problems

Springer

Editors
Reza N. Jazar
Xiamen University of Technology
Xiamen, China

School of Engineering, RMIT University
Bundoora, VIC, Australia

Liming Dai
Xiamen University of Technology
Xiamen, China

University of Regina
Industrial Systems Engineering
REGINA, SK, Canada

ISBN 978-3-030-18962-4 ISBN 978-3-030-18963-1 (eBook)
https://doi.org/10.1007/978-3-030-18963-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Masterpiece happens before 20 and after 80.
Everything in between is practice.

Dedicated to Mojgan and Xinming

Preface

This book is the sixth volume in the series of "Nonlinear Approaches in Engineering Applications," organized by the editors. This series are collecting individual application on engineering problems in which the nonlinearity is quite important. Those systems have been introduced and modeled mathematically, and the nonlinearity in their equations has been used to make the system optimized, stable, analyzed, etc. This book is also a collection of ten different important problems set in two groups: Practical System Applications and Analytical System Applications. Both groups are more or less focussed on applications of engineering problems. Chapter 1 is on the laziness of vehicle to investigate how much vehicle behavior in transient periods deviates from their steady-state behavior. Other chapters of the Practical System Applications in the first group are on autonomous vehicles, drilling dynamics and friction, micro-/nanorobotics, and modeling of sea level fluctuations. The second group on Analytical System Applications begins with an extensive article on how to model and simulate dynamic systems, methods of solutions, and different classical behaviors. It follows up with a chapter on large deformation in curvilinear coordinate systems and big data analysis, and the last two chapters are on genetic algorithm and programing.

The nonlinear analysis, techniques, and applications have been developed in the past two to three centuries when the linear mathematical modeling of natural dynamical phenomena appeared not to be exact enough for some practical applications. The positive aspects of linear approximation of dynamic phenomena are simplicity and solvability. Linear approximation of a system provides us with the simplest model acting as the base and standard for which other nonlinear models should approach when the nonlinearities become very small. Solvability is another characteristic of all linear systems. These two characteristics provide us with a great ability and desire to model dynamic systems linearly. However, there exist many systems that their linear model and solution cannot provide exact enough approximation of the real system behavior. For such systems, considering the nonlinearities of the phenomena is unavoidable. Although the nonlinear approximation of a system provides us with a better and more accurate model, it also provides us with several complications. Unsolvability is one of them that makes us to search for indirect

methods to gain some information of the possible solutions. Due to the nonlinearity and complexity of the nonlinear systems, usually, it is very difficult or impossible to derive the analytical and closed-loop solutions for the systems. In solving or simulating the nonlinear systems, we have to rely on approximate or numerical methods, which may only provide approximate results for the systems while errors are unavoidable during the processes of generating the approximate results.

Level of the Book

This book aims at engineers, scientists, researchers, and engineering and physics students of graduate levels, together with the interested individuals in engineering, physics, and mathematics. This chapter book focuses on the application of the nonlinear approaches representing a wide spectrum of disciplines of engineering and science. Throughout the book, great emphases are placed on engineering applications, physical meaning of the nonlinear systems, and methodologies of the approaches in analyzing and solving for the systems. The topics that have been selected are of high interest in engineering and physics. An attempt has been made to expose the engineers and researchers to a broad range of practical topics and approaches. The topics contained in the present book are of specific interest to engineers who are seeking expertise in vehicle- and automotive-related technologies as well as engines and alternative fuels, mathematical modeling of complex systems, biomechanical engineering approaches to robotics and artificial muscles, nonclassical engineering problems, and modern mathematical treatments of nonlinear equations.

The primary audience of this book are the researchers, graduate students, and engineers in mechanical engineering, engineering mechanics, electrical engineering, civil engineering, aerospace engineering, mathematics, and science disciplines. In particular, the book can be used for training the graduate students as well as senior undergraduate students to enhance their knowledge by taking a graduate or advanced undergraduate course in the areas of nonlinear science, dynamics and vibration of discrete and continuous system, structure dynamics, and engineering applications of nonlinear science. It can also be utilized as a guide to the readers' fulfilment in practices. The covered topics are also of interest to engineers who are seeking to expand their expertise in these areas.

Organization of the Book

The main structure of the book consists of two parts, Practical System Applications and Analytical System Applications, including ten chapters. Each of the chapters covers an independent topic along the line of nonlinear approach and engineering applications of nonlinear science. The main concepts in nonlinear science and

engineering applications are explained fully with necessary derivatives in details. The book and each of the chapters are intended to be organized as essentially self-contained. All the necessary concepts, proofs, mathematical background, solutions, methodologies, and references are supplied except for some fundamental knowledge well-known in the general fields of engineering and physics. The readers may therefore gain the main concepts of each chapter with as less as possible the need to refer to the concepts of the other chapters and references. The readers may hence start to read one or more chapters of the book for their own interests.

Method of Presentation

The scope of each chapter is clearly outlined, and the governing equations are derived with an adequate explanation of the procedures. The covered topics are logically and completely presented without unnecessary overemphasis. The topics are presented in a book form rather than in the style of a handbook. Tables, charts, equations, and references are used in abundance. Proofs and derivations are emphasized in such a way that they can be straightforwardly followed by the readers with fundamental knowledge of engineering science and university physics. The physical model and final results provided in the chapters are accompanied with necessary illustrations and interpretations. Specific information that is required in carrying out the detailed theoretical concepts and modeling processes has been stressed.

Prerequisites

The present book is primarily intended for the researchers, engineers, and graduate students, so the assumption is that the readers are familiar with the fundamentals of dynamics, calculus, and differential equations associated with dynamics in engineering and physics, as well as a basic knowledge of linear algebra and numerical methods. The presented topics are given in a way to establish as conceptual framework that enables the readers to pursue further advances in the field. Although the governing equations and modeling methodologies will be derived with adequate explanations of the procedures, it is assumed that the readers have a working knowledge of dynamics, university mathematics, and physics together with theory of linear elasticity.

Bundoora, VIC, Australia Reza N. Jazar
Regina, SK, Canada Liming Dai

Acknowledgments

This book is made available under the close and effective collaborations of all the enthusiastic chapter contributors who have the expertise and experience in various disciplines of nonlinear science and engineering applications. They deserve sincere gratitude for the motivation of creating such a book, for the encouragement in completing the book, for the scientific and professional attitude in constructing each of the chapters of the book, and for the continuous efforts toward improving the quality of the book. Without the collaboration and consistent efforts of the chapter contributors, the completion of this book would have been impossible. What we have at the end is a book that we have every reason to be proud of.

It has been gratifying to work with the staff of Springer-Verlag through the development of this book. The assistance provided by the staff members has been valuable and efficient. We thank Spinger-Verlag for their production of an elegant book.

Reza N. Jazar
Liming Dai

Contents

List of Figures

List of Tables

Part I
Practical System Applications

Chapter 1
Vehicles Are Lazy: On Predicting Vehicle Transient Dynamics by Steady-State Responses

Sina Milani, Hormoz Marzbani, Ali Khazaei, and Reza N. Jazar

1.1 Introduction

Analysis of vehicles' handling behavior in turning maneuvers requires a proper mathematical model. There are several factors affecting a vehicle's response in a turning maneuver. Apart from variations in vehicle and tire parameters, external factors such as air resistance and slope of the road make it quite a complicated task to consider all parameters in the vehicle model. The majority of the most important features of the vehicle behavior in maneuvers are observable using fairly simplified planar vehicle models. In planar modeling, we ignore the roll, pitch, and vertical motions of the vehicle and only emphasize on the longitudinal, lateral, and yaw motions.

The most famous and basic planar vehicle model is known as the *bicycle model*. Many of the vehicle handling analyses and all the basic characterizations have been derived using bicycle model throughout the course of vehicle dynamics studies [1–3]. Bicycle model is accurate enough to represent the real car behavior to a reasonable extent in normal driving conditions. This characteristic of the bicycle model makes it useful in designing and investigating new ideas on dynamics and

S. Milani · H. Marzbani
Mechanical and Automotive Engineering, School of Engineering, RMIT University, Melbourne, VIC, Australia

A. Khazaei
Mechanical Engineering, Kennesaw State University, Marietta, GA, USA

R. N. Jazar (✉)
Xiamen University of Technology, Xiamen, China

School of Engineering, RMIT University, Bundoora, VIC, Australia
e-mail: reza.jazar@rmit.edu.au

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_1

control of vehicles, such as defining the nominal vehicle response for yaw-rate and/or side-slip angle [4, 5]. In the following section, the bicycle model is presented in detail and the underlying assumptions are discussed.

In the rest of the chapter, the importance of steady-state responses of the bicycle model is discussed by comparing the steady-state and transient vehicle behaviors, characteristics of maneuvering vehicles including steady-state charts are presented, and finally, application of such an analysis on a path following strategy is explained and two examples are given to evaluate the proposed idea.

1.1.1 Bicycle Model

The first step in dynamic modeling of vehicles is to identify the main forces and moments acting on the vehicle as a rigid body. The magnitude of these forces and moments depends on the motion of the vehicle, which is described by kinematic variables of the vehicle, such as longitudinal and lateral velocities, yaw rate, side-slip angles, etc. In the next step, the forces and moments must be transferred to the center of gravity of the vehicle and, finally, put into the Newton–Euler equations of motion for a rigid body on a planar surface.

The main forces acting on the vehicle are the longitudinal and lateral tire forces. There are many other acting forces and moments such as the aerodynamic forces and rolling resistance of tires which are assumed to be negligible when dealing with motion of vehicles in normal driving conditions. The normal driving condition is defined as regular turning maneuvers with forward and lateral accelerations are kept in certain limits that represent non-emergency maneuvering (see the maneuver design in Sect. 1.3).

Longitudinal tire force is a function of a kinematic variable called the longitudinal slip. The longitudinal slip is defined as the ratio of the slip velocity at tire contact patch to the longitudinal velocity of the vehicle and, therefore, is a function of tire's rotational velocity. To be able to model the longitudinal tire force, rotational dynamics of tires must be considered separately. Also, the longitudinal forces will affect the forward velocity of the vehicle, which is normally studied in vehicle performance investigations. Therefore, the longitudinal Degree of Freedom (DoF) is of less importance when dealing with turning maneuvers and is usually ignored in vehicle handling studies to avoid over-complication. Hence, the most significant external forces to be studied are the lateral tire forces. The lateral tire force is assumed to be proportional to another kinematic variable called tire side-slip angle, in normal driving condition. The tire side-slip angle α_i is defined as the angle measured from the tire's longitudinal axis x_{ti} towards the velocity vector v_i (direction of motion) at the wheel center about the vertical tire axis z_{ti} which makes a right-handed coordinate frame with x_{ti} and y_{ti} (see Fig. 1.1a) [3].

Similarly, another side-slip angle β_i may be defined for the vehicle body at any point. The body side-slip angle is defined as the angle between longitudinal axis of the vehicle body x and the velocity vector at that point v_i about the vertical body

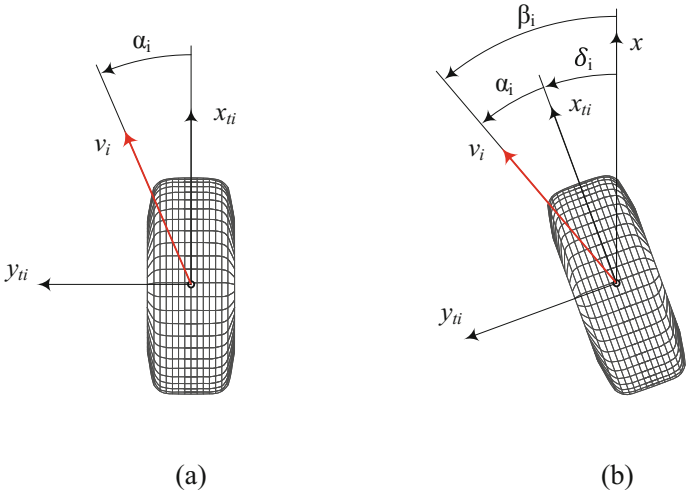


Fig. 1.1 Definition of side-slip angles and tire coordinate frame

axis z. Body side-slip angle β_i is related to longitudinal and lateral velocities of the body point expressed in vehicle body coordinate frame by (1.1). Definition of β_i allows for calculation of the tire side-slip angle for a steered wheel (see Fig. 1.1b). The tire side-slip in case of a steered wheel is obtained by the more general Eq. (1.2). One may consider steer angle of $\delta_i = 0$ for a wheel with no steering.

$$\beta_i = \arctan\left(\frac{v_{yi}}{v_x}\right) \approx \frac{v_{yi}}{v_x}, \qquad \text{for small } \beta_i \tag{1.1}$$

$$\alpha_i = \beta_i - \delta_i \approx \frac{v_{yi}}{v_x} - \delta_i \tag{1.2}$$

Note that the longitudinal velocity of any point on the vehicle body is assumed to be the same and equal to v_x in a bicycle model.

Thus, the tire lateral force is written as (1.3). The negative sign is used to match the direction of the force and the side-slip angle. When a vehicle is turning left, the lateral force is positive while the tire side-slip α_i is negative (see Fig. 1.2).

$$F_{yi} = -C_{\alpha i}\alpha_i \tag{1.3}$$

As shown in Fig. 1.2, with bicycle model of the vehicle, we assume the effect of left and right tires is lumped at the center of the axle by an equivalent tire. It is assumed that the lateral shift of the tires' vertical loads in turning maneuver is negligible and the lateral forces from left and right tires can be added up at the center of the axle. The nominal parameters of the bicycle vehicle model used in all sections of this manuscript are as follows:

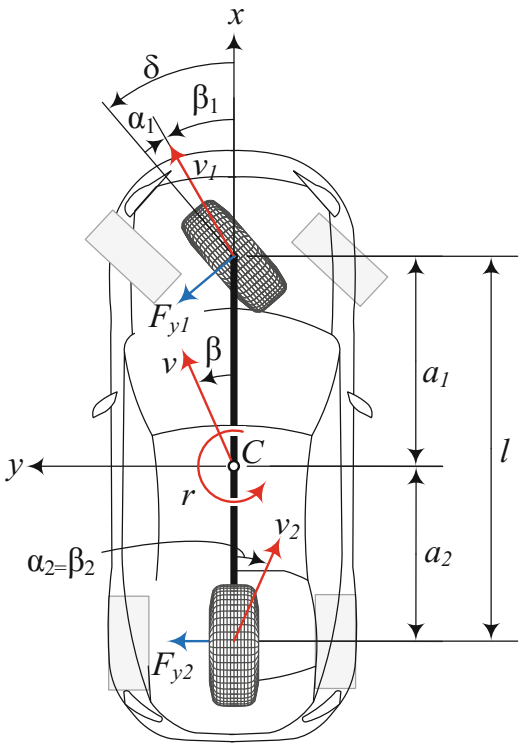


Fig. 1.2 Bicycle model and vehicle body coordinate frame

$$m = 1000\,\text{kg}$$

$$a_1 = 1\,\text{m}$$

$$a_2 = 1.5\,\text{m}$$

$$l = a_1 + a_2 = 2.5\,\text{m}$$

$$I_z = 1650\,\text{kg}\,\text{m}^2$$

$$C_{\alpha 1} = C_{\alpha 2} = 60{,}000\,\text{N/rad}$$

1.1.2 Equations of Motion

Having identified and calculated the external forces as functions of vehicle kinematic variables, we may transfer the forces to the center of gravity (point C in Fig. 1.2). Using the side-slip definition from (1.2) and assuming small δ, we may

write the total lateral force and yaw moment at vehicle's center of gravity C as (1.4), (1.5) for a front steering vehicle.

$$F_y = F_{y1} + F_{y2} = -C_{\alpha 1}\left(\frac{v_{y1}}{v_x} - \delta\right) - C_{\alpha 2}\frac{v_{y2}}{v_x} \tag{1.4}$$

$$M_z = a_1 F_{y1} - a_2 F_{y2} = -a_1 C_{\alpha 1}\left(\frac{v_{y1}}{v_x} - \delta\right) + a_2 C_{\alpha 2}\frac{v_{y2}}{v_x} \tag{1.5}$$

From the kinematics of a rigid planar vehicle, we know:

$$v_{y1} = v_y + r a_1, \qquad v_{y2} = v_y - r a_2 \tag{1.6}$$

where yaw rate r is the rotational velocity of the vehicle about z axis and v_y is the lateral velocity of the vehicle at center of gravity C along y direction. Substituting (1.6) in (1.4), (1.5), total lateral force and yaw moment become:

$$\begin{aligned} F_y &= \left(-\frac{a_1}{v_x}C_{\alpha 1} + \frac{a_2}{v_x}C_{\alpha 2}\right) r - (C_{\alpha 1} + C_{\alpha 2})\frac{v_y}{v_x} + C_{\alpha 1}\delta \\ &= C_r r + C_\beta \beta + C_\delta \delta \end{aligned} \tag{1.7}$$

$$\begin{aligned} M_z &= \left(-\frac{a_1^2}{v_x}C_{\alpha 1} - \frac{a_2^2}{v_x}C_{\alpha 2}\right) r - (a_1 C_{\alpha 1} - a_2 C_{\alpha 2})\frac{v_y}{v_x} + a_1 C_{\alpha 1}\delta \\ &= D_r r + D_\beta \beta + D_\delta \delta \end{aligned} \tag{1.8}$$

in which the *force system coefficients* $C_r, C_\beta, C_\delta, D_r, D_\beta, D_\delta$ are introduced for simplicity in the equations. These coefficients are functions of vehicle parameters, including v_x which is treated as a varying parameter.

To complete the equations of motion, we need to calculate the accelerations of the vehicle in body coordinate frame as functions of vehicle variables v_y and r. These accelerations are derived from the general Newton–Euler set of equations for 6 DoF in space. Since we limited the motion to planar, the equations simplify to longitudinal, lateral, and yaw motions [3]:

$$F_x = m(\dot{v}_x - r v_y) \tag{1.9}$$

$$F_y = m(\dot{v}_y + r v_x) \tag{1.10}$$

$$M_z = \dot{r} I_z \tag{1.11}$$

Note that the term $(\dot{v}_y + r v_x)$ is equivalent to the lateral acceleration a_y of the vehicle mass center expressed in the body frame. Equating (1.7), (1.8) with (1.10), (1.11) provides us with the equations of motion for bicycle model:

$$m(\dot{v}_y + r v_x) = C_r r + C_\beta \beta + C_\delta \delta \tag{1.12}$$

$$\dot{r} I_z = D_r r + D_\beta \beta + D_\delta \delta \tag{1.13}$$

Taking the system variables as v_y and r, we may rewrite (1.12), (1.13) in the form of a state-space representation of the system as:

$$\begin{bmatrix} \dot{v}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{C_\beta}{m v_x} & \frac{C_r}{m} - v_x \\ \frac{D_\beta}{I_z v_x} & \frac{D_r}{I_z} \end{bmatrix} \begin{bmatrix} v_y \\ r \end{bmatrix} + \begin{bmatrix} \frac{C_\delta}{m} \\ \frac{D_\delta}{I_z} \end{bmatrix} \delta \tag{1.14}$$

Note that since we assumed v_x to be known, Eq. (1.9) does not add any information to the system dynamics; however, the required F_x obtained from the same equation provides the necessary longitudinal force during the maneuver which is assumed to be supplied.

1.1.3 Steady-State Responses

Equations of motion (1.14) may be solved numerically or analytically to obtain the transient response of bicycle model to a certain steer input δ. The same equations may also be used to derive the steady-state values of the vehicle variables v_y and r to a step steer input. In steady-state condition, all of the state variables are kept constant in time, hence, their time derivatives will be equal to zero. In other words, Eq. (1.14) reduces to (1.15) in steady-state which is solved as (1.16).

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{C_\beta}{m v_x} & \frac{C_r}{m} - v_x \\ \frac{D_\beta}{I_z v_x} & \frac{D_r}{I_z} \end{bmatrix} \begin{bmatrix} (v_y)_{ss} \\ (r)_{ss} \end{bmatrix} + \begin{bmatrix} \frac{C_\delta}{m} \\ \frac{D_\delta}{I_z} \end{bmatrix} \delta \tag{1.15}$$

$$\begin{bmatrix} (v_y)_{ss} \\ (r)_{ss} \end{bmatrix} = \begin{bmatrix} \frac{C_\beta}{m v_x} & \frac{C_r}{m} - v_x \\ \frac{D_\beta}{I_z v_x} & \frac{D_r}{I_z} \end{bmatrix}^{-1} \begin{bmatrix} -\frac{C_\delta}{m} \\ -\frac{D_\delta}{I_z} \end{bmatrix} \delta \tag{1.16}$$

where "ss" subscript refers to the steady-state solution of the variables. After simplification, one may obtain v_y and r solutions as:

$$(v_y)_{ss} = \frac{D_\delta (C_r - m v_x) - D_r C_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta} v_x \delta \tag{1.17}$$

$$(r)_{ss} = \frac{C_\delta D_\beta - C_\beta D_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta}\delta \tag{1.18}$$

It can be seen that the steady-state solutions are proportional to the steer angle input. Hence, we may define steady-state responses of important vehicle variables as their ratio to the steer angle δ. To observe more details about the behavior of the vehicle in maneuver, we may introduce new dependent variables, namely body side-slip at center of gravity β, and lateral acceleration a_y which are useful in studying handling of vehicle and are defined as:

$$(\beta)_{ss} = \frac{(v_y)_{ss}}{v_x} \tag{1.19}$$

$$(a_y)_{ss} = (\dot{v}_y)_{ss} + (r)_{ss} v_x = (r)_{ss} v_x \tag{1.20}$$

Thus, the steady-state responses are obtained as:

$$S_y = \frac{(v_y)_{ss}}{\delta} = \frac{D_\delta (C_r - m v_x) - D_r C_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta} v_x \tag{1.21}$$

$$S_r = \frac{(r)_{ss}}{\delta} = \frac{C_\delta D_\beta - C_\beta D_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta} \tag{1.22}$$

$$S_\beta = \frac{(\beta)_{ss}}{\delta} = \frac{S_y}{v_x} = \frac{D_\delta (C_r - m v_x) - D_r C_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta} \tag{1.23}$$

$$S_a = \frac{(a_y)_{ss}}{\delta} = S_r v_x = \frac{C_\delta D_\beta - C_\beta D_\delta}{D_r C_\beta - C_r D_\beta + m v_x D_\beta} v_x \tag{1.24}$$

Note that the lateral and centripetal accelerations are approximately equal for small side-slip angle β and forward acceleration a_x, but in general, the following relationship is held between a_c, a_x, a_y:

$$\begin{aligned} a_c &= a_y \cos\beta - a_x \sin\beta \\ &\approx a_y \qquad \text{if } (a_x \,\&\, \beta \text{ small}) \end{aligned} \tag{1.25}$$

$$a_x = \dot{v}_x - r v_y \tag{1.26}$$

$$a_y = \dot{v}_y + r v_x \tag{1.27}$$

Denominator of (1.21)–(1.24) forms the characteristic equation of the dynamic system. A combination of the vehicle parameters dictates the general behavior of the vehicle. Substituting the vehicle parameters into the force system coefficients, the denominator D is obtained; equating it to zero, we may find a stability condition for turning maneuver:

$$D = m(C_{\alpha 2}a_2 - C_{\alpha 1}a_1)v_x^2 + l^2 C_{\alpha 1}C_{\alpha 2} = 0 \tag{1.28}$$

$$\frac{m}{l^2}\left(\frac{a_2}{C_{\alpha 1}} - \frac{a_1}{C_{\alpha 2}}\right)v_x^2 + 1 = K v_x^2 + 1 = 0 \tag{1.29}$$

$$v_x = v_c = \sqrt{-\frac{1}{K}} \tag{1.30}$$

$$K = \frac{m}{l^2}\left(\frac{a_2}{C_{\alpha 1}} - \frac{a_1}{C_{\alpha 2}}\right) \tag{1.31}$$

where v_c is the critical speed at which the denominator will become zero and the vehicle becomes unstable. The critical speed only exists when the stability factor K is negative. The stability factor determines whether the vehicle is *under-steer* ($K > 0$), *over-steer* ($K < 0$), or *neutral-steer* ($K = 0$). The behavior of the vehicle changes with different signs of K which is not in the scope of this chapter. For more information about the effect of K on vehicle responses, see [1–3].

1.2 Center of Curvature and Path of Motion

One of the main characteristics of a maneuvering vehicle which is emphasized in this chapter is the path of motion. The path of motion is directly related to the radius of turning R and the center of curvature of the vehicle at each time instance, which is also called the Instantaneous Center of Rotation (ICR). In this section, the methodology to calculate R and ICR as well as obtaining the global vehicle coordinates (X, Y) which define the path of motion will be examined. Calculation of ICR helps in understanding the similarity between transient and steady-state responses [6]. The usage of the presented calculations is detailed in Sect. 1.3.

1.2.1 Curvature and Turning Radius

We start by calculating the radius of turning R. For a particle rotating about a center with radius R, translational velocity of v, and angular velocity of ω, in 2-dimensional space, the following relationship holds:

$$v = R\omega \tag{1.32}$$

The relation between longitudinal and lateral velocities v_x, v_y and the body side-slip angle β can be written as:

$$v = v_x \cos\beta + v_y \sin\beta \tag{1.33}$$

and since β is assumed to be small:

$$v = v_x + v_y\beta \tag{1.34}$$

Assuming the vehicle to be a point mass at its mass center rotating with angular velocity of $\omega = r$, we may use (1.32) and (1.34) to write:

$$R = \frac{v}{r} = \frac{v_x + v_y\beta}{r} \tag{1.35}$$

$$\kappa = \frac{1}{R} = \frac{r}{v_x + v_y\beta} \tag{1.36}$$

where κ is called the *curvature of the path* and R is the radius of curvature. For straight driving, $\kappa = 0$ while $R \to \infty$. Using (1.36) and substituting v_y, r, β from (1.21)–(1.23), we may obtain the value of κ in steady-state which would be a nonlinear function of steer input δ. On the other hand, the multiplication of two small quantities v_y and β is negligible compared to v_x in (1.35), (1.36). So, with a highly reasonable approximation, we may ignore the term $v_y\beta$ in (1.35), (1.36) and rewrite them as:

$$R = \frac{v}{r} = \frac{v_x}{r} \tag{1.37}$$

$$\kappa = \frac{1}{R} = \frac{r}{v_x} \tag{1.38}$$

By using (1.37), (1.38) instead of (1.35), (1.36), we end up with a linear relationship between $(\kappa)_{ss}$ and δ and we are able to define an additional steady-state response S_κ called the *curvature response*:

$$(\kappa)_{ss} = \frac{1}{(R)_{ss}} = \frac{(r)_{ss}}{v_x} \tag{1.39}$$

$$S_\kappa = \frac{(\kappa)_{ss}}{\delta} = \frac{S_r}{v_x} = \frac{C_\delta D_\beta - C_\beta D_\delta}{(D_r C_\beta - C_r D_\beta + m v_x D_\beta) v_x} \tag{1.40}$$

The importance of the curvature and its steady-state response in analyzing vehicle's maneuver is due to its key role in calculating ICR as explained in the following sections.

1.2.2 ICR in Vehicle Body Coordinate Frame

Having the radius of curvature R calculated, we know how far ICR is located from vehicle's center of gravity. On the other hand, the vehicle side-slip angle determines how much rotation exists between the vehicle's heading direction and the direction

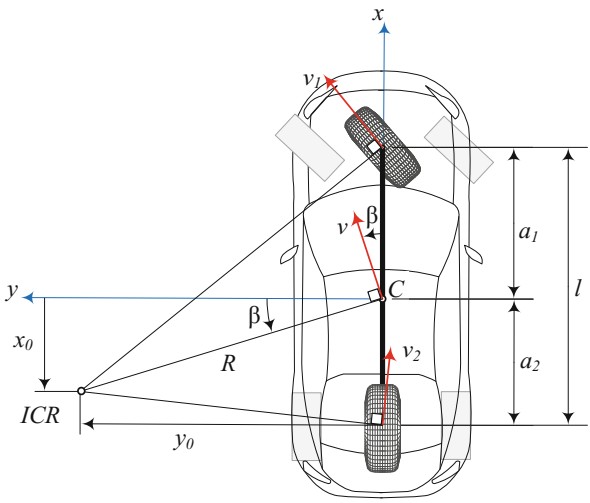


Fig. 1.3 Location of ICR in vehicle body frame

tangent to the path. Thus, the side-slip angle β affects the longitudinal and lateral coordinates of ICR (x_0, y_0) in body frame. Figure 1.3 shows the calculation of ICR location in the vehicle body coordinate frame.

Thus, x_0 and y_0 are calculated as:

$$x_0 = -R \sin \beta \tag{1.41}$$

$$y_0 = R \cos \beta \tag{1.42}$$

Note that we do not use small angle approximation here to be able to observe both the lateral and longitudinal location of ICR in the body coordinate frame.

1.2.3 ICR in Global Coordinate Frame

The location of ICR may also be calculated in the global coordinate frame. The relationship between global and body coordinate frames is defined using the rotation matrix ${}^{G}R_{B}$. Any position vector in 3-dimensional space may be expressed whether in global frame G shown by ${}^{G}\vec{d}$ or in body frame B shown by ${}^{B}\vec{d}$. Transformation between body to global coordinate is defined by [7]:

$${}^{G}\vec{d} = {}^{G}R_{B}\,{}^{B}\vec{d} + {}^{G}\vec{e}_{BG} \tag{1.43}$$

$${}^{G}R_{B} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.44}$$

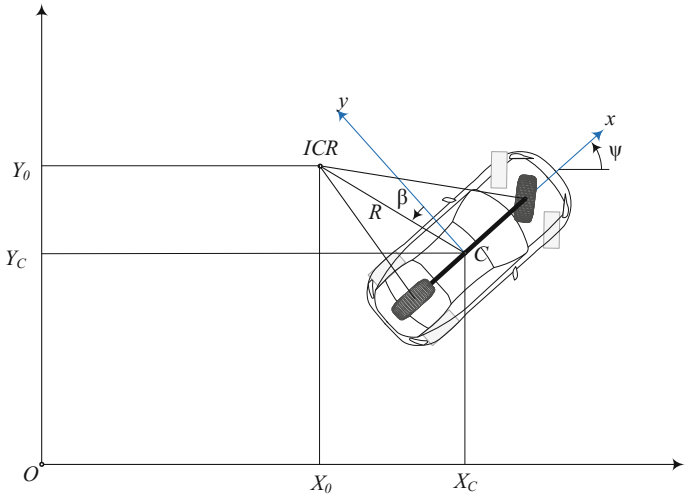


Fig. 1.4 Location of ICR in global frame

where ${}^{G}\vec{e}_{BG}$ is the position vector of the origin of body frame O_B with respect to the origin of the global frame O_G expressed in global frame G; and ψ is the heading angle of the vehicle which is measured from the global longitudinal axis X towards vehicle's longitudinal axis x about the vertical axis z. In case of a vehicle in turning maneuver, ${}^{G}\vec{e}_{BG}$ is defined by global position components of the vehicle's center of gravity X_C, Y_C. Figure 1.4 shows the global position and orientation of the vehicle in a turning maneuver.

Applying the transformation (1.43) from body frame to global frame for ICR position vector yields:

$$\begin{bmatrix} X_0 \\ Y_0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix} + \begin{bmatrix} X_C \\ Y_C \\ 0 \end{bmatrix} \tag{1.45}$$

$$X_0 = x_0 \cos\psi - y_0 \sin\psi + X_C \tag{1.46}$$

$$Y_0 = x_0 \sin\psi + y_0 \cos\psi + Y_C \tag{1.47}$$

substituting from (1.41), (1.42) the global coordinates of ICR are obtained as:

$$X_0 = -R \sin\beta \cos\psi - R \cos\beta \sin\psi + X_C \tag{1.48}$$

$$Y_0 = -R \sin\beta \sin\psi + R \cos\beta \cos\psi + Y_C \tag{1.49}$$

Calculation of the global coordinates X_C, Y_C which construct the path of motion is presented in the following section.

1.2.4 Path of Motion

We may assume that the vehicle was initially coincident with the origin of the global frame. As the vehicle moves, a combination of the longitudinal, lateral, and yaw velocities causes displacement and change of direction with respect to the global frame. To calculate the current global coordinates at any time, we need to integrate the velocities expressed in global frame, namely $\dot{X}_C = v_X$, $\dot{Y}_C = v_Y$. On the other hand, these velocities are related to local expressions of the velocities in body frame, namely v_x, v_y through a rotation of ψ about the vertical axis z. The transformation between local and global velocity vectors is defined by:

$$^{G}v = {}^{G}R_{B}\,{}^{B}v \tag{1.50}$$

$$\begin{bmatrix} \dot{X}_C \\ \dot{Y}_C \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{1.51}$$

$$\dot{X}_C = v_x \cos\psi - v_y \sin\psi \tag{1.52}$$

$$\dot{Y}_C = v_x \sin\psi + v_y \cos\psi \tag{1.53}$$

As (1.52) and (1.53) imply, derivation of the path of motion relies on calculation of heading angle ψ. On the other hand, time derivative of ψ is equal to the yaw velocity of the vehicle r:

$$\dot{\psi} = r \tag{1.54}$$

Using the expressions (1.52)–(1.54), we may increase the order of the system expressed by (1.14) and introduce new state variables X_C, Y_C, ψ in order to obtain the integrated quantities required for plotting the path, as outputs of the system. This way, any numerical integration method used for solving the differential equations of the system will also result in the path of motion. Augmenting the system variables in (1.14) by (1.52)–(1.54), the new system would be represented by the following set of differential equations:

$$\begin{bmatrix} \dot{v}_y \\ \dot{r} \\ \dot{X}_C \\ \dot{Y}_C \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{C_\beta}{m v_x} & \frac{C_r}{m} - v_x & 0 & 0 & 0 \\ \frac{D_\beta}{I_z v_x} & \frac{D_r}{I_z} & 0 & 0 & 0 \\ -\sin\psi & 0 & 0 & 0 & 0 \\ \cos\psi & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_y \\ r \\ X_C \\ Y_C \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{C_\delta}{m}\delta \\ \frac{D_\delta}{I_z}\delta \\ v_x \cos\psi \\ v_x \sin\psi \\ 0 \end{bmatrix} \tag{1.55}$$

Note that the above system is nonlinear due to potentially large values of ψ making it impossible to linearly approximate $\sin\psi$ and $\cos\psi$ terms.

1.3 Comparing Transient and Steady-State Behaviors

Steady-state relations may be used to calculate settled responses while the differential equations must be solved to obtain the transient responses. Road vehicles are most of the time performing in a steady-state condition. In all other times, the vehicle is undergoing a transition between two steady-states. Such a transition in normal driving conditions is examined in this section. The objective of this section is to show how this transition is made and how far the vehicle responses deviate from quasi-steady-state transition. A Quasi-Steady-State (QSS) transition happens when the inputs to the vehicle change at very low rates such that the vehicle responses are close to steady-state responses at each time instance. The laziness in vehicle's behavior is defined by "vehicle's tendency to get to the steady-state condition as soon as possible." Two different maneuver types are investigated in this manuscript to examine the vehicle's laziness against input changes during a turning maneuver: 1—increasing forward velocity at constant steering, and 2—increasing steer angle at constant forward velocity.

In designing the maneuvers, the maximum longitudinal acceleration is set to be $a_{x\cdot\max} = \pm 7.5\,\mathrm{m/s^2}$ since the maximum achievable acceleration or deceleration for a vehicle with perfect traction management is $a_x = \pm\mu g$ and for a passenger vehicle $\mu \approx 0.75$. Maximum lateral acceleration is set at $a_{y\cdot\max} = \pm 0.5\,\mathrm{g} = \pm 4.91\,\mathrm{m/s^2}$. Such a limitation avoids creation of large angles and considerable lateral load shift in order for the bicycle model to remain valid. Both longitudinal and lateral accelerations are well-above the limits of regular driving with passenger vehicles.

1.3.1 Time Response of System Variables

In this section, we examine the time responses of the main vehicle variables in time domain and see how they behave in transient and steady-state conditions. As (1.14) implies, the lateral velocity v_y and yaw velocity r are the main variables of the vehicle system. It is also useful to monitor the lateral acceleration of the vehicle a_y. Lateral acceleration is approximately equal to the centripetal acceleration of the vehicle in a turning maneuver when side-slip angle β is small, and it is an indication of lateral tire forces. To start examining the laziness of vehicles in maneuvers, we may start by calculating the QSS response of the above-mentioned variables between two steady-state conditions.

Using the vehicle parameters provided in Sect. 1.1.1, we first calculate the force system coefficients in (1.7), (1.8), and then all of the steady-state responses in (1.21)–(1.24) and (1.40) are evaluated for the corresponding range of v_x and δ:

$$C_r = -\frac{30{,}000}{v_x}\,\mathrm{N/rad} \tag{1.56}$$

$$C_\beta = -120{,}000\,\mathrm{N/rad} \tag{1.57}$$

$$C_\delta = 60{,}000\,\mathrm{N/rad} \tag{1.58}$$

$$D_r = -\frac{195{,}000}{v_x}\,\mathrm{Nm/rad} \tag{1.59}$$

$$D_\beta = 30{,}000\,\mathrm{Nm/rad} \tag{1.60}$$

$$D_\delta = 60{,}000\,\mathrm{Nm/rad} \tag{1.61}$$

$$S_y = -\frac{2v_x(v_x^2 - 225)}{v_x^2 + 750}\,\mathrm{m/s\,rad} \tag{1.62}$$

$$S_r = \frac{300v_x}{v_x^2 + 750}\,\mathrm{rad/s\,rad} \tag{1.63}$$

$$S_\beta = -\frac{2v_x^2 - 450}{v_x^2 + 750}\,\mathrm{rad/rad} \tag{1.64}$$

$$S_a = \frac{300v_x^2}{v_x^2 + 750}\,\mathrm{m/rad\,s^2} \tag{1.65}$$

$$S_\kappa = \frac{300}{v_x^2 + 750}\,\mathrm{1/m\,rad} \tag{1.66}$$

For transient simulation, system (1.14), or alternatively (1.55), is solved numerically for v_y and r. The lateral acceleration in transient maneuver can be calculated from (1.10) as:

$$a_y = \dot{v}_y + r v_x \tag{1.67}$$

where $\dot{v}_y$ is found from (1.14) as:

$$\dot{v}_y = \frac{C_\beta}{m v_x} v_y + \left(\frac{C_r}{m} - v_x\right) r + \frac{C_\delta}{m}\delta \tag{1.68}$$

substituting in (1.67):

$$a_y = \frac{1}{m}\left(\frac{C_\beta}{v_x} v_y + C_r r + C_\delta \delta\right) \tag{1.69}$$

Equivalently, we may obtain a_y directly from (1.7) as:

$$a_y = \frac{1}{m} F_y = \frac{1}{m} \left(C_\beta \beta + C_r r + C_\delta \delta \right) \tag{1.70}$$

Manuver 1: Increasing Velocity

The first maneuver consists of both increasing and decreasing the forward velocity v_x at the rate of maximum longitudinal acceleration, while keeping the steer angle δ constant. The maneuver starts from an initial velocity of $v_{x0} = 10\,\text{m/s}$ with a large constant large acceleration of $a_{x\cdot\max} = 7.5\,\text{m/s}^2$ up to $v_{x1} = 25\,\text{m/s}$ in 2 s. After another 2 s (to ensure reaching steady-state) vehicle then decelerates at the opposite rate back to $v_{x2} = 10\,\text{m/s}$ in 2 s. The maneuver continues up to $t = 8\,\text{s}$ to ensure reaching steady-state. We may then calculate the constant steer angle at which the maximum lateral acceleration of $a_{y\cdot\max} = 4.91\,\text{m/s}^2$ is created in the middle of the maneuver:

$$\delta = \frac{a_{y\cdot\max}}{S_{a1}} = \frac{4.91}{136.36} = 0.036\,\text{rad} = 2.06\,\text{deg} \tag{1.71}$$

Inputs are plotted in Fig. 1.5.

The initial values of the variables v_y, r are set to their steady-state values at v_{x0} to realize the first steady-state condition at the beginning of the maneuver when solving the differential equations of motion:

$$v_{y0} = S_{y0}\delta = (2.94)(0.036) = 0.11\,\text{m/s} \tag{1.72}$$

$$r_0 = S_{r0}\delta = (3.53)(0.036) = 0.13\,\text{rad/s} \tag{1.73}$$

Figure 1.6 shows the variation of lateral velocity v_y in the QSS case (gradual increase of v_x) as well as the transient response. The difference between the two is

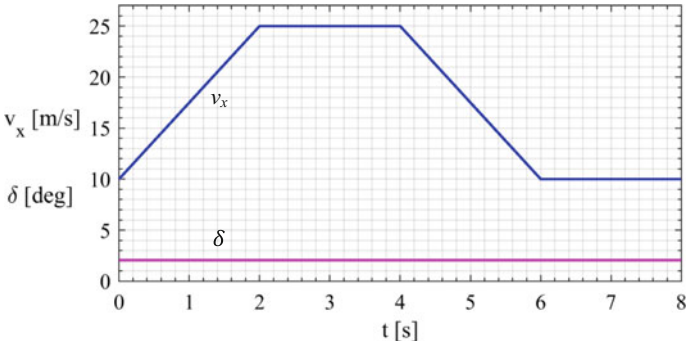


Fig. 1.5 Steering and velocity inputs for maneuver 1

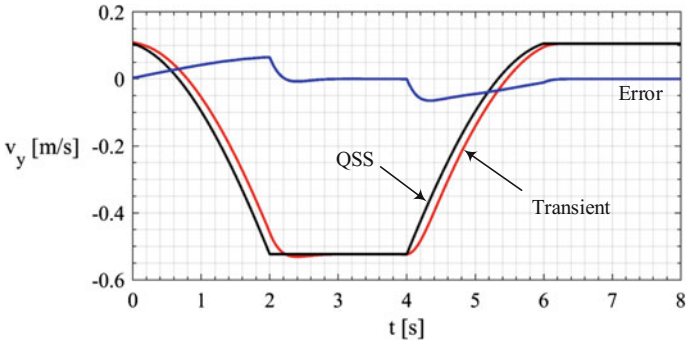


Fig. 1.6 QSS versus transient response of v_y for increasing v_x at constant δ

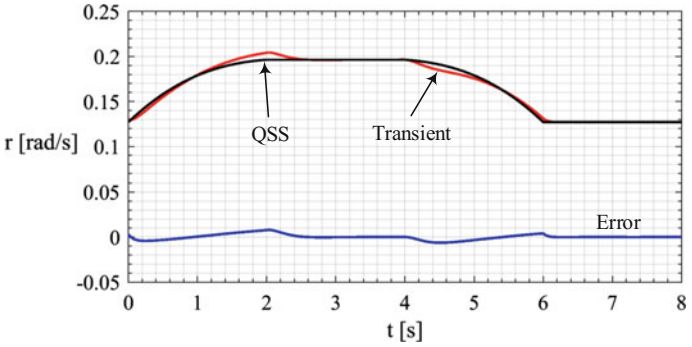


Fig. 1.7 QSS versus transient response of r for increasing v_x at constant δ

also plotted as the error. The plot shows a reasonable level of error at each time. Error magnitude is larger at higher velocities, but settles very quickly when motion becomes steady. It is also observed that the transient curve of v_y is slightly shifted (about 0.2 s) in time domain which is negligible compared to the response time, indicating the laziness of v_y response in this maneuver, both for acceleration and deceleration.

Figure 1.7 depicts the variation of yaw velocity r for both conditions. Maximum error between transient and QSS responses is observed at around $t = 2$ right at the end of transition to higher velocity, but it shows very quick elimination of the error and drop of the yaw velocity to its steady-state in around 0.5 s afterwards. Yaw velocity shows a very similar response to the QSS.

The lateral acceleration plot is shown in Fig. 1.8. It can be seen that the deviation between the responses is almost uniform during transition and it does not exceed $0.2\,\mathrm{m/s^2}$ which is around 4% of the maximum acceleration during such a quick transition, proving a close response to QSS.

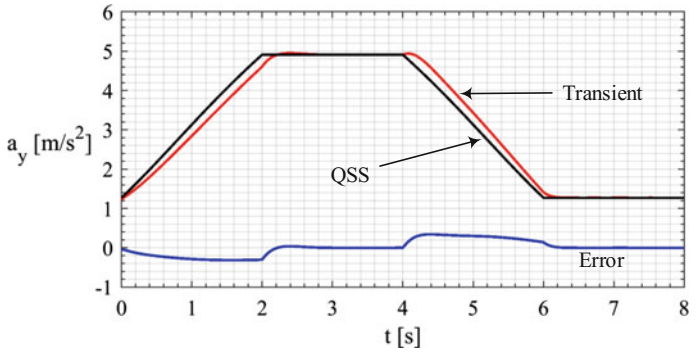


Fig. 1.8 QSS versus transient response of a_y for increasing v_x at constant δ

Maneuver 2: Increasing Steer Angle

The second maneuver is defined as increasing steer angle δ at constant velocity of $v_x = 20\,\text{m/s}$ from an initial value of $\delta_0 = 0$ with constant rate of $1\,\text{deg/s}$ up to the final value of δ_1. The final value of the steer angle is calculated such that the final lateral acceleration is approximately equal to $a_y \approx a_{y\cdot\max} = 4.91\,\text{m/s}^2$ as the limiting value.

$$\delta_1 = \frac{a_{y\cdot\max}}{S_{a1}} = \frac{4.91}{104.35} = 0.047\,\text{rad} = 2.70\,\text{deg} \tag{1.74}$$

$$t_1 = \frac{\delta_1 - \delta_0}{1} = 2.7\,\text{s} \tag{1.75}$$

Similar to the previous maneuver, the initial values of the variables v_y, r are set to their steady-state values at δ_1 to realize the first steady-state condition at the beginning of the maneuver:

$$v_{y0} = S_{y0}\delta_0 = 0\,\text{m/s} \tag{1.76}$$

$$r_0 = S_{r0}\delta_0 = 0\,\text{rad/s} \tag{1.77}$$

The simulation continues up to $t = 5\,\text{s}$ after $t = t_1$ to damp any transient behavior. Velocity and steer inputs are plotted in Fig. 1.9.

Variation of v_y for QSS and transient maneuvers for the second maneuver are shown in Fig. 1.10. It is observed that the transient value of v_y shows less agreement with QSS response, indicating some expected side-slip difference between transient and QSS during the transition. However, the effect of v_y in total velocity v is negligible.

Variation of yaw velocity r is shown in Fig. 1.11. There is a high level of agreement between plots and the steady-state behavior seems to be dominant.

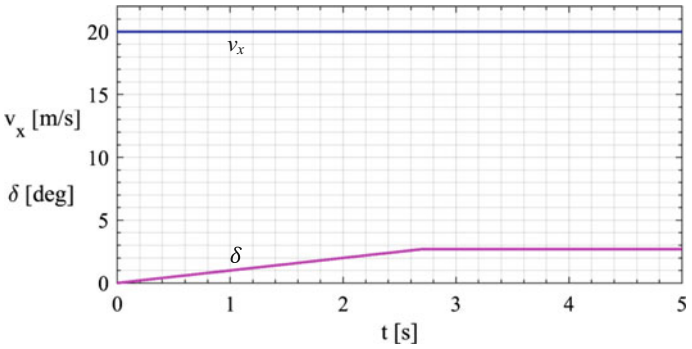


Fig. 1.9 Steering and velocity inputs for maneuver 2

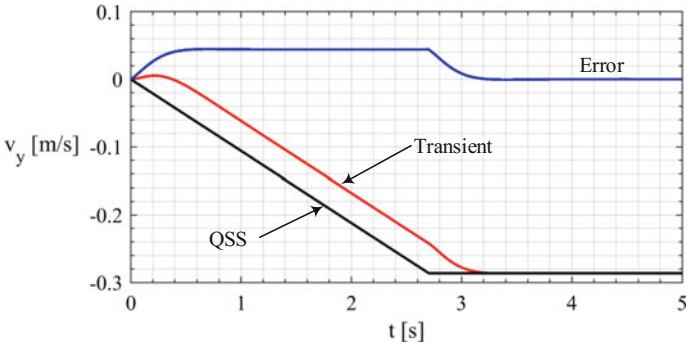


Fig. 1.10 QSS versus transient response of v_y for increasing δ at constant v_x

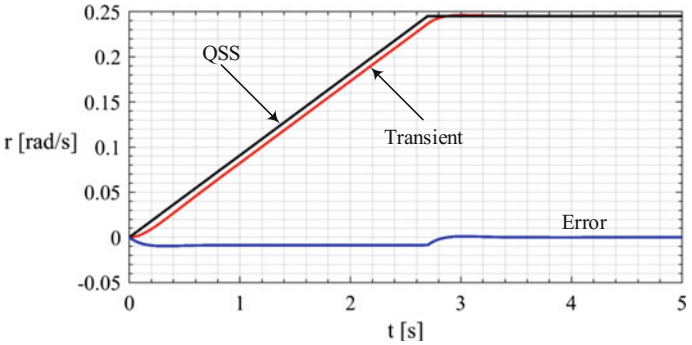


Fig. 1.11 QSS versus transient response of r for increasing δ at constant v_x

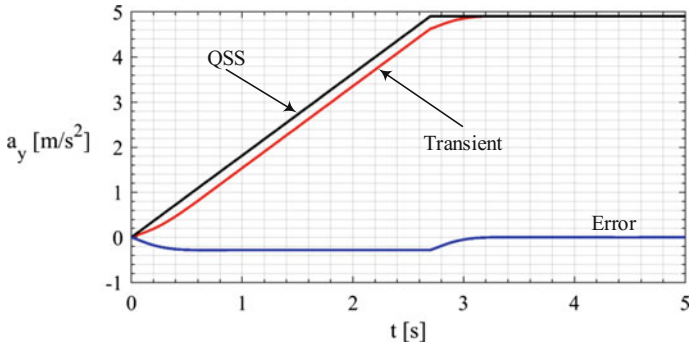


Fig. 1.12 QSS versus transient response of a_y for increasing δ at constant v_x

Figure 1.12 shows how the lateral acceleration varies during the maneuver in transient and QSS conditions. Similar to yaw velocity, the lateral acceleration response shows a high tendency to QSS case, proving laziness of the vehicle with respect to a_y.

Steady-State Surface Maps

It was shown in sections "Manuver 1: Increasing Velocity" and "Maneuver 2: Increasing Steer Angle" that the time response of the vehicle variables is close to their QSS responses. Although v_x is a varying parameter of system (1.14), we may also treat the steer angle δ and the forward velocity v_x as the inputs from the driver to the vehicle system. Having v_x and δ as inputs and using the conclusion above, we may introduce 3-dimensional surfaces consisting of steady-state variables $(v_y)_{ss}, (r)_{ss}, (a_y)_{ss}$ for each pair of v_x, δ. Such a surface is called a *steady-state surface map* of the variable of interest which indicates the steady-state value of that variable in the input domain, instead of time domain.

We expect the transient response plots to lie very close to the steady-state surface maps, because of the transient time response of the vehicle being close to its QSS response. To plot the surface maps, we create a mesh for a range of v_x and a range of δ values and calculate the steady-state variable for each point.

Figures 1.13, 1.14, and 1.15 show the steady-state surface maps for v_y, r, a_y respectively. They include full information about all possible responses of variables with any set of inputs. It is important to note that surface maps do not include any information about the time. To find the time duration in which the transient results are obtained, one must refer to the definition of the input as a function of time, presented in sections "Manuver 1: Increasing Velocity" and "Maneuver 2: Increasing Steer Angle".

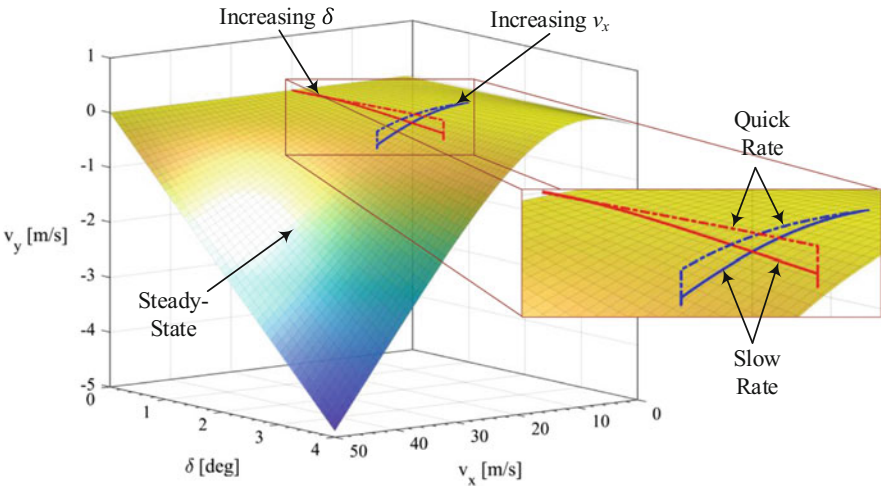


Fig. 1.13 Steady-state surface map of v_y

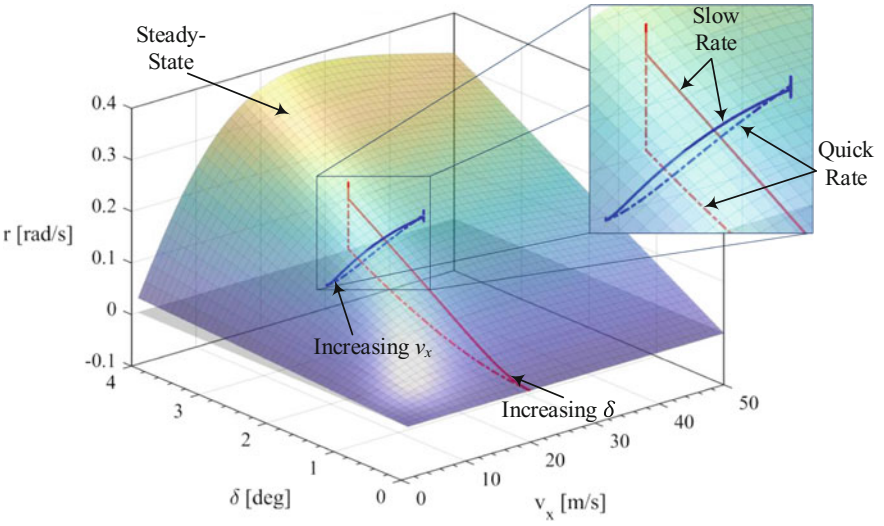


Fig. 1.14 Steady-state surface map of r

Similar to sections "Manuver 1: Increasing Velocity" and "Maneuver 2: Increasing Steer Angle", two maneuvers of increasing $v_x : 10 \rightarrow 25\,\text{m/s}$ at $\delta = 2.06\,\text{deg}$ in two different time durations of 2 s and 4 s are plotted by the blue curves. Another maneuver of increasing $\delta : 0 \rightarrow 2.7\,\text{deg}$ at $v_x = 20\,\text{m/s}$ in two different time durations of 1 s and 2 s is plotted by the red curves. As expected, it is observed that as the time duration of the transition between two steady-state cases gets smaller, the transient behavior becomes more visible and the deviation of the transient plot

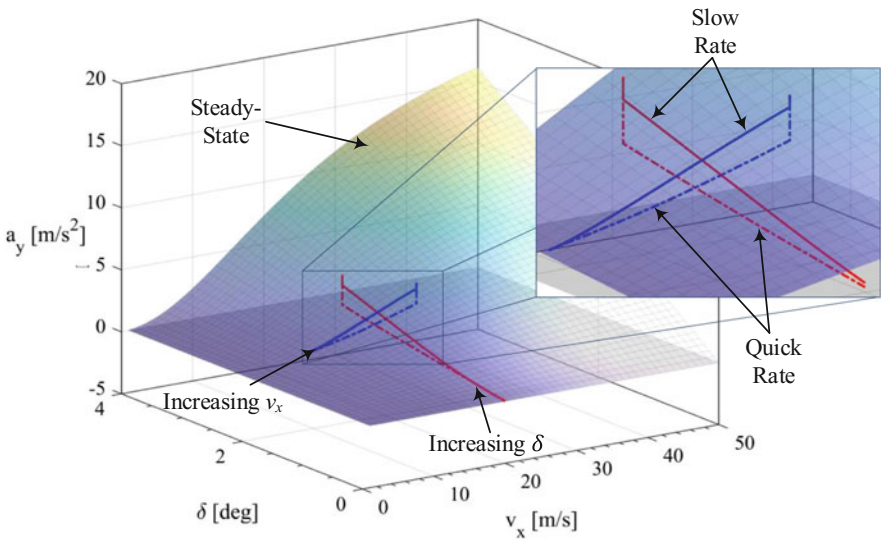


Fig. 1.15 Steady-state surface map of a_y

from steady-state surface map increases. Nevertheless, it can be seen that even the limiting quick maneuver lies reasonably close to the steady-state surface.

Although the focus of this manuscript is on investigating maneuvers with only v_x or δ being variable to assess the sensitivity with respect to each input, let us consider a special case of decreasing steer angle δ and increasing velocity v_x at the same time, which represents a condition of merging into a new road. The transient response of such a maneuver with $\delta : 3\,\text{deg} \rightarrow 0\,\text{deg}$ and $v_x : 10\,\text{m/s} \rightarrow 30\,\text{m/s}$ in 3 s is plotted in Fig. 1.16. It can be seen that the response of this maneuver is also very close to the steady-state.

1.3.2 Center of Curvature Response (ICR Map)

So far, the responses of vehicle body variables are investigated and found to be acting close to their steady-state values in normal driving conditions. It is expected that such a lazy behavior will also be observable in the location of ICR. If we are able to reasonably approximate the location of ICR by steady-state calculation, we may directly relate the ICR response of a certain vehicle only to steer input and forward velocity of that vehicle at any time instance. Hence, we can generate a look-up table for vehicle turning maneuver so that any feasible turning demand (ICR location in body coordinate frame) is translated to a pair of (v_x, δ) for autonomous maneuvering of vehicles.

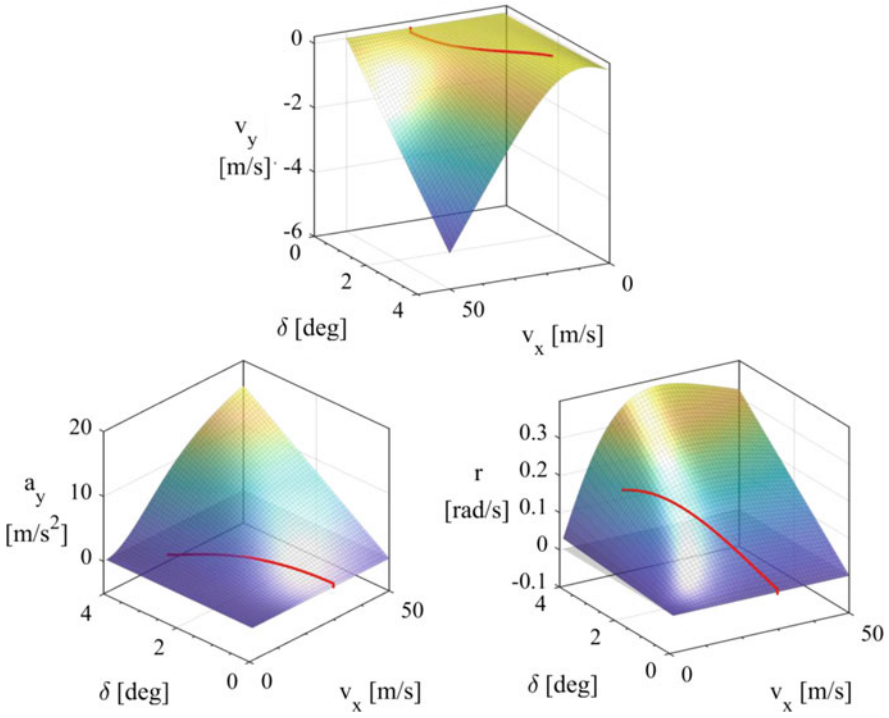


Fig. 1.16 Steady-state surface maps for special maneuver of turning into a road

For a pair of (v_x, δ), the steady-state ICR coordinates in body frame are calculated using (1.41), (1.42):

$$(x_0)_{ss} = -(R)_{ss} \sin(\beta)_{ss} = -\frac{1}{\delta S_\kappa} \sin(\delta S_\beta) \tag{1.78}$$

$$(y_0)_{ss} = (R)_{ss} \cos(\beta)_{ss} = \frac{1}{\delta S_\kappa} \cos(\delta S_\beta) \tag{1.79}$$

Considering specific ranges for v_x and r, a steady-state chart is calculated that contains all the possible ICR locations in body coordinate. Such a chart is called the *ICR map*. Figure 1.17 shows the ICR map for the vehicle of interest in this manuscript for prescribed ranges of v_x, δ.

In Fig. 1.17, the horizontal black lines indicate constant velocity curves and the green lines indicate constant steer angle curves. Note that these curves are nonlinear functions of v_x and δ as expressions (1.78), (1.79) imply, but the range of feasible input values v_x, δ, limits the curves to regions in which they look linear. For any point in between the plotted points, an interpolation may be used to calculate the required (v_x, δ). Analytical calculations are also possible for higher accuracy.

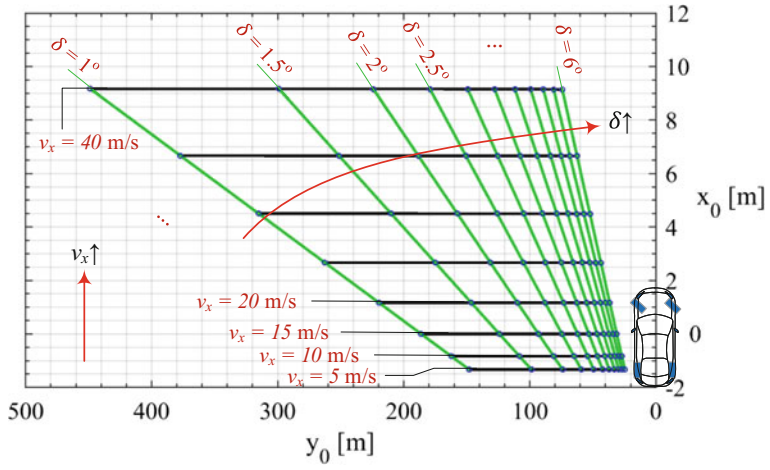


Fig. 1.17 ICR map (loci of possible steady-state ICRs in body coordinate)

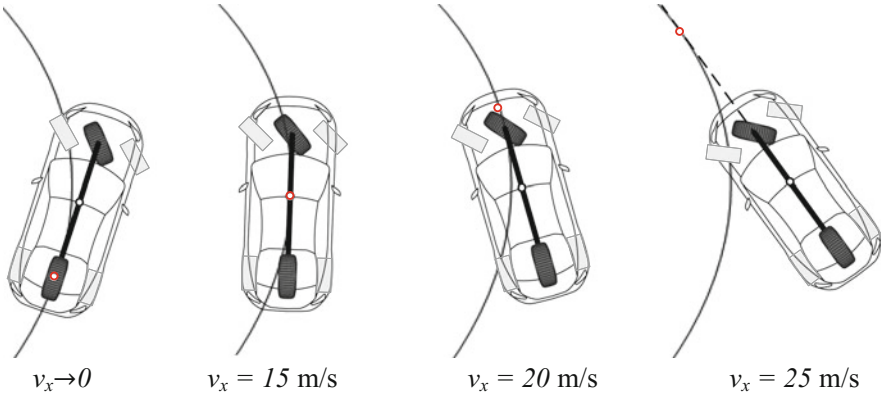


Fig. 1.18 Variation of the tangent point at different velocities

Figure 1.17 indicates that at a certain forward velocity, the longitudinal position of ICR is almost constant for any steer angle. Connecting the point ${}^{B}(x_0, 0)$ in the body coordinate to ICR, we can conclude that the velocity of the body at that point is perpendicular to the radius of curvature and tangent to x. In other words, the point ${}^{B}(x_0, 0)$ is the only point on the body frame that travels tangent to the path. Such a location is called the *tangent point*. Figure 1.18 shows how the tangent point varies at different velocities.

At very low velocities (kinematic turning), the tangent point is on the extension of the rear axle. As the speed increases, the tangent point reaches the center of gravity ($\beta = 0$); this velocity at which the side-slip angle becomes zero is also called the

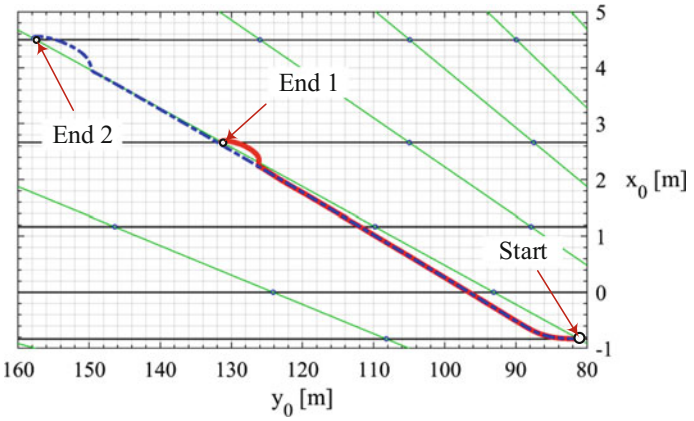


Fig. 1.19 Variation of ICR in body coordinate: effect of Δv_x magnitude at constant δ

tangent speed [2]. It moves further forward with increasing speed. Note that the tangent point is not necessarily on the vehicle body and it may go further beyond the size of actual vehicle body. The tangent point does not go behind the rear axle when the vehicle is moving forward.

Let us compare characteristic transient maneuvers with ICR map to see how the magnitude and the rate of change in the inputs affect the location of ICR when compared to steady-state.

Figure 1.19 shows two different scenarios applied on the same vehicle at constant steer angle of $\delta = 2\,\text{deg}$. The first scenario represents an increase of velocity from $v_{x0} = 10\,\text{m/s}$ to $v_{x1} = 25\,\text{m/s}$ in 2 s. The second scenario represents an increase of velocity from $v_{x0} = 10\,\text{m/s}$ to $v_{x1} = 30\,\text{m/s}$ in 2.7 s. Note that the rate of change in forward velocity is equal for both scenarios. It is observed that, as the rate is equal in both cases, the proximity to QSS is almost equal, but the larger magnitude of Δv_x in the second scenario causes more time of deviation between transient and QSS. Hence, the smaller the change in Δv_x, the lazier the vehicle would be.

Figure 1.20 shows two different scenarios applied on the same vehicle at constant steer angle of $\delta = 2\,\text{deg}$. The first scenario represents an increase of velocity from $v_{x0} = 10\,\text{m/s}$ to $v_{x1} = 25\,\text{m/s}$ in 2 s. The second scenario represents the same increase of velocity in 1 s. It is observed that, as the magnitude of change is equal in both cases, the beginning and end points are the same in both scenarios, but the larger rate of $\Delta v_x/\Delta t$ in the second scenario causes more deviation of transient from QSS. Hence, the smaller the rate of change $\Delta v_x/\Delta t$, the lazier the vehicle would be.

Figure 1.21 shows two different scenarios applied on the same vehicle at constant velocity of $v_x = 20\,\text{m/s}$. The first scenario represents an increase of steer angle from $\delta_0 = 1\,\text{deg}$ to $\delta_1 = 2\,\text{deg}$ in 1 s. The second scenario represents an increase of steer angle from $\delta_0 = 1\,\text{deg}$ to $\delta_1 = 3\,\text{deg}$ in 2 s. Note that the rate of change in steer angle is equal for both scenarios. It is observed that, as the rate is equal in both

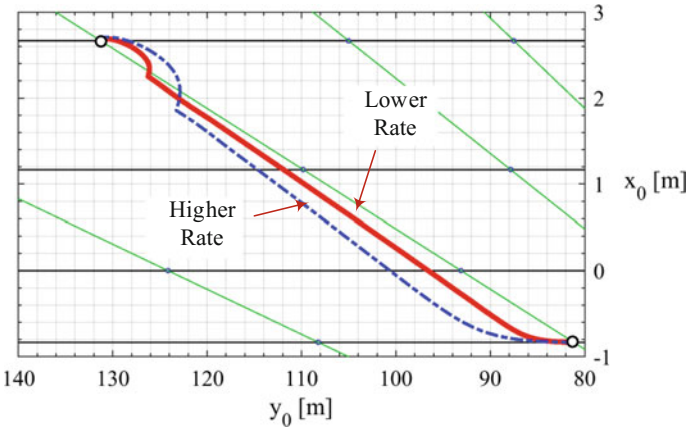


Fig. 1.20 Variation of ICR in body coordinate: effect of $\Delta v_x/\Delta t$ rate at constant δ

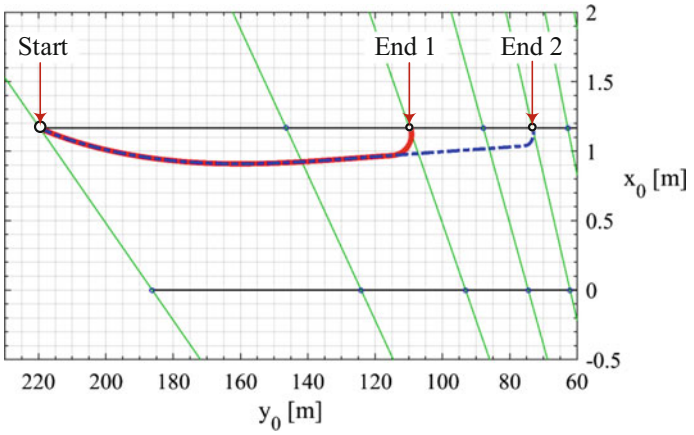


Fig. 1.21 Variation of ICR in body coordinate: effect of $\Delta\delta$ magnitude at constant v_x

cases, the proximity to QSS is almost equal, but the larger magnitude of $\Delta\delta$ in the second scenario causes more time of deviation between transient and QSS. Hence, the smaller the change in $\Delta\delta$, the lazier the vehicle would be.

Figure 1.22 shows two different scenarios applied on the same vehicle at constant velocity of $v_x = 20\,\text{m/s}$. The first scenario represents an increase of steer angle from $\delta_0 = 1\,\text{deg}$ to $\delta_1 = 2\,\text{deg}$ in 1 s. The second scenario represents the same increase of steer angle in 0.5 s. Magnitude of change is equal in these cases, the beginning and end points are the same in both scenarios, but the larger rate of $\Delta\delta/\Delta t$ in the second scenario causes more deviation of transient from QSS. Hence, the smaller the rate of change $\Delta\delta/\Delta t$, the lazier the vehicle would be.

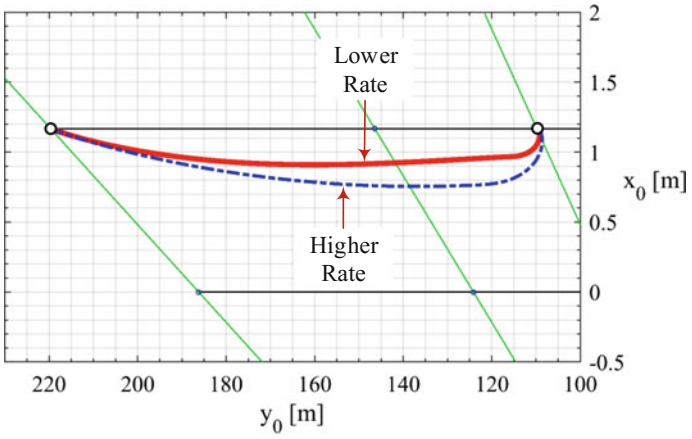


Fig. 1.22 Variation of ICR in body coordinate: effect of $\Delta\delta/\Delta t$ rate at constant v_x

1.3.3 Reference Road Profile

In Sect. 1.3.1, the time response of variables was shown to act very close to their steady-state solutions at each time sample. This step proves that the primary responses act lazily. Such a conclusion allowed us to continue a step further and see how the geometry of turning is affected by examining the location of ICR in Sect. 1.3.2. The transient location of ICR in body frame was shown to follow the ICR map with high level of similarity when the steer angle δ is kept constant while the vehicle accelerates. On the other hand, for variation of δ at constant forward velocity, the deviation between transient location and the ICR map is not that small. Although the scale of x in the ICR map plot is much larger than the scale of y, this deviation needs more investigation to be accepted as reasonably small. For this reason, in order to come up with a stronger conclusion about similarity of turning behavior between transient and QSS cases, we may continue one step further and examine the final path traveled by the vehicle to make a better judgment about the vehicle's behavior in turning maneuvers. As the path is, in fact, the coordinates of the vehicle's center of gravity in global frame, investigation of the path can be viewed as an equivalent to analyzing the ICR location in global frame.

In the previous sections, there was a QSS response defined for each pair of inputs (δ, v_x). In case of path, the QSS response will define an expected path with the given inputs. So, it would be more realistic for applications to start from a given intended path (instead of an expected path) and derive the required inputs (δ, v_x) which are expected to result in the intended path called the *reference road profile*. In fact, it is also highly reasonable to assume v_x to be known from the separately analyzed road in terms of safety considerations. Such safety considerations may include calculation of suggested speed range as a function of road curvature at any point on the road.

For a vehicle to stay on a given road at any time, ICR of the vehicle in global frame must coincide with the center of curvature of the road at that specific time [8]. As observed in Sect. 1.2, the ICR, whether expressed in body or global coordinates, moves on a smooth curve as the inputs to the vehicle change. In other words, there is no jump in the ICR location if the inputs are continuous. As a result, one expects continuous variation of rotation center of the traveled path as well. This implies that a feasible road profile must have continuous loci of center of rotation in global frame. Such a road profile might be feasible to be followed by a vehicle if the velocity and curvature conditions fit in the achievable limit. Such a feasibility limit may be translated to the following statement: "If the location of curvature center fits in the boundaries of Fig. 1.17 at every time, there will be a reasonably accurate solution for inputs (δ, v_x) to make the vehicle stay on the intended road."

A road with non-continuous loci of curvature center is theoretically impossible to be followed continuously. Although there will be approximate input solutions to keep the vehicle in an acceptable error boundary, but since there is discontinuity in the mathematical solution, the accuracy of path following will drop and sudden changes of inputs δ and/or v_x will also be necessary as an undesirable condition. Although the overall geometry of roads with continuous curvature do not visibly vary much from the ones with non-continuous curvature, but the consequences in vehicle control during maneuvering are critical. Note that different sections of the road might be tangent to each other, but have discontinuous curvature.

In the remaining sections we first design a proper sample road, and then use it as a reference road profile for the vehicle to follow.

Clothoid

For design of a road with continuous curvature, we need a mathematical function that acts as a transition curve between two steady-state curves (two circles or a circle and a line). Let us assume that the first section of the road is a circular curve with curvature $\kappa_1 = 1/R_1$ and the second section is another circle with curvature $\kappa_2 = 1/R_2$. Any of κ_1 or κ_2 may be zero in case of straight road.

A preferable transition between the two circular sections is the one with a constant increasing rate for κ with respect to the arc length. Such a transition also requires that beginning and end points coincide with the circular sections and the slope of the transition curve and departure and destination curves match each other. The solution to the constant rate of increase in curvature is the *Euler spiral* which is also called a *clothoid*. The parametric equation for a clothoid is given as [9]:

$$X(t) = a \int_0^t \cos\left(\frac{\pi}{2}u^2\right) du \tag{1.80}$$

$$Y(t) = a \int_0^t \sin\left(\frac{\pi}{2}u^2\right) du \tag{1.81}$$

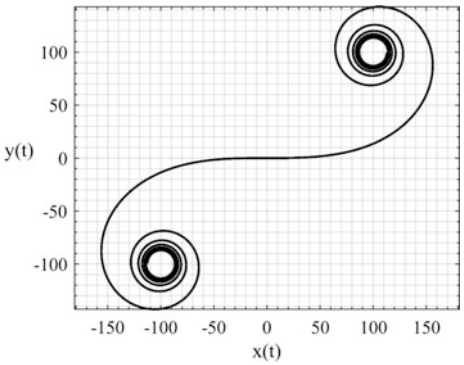


Fig. 1.23 Example of the Euler spiral

where the parameter a which defines the expansion of the curve, if we take t analogous to time, can be viewed as the speed of the pen that slides on the paper to draw the clothoid. One may convert the independent variable from t to arc length s by the following relationship:

$$s = at \tag{1.82}$$

The curvature and the slope of the clothoid along its length are also given as [9]:

$$\kappa = \frac{1}{R} = \frac{\pi t}{a} = \frac{\pi s}{a^2} \tag{1.83}$$

$$\theta = \frac{\pi}{2} t^2 = \frac{\pi s^2}{2a^2} \tag{1.84}$$

An example of a clothoid with $a = 200$ is plotted for $-5 < t < 5$ in Fig. 1.23 which is a representation of the Euler spiral.

Integrals of (1.80), (1.81) are known as Fresnel cosine and Fresnel sine integrals.

Sample Road 1

The first road profile is a road with a circular beginning with radius $R_1 = 160\,\text{m}$ and a second circular section with radius $R_2 = 80\,\text{m}$ at the end. These radii are calculated to approximately match the first scenario in Fig. 1.21. We start by calculating the transition clothoid and then match the circular paths with it, for simplicity. Assuming a constant value for $a = 200$ and using (1.83), (1.84), the clothoid expressions may be found by the following procedures to match the calculated curvatures in the beginning and at the end of the transition:

$$t_1 = \frac{\kappa_1 a}{\pi} = \frac{a}{\pi R_1} = 0.398, \qquad t_2 = \frac{\kappa_2 a}{\pi} = \frac{a}{\pi R_2} = 0.796 \tag{1.85}$$

$$\theta_1 = \frac{\pi}{2} t_1^2 = 0.249\,\text{rad}, \qquad \theta_2 = \frac{\pi}{2} t_2^2 = 0.995\,\text{rad} \tag{1.86}$$

$$X(t) = 200 \int_0^t \cos\left(\frac{\pi}{2} u^2\right) du \qquad (0.398 \leq t \leq 0.796) \tag{1.87}$$

$$Y(t) = 200 \int_0^t \sin\left(\frac{\pi}{2} u^2\right) du \qquad (0.398 \leq t \leq 0.796) \tag{1.88}$$

The centers of the curvatures at the beginning and at the end of the clothoid are located at:

$$X_{O1} = X(t_1) - R_1 \sin\theta_1 = 39.71\,\text{m} \quad Y_{O1} = Y(t_1) + R_1 \cos\theta_1 = 161.65\,\text{m} \tag{1.89}$$

$$X_{O2} = X(t_2) - R_2 \sin\theta_2 = 77.02\,\text{m} \quad Y_{O2} = Y(t_2) + R_2 \cos\theta_2 = 92.74\,\text{m} \tag{1.90}$$

Assuming the same t and a for the circular sections, the parametric equations of the circular sections are given as:

$$X(t) = X_{O1} + R_1 \cos\left(\frac{at}{R_1} + \alpha_0\right) \qquad (t_0 < t < 0.398) \tag{1.91}$$

$$Y(t) = Y_{O1} + R_1 \sin\left(\frac{at}{R_1} + \alpha_0\right) \qquad (t_0 < t < 0.398) \tag{1.92}$$

$$X(t) = X_{O2} + R_2 \cos\left(\frac{at}{R_2} + \beta_0\right) \qquad (0.796 < t < t_3) \tag{1.93}$$

$$Y(t) = Y_{O2} + R_2 \sin\left(\frac{at}{R_2} + \beta_0\right) \qquad (0.796 < t < t_3) \tag{1.94}$$

Intersecting the curves described in (1.87), (1.88) and (1.91), (1.92) at $t = t_1$ and doing the same for curves (1.87), (1.88) and (1.93), (1.94) at $t = t_2$ provides us with the values of α_0, β_0. We may then manually define the beginning point at $t_0 = -1$ and the end point at $t_3 = 3$ of the total road. Note that the independent variable t in the road design is different from the time it takes for the vehicle to travel the path and it is only used to create the reference path; thus, t might even take negative values as in here.

$$\alpha_0 = -1.82\,\text{rad} \tag{1.95}$$

$$\beta_0 = -2.57\,\text{rad} \tag{1.96}$$

$$t_0 = -1 \tag{1.97}$$

$$t_3 = 3 \tag{1.98}$$

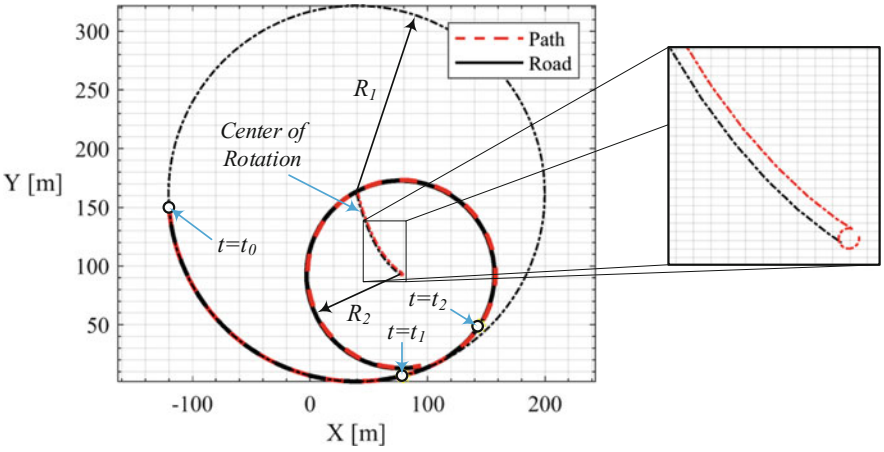


Fig. 1.24 Actual path of motion versus the reference road profile for road 1

Finally, the set of curves described in (1.87), (1.88) and (1.91)–(1.94) are used to draw the reference road profile *sample road 1* as shown in Fig. 1.24. Center of curvature in global frame at any point is also found and plotted from:

$$X_O(t) = X(t) - R(t)\sin\theta(t), \qquad Y_O(t) = Y(t) + R(t)\cos\theta(t) \tag{1.99}$$

where $\theta(t)$ and $R(t)$ are the slope and the radius of the curve at each point which depend on type of the curve at each section.

Sample Road 2

To further evaluate the idea, we introduce another road profile which generates a more complicated maneuver. Assume a straight road which eventually starts to bend in until it merges with a circular path. The road continues to almost complete a circle and then starts to decrease its curvature until it reaches back to its straight position so that it continues the same straight line in the beginning as shown in Fig. 1.25. Such a road is a good example for evaluating laziness of vehicles in maneuvering; if the calculated inputs to the vehicle cause considerable error, the vehicle will not get back to its initial orientation at the end of the road and we will observe an angle made between the initial and the final directions of motion.

The derivation of the road profile is similar to section "Sample Road 1". The second half of the profile may be calculated using a symmetry about the middle axis passing through the center of the circle. To be concise, the derivations are not described here and only the final equations of the curves are given:

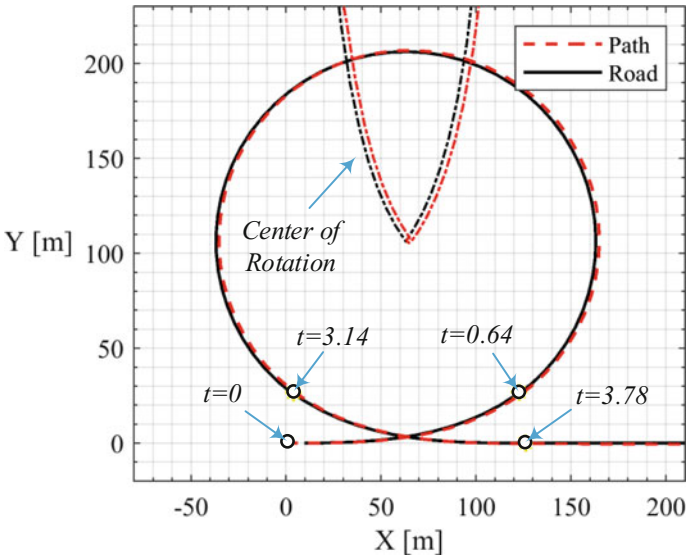


Fig. 1.25 Actual path of motion versus the reference road profile for road 2

$$X(t) = 200 \int_0^t \cos\left(\frac{\pi}{2}u^2\right) du \qquad (0 \le t \le 0.64) \qquad (1.100)$$

$$Y(t) = 200 \int_0^t \sin\left(\frac{\pi}{2}u^2\right) du \qquad (0 \le t \le 0.64) \qquad (1.101)$$

$$X(t) = 63.13 + 100 \cos\left(\frac{200t}{100} - 0.93\right) \qquad (0.64 < t < 3.14) \qquad (1.102)$$

$$Y(t) = 106.23 + 100 \sin\left(\frac{200t}{100} - 0.93\right) \qquad (0.64 < t < 3.14) \qquad (1.103)$$

$$X(t) = 126.25 - 200 \int_0^{3.78-t} \cos\left(\frac{\pi}{2}u^2\right) du \qquad (3.14 \le t \le 3.78) \qquad (1.104)$$

$$Y(t) = 200 \int_0^{3.78-t} \sin\left(\frac{\pi}{2}u^2\right) du \qquad (3.14 \le t \le 3.78) \qquad (1.105)$$

$$X(t) = 200t + 126.25 \qquad (t > 3.78) \qquad (1.106)$$

$$Y(t) = 0 \qquad (t > 3.78) \qquad (1.107)$$

1.3.4 Actual Path

The first step in determining the vehicle's path is to calculate the radius of curvature of the road at each point. Radius of curvature is calculated from (1.83) for clothoid sections and is a constant value for circular sections. The intended center of curvature at every point of the road may be translated into a desired ICR in body frame as in the ICR map in Fig. 1.17, if an allowable range of side-slip angle β is known as shown in Sect. 1.2.2, which can be used to solve for a non-unique pair of inputs (δ, v_x). In another approach, a constant velocity v_x may be assumed throughout the road by making sure that the centripetal acceleration required $a_c \approx v_x^2/R$ does not exceed the feasible limit of the vehicle. Then, the only unknown to be calculated is the steer angle δ and the problem is solved by obtaining a vector of δ associated with the vector of time t or distance traveled s. This may be done geometrically using Fig. 1.17 or using the curvature response S_κ from (1.40).

In this section, for road 1, we use the same constant velocity of $v_x = 20\,\mathrm{m/s}$ used in maneuver 1 of section "Manuver 1: Increasing Velocity". For road 2, we select the constant forward velocity of $v_x = 22\,\mathrm{m/s}$ considering the fact that the maximum centripetal acceleration required during the circular motion would be $a_c \approx v_x^2/R = 4.84\,\mathrm{m/s^2}$ which is below the limit of $4.91\,\mathrm{m/s^2}$.

Figure 1.24 and 1.25 show the actual path of the vehicle on road 1 and road 2 by inputting the steer angle calculated from the steady-state response. The location of ICR in global frame is also plotted with is calculated from (1.48), (1.49). It can be seen that the path of motion is very close to the reference road in both cases. These results show that there is a great agreement between path of motion of transient and QSS responses. It also shows that steady-state responses are well-suited for control of the vehicle and keep it on the road on properly designed roads, proving laziness of vehicles in turning maneuvers.

The maximum perpendicular distance (error) between the path and the road for road 1 is around 1 m and for road 2 is around 1.5 m which are negligible compared to the radii of roads. It also shows that the vehicle will almost stay inside the intended lane, which is typically around 3 m wide.

It is important to note that there is a visible difference between the center of curvature of the road and global ICR location which is almost eliminated in the path of motion. Such a deviation in rotation centers is due to the fact that the heading direction of the vehicle is not necessarily tangent to the road. In other words, there is a nonzero side-slip angle β present, as expected from Fig. 1.10, that causes ICR to rotate on a circle instead of staying at a fixed point, at steady-state. The value of the side-slip angle is dictated by the imposed forward velocity v_x as discussed in Sect. 1.3.2; however, radius of rotation of the vehicle and the path are equal at steady-state. Figure 1.26 shows this in more detail in an exaggerated manner.

To see the effect of reference velocity v_x on the imposed side-slip angle β and ICR deviation from road center of curvature, we may conduct the same simulation on road 1 with different forward velocities and plot them together as shown in Fig. 1.27.

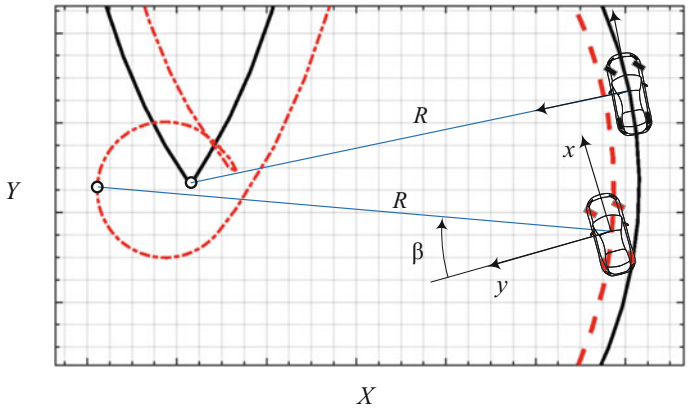


Fig. 1.26 Detailed view of the side-slip angle while maneuvering on the road

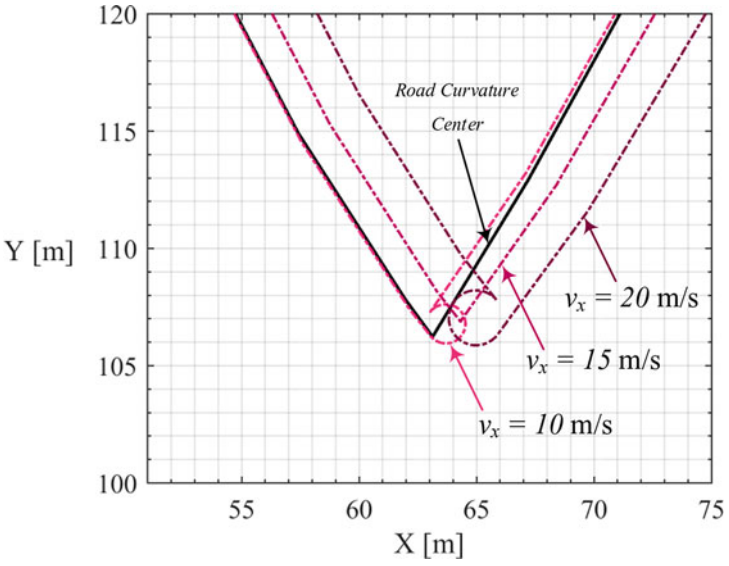


Fig. 1.27 Effect of v_x on side-slip angle β and ICR deviation

It can be seen in Fig. 1.27 as the velocity gets closer to the zero side-slip velocity explained in Sect. 1.3.2, the deviation of ICR from road center of curvature is decreased, implying that the vehicle is moving with more similar heading direction to the road. The side-slip angle is positive for $v_x = 10$ m/s, zero for $v_x = 15$ m/s, and negative for $v_x = 20$ m/s.

1.4 Summary and Conclusions

The objective of this chapter was to show how close vehicles act to their steady-state when undergoing a transient maneuver in normal driving conditions. The well-known bicycle model was chosen to simulate vehicle behavior as it has proven to be a reasonable approximate modeling for normal driving conditions with relatively low accelerations and slip angles. Detailed equations of motion of the bicycle model were given in Sect. 1.1 and steady-state solutions were extracted and summarized in Eqs. (1.21)–(1.24) and (1.40). Calculations of curvature, turning radius, the location of instantaneous center of rotation, and the calculation of path of motion were explained in Sect. 1.2.

In Sect. 1.3.1, transient and quasi-steady-state solutions were compared in time domain for two main transient maneuvers which reveal the overall behavior of the vehicle variables in time domain and also in inputs' domain using surface maps. The similarity between the transient and quasi-steady-state responses were promising, except v_y response when δ was increasing, and we concluded that overall vehicle behavior is lazy in turning maneuvers between two steady-states, as the major activity in normal driving.

Investigation on the laziness of vehicles needed more evaluation on the next level of vehicle response, namely the location of instantaneous center of rotation. This investigation was performed in Sect. 1.2 by calculating body and global expressions of the point. A useful chart was presented in Sect. 1.3.2 as the "ICR map" of a vehicle. The concept of tangent point was explained in detail, clarifying the quality of turning maneuver with geometrical expressions. Examples of basic transient maneuvers were also investigated on the "ICR map" and it was found that the vehicles also act lazy in that regard, specially when forward velocity changes at constant steer input.

For better understanding the vehicle's behavior when the steer angle changes, road profile and its creation as an expected path of motion were discussed in Sect. 1.3.3 and two sample roads were designed. Such road profiles enabled us to apply an open loop steering control algorithm based on steady-state responses and the results were satisfying, justifying that vehicles are lazy during turning maneuvers in normal driving conditions. It was also found that although the instantaneous center of rotation of the vehicle and the curvature center of the road may not exactly coincide because of the velocity and side-slip relation, the path-following error is kept minimum. It is important to note that such a conclusion does not imply that vehicle path following can be simply achieved by an open-loop control strategy, as vehicles may be driven at more severe circumstances at specific moments. Also, the availability of road and vehicle data, vehicle variable measurements, etc. with high-certainty is not guaranteed, the vehicle model might include unmodeled dynamics, and roads might be imperfectly designed.

As a result, it is concluded that steady-state responses, including the maps presented in this document, may be used as a feed-forward strategy to strongly enhance control quality with high accuracy; second usage of steady-state responses is to get

insight about vehicle's maneuvering behavior and possible operating ranges; another important usage is to design roads based on steady-state characteristic of vehicles. In parallel to steady-state feed-forward control, a feedback control is always needed to compensate for errors caused by different sources including uncertainties and unmodeled dynamics. Such a feedback control in a path-following strategy might be tuned to keep the vehicle in a boundary around the road profile at each time instant, such as a lane width.

List of Symbols

α_i	Tire side-slip angle for tire number i
β_i	Body side-slip angle at the center of wheel number i
v_x	Longitudinal velocity of center of gravity in body frame
v_y	Lateral velocity of center of gravity in body frame
v_{yi}	Lateral velocity of wheel center i in body frame
v	Total velocity of center of gravity
δ	Steer angle
$C_{\alpha i}$	Cornering stiffness of tire number i
m	Vehicle's mass
a_1	Distance from center of gravity to front axle
a_2	Distance from center of gravity to rear axle
l	Wheelbase
I_z	Yaw moment of inertia
F_{yi}	Lateral force of tire number i
F_y	Total lateral force at center of gravity
M_z	Total yaw moment at center of gravity
S_y	Lateral velocity response
S_r	Yaw velocity response
S_β	Side-slip response
S_a	Lateral acceleration response
S_κ	Curvature response
a_x	Forward acceleration in body frame
a_y	Lateral acceleration in body frame
a_c	Centripetal acceleration
K	Stability factor
R	Radius of curvature
κ	Curvature
${}^G R_B$	Rotation matrix from body to global frame
x_0, y_0	Coordinates of instantaneous center of rotation in body frame
X_0, Y_0	Coordinates of instantaneous center of rotation in global frame
X_C, Y_C	Coordinates of center of gravity global frame
X, Y	Road profile coordinates in global frame
X_O, Y_O	Coordinates of road center of curvature in global frame

References

1. Ellis, J. (1969). *Vehicle dynamics*. Business Books.
2. Milliken, W. F., & Milliken, D. L. (1995). *Race car vehicle dynamics*. Warrendale, PA: Society of Automotive Engineers.
3. Jazar, R. N. (2017). *Vehicle dynamics: Theory and application*. Berlin: Springer.
4. Van Zanten, A. T. (2000). *Bosch esp systems: 5 years of experience*. SAE Technical Paper.
5. Rajamani, R. (2011). *Vehicle dynamics and control*. Berlin: Springer.
6. Marzbani, H., Vo, D. Q., Khazaei, A., Fard, M., & Jazar, R. N. (2017). Transient and steady-state rotation centre of vehicle dynamics. *International Journal of Nonlinear Dynamics and Control, 1*(1), 97–113.
7. Jazar, R. N. (2011). *Advanced dynamics: Rigid body, multibody, and aerospace applications*. London: Wiley.
8. Jazar, R. N. (2010). Mathematical theory of autodriver for autonomous vehicles. *Journal of Vibration and Control, 16*(2), 253–279.
9. Marzbani, H., Jazar, R. N., & Fard, M. (2015). Better road design using clothoids. In: *Sustainable automotive technologies 2014* (pp. 25–40). Berlin: Springer.

Chapter 2
Artificial Intelligence and Internet of Things for Autonomous Vehicles

Hamid Khayyam, Bahman Javadi, Mahdi Jalili, and Reza N. Jazar

2.1 Introduction

Artificial Intelligence (AI) is a machine intelligence tool providing enormous possibilities for smart industrial revolution. It facilitates gathering relevant data/information, identifying the alternatives, choosing among alternatives, taking some actions, making a decision, reviewing the decision, and predicting smartly. On the other hand, Internet of Things (IoT) is the axiom of industry 4.0 revolution, including a worldwide infrastructure for collecting and processing of the data/information from storage, actuation, sensing, advanced services and communication technologies. The combination of high-speed, resilient, low-latency connectivity, and technologies of AI and IoT will enable the transformation towards fully smart Autonomous Vehicle (AV) that illustrate the complementary between real world and digital knowledge for industry 4.0. The purpose of this book chapter is to examine how the latest approaches in AI and IoT can assist in the search for the AV. It has been shown that human errors are the source of 90% of automotive crashes, and the safest drivers drive ten times better than the average [1]. The automated vehicle safety is significant, and users are requiring 1000 times smaller acceptable risk level. Some of the incredible benefits of AVs are: (1) increasing

H. Khayyam (✉) · M. Jalili
School of Engineering, RMIT University, Melbourne, VIC, Australia
e-mail: hamid.khayyam@rmit.edu.au

B. Javadi
School of Computing, Engineering, and Mathematics, Western Sydney University, Sydney, NSW, Australia

R. N. Jazar
Xiamen University of Technology, Xiamen, China

School of Engineering, RMIT University, Bundoora, VIC, Australia

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_2

vehicle safety, (2) reduction of accidents, (3) reduction of fuel consumption, (4) releasing of driver time and business opportunities, (5) new potential market opportunities, and (6) reduced emissions and dust particles. However, AVs must use large-scale data/information from their sensors and devices.

The complexity of AV data/information (processing 1 GB per second) is increasing which is used for Advanced Driver Assistance Systems (ADAS) and entertainment. Therefore, it is needed to grow hardware and software requirements, which use sensors, actuators devices and software, to compete the functions similar to the superhuman brain as aimed through AI. AV sensors and devices produce data containing information such as time, date, motion detection, navigation, fuel consumption, voice recognition, vehicle speed with acceleration, deceleration, cumulative mileage, voice search, recommendation engines, eye tracking and driver monitoring, image recognition, sentiment analysis, speech recognition and gesture, and virtual assistance. The total data is thus over a 100 terabyte per year for 100,000 vehicles [2, 3].

This data is predictable to increase further due to the growing adoption of Connected Vehicles (CVs). The ascension of the AV brings new opportunities for industrial manufacturers and dealerships, enabling companies to use AI to increase value for their customers. When it comes to processing this data/information by AI, the most efficient approach is to use Machine Learning (ML) algorithms. The ML algorithms help form behavioural patterns for certain driver profiles and also offer vehicle owners exactly what they need both in the vehicle and through their mobile phones via a corresponding application. They accomplish this by remembering their behaviour and analysing their driving history and the situation on the road.

Although AI can handle the AV big data, some of the extra data conditions, such as traffic, pedestrians, and experiences, will need to be collected through various IoT networks, such as Local Area Network (LAN), Wide Area Network (WAN), Wireless Sensor Network (WSN), and Personal Area Network (PAN). This huge data/information needs to have some substances, such as embedded electronics devices, sensors, vehicles, buildings, software, and network connectivity, that enable them to collect and share the data. These IoT-enabled AVs utilize a number of integrated devices to provide many real-time assistance such as improving safety, reduction of fuel consumption, and security for a vehicle. Both IoT and automotive industry 4.0 will be transformed to provide a big boost through reducing machine failure, improving quality control, increasing productivity, and lowering costs at the same time. The potential and the predictions of IoT technology is astonishing. A report by Morgan Stanley Research [4] shows that at least nine industrial manufacturers will benefit from AVs through providing a number of superior technology, key features, and services:

(1) Original Equipment Manufacturers (OEM), (2) Auto dealers, (3) Autonomous trucks, (4) Chemical engineering, (5) Electric utilities, (6) Semiconductor, (7) IT hardware-software, (8) Telecom and communications, and (9) Beverage and restaurant sectors.

This chapter observes the technical trend towards AV with some discussion about key issues and challenges faced by the automotive industry and Sect. 2.2 explains

the AI field in detail with their approaches. In Sect. 2.3, the AV is described with challenges and opportunities. In Sect. 2.4, the IoT network including cloud and edge computing is demonstrated that allows us to use the generated huge amounts of data from networked and connected devices. Finally, the combination of AI approaches and IoT for AVs is concluded.

2.2 Artificial Intelligence (AI) Approaches

AI is a field of computer science and engineering used for different smart applications aiming to make intelligent machines. AI works and responds like humans, intelligently and independently through learning from experience and adjusting to new participations.

2.2.1 Introduction to Artificial Intelligence: Benefits and Challenges

The trend of industrial revolution such as technologies, automation, and data exchange is shown in Fig. 2.1. Current industries have new challenges in terms of competition and market demand and they must take radical change to Industry 4.0 evolution. Artificial Intelligence (AI) is one of the capabilities that enables

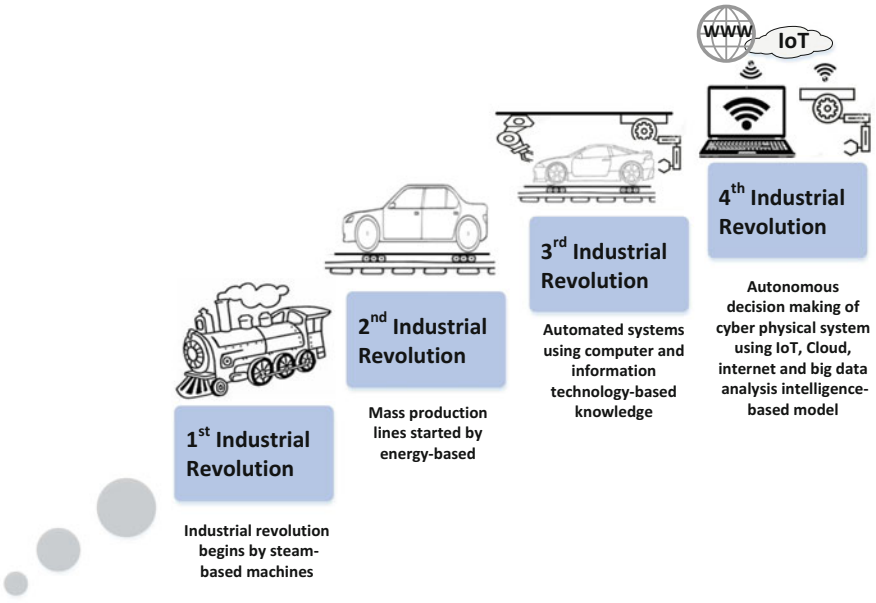


Fig. 2.1 The fourth industrial revolution

improved decision dynamics and better decision precision for industry 4.0, resulting in better business performance, reducing machine failure, improving quality control, increasing productivity, and lowering costs. AI has a number of benefits including: (1) using data to automate learning, (2) enhancing the intelligence abilities to current products, (3) adapting intelligent learning algorithms to do the programming by the data, (4) analysing rationally of data, and (5) improving data accuracy [5]. Although AI most likely changes today's world, it has its own limitations. The biggest challenges of AI are about learning from the experience, and there is no way that the knowledge can be incorporated in the learning. Besides that, any inaccuracies in the data will be reflected in the results and are very challenging.

2.2.2 Artificial Intelligence: History and Approaches

AI is based on combining large amounts of data. It processes the data very fast with iterative processing through the intelligent algorithms, which allow the software to learn from features or patterns of the data. AI has become more widespread since recently it substantially decreased the vision error (less than 5%) in comparison with human vision error as shown in Fig. 2.2 [6, 7].

The history of AI began in antiquity, but it was introduced by John McCarthy in 1950s. A brief evolution of AI is schematically shown in Fig. 2.3 [8]

AI was invented in three main areas: (1) Neural Networks (NNs) from 1950s to 1970s that were mainly focused based on stirs excitement for 'thinking machines', (2) Machine Learning (ML) from 1980s to 2010s that became popular approaches of AI, and (3) Deep Learning (DL) at the present decade that drives the breakthroughs. Our schematic diagram of AI approaches is shown in Fig. 2.4. As can be seen, in general, AI can be divided into three main fields: symbolic learning, statistical learning, and machine learning. These and briefly explained as follows:

Symbolic Learning: Symbolic learning is based on human readable symbols of logic, problems, search and the symbolic learning rules, which are created

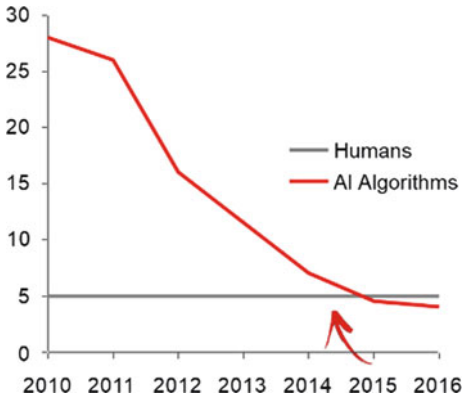


Fig. 2.2 The vision error rate (%) from human and AI algorithm [7]

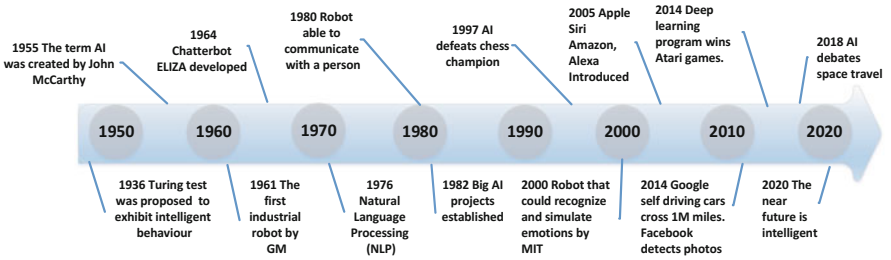


Fig. 2.3 A schematic evolutionary diagram of Artificial Intelligence (AI)

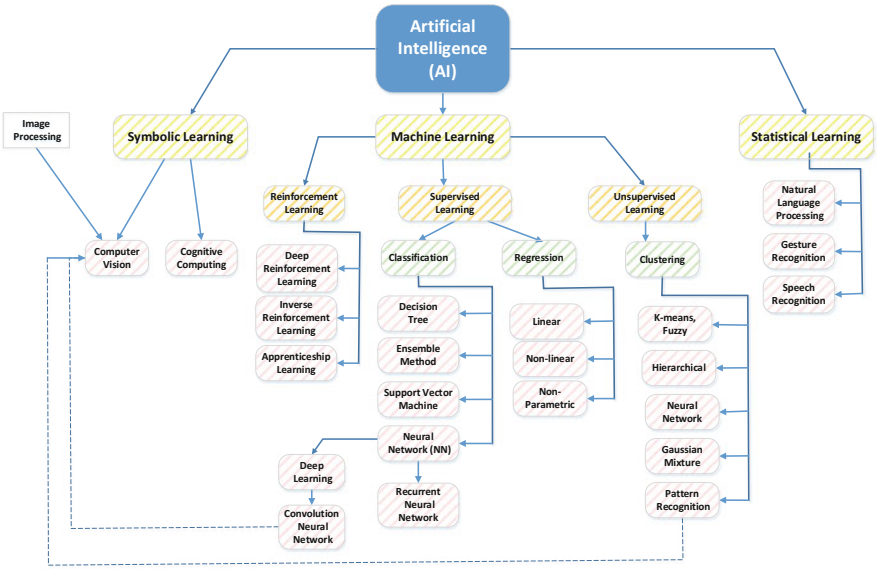


Fig. 2.4 Artificial intelligence approaches/apparatuses

through human intervention. Mixtures of symbols with their interrelations is called reasoning. In order to construct a symbolic reasoning, humans start to learn the rules of the phenomena relationships, and then the code of those relationships transfer into a program. Symbolic learning can be divided into cognitive computing and computer vision.

Statistical Learning: Statistical learning is mathematics intensive and deals with the problem of finding a predictive function based on data. It involves forming a hypothesis before proceeding to building a model. Statistical learning relies on rule-based programming and is formalized in the form of relationship between variables. Statistical learning is also based on a smaller dataset with some attributes, operates on assumptions, such as normality, no multi-collinearity, and homoscedasticity.

Machine Learning: Machine Learning (ML) creates and automates analytical/numerical models and algorithms that can be used to improve the system performance in a specific task. ML uses approaches from heuristic methods, operations research, and statistics, and finds hidden insights in data without explicitly being planned where to look or what to accomplish. The major ML subfields are unsupervised, supervised, and reinforcement learning, which are explained as follows:

Unsupervised Learning: Unsupervised learning is a group of understanding data created based only on input data. One of the unsupervised learning techniques is clustering that involves the grouping of data points through a set of data points. The clustering algorithm can classify each set of data points into a cluster group (see Fig. 2.5a).

Supervised Learning: Supervised Learning (SL) develops a model to predict based on input and output data. SL approaches can be divided into: (1) *Regression* that is an approach to find the relationship between variables. In machine learning, this is used to predict the outcome of an event based on the relationship between variables obtained from the dataset (see Fig. 2.5b). (2) *Classification* that is trying to identify to which of a set of categories for a new observation belongs accurately and it also attempts to predict the target class for each category of the data (see Fig. 2.5c).

Reinforcement Learning: Reinforcement Learning (RL) is a new AI technology based on decision-making that will help AI to advance extremely into the area of machine learning of the real world. A brief comparison of unsupervised, supervised, and reinforcement learning are listed in Table 2.1.

RL uses an agent that learns interactively with the environment through trial-and-error response from its experiences and own actions. Fig. 2.6 shows a simple (RL) framework that can solve the decision problem by using a sequential decision problem through interactive to measure of the feedback performance [9]. In general, RL tries to find a decent mapping that defines perceptions to do any actions for

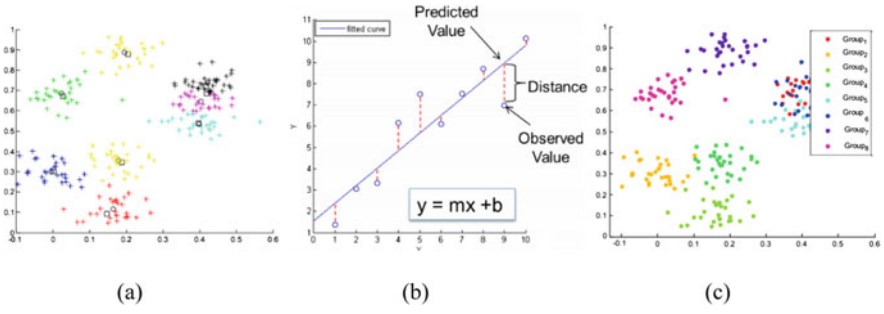


Fig. 2.5 A sample unsupervised and supervised learning methods: (**a**) clustering, (**b**) regression, and (**c**) classification

Table 2.1 Comparison methods of machine learning

	Unsupervised learning	Supervised learning	Reinforcement learning
Model affection	Model does not affect the input data	Model does not affect the input data	Agent can affect its own observations
Learning structure	Learning underlying data structure	Learning to approximate reference answers	Learning optimal strategy by trial and error
Feedback	No feedback required	Needs correct answers	Needs feedback on agent's own actions

Fig. 2.6 A simple reinforcement learning framework

addressing situations for the decision-maker cooperating with an environment. RL is a powerful method to speed up initial learning with remarkable results [10, 11] that has become popular within the community of computer science, automation, control, and mechatronics, and recent years have perceived to widely use especially to solve the multi-agent problems.

The RL can be mathematically formulated as follows. The environment is a current state s out of a set S, and the agent action is a subset of an action set A, where S: $S \times A \rightarrow S$ and R: $S \times A \rightarrow \mathbb{R}$, action a in state s, the state $S(s, a)$, a reward $R(s, a)$.

The goal function is to learn a policy function p: $S \rightarrow A$, by choosing actions that maximize expected future rewards. This is typically done by defining a discount factor $\gamma \in [0, 1)$, used to scale future rewards in the total value of a policy:

$$V^p(s) = \sum_{k=1}^{+\infty} \gamma^{k-1} r^k = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \tag{2.1}$$

where rk is the reward obtained after k steps, starting from state s and following policy p thereafter.

2.2.3 *State of the Art of Artificial Intelligence Approaches*

One of the new powerful and flexible machine learning that represents the world with hierarchy of implications with more abstract representations is Deep Learning (DL).

Deep Learning

In most of machine learning techniques, in order to reduce the complexity of the data and make patterns more visible to learning algorithms to work, the applied features need to be identified by a domain expert. The main gain of Deep Learning (DL) algorithms are to learn in high-level features from data in an incremental way [12]. DL is based on conception, perception, and decision-making. It uses huge neural network layers by using many processing units that has many advantages of advances to improve training techniques for learning complex patterns in lots of data. Common complex engineering applications include activity recognition, video labelling, image, speech recognition, object recognition, and several types. DL is also transmitting significant inputs to other areas of perception, such as audio, speech, and natural language processing.

Deep Reinforcement Learning

Although AI has many successful approaches, the essential technology of AI is the combination of deep learning and Reinforcement Learning (RL) which produce inspiring results in learning. Deep RL approach extends reinforcement learning by using a deep neural network and without explicitly designing the state space [13, 14]. Thus, Deep RL refers to goal-oriented algorithms to open up many new applications in areas such as engineering, and many more.

2.3 Autonomous Vehicle

2.3.1 Introduction

An Autonomous Vehicle (AV) is a vehicle that can guide itself, as opposed to being controlled by human. The AV is a kind of driverless vehicle that has become in reality and is the art of driving using computers for future. AVs have been targeted due to: (1) increasing vehicle safety, (2) reduction of accidents, (3) reduction of fuel consumption, (4) releasing of driver time and business opportunities, (5) new potential market opportunities, and (6) reduced emissions and dust particles. It is planned that around ten million AVs will be on to the roads by 2020 and is expected that AVs produce $7 trillion annual revenue stream by 2050 [15].

2.3.2 Automated Vehicle: Levels and History

Vehicles have six levels for Advanced Driver Assistance Systems (ADASs) for automated vehicles. The journey level of automation to fully autonomous vehicle as shown in Fig. 2.7 are: Level zero—no automation and the human executes

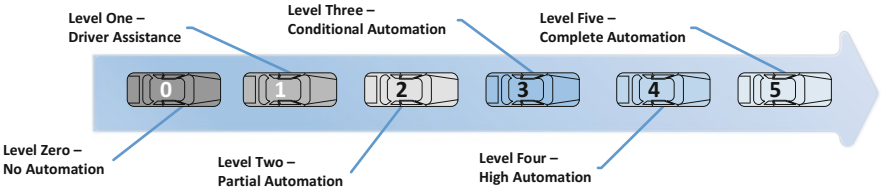


Fig. 2.7 The journey of automation to fully autonomous vehicle

to operate all the dynamic driving tasks like accelerating or slowing down, steering, braking, and so forth. Level one—driver assistance by system of either acceleration/deceleration or steering using information about driving conditions. Level two—partial automation of vehicle which combined automated functions both acceleration/deceleration and steering. Level three—conditional automation of the driving mode that is precise performance by an automated driving system when the driver response to request. Level four—high automation is the vehicle capable of performing all driving functions under certain conditions even if a human driver does not reply to a request. Level five—complete automation is the vehicle accomplished to perform all driving jobs/functions under all conditions [16].

In level four and five, AVs are capable of performing all driving functions by conjunction of many systems and sensors with each other to control a driverless car. Table 2.2 listed the six levels of automated vehicle including the ADAS technologies, sensors, and actuators (detailed in section "AV Objective Sensors") which have been constructed so far [16].

The idea of Autonomous Vehicles (AVs) is started from 1930s when science fiction writers visualized and innovated the self-driving cars as a new challenge for automotive industries. A brief history of autonomous driving is listed in Table 2.3.

In the near future, AV will reach fantastic human performance for competences compulsory for driving by using the sensing algorithms. Intelligent perception is close to do human tasks such as recognition, localization, path tracking, and tracking for the AV. A new report predicts AVs will be widely adopted by 2020 and the adoption of AV competences won't be restricted to individual transportation [17]. As of 2016, many countries such as the USA (Nevada, Florida, California, and Michigan states), Canada, France, the United Kingdom, and Switzerland have approved some laws and regulations for the testing of AVs on public roads.

2.3.3 Autonomous Vehicle: Key Issues and Complexities

In general, Autonomous Vehicle (AV) needs autonomous mobile navigation to find its: (1) localization, (2) map building, (3) path planning, and (4) path tracking. In addition, it is required the AV obstacle avoidance through detection and classification.

Table 2.2 The six levels of automated vehicle including the ADAS technologies, sensors, and actuators [16]

Level 5	Fully autonomous driving										
Level 4	Automated route (trained)					Automated route (destination)					
Level 3	Automated highway driving			Automated city driving		Automated valet parking			Evasive manoeuvres		
Level 2	AEB-city			AEB-urban		AEB pedestrian/ cyclist		Intersection assist	Overtaking assist		
Level 1	Acc Stop and Go	Lane keep		Cross-traffic assist		Real-collision mitigation		Auto parking assist	Navigation		Electric horizon
		Assist									
Level 0	Anti-Lock	Stability	Cruise	Electric power steering	Blind spot morning	Pedestrian/ cyclist		Parking assist	Traffic sign	Lane departure warning	Forward collision warning	Surround/rear view
	Brakes	Control	Control			Detection			Recognition			
Sensors/ actuators	Propulsion controls	Steering	Brake					Inertial	Ultrasonic			Driver
		Controls	Controls	Radar	Cameras	LiDAR	GPS	Sensors	Sensors	MAPS	V2X	Monitoring

Table 2.3 A brief history of autonomous driving by various research and development projects

Year	Companies/projects	Activity
1925	Houdina Radio Control	Demonstrates a radio-controlled 'driverless' car
1939	General Motors	Exhibit 'Futurama' model
1949	RCA	Begin the technical explorations
1950s	General Motors /RCA	Research collaborative a large project
1950s	General Motors	The concept car called Firebird II
1956	General Motors	The Firebird II exhibited is equipped with receivers for detector circuits in roadways
1958	Chrysler	The first car with cruise control called imperial
1960s	Kikuchi and Matsumoto	Wire following in Japan
1964	General Motors	Futurama II exhibit
1964	OSU	Research by Fenton
1970s	Tsugawa	Vision guidance in Japan
1979	Stanford Cart	Used a video processing to navigate a cluttered room without human input
1980s	Dickmanns	Vision guidance in Germany
1986	California PATH and PROMETHEUS	Programs start
1994	PROMETHEUS	Demo in Paris
1995	VaMP	Autonomous vehicle drivers (almost) completely autonomously for 2000 km
1995–1998	National AHS Consortium	Demo '97
2003	PATH	Automated bus and truck demos
2004–2007	DARPA	Grand challenges is founded to incentivise autonomous vehicle development
2009	Google	Self-driving car project begins
2015	Tesla	Release its Autopilot software update
2016	Google	Self-driving car has its accident
2017	General Motors	Plans to include autonomous controls in the Bolt and Super cruise in Cadullic Ct6
2017	Volvo	Plans to launch 100 self-driving vehicles to customers

Some of mobility key issues of AVs are: (**1**) **Software accuracy** and fail proof software is needed to make sure no problems will happen, (**2**) **Map completeness and correctness** through improved features on maps with some additional details such as identifying the surrounding objects and generating some virtual maps to assist the AVs in finding the correct way and looking at dynamic obstacles (pedestrians and vehicles), (**3**) **Sensor fusion and estimation** to sense diverse unpredicted conditions to calibration be able to distinguish between very dangerous situations from those less dangerous are needed.

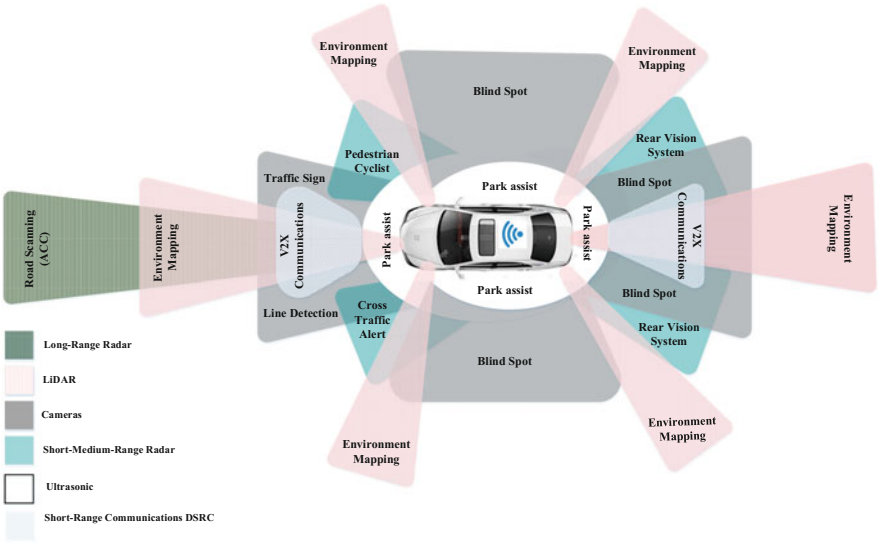


Fig. 2.8 The complexity situation awareness of autonomous vehicle caused by using multi-sensors

The sensors, awareness area, architecture, and software of AVs become quite complex due to the difficulty of the tasks. Among the above AVs issues, currently sensors cannot process quickly to distinguish dangerous situations. A variety of sensors and devices are required in order to keep the vehicle on path and avoid obstacles. The huge information then generate the situational awareness of the vehicle and its surroundings and make appropriate decisions while driving (Fig. 2.8). The combination of sensors with different situational awareness, failures, and real-time response shows the AVs complexities that they need to have a comprehensive software. One approach to reduce complexity of AVs is logical development of actions. Additional approach is to minimize the amount of state information and the duration of the retaining of information. A limited inputs data to the AV system make its behaviour more deterministic. Nonetheless, the main difficulty to reduce data is that the vehicle has limited ability to navigate and manoeuvre. Therefore, there are many AV challenges to be considered and can be solved through a design of AV system architecture and software.

AV Objective Sensors

The AV objective sensors are:

Ultrasonic: The sensor uses sound waves with high-frequency that bounce back to measure the objective distance of a vehicle. It releases sound waves (50 kHz) and listens for bounce back. Then it calculates to determine range based on time-of-flight.

Cameras: Cameras detect the real-time obstacle to enable lane departure and track roadway information (similar to road signs). An image created from camera includes a huge array of values of individual pixels taken individually; these numbers are almost worthless. The image must be understand by conversion of low level information image information into high-level image information by using computer vision algorithms. Computer vision includes analysis of signals from: (a) thermal sensors, (b) cameras, (c) laser range finders, (d) X-ray detectors. Computer vision has three components: (1)Segmentation is where the physical objects are, (2) Classification is what these objects are, and (3) 3D reconstruction is estimating ranges from 2D pictures.

Radar: The sensor releases radio waves that detect short- and long-range depth. Radar sensors dotted around the vehicle monitor the position of vehicles nearby. The radar emits a radio signal (green) which is scattered in all directions (blue). The time-of-flight t for the signal returns the signal to the radar and gives the distance d.

LiDAR: This sensor measures the distance by target brightness with pulsed laser light and measures reflected pulses with sensors to create 3D map of area. LiDAR sensors help to detect the boundaries of roads and identify lane markings by active pulses of light off the vehicle's environments.

DSRC: Dedicated Short-Range Communications (DSRC) is one-way or two-way short-range to medium-range wireless communication channels precisely designed for vehicle use and a consistent set of standards and protocols. DSRC can use as 4G, Wi-Fi, Bluetooth, etc. to Vehicle to Infrastructure (V2I) communication, Vehicle-to-Vehicle (V2V), and Vehicle-to-everything X (V2X). The suitable device is to have lowest latency.

AV Pose Sensors

A vehicle has (at least) six degrees of freedom stated by the pose: $(x, y, z, \varphi, \theta, \psi)$ shown in Fig. 2.9 and AV needs to have some sensors,

Where position $= (x, y, z)$ and attitude: roll is the angle between y' and the x–y plane, $-\pi < \varphi < \pi$, which is the angle between x' and the x–y plane $-\pi < \theta < \pi$, and

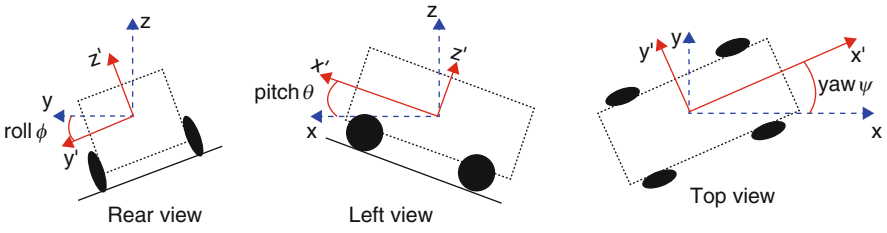


Fig. 2.9 Six degrees of freedom of vehicle dynamics

yaw is the angle between x and the projection of x' on the x–y plane. $-\pi < \psi < \pi$ as shown in Fig. 2.9.

GPS: Triangulates position for a moving receiver: latitude, longitude, altitude, and also speed and direction of movement can be estimated position of vehicle using satellites. Current GPS technology is limited to a certain distance.

Wheel Odometry: It computes changes in the 2D-pose (x, y, θ) from vehicle steering angle and velocity (steering angle from angle sensor and velocity from shaft encoders or speed sensor), it also translates steering angle and velocity to kinematics equations (x, y, θ).

Accelerometers: A variant for measuring change in position (x, y, z) and force F.

Gyroscopes: Measures rotation with one, two, or three degrees of freedom. It estimates (φ, θ, ψ) by summing up gyroscope rotations.

The foremost above AV issues is sensor fusion, as AV needs to have multi-various homogeneous as well as heterogeneous sensors for detection and identification of AV's objectives which are compulsory to design AVs.

Vehicle Dynamics: While pose sensors to be developed rely only on the kinematics of vehicle motion, a dynamic model is required for validating the vehicle performance. The lateral dynamics of a vehicle in the horizontal plane are represented here by the single track, or bicycle model with states of lateral velocity, u_y, and yaw rate, r. The bicycle model (see in Fig. 2.10) is a standard representation in the area of ground vehicle dynamics and has been used extensively in [18, 19]. While detailed derivation and explanation can be found in many textbooks [20, 21], the underlying assumptions are that the slip angles on the inside and outside wheels are approximately the same and the effect of the vehicle roll is small.

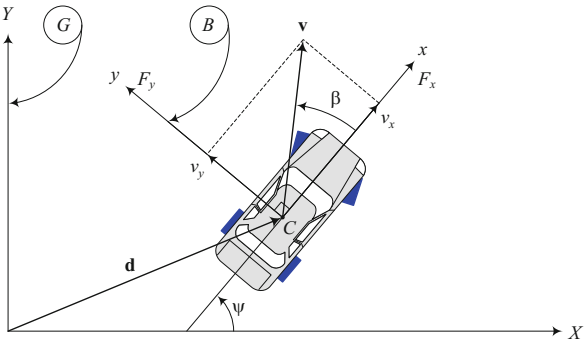


Fig. 2.10 Bicycle model [21]

Data Fusion

As mentioned in the above section, the local situational awareness of Autonomous Vehicle (AV) mostly depends on extracting information from a variety of sensors (e.g. camera, LiDAR, Radar) each of which has its own operating conditions (e.g. lighting, range, power). One of the open issues in the reconstruction and understanding of the environment of AV is how to fuse locally sensed data to support a specific decision task such as vehicle detection [22]. Sensor fusion is a software for combining data from multi-sensors for the determination of improving system performance. The accuracy independent vehicle position and orientation of combining data from the separated sensors will be calculated [23].

Data fusion techniques can be categorized into the following methods:

- Estimation: The method optimally performs the estimation task by using a well-defined statistical framework [24] such as: (1) Weighted Averaging (WA) and (2) Kalman Filtering (DF) [25].
- Classification: It can be used in order to solve classification problems. The challenge is to partition a multi-dimensional feature space into distinct regions where each represents a specific class (group) such as: (1) Density Estimation (DE), (2) K-Nearest Neighbour, (3) Discriminant Analysis (DA), (4) Support Vector Machines (SVM), (5) Decision Trees (DT)[26].
- Inference: This method sets up a further category of fusion techniques based on probability theory such as: (1) Naive Bayesian Inference (NBI), (2) Dempster-Shafer Evidential Reasoning (DSER) [26].
- Artificial intelligence: These approaches are based on heuristic methods such as: (1) Fuzzy logic (FL), (2) Artificial Neural Networks (ANN)[27].

Table 2.4 provides[28] a coarse overview of the advantages and disadvantages of the selected algorithms concerning the applicability to embedded real-time processing.

2.3.4 Recent Autonomous Vehicle Developments

Current vehicles use a wide variety of sensing competences. Moderately, these days a vehicle has seventy sensors including ambient light sensors, accelerometers, gyroscopes, and moisture sensors in the USA [29]. Vehicle sensors are not new components and they were constructed before 2000; the sensors were used for the internal state of the vehicle such as its wheel position, acceleration, and speed. Vehicles already had a number of functionalities such as Anti-lock Braking Systems (ABS), Airbag Control (AC), Traction Control Systems (TCS), and Electronic Stability Control (ESC) for combining real-time sensing with perception and decision-making. The recent commercialized automated competences functions are listed in Table 2.5.

Table 2.4 Overview of advantages and disadvantages of selected algorithms of fusion data [28]

Methods	Advantages	Disadvantages
K-Nearest Neighbour (KNN)	Notable classification results No (re)training phase Distance metrics Error probability bounded	Time consuming Classification time Memory utilization Finding optimal k no online learning
Mahalanobis Distance Classifier (MDC)	Notable classification results Approximative online learning	Estimation of statistics Complex matrix ops
Linear Discriminant Analysis (LDA)	Linear decision boundary Fast classification Fast parameter estimation online learning	Gaussian assumptions Training time Complex matrix ops
Quadratic Discriminant Analysis (QDA)	Quadratic decision boundary Fast classification Fast parameter estimation Online learning	Gaussian assumptions Training time Complex matrix ops
Naive Bayes Classifiers (NBC)	Fast execution ability Fast classification Online learning	Gaussian assumptions
Artificial Neural Networks (ANN)	Fast execution ability Fast classification Arbitrary decision boundaries Online learning	Training time Use of heuristics
SVM	Fast execution ability Fast classification Online learning	Training time Limited decision boundaries

Table 2.5 Recent AV automated functions [29]

Context	Automated functionality	Date
Parking	Intelligent Parking Assist System	Since 2003
Parking	Summon Since	2016
Arterial & Highway	Lane departure system	Since 2004 in North America
Arterial & Highway	Adaptive cruise control	Since 2005 in North America
Highway	Blind spot monitoring	2007
Highway	Lane changing	2015

The automated functionalities help drivers or totally take over well-defined actions for increased comfortability and safety. Current vehicles can perform adaptive cruise control on highways, park themselves, alert drivers about objects in blind spots during lane changes and steer themselves during stop-and-go traffic. Vision and radar technology are used to avoid collision by autonomously brake when risk of a collision is detected for vehicles. By using deep learning, vehicles are able to detect objects in the environment and recognize sound [29].

The automotive industry aimed to continuously develop Autonomous Vehicle (AV) in the last few years [30]. Around 46 private companies work in auto tech on Autonomous Vehicle (AV). A report by Gartner shows that it is expected that by 2020 around 250 million vehicles will be connected with each vehicle, Vehicle to Everything (V2X) or Vehicle to Infrastructure (V2I) systems [30]. Therefore, vehicles will be able to capture and share not only vehicle's situations and location data but also the road conditions (such as weather, traffic congestion and accidents, road geometry, wind...), completely in real time. Although AVs are equipped with sensors and cameras, but the communication systems enable the AVs to generate enormous amounts of data and information. Some of the recent vehicle hardware and software developments are briefly given as follows [31]:

1. Keolis and NAVYA (2017), in partnership with the city of Las Vegas, launched the first autonomous, fully electric shuttle to be deployed on a public roadway in the United States (2017).
2. Toyota (2018) announces 'e-Palette' concept vehicle which is a fully electric autonomous vehicle that can be customized by a partner for applications such as food deliveries (Pizza Hut), ride-sharing (Uber), or store fronts (Amazon).
3. Udelv (2018), a Bay Area tech company, completed the first delivery of goods by a self-driving car when it delivered groceries in San Mateo.
4. Hyundai (2018) announced that a fleet of its fuel cell electric cars made a fully successful automated trip from Seoul to Pyeongchang. This is the first time a Level 4 car has been operated with fuel cell electric cars.

2.3.5 *Artificial Intelligence in Autonomous Vehicle*

An Artificial Intelligence (AI) model for Autonomous Vehicle (AV) includes three steps: (1) data collection, (2) path planning, (3) act as illustrated in Fig. 2.11.

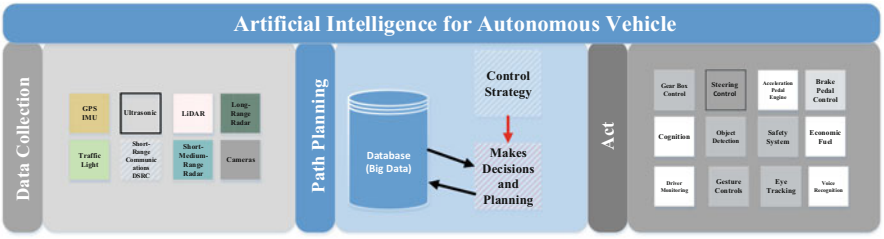


Fig. 2.11 An artificial intelligence model for autonomous vehicle including data collection, planning, and act

Step 1: Data Collection

AVs are equipped with multi-sensors and devices such as radars, cameras, and communication to produce a huge data from its vehicle and environment. These AVs data include the road, road infrastructure, other vehicles and every other object on/near the road, parking, traffic information, transport and environmental information just similar to a human driver. These data are then will be sent to be processed as AV updated information. This is the first AV communication with specific vehicle situations and environment conditions.

Step 2: Path Planning

The huge data from AV system will store and add with pervious driving experiences from every ride in a database called Big Data. Also, an AI agent acts on the Big Data to make meaningful decisions through strategy control. The control strategy of path planning for AVs enables self-driving vehicles to find the safest, most convenient, and most economically beneficial routes from point A to point B by using the pervious driving experiences which help the AI agent make much more accurate decisions in the future. Finding routes is complicated by all the static and manoeuvrable obstacles that a vehicle must identify and bypass. Path planning control strategy involves finding a geometric path from an initial configuration to a given configuration so that each configuration and state on the path is feasible (if time is considered). Path planning control strategy is involved with *Manoeuvre* planning which aims at taking the best high-level decision for a vehicle while considering the path specified by path planning mechanisms and *Trajectory* planning which is the real-time planning of a vehicle's move from one feasible state to the next, satisfying the vehicle's kinematic limits based on its vehicle dynamics and as constrained by the navigation mode. The AV knows exactly what to do in this driving environment and/or driving situation.

Step 3: Act

Based on the decisions made by the AI agent, the AV is able to detect objects on the road, manoeuvre through the traffic, parking spot, obstacles, entertainment, traffic lights, bicycle, pedestrians, working areas, weather conditions, and other vehicles without human driver interposition and goes to the destination safely. AVs are also being equipped with AI-based control and functional systems such as steering control, acceleration by pedal engine, voice and speech recognition, brake pedal control, eye tracking, safety system, gesture controls, economic fuel, and other driving assistance/monitoring systems. These AV process loop including data collection, path planning, and act will take place repetitively. The more the number of data loop takes place, the more intelligent the AI agent becomes, resulting in a higher accuracy of making decisions, especially in complex driving situations.

2.3.6 Recent Autonomous Vehicle Challenges

Though today, Autonomous Vehicles (AVs) have become a reality after many years of research and development, but still, there are a huge mountainous challenges in completely designing autonomous system for the AVs such as: engineering technologies, regulatory, lack of industry standardized technology and tools, consumer trust and acceptance, to name a few. At each progressive level of autonomy, the challenges become more difficult. But, among of the challenges, still engineering technologies especially in Perception, Localization, Planning, Control, and Prediction (PLPCP) of data/information for following conditions/areas are required to be improved [32]:

Road Conditions: Road conditions are extremely changeable and vary uncertain from point to point. In some areas, there are smooth and marked broad highways. But in some other areas, road conditions are highly deteriorated—no lane marking. Lanes are not defined, there are potholes, mountainous and tunnel roads where external signals for direction are not very clear and likewise.

Weather Conditions: Weather conditions play another spoilsport. There could be a sunny and clear weather or rainy and stormy weather. AVs should work in all sorts of weather conditions. There is absolutely no scope for failure or downtime.

Traffic Conditions: AVs would have to get onto the road where they would have to drive in all sorts of traffic conditions. They would have to drive with other AVs on the road, and at the same time, there would also be a lot of humans. Wherever humans are involved, there a lot of emotions are involved. Traffic could be highly moderated and self-regulated. But often there are cases where people may be breaking traffic rules. An object may turn up in unexpected conditions. In the case of dense traffic, even the movement of few centimetres per minute does matter. One can't wait endlessly for traffic to automatically clear and have some precondition to start moving. If more of such cars on the road are waiting for traffic to get cleared, ultimately that may result in a traffic deadlock.

Accident Liability: The most important aspect of AVs is liability for accidents. Who is liable for accidents caused by an AV? In the case of AVs, the software will be the main component that will drive the vehicle and will make all the important decisions. While the initial designs have a person physically placed behind the steering wheel, newer designs showcased by Google, do not have a dashboard and a steering wheel. In such designs, where the car does not have any control like a steering wheel, a brake pedal, an accelerator pedal, how is the person in the vehicle supposed to control the vehicle in case of an untoward incident? Additionally, due to the nature of AVs, the occupants will mostly be in a relaxed state and may not be paying close attention to the traffic conditions. In situations where their attention is needed, by the time they need to act, it may be too late to avert the situation.

Radar Interference: AVs use lasers and radar for navigation. The lasers are mounted on roof top while the sensors are mounted on the body of the vehicle. The

principle of radar works by detecting reflections of radio waves from surrounding objects. When on the road, a vehicle will continuously emit radio frequency waves, which get reflected from the surrounding vehicles and other objects near the road. The time taken for the reflection is measured to calculate the distance between the vehicle and the object. Appropriate action is then taken based on the radar readings. When this technology is used for hundreds of vehicles on the road, will a vehicle be able to distinguish between its own (reflected) signal and the signal (reflected or transmitted) from another vehicle? Even if multiple radio frequencies are available for radar, this frequency range is unlikely to be insufficient for all the vehicles manufactured.

Big Data Analytics: It is required that both training systems and real-time decision-making of AV volumes of data are deployed. Without efficient data management, the sheer resources the process will consume can dramatically slow down innovation. Explore the four data considerations in the AV: (1) data acquisition, (2) data storage, (3) data management, and (4) data labelling. For those early in the data collection process, consideration of one's data approach and thoughtful decision-making regarding relevant tradeoffs will help ensure an action plan that is both executable and expeditious. For those where data collection is becoming increasingly precarious, a careful retrofit that leverages what is already in place can take the organization to a more secure, accessible, and sustainable data approach.

A review of the six intelligent approaches: Representation Learning, Deep Learning, Distributed and Parallel Learning, Transfer Learning, Active Learning, and Kernel-Based Learning for applying scalable machine learning solutions to big data are presented and remarked in detail in this book [33].

Vehicular Communication: In order to resolve PLPCP, AVs need to have a network platform through the Internet communication with a huge data/information for staging and deploying of: (1) vehicle side: vehicle diagnostics data, vehicle real-time location, acceleration, speed, fuel consumption and emissions, and (2) environment side: real-time traffic information, traffic signal messages, safety messages, eco-routes, eco-speed limits, parking information, etc., and (3) energy efficiency powertrain side: hybrids, electric vehicles, and other alternative power sources.

2.4 Internet of Things

2.4.1 Introduction

The technical term Internet of Things (IoT) has been suggested by Kevin Ashton in 1999 [34]. The meaning of 'Things' has changed as technology evolved in last decade, but the main goal which is a digital device can make sense of information without the human intervention remains the same. The Internet made

the interconnection between people possible at an unprecedented scale and pace. The next wave of connectivity is coming much faster to interconnect objects and create a smart environment. There are currently nine billion interconnected devices, more than the number of people in the world and it is expected to reach 24 billion devices by 2020. The main advantage of such a massive number of connected devices is accessing to big datasets which can be utilized in smart applications [35]. Several industries including agriculture, mining, manufacturing, and automotives have already adopted this technology to improve the efficiency and control of their processes. In this section, we will introduce the utilization of IoT technologies in automotive industry which is mainly used for autonomous vehicles.

Autonomous Vehicles (AVs) must have a number of abilities for generation, collection, analysis, processing, and storage of vehicular data from various sources road conditions such as traffic congestion and accidents through a communication network. The Internet of Things (IoT) is an emerging technology which includes a network of physical objects such as: buildings and other items, vehicles, embedded with hardware devices, software, sensors, and network connectivity that enables these substances to collect and exchange data to information without human interaction [34]. The concept of IoT is evolving from Machine-to-Machine (M2M) connectivity as shown in Fig. 2.12. M2M connects isolated systems of sensors to servers with no (or little) human intervention, whereas IoT takes machine-to-machine connectivity, integrates web applications, and connects it to cloud computing systems. Adoption of IoT in AVs has several technical benefits including the capability to monitor vehicles to improve fleet efficiency, safety factors, reduce the vehicle crash, vehicle usage, and provide more responsive service to customers, more drivers interact with the environment around them.

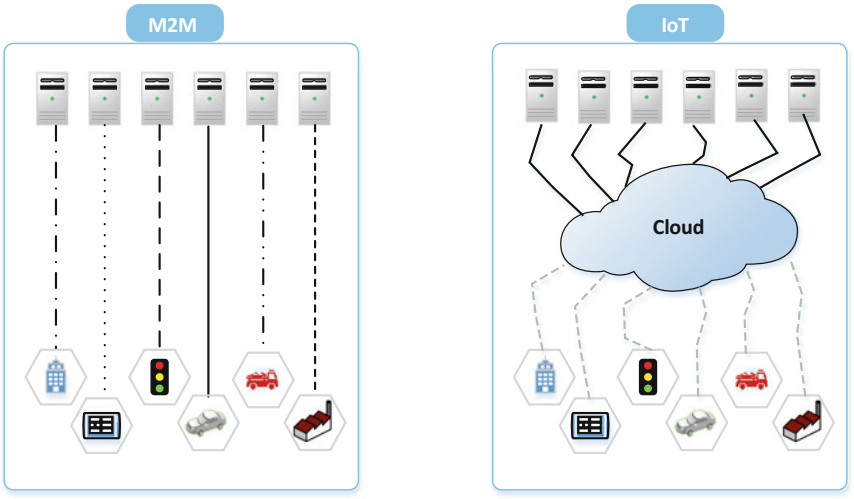


Fig. 2.12 Machine-to-Machine (M2M) and Internet of Things (IoT) connectivity for AVs

2.4.2 *Internet of Things Platform for Autonomous Driving*

A typical IoT platform is an integrated system which is capable of supporting millions of simultaneous device connections to generate a large volume of data to be transferred and processed in cloud computing. There are four main components in a typical IoT platform as depicted in Fig. 2.13. Four components are involved with IoT-AV platform: (1) first component is sensors and hardware devices which are the fundamental components and collect various data types from physical world, (2) the second component is the communication network which is normally based on wireless technologies such as Wi-Fi or cellular technologies (3G, 4G, 5G), (3) the third component is big data which represents the volume, velocity, and variety of data being generated; this data needs to be transferred, stored, and proceed, (4) the forth component of the platform is cloud where the data will be stored and processed as cloud provides several processing, analytics and storage services. IoT applications are traditionally hosted in the cloud and can provide feedbacks and decisions to the physical systems. In the case of AVs, cloud will be the centralized management system where all the software components and monitoring tools will be implemented.

IoT for autonomous driving is transforming the transportation system into a global heterogeneous vehicular network. IoT provides several benefits including dynamic information services, smart vehicle control, and applications to reduce

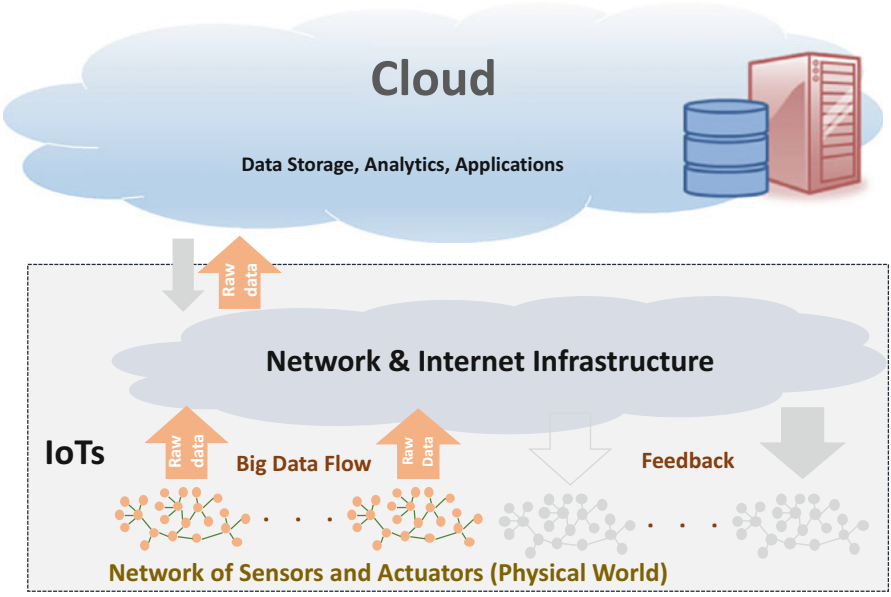


Fig. 2.13 Typical components of an IoT platform

insurance rates and reduce traffic congestions and possibly accidents. In order to create an IoT platform for vehicle, we need to extend the platform in Fig. 2.13 to make up a new ecosystem including the main elements and their interactions and then introduce a network model consisting of the intra-vehicular model and the environmental model. As mentioned before, there are a large number of sensors and hardware devices in the typical IoT platform. In this platform, there are two separate data acquisition components of an AV which are including (1) using data from own sensors, exchanges data with others in neighbourhood, and (2) IoT platform as collection place for large amounts of data from different gateway (parking, traffic information, transport, environmental information) by connected devices (parking spot, train, entertainment, traffic lights, bicycle, pedestrians, working areas, weather conditions, other vehicles). This is where the power of heterogeneous data sources and big data analytics come to picture to provide more comprehensive intelligence for AVs.

2.4.3 Internet of Things Ecosystem for Autonomous Driving

An IoT-based vehicle ecosystem is comprised of six components that interact with each other including: (1) vehicle, (2) person, (3) personal device, (4) network infrastructure, (5) sensing device, and (6) roadside device. Vehicles can be all nearby vehicles which can create a communication link to exchange relevant information such as traffic and road conditions, alerts, and other physical parameters in the ecosystem. Person includes people that request or access a service in the IoT ecosystem. Personal device is a device that belongs to any person in the ecosystem person (e.g. driver, passenger, cyclist) and uses or provides a service. Network infrastructure refers to all devices in the communication network that are used to transfer data in the ecosystem. Sensing device can be sensors and actuators that collect data about the vehicle's parameters, person's health levels, and environmental variables. For instance, this information can include tire pressure, fuel consumption, vehicle temperature for the cars and blood pressure, heart rate of the person and pollution, noise level, and weather conditions. Finally, roadside device is the transportation environment such as traffic lights, information screens or radars that have the ability to disseminate relevant information about traffic and road conditions, accidents, or possible detours.

The essential part of this IoT-based ecosystem is that the interaction among all IoT elements will cause a multi-level data exchange. This interaction, known as Device-to-Device (D2D) interaction may involve many devices (inside and outside of the vehicle) which can communicate, collect, store, and process information or make decisions with no or less human interventions. As proposed in[36], six types of D2D interactions have been identified as illustrated in Fig. 2.14. These interactions are Vehicle-to-Vehicle (V2V), Vehicle and Personal (V&P) device, Vehicle and Roadside (V&R), Vehicle and Sensor (V&S), Vehicle and Infrastructure (V&I),

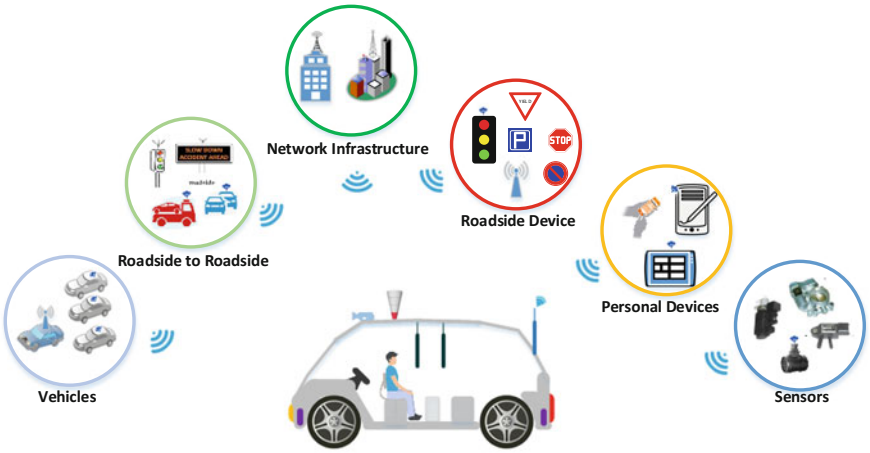


Fig. 2.14 Interaction model for IoT-based ecosystem for an autonomous vehicle

and Roadside and Personal (R&P) device. Additionally, there are two internal interactions namely, Roadside-to-Roadside (R2R) and Sensor & Actuator (S&A).

As can be seen in Fig. 2.14, sensors and some of the personal devices are within the AV and considered as internal interactions and the rest are more external interactions and can be considered as environmental information.

2.4.4 Edge Computing for Autonomous Vehicles

Current IoT platforms for AVs do not enable low-latency and real-time data processing and require offloading data processing to the cloud as shown in Fig. 2.13. The cloud allows access to storage and computing resources from anywhere and facilitates development and maintenances of applications, and related data. Although cloud computing optimizes resource utilization, it cannot provide an effective solution for hosting smart applications required in AVs. These bring several issues and challenges which hinder adopting IoT-driven services for AVs, namely:

- Transferring a large amount of data over the cloud network may incur significant overhead in terms of time, throughput, energy consumption, and cost.
- The cloud may be physically located in a different geographical region, so it is not possible to provide required services for AVs with reasonable latency and throughput.
- Real-time processing of large quantities of IoT data will increase the workload for providers and cloud data centre, with no benefit for the applications and users.
- Existing heterogeneity in hardware and software components of IoT sensors and devices.

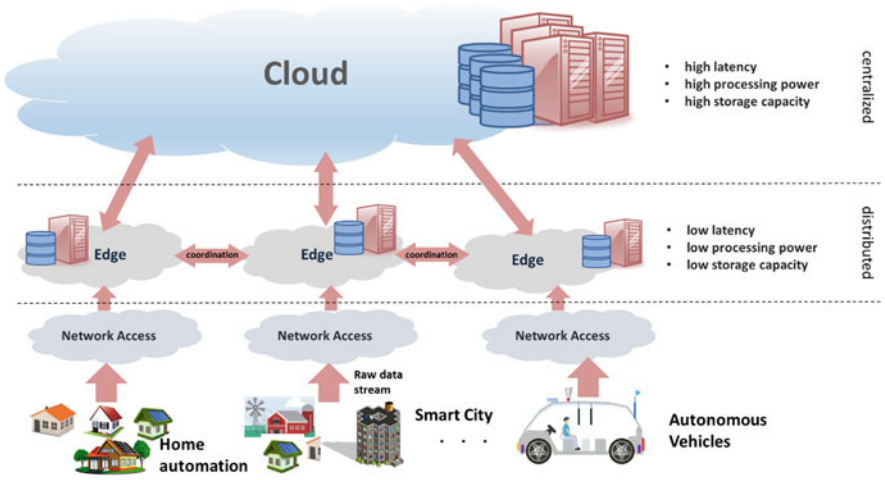


Fig. 2.15 Edge computing for IoT-based autonomous vehicles ecosystem

- Do not always integrate well together.

To address these challenges data analytics could be performed at the network edge near where the data is generated to reduce the amount of data and communications overhead [37]. This concept is called *Edge computing* (or Fog computing) as illustrated in Fig. 2.15. This emerging technology promises to deliver highly responsive computing services, scalability and privacy enforcement, and the ability to mask transient cloud outages [38].

As can be seen in Fig. 2.15, AVs will be connected to the edge devices using wireless communication network to access to real-time data analytics for required applications. Edge devices can collaborate with other edges in the vicinity, thereby creating a local peer-to-peer network beneath the cloud.

Data analytics at the edge of the physical world, where the IoT and data reside introduces an intermediate layer between the data source and the cloud as depicted in Fig. 2.16. By comparing the architecture of cloud computing in Fig. 2.12 and edge computing in Fig. 2.16, the edge computing provides on premise data analytics as well as the capabilities for IoT devices to communicate and coordinate with each other in a distributed environment and with the cloud [39].

Edge computing can be considered as an extension of older technologies such as peer-to-peer networking, distributed data, self-healing network technology, and remote cloud services.

It provides several advantages over standard centralized cloud architectures such as optimizing resource usage in a cloud computing system and reducing network traffic, which reduces the risk of a data bottleneck. Edge computing also improves security and privacy by encrypting data closer to the network core and keeps the private data away from shared cloud environments.

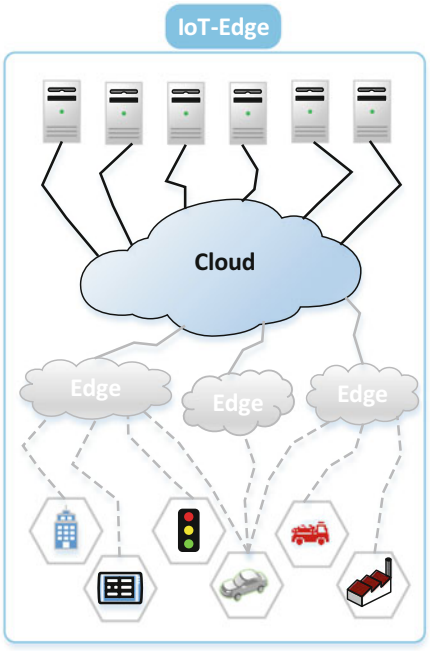


Fig. 2.16 Edge computing for IoT connectivity in AVs

In order to have a better view, Table 2.6 compares edge with cloud computing. As the data are pre-processed, filtered, and cleaned in the edge prior to offloading to the cloud, the amount of transmitted data is much less than the data collected by IoT devices. Also, the analytics on the edge is real-time while the analytics on the cloud is offline. Edge generally has limited computing power and storage compared with the cloud, however, processing on the cloud incurs higher computation latency. The edge offers a high level of fault-tolerance as the tasks can be migrated to the other edge in the vicinity in the event of a failure which is an important factor for AVs as the reliability is one of the main requirements.

Edge device may employ various types of hardware such as computing boards (e.g. Raspberry Pis), multi-core processor, FPGA, or GPU with fine granularity versus a cluster of homogenous nodes in the cloud [40]. Each edge device employs fixed hardware resources that can be configured by the user for each application, whereas the allocated resources are mainly intangible and out of user's control in the cloud. An advantage of edge is the ability of integration to mobile IoT nodes which is essential for AVs. In this case, multiple edge devices in close proximity dynamically build a sub-system in which edge devices can communicate and exchange data. Cloud offers a proven economic model of pay-as-you-go while edge is a property of the user. Edge devices could be battery-powered, so they need to be energy-efficient while the cloud is supplied with a constant source of power with possible energy-efficient resource management.

Table 2.6 Edge vs. Cloud computing

Characteristic	Edge	Cloud
Processing hierarchy	Local data analytics	Global data analytics
Processing fashion	In-stream processing	Batch processing
Computing power	GFLOPS	TFLOPS
Network Latency	Milliseconds	Seconds
Data storage	Gigabytes	Infinite
Data lifetime	Hours/days	Infinite
Fault-tolerance	High	High
Processing resources and granularity	Heterogeneous (e.g. CPU, FPGA, GPU) and fine-grained	Homogeneous (Data centre) and coarse-grained
Versatility	Only exists on demand	Intangible servers
Provisioning	Limited by the number of edge in the vicinity	Infinite, with latency
Mobility of nodes	Maybe mobile (e.g. in the vehicle)	None
Cost Model	Pay once	Pay-as-you-go
Power model	Battery-powered/Electricity	Electricity

2.4.5 Integrating Artificial Intelligence with Edge Computing for Autonomous Vehicles

In order to have AI model for AVs using edge computing, we need to modify the traditional cloud-based model where all data storage and analytics are happening in cloud. This traditional model is presented in Fig. 2.11 where the control mechanisms and database system will be implemented in cloud. In order to improve this model, we need to divide the process and planning section (Fig. 2.11b) into two modules to be handled by edge and cloud collaboratively. Figure 2.17 shows AI-based AV using edge computing where collected data from AV will be transferred to the edge node for pre-processing and decision-making. The data from IoT sensors will be analysed locally in the edge while data of the edge nodes is collected and transmitted to the cloud for offline global processing and less time sensitive decision-making. So, time-sensitive decisions such as obstacle detection or crash avoidance will be performed in the edge node in much shorter time. Whereas, the data about road, traffic and driving pattern are analysed in the cloud to improve the road safety and better driving experience. The AI models implemented in the edge node can be dynamic and updated based on the policies, and relevant rules and regulations and customer requirements. As the data are pre-processed, filtered, and cleaned in the edge node prior to offloading to the cloud, the amount of transmitted data is lower than the data generated by IoT sensors in AVs. This can save considerable amount of bandwidth and cost.

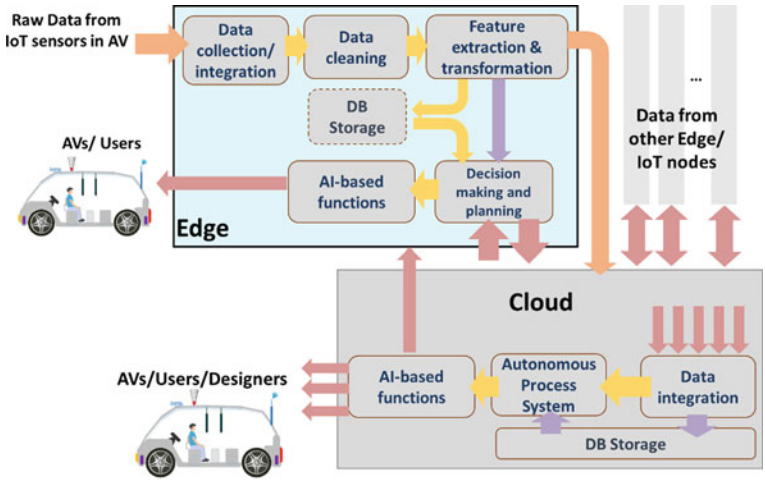


Fig. 2.17 AI-based autonomous vehicles using edge computing

2.5 Conclusions

The growth of Autonomous Vehicle (AV) in recent years creates a new trend to adopt various smart techniques and technologies to improve the performance and quality of automatic decision-making. The integration of Artificial Intelligence (AI) and Internet of Things (IoT) for AV provides high-performance embedded systems that can be utilized in environment to enable more dynamic and robust control systems. While the main software components of AVs are traditionally host by cloud computing systems, new edge computing paradigm has emerged to address some technical challenges such as latency, network bandwidth, and security. The concept architecture of a new AI-based AV using edge computing is proposed in this chapter and can be considered as a fundamental architecture for future research.

References

1. Wu, K.-F., Sasidharan, L., Thorc, C. P., & Chena, S.-Y. (2018). Crash sequence based risk matrix for motorcycle crashes. *Accident Analysis and Prevention, 117*, 21–31.
2. Sherif, A. B. T. (2017). Privacy-preserving ride sharing scheme for autonomous vehicles in big data era. *IEEE Internet of Things Journal, 4*, 611–618.
3. Xu, W. (2018). Internet of vehicles in big data era. *IEEE/CAA Journal of Automatica Sinica, 5*, 19–35.
4. Morgan Stanley Research. (2018). *Car of the future is shared, autonomous, electric*. Retrieved from https://www.morganstanley.com/ideas/car-of-future-is-autonomous-electric-shared-mobility/.
5. Lin, P.-H., Wooders, A., Wang, J. T.-Y., & Yuan, W. M. (2018). Artificial intelligence, the missing piece of online education? *IEEE Engineering Management Review, 46*, 25–28.

6. Harvard Business Review. (2018). *The business of artificial iintelligence*. Retrieved from https://hbr.org/cover-story/2017/07/the-business-of-artificial-intelligence.
7. Elsayed, G. F., et al. (2018). *Adversarial examples that fool both computer vision and time-limited humans*. Cornell University, arXiv:1802.08195v3.
8. McCorduck, P. (2004). *Machines who think: A personal inquiry into the history and prospects of artificial intelligence*. Natick: A.K. Peters Ltd.
9. Sutton, R. S., & Barto, A. G. (2014). *Reinforcement learning: An introduction*. Cambridge: The MIT Press.
10. Szepesvári, C. (2009). *Algorithms for reinforcement learning*. San Rafael: Morgan & Claypool Publishers.
11. Kusy, M., & Zajdel, R. (2015). Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network. *IEEE Transactions on Neural Networks and Learning Systems, 26*, 2163–2175.
12. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge: MIT Press.
13. Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning, 11*(3–4), 219–354.
14. Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica, 3*, 247–254.
15. Lanctot, R. (2017). *Accelerating the future: The economic impact of the emerging passenger economy*. Retrieved from https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/05/passenger-economy.pdf
16. Saiai Company. (2018). *Automated & unmanned vehicles*. Retrieved from https://www.sae.org/automated-unmanned-vehicles/.
17. Lavasani, M., Jin, X., & Du, Y. (2016). Market penetration model for autonomous vehicles on the basis of earlier technology adoption experience. *Transportation Research Record: Journal of the Transportation Research Board*. https://doi.org/10.3141/2597-09.
18. Marzbani, H., Khayyam, H., To, C. N., Voquoc, D., & Jazar, R. N. (2019). Autonomous vehicles, autodriver algorithm, and vehicle dynamics. *IEEE Transactions on Vehicular Technology, 68*(4), 3201–3211.
19. To, C., Quốc, D., Simic, M., Khayyam, H., & Jazar, R. (2018). Autodriver autonomous vehicles control strategy. *Procedia Computer Science, 126*, 870–877.
20. Jazar, R. N. (2018). *Vehicle dynamics: Theory and application*. New York: Springer.
21. Jazar, R. N. (2019). *Advanced vehicle dynamics*. New York: Springer.
22. Rubaiyat, A. H. M., Fallah, Y., Li, X., Bansal, G., & Infotechnology, T. (2018). Multi-sensor data fusion for vehicle detection in autonomous vehicle applications. *Electronic Imaging, Autonomous Vehicles and Machines, 6*, 257–267.
23. Kocić, J., Jovičić, N., & Drndarević, V. (2018). Sensors and sensor fusion in autonomous vehicles. In *26th Telecommunications forum TELFOR*.
24. Loebis, D., Sutton, R., & Chudley, J. (2002). Review of multisensor data fusion techniques and their application to autonomous underwater vehicle navigation. *Journal of Marine Engineering and Technology, 1*, 3–14.
25. Viegas, P. B., Oliveira, P., & Silvestre, C. (2018). Discrete-time distributed Kalman filter design for formations of autonomous vehicles. *Control Engineering Practice, 75*, 55–68.
26. Hardt, P. E., Duda, R. O., & Stork, D. G. (2001). *Pattern classification*. New York: Wiley.
27. Kong, Y., & Guo, S. (2006). Urban traffic controller using fuzzy neural network and multisensors data fusion. *International Conference on Information Acquisition, 2006*, 404–409.
28. Starzacher, A. (2010). *Multi-sensor real-time data fusion on embedded computing platforms*. Klagenfurt: Alpen-Adria-Universitat Klagenfurt Fakultat fur Technische Wissenschaften.
29. Stanford University. (2016). *Artificial intelligence and life in 2030. One hundred year study on artificial intelligence (AI100)*. Retrieved from https://ai100.stanford.edu.

30. Gadam, S. (2018). *Artificial intelligence and autonomous vehicles*. Retrieved from https://medium.com/datadriveninvestor/artificial-intelligence-and-autonomous-vehicles-ae877feb6cd2
31. MCCA Global TEC Forum. (2018). *Autonomous vehicles: Navigating the legal and regulatory issues of a driverless world*. Retrieved from https://www.mcca.com/wp-content/uploads/2018/04/Autonomous-Vehicles.pdf.
32. IIoT World. (2018). *Five challenges in designing a fully autonomous system for driver less cars*. Retrieved from https://iiot-world.com/artificial-intelligence/five-challenges-in-designing-a-fully-autonomous-system-for-driverless-cars/
33. Crawford, B., Khayyam, H., Milani, A. S., & Jazar, R. (2019). Big data modelling approaches for engineering applications. In R. N. Jazar (Ed.), *Nonlinear approaches in engineering applications*. New York: Springer.
34. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*, 1645–1660.
35. Javadi, B., Calheiros, R. N., Matawie, K. M., Ginige, A., & Cook, A. (2018). Smart nutrition monitoring system using heterogeneous internet of things platform. In G. Fortino, A. B. M. S. Ali, M. Pathan, A. Guerrieri, & G. Di Fatta (Eds.), *Internet and distributed computing systems* (pp. 63–74). Fiji: Springer.
36. Vermesan, O., et al. (2018). Automated driving progressed by internet of things. In: *European Union's Horizon 2020 Research and Innovation Programme (2014-2020), SINTEF*.
37. Mehdipour, F., Javadi, B., & Mahanti, A. (2016). FOG-engine: Towards Big data analytics in the fog. In *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Internarional Conference on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 640–646.
38. Satyanarayanan, M. (2017). The emergence of edge computing. *Computer, 50*, 30–39.
39. Hu, P., Dhelim, S., Ning, H., & Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications, 98*, 27–42.
40. Mehdipour, F., Javadi, B., Mahanti, A., & Ramirez-Prado, G. (2019). Fog computing realization for big data analytics. In *Fog and edge computing: Principles and paradigms* (pp. 259–290). New York: Wiley.

Chapter 3
Nonlinear Drilling Dynamics with Considerations of Stochastic Friction and Axial and Tangential Coupling

Jialin Tian, Yinglin Yang, Liming Dai, and Lin Yang

3.1 Introduction

With the development of the global economic, the demand for energy has increased gradually in various countries in the world. In particular, the consumption of oil and natural gas is increasing, which leads to an increase in the exploration and development of oil and natural gas. At present, with the continuous development and improvement of modern drilling technologies, drilling technology has been developed from efficient drilling to enhanced oil recovery, increased oil and gas production, and reduced mining costs. During the actual drilling of oil and gas wells, the downhole work conditions are complicated due to the difference in the formation environment. With the development of new drilling technologies, various new drilling technologies are being developed and applied, which makes drilling technology face more challenges that include the safe evaluation of downhole tool, the reduction friction in the drilling process, and the increase of speed and efficiency.

J. Tian · L. Yang
School of Mechanical Engineering, Southwest Petroleum University, Chengdu, China

University of Regina, Regina, SK, Canada

Y. Yang
School of Mechanical Engineering, Southwest Petroleum University, Chengdu, China

L. Dai (✉)
Xiamen University of Technology, Xiamen, China

University of Regina, Industrial Systems Engineering, REGINA, SK, Canada
e-mail: liming.dai@uregina.ca

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_3

For the spiral buckling of the drill string in the horizontal section, the finite-difference method and the Newton iteration method are used to obtain the calculation model about the spiral buckling critical force and the contact force between the drill string and the borehole wall in Gao Deli et al. [1]. In the drilling process, the spiral buckling phenomenon leads to an increase of the frictional resistance. And when the drill string is subjected to spiral buckling, once the drill string is reversed, the frictional resistance caused by the rotation of the drill string has to be considered to influence the mechanical properties of the drill string.

With the combination of the actual drilling characteristics of large-displacement wells, a three-dimensional soft-pole calculation model is established in He Zhigang et al. [2]. The calculation model considers the change of hole deviation angle and azimuth angle in the actual drilling process, and the effect of drill string stiffness on the calculation model is ignored. The drilling friction model based on large displacement wells is modeled by using the space-oblique plane hypothesis. The three-dimensional soft-pole calculation model can provide a theoretical reference for drilling friction resistance and drilling torque in the design of the drilling trajectory and it can provide help for field monitoring analysis of drilling friction torque.

The influence on the friction resistance of horizontal wells from the effect of lateral vibration of the drill string is conducted in Zhang Huizeng et al. [3]. And the influence law of excitation amplitude and excitation frequency on the friction resistance of the friction pair is obtained. If the excitation frequency is same, there is a linear negative correlation between the friction coefficient and the excitation force. If the excitation force is same, the logarithm of the friction coefficient is negatively related to the excitation frequency. The research conclusions can provide a design basis for the design of the transverse vibration tool.

The effect of frictional resistance on the drill string buckling is studied in Valery Gulyayev et al. [4]. The Euler instability of drill string is predicted by the computer simulation method. And on the basis of the curve elastic rod theory, the Euler stability theory, the channel surface theory, and the classical mechanic method with nonlinear constraints, a modeling method of the drill string critical buckling is proposed. The frictional buckling of drill string is researched in Marcelo A. Jaculli et al. [5] to establish the dynamic model of the drilling process, which includes the problem of not considering the friction between the drill string and the borehole wall or ignoring it, considering the hole deviation angle and the heave motion of drill string. And the research result shows that the friction between the drill string and the borehole wall is an important factor in the drill string buckling. The Aadnoy friction model is established in Ahmed A. Elgibaly et al. [6], based on the friction factors of directional well, which is applied to borehole trajectory design and friction analysis of drilling process, but the model is not accurate for back-pressure and buckling calculations.

The drill string vibration model with the consideration of the axial load is established, and vibration displacement and vibration velocity of different drill string nodes are solved, which are discretized to obtain the solution results of the vibration model in different nodes. Then, the influence analysis of radial inertia effect is conducted, which influences the drill string dynamics to get the influence analysis of parameters. Combined with the axial vibration model and considering the stochastic friction force, the dynamic model with the consideration of stochastic load is developed. And the model solution is carried out to conduct random filed research between the drill string and the wellbore. In addition, the wear research on cutting teeth due to torsional load is conducted and the geometry equation is set up. According to the theoretical research, the calculation result of cutting teeth wear is obtained. The research results are of significance to conduct the study of drill string dynamics under different working conditions and can provide theoretical references for further study of drill string dynamics.

3.2 Dynamics Model

3.2.1 Axial Vibration Model in Drill String Dynamics

Model Assumptions

During the axial vibration analysis model establishment for the downhole drill string, the following model assumptions are made based on the existing drill string vibration research:

1. The drill string is a small-deformed elastic body that can be reduced to an elongate rod with uniform mass, and its axis coincides with the wellbore axis. This chapter only discusses the problem of vibration and friction reduction before the buckling of the drill string.
2. The inner and outer boundaries of the drill string section and the inner wall of the wellbore are both rigid. The cross sections of both the wellbore and the drill string are circular.
3. The speed is continuous during the delivery of the drill string, which does not consider the mutual conversion process of static friction and dynamic friction at each node of the drill string.
4. It is assumed that the drill string is in uniform contact with the well wall, and the gravity, positive pressure, and frictional force on the drill string are uniformly distributed.
5. The drill string in the studied horizontal section is long enough, and the spring short section will greatly reduce the upward propagation of the vibration, so the simplification of the upper boundary condition will not affect the propagation of the vibration wave of the drill string.

Vibration Model

A finite element model for axial vibration analysis in horizontal well is established based on the above model assumptions, as shown in Fig. 3.1. Simplify the drill string into n equal-spaced nodes, each with equal mass.

The axial vibration analysis model is discretized, and the vibration equation of each node is obtained according to Newton's second law. All nodes are considered using the finite element method. The vibration equation of the vibration model can be written as [7]

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = F(t) \tag{3.1}$$

where $x(t)$ is the displacement vector of the node, $\dot{x}(t)$ is the speed vector, $\ddot{x}(t)$ is the acceleration vector, $F(t)$ is the force vector, M is the overall mass matrix, C is the overall damping matrix, and K is the overall stiffness matrix.

According to the node's degree of freedom, the displacement vectors, velocity vectors, acceleration vectors, and force vectors of the n nodes are determined as

$$\begin{aligned} x(t) &= \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \ \dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix} \\ \ddot{x}(t) &= \begin{bmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \\ \vdots \\ \ddot{x}_n(t) \end{bmatrix}, \ F(t) = \begin{bmatrix} F_1(t) \\ F_2(t) \\ \vdots \\ F_n(t) \end{bmatrix} \end{aligned} \tag{3.2}$$

The overall mass matrix and the overall stiffness matrix consisting of n nodes are obtained as follows

$$M = \begin{bmatrix} m_1 & 0 & 0 & \cdots & 0 \\ 0 & m_2 & 0 & \cdots & 0 \\ 0 & 0 & m_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & m_n \end{bmatrix} \tag{3.3}$$

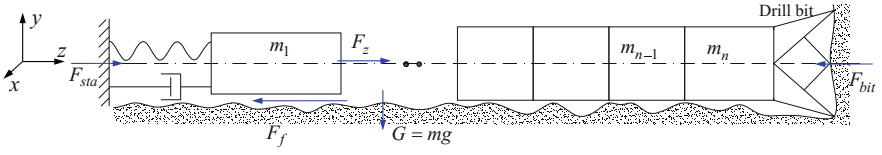


Fig. 3.1 Drill string vibration model in horizontal wells

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & \cdots & 0 \\ -k_2 & k_2 + k_3 & -k_3 & \cdots & 0 \\ 0 & -k_3 & k_3 + k_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -k_n & k_n \end{bmatrix} \tag{3.4}$$

The object will have the effect of damping force as long as there is vibration. The damping force will make the energy dissipation of the vibration system. There are many reasons for the damping force, and the mechanism of damping is also complicated. It is difficult or even impossible to calculate the damping by considering various factors. According to macroscopic studies, there are two main forms of damping force: the first is the damping force generated by the fluid around the object, which is called viscous damping; the second is the damping force generated by the internal friction of the object, which is called the structural damping.

To simplify the calculation, Rayleigh damping is used. It is a kind of linear damping mode, which is a linear combination of mass matrix and stiffness matrix in this chapter. The damping matrix can be expressed as [8]

$$C = \alpha M + \beta K \tag{3.5}$$

where α and β can be valued according to the method provided in document.

Since the drill string is simplified as a rod member with uniform mass distribution and its cross section is circular, the stiffness of each node is calculated as follows according to Hooke's law of material mechanics

$$k_i = \frac{E_1 \left(D_1^2 - D_0^2\right)}{4l} \tag{3.6}$$

where E_1 is the elastic modulus (Pa), D_1 is the drill pipe outside diameter (mm), D_0 is the drill pipe outside diameter (mm), l is the node length (m), and i is the node label, take $2 \sim n$.

The first node of the model is the hydraulic oscillator, k_1 is the stiffness of the spring peg in the hydraulic oscillator, and m_1 is the mass of the hydraulic oscillator. The quality of each remaining node can be written as

$$m_i = \frac{1}{4}\rho l \left(D_1^2 - D_0^2\right) \tag{3.7}$$

According to the force condition of the drill string system, the external force on the first node is obtained as

$$F_1(t) = F_{\text{sta}} + F_z \tag{3.8}$$

where F_{sta} is the axial thrust on the drill string (N) and F_z is the axial impact force generated by hydraulic oscillator (N).

The external force from the second node to the $n-1$ node is the friction between the drill string and the borehole wall, i.e.,

$$F_i(t) = F_{if}(t) = -\mu F_{iN} \operatorname{sgn}(\dot{x}_i(t)) = -\mu m_i g \operatorname{sgn}(\dot{x}_i(t)) \tag{3.9}$$

where μ is the sliding coefficient of friction, F_{if} is the friction at each node (N), F_{iN} is the positive pressure for each node (N), i is the node label, take $2 \sim n$, and $\dot{x}_i(t)$ is the speed of the drill string at the ith node.

The external force received at the nth node includes the friction between the drill string and the borehole wall and the reaction force of the rock on the roller cone bit. The friction force is calculated as

$$F_{nf}(t) = -\mu F_{nN} \operatorname{sgn}(\dot{x}_n(t)) = -\mu m_i g \operatorname{sgn}(\dot{x}_n(t)) \tag{3.10}$$

The reaction force of the rock to the roller cone bit can be calculated by referring to the formula given in document [9]

$$F_{\text{bit}} = W_0 + \mathrm{W}_1 \sin\left(\frac{\pi N_{\text{b}} N_{\text{r}}}{30} t\right) \tag{3.11}$$

where N_{b} is the number of cones, N_{r} is the bit speed, W_0 is the static pressure value for drilling, and W_1 is the axial excitation amplitude.

So, the external force to the nth node can be expressed as

$$F_n = F_{nf} - F_{\text{bit}} \tag{3.12}$$

Calculation and Solution of Vibration Model

The result of vibration displacement of the bottom drill string system is shown in Fig. 3.2, which includes the first, seventh, 11th, and 13th node. Figure 3.3 shows the calculation results of the vibration speeds of the first, seventh, 11th, and 13th node of the lower drill string system. The first node is the hydraulic oscillator, and the thirteenth node is the position of the drill bit. From Fig. 3.2 we can see that the amplitude of the vibration displacement and vibration velocity from the first node to the 13th node gradually becomes smaller, and the first node is the point of excitation; the farther the distance to the excitation, the more attenuated the vibration. Figure 3.4 shows the spectrum of the vibration displacement obtained by the Fourier transform of the vibration displacement. The maximum response frequency in the figure is 12.65 Hz, which is the same as the excitation frequency with an input flow rate of 30 L/s.

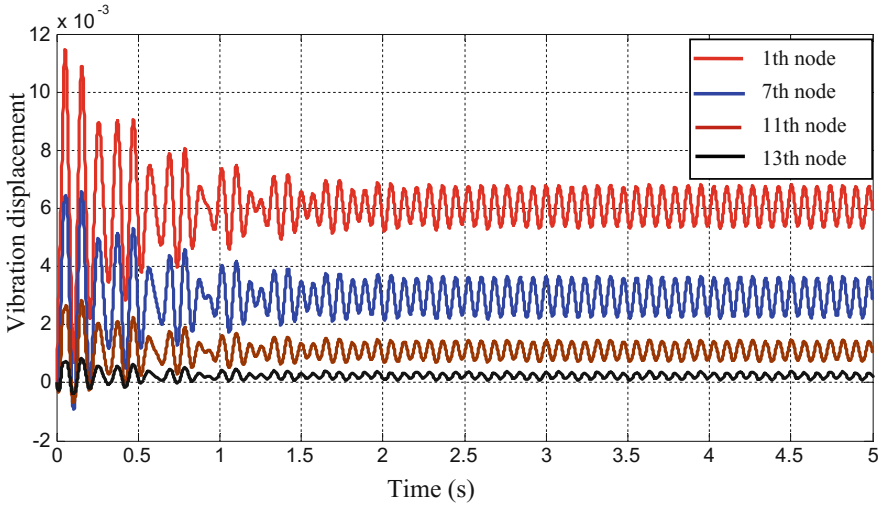


Fig. 3.2 Vibration displacement of different nodes

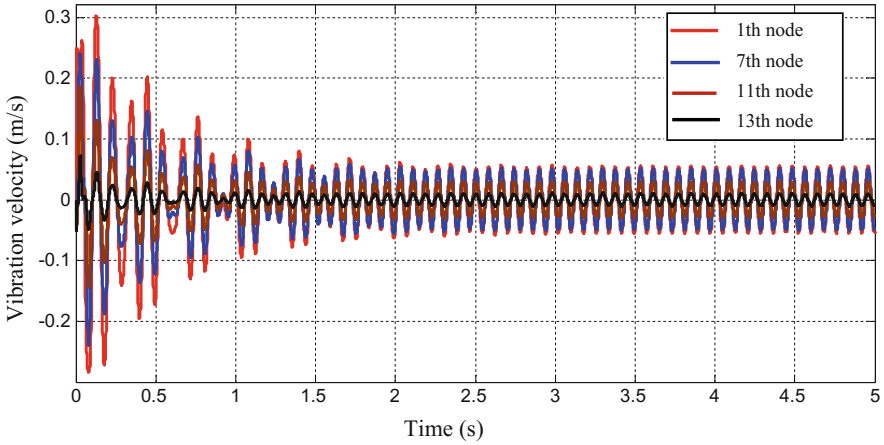


Fig. 3.3 Vibration speed of different nodes

3.2.2 Radial Inertia Effect on Vertical Vibration of Drill String

Vertical Vibration Model

According to drilling conditions, related assumptions are proposed to research the vertical vibration of the drill string, as

1. The cross-sectional surface of the drill string is equivalent circular, and the axis of the drill string coincides with borehole axis.

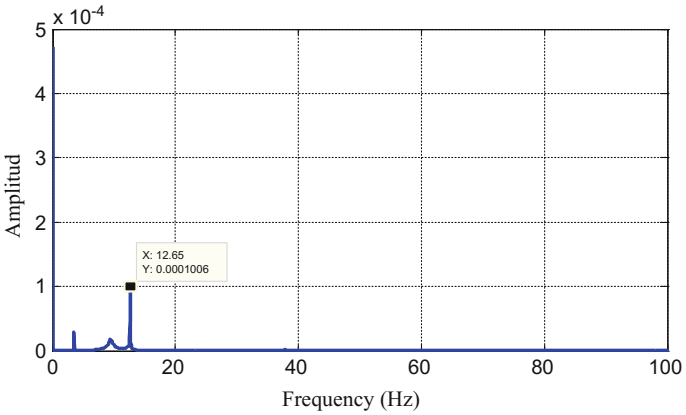


Fig. 3.4 Spectral map of vibration displacement

2. The influence of the gravity of the drill string and the static including the drilling pressure and drilling fluid buoyancy are ignored.
3. Drilling fluid is regarded as viscoelastic layer, which can be divided into several parts, and the properties of drilling fluid inside and outside of the drill string are of identical characteristics whose effect on inside and outside of the drill string are equal.

The vibration model is shown in Fig. 3.5. Where h_k and H denote the location and the total length of the kth drill string segment, respectively; r_{Pk} and r'_{Pk} denote the outer radius and the inner radius of the kth drill string segment, respectively; τ_{Pk} is the shear stress of drilling fluid inside and outside of the drill string at the interface; and k_{PK} and δ_{PK} denote the stiffness coefficient and the damping coefficient of the bottom-hole to the drill string, respectively.

Vibration Equation and Solution

Not Considering the Effect of Radial Inertia

For the vertical vibration of the drill string taking into no account the radial inertia effect, regarding the drill string as one-dimensional sticky elastomer, the vibration equation of the kth drill string segment is described as [10]

$$E_{Pk}A_k\frac{\partial^2 w_k}{\partial z^2} + 2\pi\tau_{Pk}\left(r_{Pk} - r'_{Pk}\right) = \rho_{Pk}A_k\frac{\partial^2 w_k}{\partial t^2} \tag{3.13}$$

In which, $w_k = w_k(z,t)$ denotes the radial displacement of the kth drill string segment; E_{Pk}, ρ_{Pk}, r_{Pk}, and r'_{Pk} denote the elastic modulus, density, and outside and inside radii of the kth drill string segment, respectively; K_k is the vertical shear

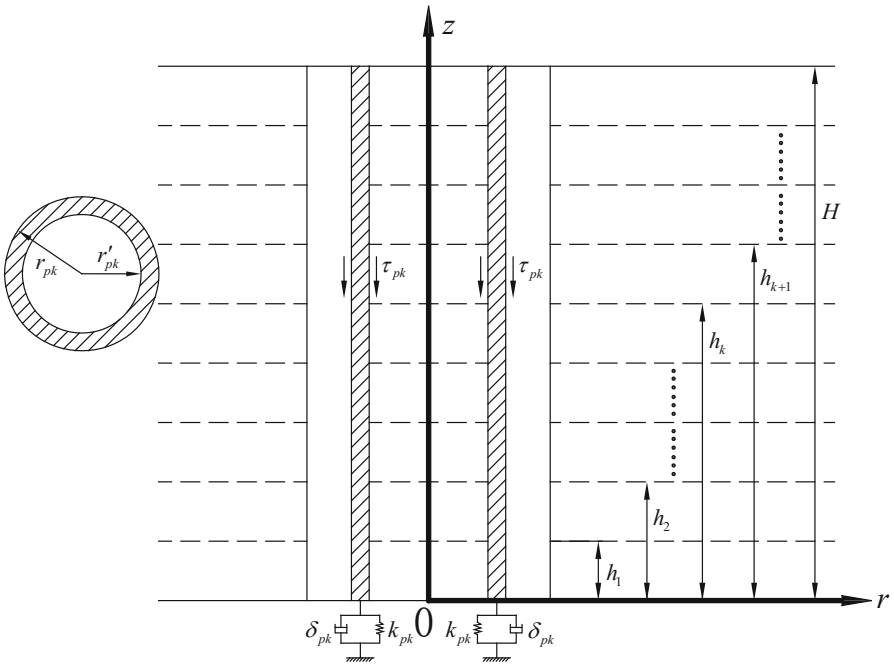


Fig. 3.5 Longitudinal vibration model of drill string

stiffness of the drilling fluid in the unit length direction; and A_k represents the cross-sectional area of the kth drill string segment. And the expressions of τ_{Pk} and A_k are

$$\tau_{Pk} = K_k w_k(z,t) \tag{3.14}$$

$$A_k = \pi\left(r_{Pk}^2 - r_{Pk}'^2\right) \tag{3.15}$$

Laplace transform is applied to Eq. (3.13), taking account of the assumptions, to give

$$E_{Pk}A_k\frac{\partial^2 W_k}{\partial z^2} - \left[\rho_{Pk}A_k s^2 - 2\pi K_k\left(r_{Pk} - r'_{Pk}\right)\right]W_k = 0 \tag{3.16}$$

In which, the Laplace transform with respect to time of $w_k = w_k(z,t)$ represented by$W_k = W_k(z,s)$, and the transform relation is

$$W_k(z,s) = L\left[w_k(z,t)\right] = \int_0^{+\infty} w_k(z,t)\,e^{-st}dt \tag{3.17}$$

Then, Eq. (3.16) general solution has the following form:

$$W_k(z,s) = C_k e^{\alpha z} + D_k e^{-\alpha z} \tag{3.18}$$

where

$$\alpha = \sqrt{\frac{\rho_{Pk} A_k s^2 - 2\pi K_k \left(r_{Pk} - r'_{Pk}\right)}{E_{Pk} A_k}} \tag{3.19}$$

C_k and D_k denote undetermined coefficients determined by boundary conditions, respectively.

In the process of drilling, the bottom is uneven, which leads to drill bit vibration up and down. Assuming that the force of bottom hole to the drill bit is $f_k(z,t)$, according to the transitivity of force, the force on the bottom of the drill string can be also denoted as $f_k(z,t)$. So, the bottom and top of the drill string meet

$$E_{Pk} A_k \frac{\partial W_k(z,s)}{\partial z}\bigg|_{z=0} = f_k(z,s) \tag{3.20}$$

$$E_{Pk} \frac{\partial W_k(z,s)}{\partial z}\bigg|_{z=H} = -\left(k_{PK} + \delta_{PK} s\right) W_k(z,s) \tag{3.21}$$

where the Laplace transform with respect to time of $f_k(z,t)$ is represented by $f_k(z,s)$; k_{Pk} and δ_{Pk} denote the stiffness coefficient and the damping coefficient of the bottom hole to the drill string, respectively.

Furthermore, Eqs. (3.20) and (3.21) are substituted into Eq. (3.18) to get the following two equations:

$$C_k - D_k = \frac{f_k(z,s)}{\alpha E_{Pk} A_k} \tag{3.22}$$

$$C_k = \frac{E_{Pk}\alpha - \left(k_{Pk} + \delta_{Pk} s\right)}{E_{Pk}\alpha + \left(k_{Pk} + \delta_{Pk} s\right)} e^{-2\alpha H} D_k \tag{3.23}$$

It is assumed that

$$\frac{E_{Pk}\alpha - \left(k_{Pk} + \delta_{Pk} s\right)}{E_{Pk}\alpha + \left(k_{Pk} + \delta_{Pk} s\right)} e^{-2\alpha H} = \gamma \tag{3.24}$$

Then, C_k and D_k can be calculated by Eqs. (3.22) and (3.23), which are substituted into Eq. (3.18), and the displacement of the drill string can be denoted as follows:

$$W_k(z,s) = \frac{f_k(z,s)}{(\gamma - 1)\alpha E_{Pk} A_k} \left(\gamma e^{\alpha z} + e^{-\alpha z}\right) \tag{3.25}$$

It is assumed that

$$s = i\omega \tag{3.26}$$

Then, the Laplace transform is equivalent to the Fourier transform of unilateral, so the response of displacement frequency can be expressed as $W_k(z, i\omega)$. The complex impedance of the drill string is derived as follows:

$$K_{Pd} = \frac{\alpha A_k (\gamma - 1) E_{Pk}}{\gamma + 1} \tag{3.27}$$

The complex impedance of the drill string bottom is equivalent to the complex stiffness, which can be expressed in the plural form as follows:

$$K_{Pd} = K_{\mathrm{r}} + iC_i \tag{3.28}$$

In which, the real component K_{r} denotes the real dynamic stiffness, which reflects the ability of the drill string system to fight the vertical deformation; the imaginary component C_i denotes the dynamic damping, which reflects the energy dissipation.

Considering the Effect of Radial Inertia

Combined with actual situation to allow for the impact of the radial inertia effect, the vibration problem of the kth drill string segment can be described by the theory of Rayleigh–Love model as follows:

$$E_{Pk}A_k\frac{\partial^2 w_k}{\partial z^2}+2\pi\tau_{Pk}\left(r_{Pk}-r'_{Pk}\right)=\rho_{Pk}A_k\left(\frac{\partial^2 w_k}{\partial t^2}-\nu^2{}_{Pk}\left(r_{Pk}^2-r_{Pk}'^2\right)\frac{\partial^4 w_k}{\partial z^2\partial t^2}\right) \tag{3.29}$$

where ν_{Pk} denotes the Poisson's ratio of the kth drill string segment.

Without considering the impact of the above-described lateral inertia, according to the initial and continuity conditions, the vibration equation of the drill string is solved by Laplace transform, and the displacement expression of the drill string is obtained considering the radial inertia effect, which can be introduced as:

$$W_k(z,s) = \frac{f_k(z,s)}{(\gamma-1)\alpha(E_{Pk}A_k + A_k\beta)}\left(\gamma e^{\alpha z} + e^{-\alpha z}\right) \tag{3.30}$$

where

$$\alpha = \sqrt{\frac{\rho_{Pk} A_k s^2 - 2\pi K_k \left(r_{Pk} - r'_{Pk}\right)}{E_{Pk} A_k + \rho_{Pk} A_k v^2{}_{Pk} \left(r_{Pk}^2 - r_{Pk}'^2\right) s^2}} \tag{3.31}$$

$$\beta = \rho_{Pk} v^2{}_{Pk} \left(r_{Pk}^2 - r_{Pk}'^2\right) s^2 \tag{3.32}$$

$$\gamma = \frac{\left(E_{Pk} + \beta\right)\alpha - \left(k_{Pk} + \delta_{Pk} s\right)}{\left(E_{Pk} + \beta\right)\alpha + \left(k_{Pk} + \delta_{Pk} s\right)} \mathrm{e}^{-2\alpha H} \tag{3.33}$$

Then, the analytical solution of the complex impedance at the drill string bottom can be expressed as:

$$K_{Pd} = \frac{\alpha A_k \left(\gamma - 1\right)\left(E_{Pk} + \beta\right)}{\gamma + 1} \tag{3.34}$$

Parameter Impact Analysis

To analyze contrastively the influence of the drill string design parameters on the characteristics of its vertical vibration, its characteristics of change are discussed and studied by using the analytical solution of the complex impedance when one of the related parameters changes. In the figure, the horizontal axis is the vibration frequency, and the vertical axis is the dynamic stiffness at the bottom of the drill string. The material parameters of the drill string are shown in Table 3.1.

Impact of the Length of Drill String on Vertical Vibration

The length of the drill string is dependent on the well depth. The larger the diameter of the drill string, the poorer is its stability about the vibration. The impact of the drill string length on the dynamic stiffness is analyzed with the consideration of the radial inertia effect. The calculation parameters influencing the length of drill string are shown in Table 3.2.

In Fig. 3.6, H denotes without considering the radial inertia effect, m; and h denotes considering the radial inertia effect, m. Figure 3.2 shows the influence of the length of the drill string on the dynamic stiffness in the frequency domain. It

Table 3.1 Material parameters of the drilling string

Name of parameter	Result
Density of the drill string ρ_{Pk} (kg/m^3)	8000
Modulus of elasticity of the drill string E_{Pk} (MPa)	2.06×10^3

Table 3.2 Calculated parameters of the impact of the length of drill string

Name of parameter	Result
Poisson's ratio of the drill string, ν	0.3
Length of the drill string, H_1 (m)	8
Length of the drill string, H_2 (m)	10
Outside radius of the drill string, r_{Pk} (m)	0.1
Inside radius of the drill string, r'_{Pk} (m)	0.08
Damping coefficient, δ_{PK} (N · s/m^3)	10,000

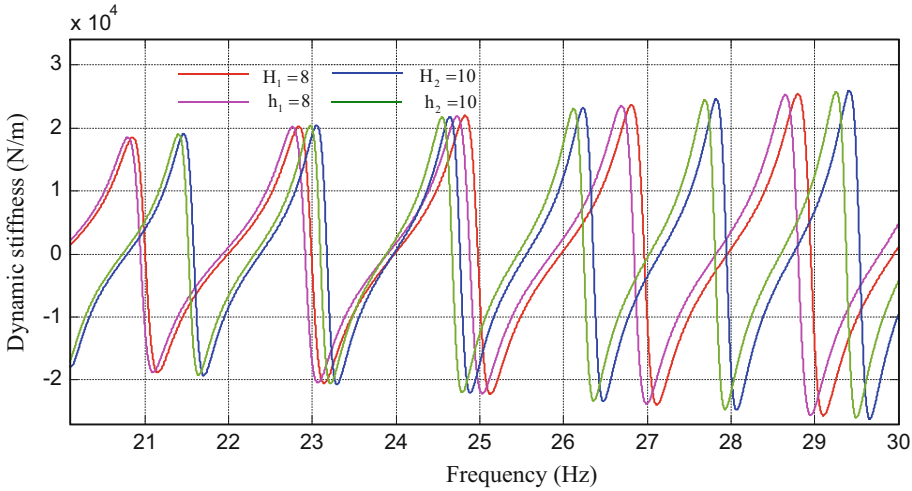


Fig. 3.6 Impact of the length of drill string on its bottom dynamic stiffness

can be observed that the increasing trend of the dynamic stiffness of the drill string becomes more notable when the length of the drill string and the cyclic frequency increase whether allowing for the radial inertia effect or not. Meanwhile, the radial inertia effect on the dynamic stiffness can be almost neglected only when the cyclic frequency is small or zero, and which becomes more and more evident with the increase in frequency. When considering the radial inertia effect, the absolute value of dynamic stiffness in increasing period is bigger than the condition where the radial inertia effect is ignored. The shorter the length of the drill string, the more evident is the radial inertia effect.

Impact of the Inside and Outside Radii of Drill String on Vertical Vibration

The inside and outside radii are determined by the specification of the selected drill string. The influence of the inside and outside radius of the drill string on the dynamic stiffness is analyzed with taking account of the radial inertia effect. The calculated parameters to study the influence of the inside and outside radii of the drill string are shown in Table 3.3.

Table 3.3 Calculated parameters of the impact of inside and outside radii

Name of parameter	Result
Poisson's ratio of the drill string, ν	0.3
Length of the drill string, H (m)	10
Outside radius of the drill string, r_{Pk1} (m)	0.1
Outside radius of the drill string, r_{Pk2} (m)	0.11
Inside radius of the drill string, r'_{Pk1} (m)	0.08
Inside radius of the drill string, r'_{Pk2} (m)	0.09
Damping coefficient, δ_{PK} (N · s/m^3)	10,000

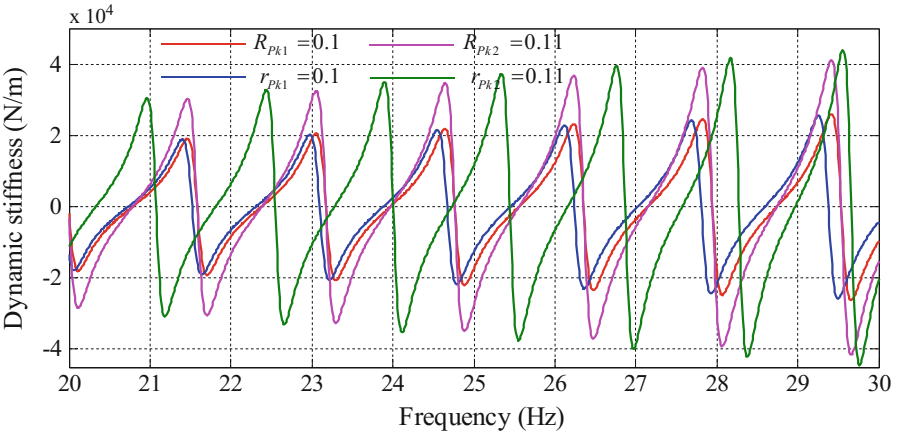


Fig. 3.7 Impact of outside radius on its bottom dynamic stiffness

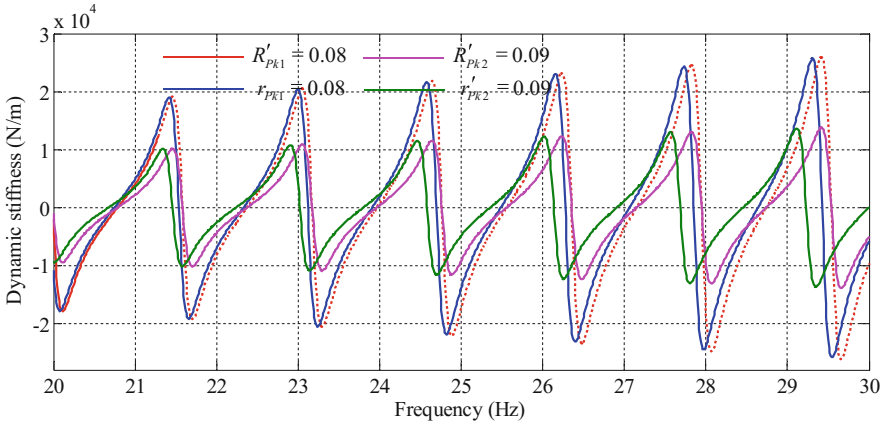


Fig. 3.8 Impact of inside radius on its bottom dynamic stiffness

In Figs. 3.7 and 3.8, R_{Pk} and R'_{Pk} denote not considering the radial inertia effect, and r_{Pk} and r'_{Pk} denote considering the radial inertia effect.

Table 3.4 Calculated parameters of the impact of the damping coefficient

Name of parameter	Result
Poisson's ratio of the drill string, ν	0.3
Length of the drill string, H (m)	10
Outside radius of the drill string, r_{Pk} (m)	0.1
Inside radius of the drill string, r'_{Pk} (m)	0.08
Damping coefficient, δ_{PK1} (N · s/m^3)	8000
Damping coefficient, δ_{PK2} (N · s/m^3)	10,000

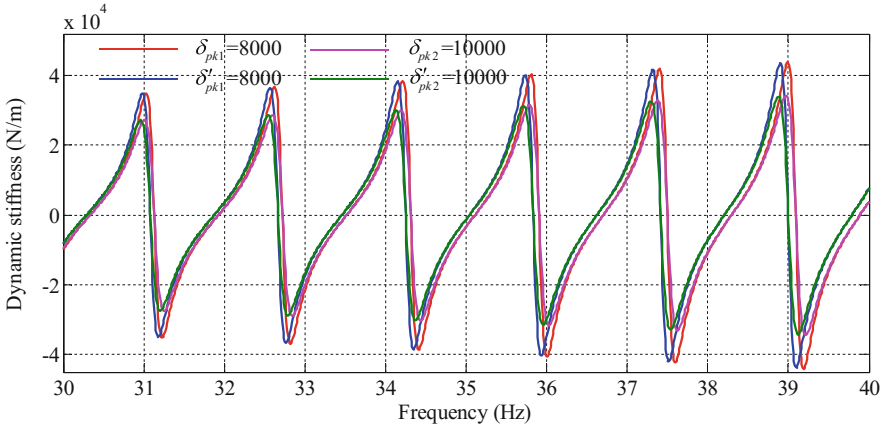


Fig. 3.9 Impact of the damping coefficient on its bottom dynamic stiffness

Figures 3.7 and 3.8 show the influence of the inside and outside radii of the drill string on the dynamic stiffness in the frequency domain. The increase trend of the dynamic stiffness becomes more remarkable when the outside radius of the drill string and the cyclic frequency increase or the inside decrease. When considering the radial inertia effect, the absolute value of the dynamic stiffness in increasing period is bigger than the condition where the radial inertia effect is ignored. Meanwhile, the radial inertia effect has more important influence on the dynamic stiffness of the drill string if the difference between the outside and inside radii of the drill string is greater.

Impact of the Damping Coefficient on Vertical Vibration

The damping coefficient of the bottom hole to the drilling bottom should have certain influence on the vertical vibration of the drill string. The calculation parameters influencing the damping coefficient are shown in Table 3.4.

In Fig. 3.9, δ_{PK} denotes not considering the radial inertia effect and δ'_{Pk} denotes considering the radial inertia effect.

Figure 3.9 shows the influence of the damping coefficient on the dynamic stiffness in the frequency domain. With the increase of the damping coefficient

Table 3.5 Calculated parameters of the impact of the Poisson's ratio of drill string

Name of parameter	Result
Poisson's ratio of the drill string, ν_1	0.1
Poisson's ratio of the drill string, ν_2	0.3
Length of the drill string, H (m)	10
Outside radius of the drill string, r_{Pk} (m)	0.1
Inside radius of the drill string, r'_{Pk} (m)	0.08
Damping coefficient, δ_{PK} (N · s/m^3)	10,000

and the cyclic frequency, the dynamic stiffness increase more obviously. In a single cycle, the closer to the crest and trough the curve is, the bigger the difference of the dynamic stiffness of the drill string between the two conditions. When considering the radial inertia effect, the absolute value of the dynamic stiffness in increasing period is bigger than that in the condition where the radial inertia effect is ignored; the bigger the damping coefficient, the less obvious is the effect, which indicates that the damping of the bottom hole to the drill string could weaken the influence of the radial inertia effect on the dynamic stiffness.

The Impact of the Poisson's Ratio of the Drill String on Vertical Vibration

Poisson's ratio is the lateral deformation coefficient of the material, which reflects the lateral deformation elastic constants of the material. In the static case, the Poisson's ratio change of the drill string is very small, so it is generally taken as $\nu = 0.25$. However, in the dynamic case, the dynamic Poisson's ratio is commonly taken as $\nu = 0.1 \sim 0.30$. The calculated parameters to study the influence of the Poisson's ratio are shown in Table 3.5.

In Fig. 3.10, ν denotes not considering the radial inertia effect and ν' denotes considering the radial inertia effect. Figure 3.10 shows the influence of the Poisson's ratio of drill string on the dynamic stiffness in the frequency domain. It can be seen that the increasing trend of the dynamic stiffness of the drill string becomes more notable when the Poisson's ratio of the drill string and the cyclic frequency increase whether allowing for the radial inertia effect or not. The change of the Poisson's ratio of drill string has influence on the condition where the radial inertia effect is considered rather than the other case, which indicates that radial inertia effect includes the effect of the Poisson's ratio. When considering the radial inertia effect, the absolute value of the dynamic stiffness in increasing period increases with the Poisson's ratio of the drill string.

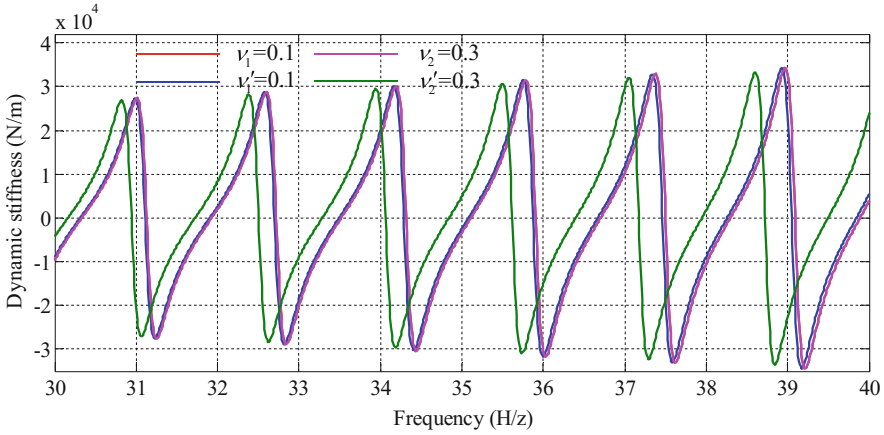


Fig. 3.10 Impact of the Poisson's ratio of drill string on its bottom dynamic stiffness

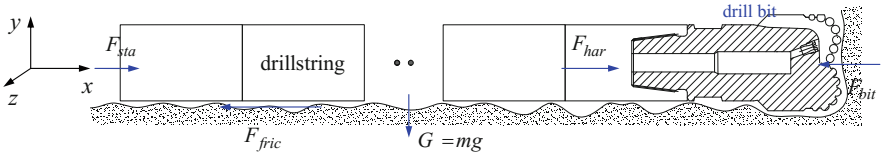


Fig. 3.11 Force analysis of drill string

3.2.3 *Drill String Dynamics with Wellbore Random Friction Forces*

Drill String Dynamics Model

According to the working conditions in horizontal well, the force analysis of drill string is conducted, as shown in Fig. 3.11, where the left boundary is the origin of coordinates of analysis segment. F_{sta} is the equivalent force of the left boundary, F_{fric} is the friction force, G is the gravity of drill string, m is the mass, g is the acceleration of gravity, F_{har} is the force of the drilling fluid, and F_{bit} is the reaction force of the rock on the drill bit, which is equal to weight on bit (WOB).

Based on the force analysis, the vibration analysis model of the horizontal drill string is established to research random wellbore friction. The model is expressed as [11]

$$\begin{cases} \mathbf{M}\ddot{u}(t,\xi)+\mathbf{C}\dot{u}(t,\xi)+\mathbf{K}u(t,\xi)=\mathbf{F}_{\text{sta}}+\mathbf{F}_{\text{har}}(t)+\mathbf{F}_{\text{bit}}(\dot{u}(t,\xi))+\mathbf{F}_{\text{fric}}(\dot{u}(t,\xi),\xi) \\ u(0)=u_0 \\ \dot{u}(0)=\dot{u}_0 \end{cases} \tag{3.35}$$

In which, **M**, **C**, and **K** are the mass matrix, damping matrix, and stiffness matrix, respectively, u is the displacement function, $\dot{u}$ is the corresponding speed, t is the time coordinate, ξ is the friction coefficient, $\mathbf{F}_{\text{sta}}(x,t)$, $\mathbf{F}_{\text{har}}(x,t)$, $\mathbf{F}_{\text{bit}}(\dot{u}(x,t))$, $\mathbf{F}_{\text{fric}}(\dot{u}(x,t))$, and $\mathbf{F}_{\text{mass}}(\ddot{u}(x,t))$ are the corresponding matrix of the force, respectively.

In order to analyze drilling power consumption, the efficiency equation is defined as

$$\eta = \frac{\frac{1}{t_1-t_0}\int_{t_0}^{t_1} f_{\text{sta}}\dot{u}(0,t) + f_{\text{har}}\dot{u}(L,t)\,\mathrm{d}t}{\frac{1}{t_1-t_0}\int_{t_0}^{t_1} f_{\text{bit}}(t)\dot{u}(L,t)\,\mathrm{d}t} \tag{3.36}$$

With the combination of the importance and actual significance of the drilling efficiency, the mean square value analysis of the drilling efficiency is conducted to ensure the analysis result of the efficiency. And the solution method is given as

$$\text{con}\left(n_\eta\right) = \frac{1}{n_\eta}\sum_{i=1}^{n_\eta}(\eta_i)^2 \tag{3.37}$$

where n_η is different solution counts.

Friction Randomness Research and Model Solution

In order to solve the vibration equation, the representation of each force needs to be determined. According to the research innovation, the wellbore friction randomness is studied first.

The randomness is an important feature of the drill string friction, which is reflected in the random change of the contact position and the value of the friction coefficient. The accuracy of mathematical method has a direct decisive effect on the precision of the calculation results. Considering the characteristics of the wellbore friction on drill string, the random variable ξ, corresponding to the displacement u, is introduced to research the randomness of the generation location of the friction pair. For the friction coefficient, through random statistical analysis, the distribution law of the random data can be described by Gaussian distribution. The analysis method is established, and the analysis model can be defined as follows.

$$F_{\text{fric}}(\dot{u}(x,t),\xi) = \rho_{\text{L}} g(-f(x,\xi))\,\text{sgn}(\dot{u}(x,t)) \tag{3.38}$$

where x is the coordinate value of the analysis position, ρ_{L} is the mass of per unit length, $f(x,\xi)$ is the friction coefficient based on x and ξ, the mean value is $\overline{f}(x,\xi)$, and $\text{sgn}(\dot{u}(x,t))$ is the judgment function, which can be expressed as

$$\text{sgn}(\dot{u}(x,t)) = \begin{cases} 1 & (\dot{u}(x,t) > 0) \\ 0 & (\dot{u}(x,t) = 0) \\ -1 & (\dot{u}(x,t) < 0) \end{cases} \tag{3.39}$$

The probability density of $f(x, \xi)$ is calculated by

$$p\left(f\left(x, \xi\right)\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{f\left(x, \xi\right) - \overline{f}\left(x, \xi\right)}{\sigma}\right)^2\right) \tag{3.40}$$

In which, σ is the standard deviation of the random variable.

In order to describe the randomness of the friction behavior, the random variable needs to be converted. Since the Gaussian distribution characteristic of the stochastic friction, Karhunen–Loève (KL) method is used to convert the random variable. According to KL expansion, $f(x, \xi)$ is expanded at the mean value $\overline{f}\left(x, \xi\right)$, and the expansion can be written as

$$f\left(x, \xi\right) = \overline{f}\left(x, \xi\right) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(x) Z_i\left(\xi\right) \tag{3.41}$$

where $\{\lambda_i\}$ and $\{\phi_i(x)\}$ are eigenvalue and eigenfunction sequences, respectively, and $Z_i(\xi)$ is uncorrelated random variable.

The expansion of Eq. (3.41) should be convergent in the Gaussian stationary field. Relative to other types of expansion, there is a minimum mean square error when the expansion term is same and limited. According to the downhole conditions during the drilling process, x is defined as the spatial coordinate and $x = (x_1, y_1)$, which can be used to research the two-dimensional random field. Similarly, based on the construction method of the two-dimensional random field, the result of the one-dimensional random field can be obtained.

Corresponding to the matrix form of $Z_i(\xi)$, each row consists of an implementation of random variable vector and each line is the value of different application for the same random variable and is mutually independent and satisfies the normal distribution. For Gaussian random field, the independent random variable $Z_i(\xi)$ remains the Gaussian model, and the mathematical expectation satisfies the condition, which can be expressed as

$$\begin{cases} E\left[Z_i\left(\xi\right)\right] = 0 \\ E\left[Z_i\left(\xi\right) Z_j\left(\xi\right)\right] = \delta_{ij} \end{cases} \tag{3.42}$$

In the two-dimensional area, the characteristic function $\{\phi_i(x)\}$ is satisfied with the following conditions.

$$\int_{\Omega} \phi_i\left(x_1, y_1\right) \phi_j\left(x_2, y_2\right) d\Omega = \delta_{ij} \tag{3.43}$$

In which, Ω is a two-dimensional and closed region, x and y are two-dimensional region coordinates, respectively, and δ_{ij} is the K-delta function.

The analytic expression of the eigenvalue $\{\lambda_i\}$ and the eigenfunction $\{\phi_i(x)\}$ in the expansion equation (3.41) can be obtained as [12]

$$\lambda_i = \frac{4\gamma_1\gamma_2\sigma_\phi^2}{\left(\gamma_1^2\omega_{1,m}^2 + 1\right)\left(\gamma_2^2\omega_{2,n}^2 + 1\right)} \tag{3.44}$$

$$\phi_i^* = \phi_{1,m}(x_1)\,\phi_{2,n}(x_2) \tag{3.45}$$

wherein γ_1 and γ_2 are the correlation lengths in x and y, respectively, and when there is isotropic, for that is $\gamma_1 = \gamma_2$, σ_ϕ^2 is the variance of the randomness field.

In fact, during the calculation process, the eigenvalue $\{\lambda_i\}$ and the eigenfunction $\{\phi_i(x)\}$ can be achieved by the solution of the second Fredholm equation as follows [13].

$$\int_\Omega C(x_1, y_1; x_2, y_2)\,\phi_i(x_2, y_2)\,d\Omega = \lambda_i\phi_i(x_1, y_1) \tag{3.46}$$

where $C(x_1, y_1; x_2, y_2)$ is the covariance function of the two-dimensional randomness field and its expression can be written as

$$C(x_1, y_1; x_2, y_2) = \sigma^2 \exp\left(-\frac{|x_1 - x_2|}{\gamma_1} - \frac{|y_1 - y_2|}{\gamma_2}\right) \tag{3.47}$$

In which, $|x_2 - x_1|$ and $|y_1 - y_2|$ are the distances of the sample points in the two-dimensional randomness field.

When Eq. (3.41) is work, the covariance function can be expressed as the following spectral expansion.

$$C(x_1, y_1; x_2, y_2) = \sum_{i=0}^{\infty} \lambda_i\phi_i(x_1, y_1)\,\phi_i(x_2, y_2) \tag{3.48}$$

Meanwhile, the theoretical covariance function can be represented approximately by M-order truncation analog value in the actual calculation. And the results can be obtained as follows.

$$C_M(x_1, y_1; x_2, y_2) = \sum_{i=0}^{M} \lambda_i\phi_i(x_1, y_1)\,\phi_i(x_2, y_2) \tag{3.49}$$

On the basis of the above process, the matrix relationship, corresponding to the covariance function of the multidimensional random variables, can be expressed as

$$[C(x_1, x_2, \ldots, x_m)] = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,m} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,m} \end{bmatrix} \tag{3.50}$$

In order to conduct the error analysis, different expansion orders can be used to analyze the changing trend of the eigenvalues after the solution of the above method. And the weight of orders is analyzed throughout the KL expansion. The calculation results of the covariance function can also be used to analyze the error. When Eq. (3.49) holds, the relationship of covariance, correlation length, and expansion order can be obtained by comparing the calculation results of the M-order covariance function with the theoretical covariance function as Eq. (3.47).

With this, the method is achieved that is used for constructing the randomness field of the wellbore friction. The steps, using the KL expansion, are as follows that can generate the Gaussian friction randomness field between the drill string and the wellbore. The independent random variables $f_i(x, \xi)$ are generated whose number is M. Then, the eigenvalue $\{\lambda_i\}$ and the eigenfunction $\{\phi_i(x)\}$ of the integral equation are solved and the results are taken into Eq. (3.41). Finally, the M-order truncation is taken to generate a randomness field, and the P-time truncation is taken to generate the P-time randomness field; then, $\left[Z_M^P(\xi)\right]$ can be written as

$$\left[Z_M^P(\xi)\right] = \begin{bmatrix} Z_1^1(\xi) & Z_2^1(\xi) & \cdots & Z_M^1(\xi) \\ Z_1^2(\xi) & Z_2^2(\xi) & \cdots & Z_M^2(\xi) \\ \vdots & \vdots & \ddots & \vdots \\ Z_1^P(\xi) & Z_2^P(\xi) & \cdots & Z_M^P(\xi) \end{bmatrix} \tag{3.51}$$

For $\left[Z_M^P(\xi)\right]$, each row consists of an implementation of random variable vector, and each line is the value of different implementation for the same random variable such that each line is mutually independent and satisfies the normal distribution, where each element is independent Gaussian random number. Thus, the wellbore friction randomness field of M-order and P-time is established.

The description and processing method of the friction force are given. The expression and the solution method of other forces are determined in this section.

$$F_{\text{sta}}(x, t) = F_{\text{sta}}\delta(x) \tag{3.52}$$

where F_{sta} is the amplitude of $F_{\text{sta}}(x, t)$ and $\delta(x)$ is the Dirac function δ for the displacement of the analysis point.

For $F_{\text{har}}(x, t)$, the corresponding solution equation can be expressed as

$$F_{\text{har}}(x, t) = F_0 \sin\left(\omega_f t\right) \delta(x - L) \tag{3.53}$$

In which, F_0 is the resonant force amplitude of $F_{\text{har}}(x, t)$, ω_f is the rotation angular velocity, $x = L$ is the corresponding vibration equilibrium position of $F_{\text{har}}(x, t)$, and the Dirac function $\delta(x - L)$ is defined based on this.

About $F_{\text{bit}}(\dot{u}(x, t))$, the solution steps and results are more complex and the influence factors are numerous. According to the actual condition, the solution

methods of the Polycrystalline Diamond Compact (PDC) drill bit and the roller cone bit are given, respectively. For the PDC drill bit, the expression is defined as [14]

$$\begin{cases} F_{\text{bit}}\left(\dot{u}\left(x,t\right)\right) = \left[C_1 \exp\left(-C_2 \dot{u}\left(x,t\right)\right) - C_1\right] \delta\left(x-L\right) & \dot{u}\left(L,t\right) > 0 \\ F_{\text{bit}}\left(\dot{u}\left(x,t\right)\right) = 0 & \dot{u}\left(L,t\right) \le 0 \end{cases} \tag{3.54}$$

where C_1 is the coefficient that is aiming at the rate of penetration (ROP) and C_2 is the coefficient to describe the nonlinear characteristic between the rock and the drill bit.

For roller cone bit, according to the boundary condition, the calculation equation can be written as [15]

$$\begin{cases} F_{\text{bit}}\left(\dot{u}\left(x,t\right)\right) = -\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J(i)}\sum\limits_{k=1}^{K(i,j)} F_{\text{bit}-X}\left(i,j,k\right) + W_0 & \dot{u}\left(L,t\right) > 0 \\ F_{\text{bit}}\left(\dot{u}\left(x,t\right)\right) = 0 & \dot{u}\left(L,t\right) \le 0 \end{cases} \tag{3.55}$$

where I is the number of cones, $J(i)$ is the ring gear of the ith cone, $K(i,j)$ is the teeth number of the ith cone and the jth ring gear, $F_{\text{bit}-X}(i,j,k)$ is the contact force between single tooth and rock, $\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J(i)}\sum\limits_{k=1}^{K(i,j)} F_{bit-X}\left(i,j,k\right)$ is a dynamic WOB, and W_0 is static WOB.

For $F_{\text{mass}}\left(\ddot{u}\left(x,t\right)\right)$, the expression is

$$F_{\text{mass}}\left(\ddot{u}\left(x,t\right)\right) = -m_{\text{bit}}\ddot{u}\left(x,t\right)\delta\left(x-L\right) \tag{3.56}$$

where m_{bit} is the mass of drill bit.

Based on the completion of forces and the determination of the solution method, the dynamics vibration model of the drill string can be solved. According to Fig. 3.1, the discrete solution of the model is researched. With the downhole working conditions of the drill string, the discrete analysis model is established and the drill string is dispersed. The number of discrete units is n and the length of each unit is l, corresponding to $m_1, m_2, \ldots, m_{n-1}, m_n$ as shown in Fig. 3.12. The discrete results can be obtained, which correspond to the mass matrix, the damping matrix, and the stiffness matrix, respectively, in Eq. (3.35).

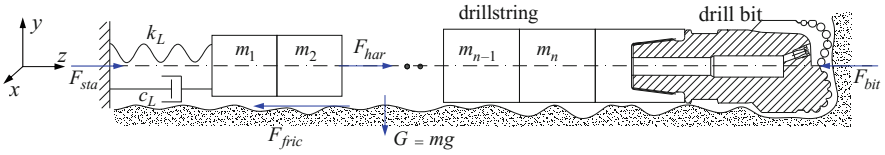


Fig. 3.12 Discrete method and model of dynamics solution

The expression of unit mass matrix can be expressed as

$$[M]^e = \frac{\mathrm{lm}^*}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{3.57}$$

The expression of the overall mass matrix can be written as

$$[M] = \frac{l}{6}\begin{bmatrix} 2m_n^* & m_n^* & & & & & & \\ m_n^* & 4m_n^* & m_n^* & & & & & \\ & & m_n^* & 2\left(m_n^* + m_{n-1}^*\right) & \cdots & & & \\ & & \cdots & & \cdots & & & \\ & & \cdots & & \cdots & & & \\ & & \cdots & & 2\left(m_2^* + m_1^*\right) & m_1^* & & \\ & & & & m_1^* & 4m_1^* & m_1^* & \\ & & & & m_1^* & 4m_1^* & m_1^* & \\ & & & & & & m_1^* & 2m_1^* \end{bmatrix} \tag{3.58}$$

The expression of the unit stiffness matrix is defined as follows.

$$[K]^e = \frac{EA}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.59}$$

The expression of the overall stiffness matrix is achieved by

$$[K] = \frac{1}{l}\begin{bmatrix} k_{\mathrm{L}} + E_n A_n & -E_n A_n & & & & & & \\ -E_n A_n & 2E_n A_n & -E_n A_n & & & & & \\ & & & -E_n A_n & & & & \\ & & & \cdots & & & & \\ & & & & (E_2 A_2 + E_1 A_1) & -E_1 A_1 & & \\ & & & & -E_1 A_1 & 2E_1 A_1 & -E_1 A_1 & \\ & & & & -E_1 A_1 & 2E_1 A_1 & -E_1 A_1 & \\ & & & & & & -E_1 A_1 & 2E_1 A_1 \end{bmatrix} \tag{3.60}$$

The expression of the damping matrix is obtained by

$$[C] = \alpha\,[M] + \beta\,[K] \tag{3.61}$$

wherein α and β are the scale coefficients, which can be determined by simulation. Different values are given to α and β. When the fluctuation of the WOB is within acceptable range, values of coefficients are determined.

Analysis of Calculation and Experimental Test Results

Because of the well structure design, some parameters are not tested easily in experiment test. Therefore, referring to the above input parameters, the wellbore friction random field is analyzed according to the established theoretical model, and then the displacement of the test point is calculated. Combined with the analysis method of the wellbore random friction forces, the friction coefficient is obtained as shown in Fig. 3.13. On the basis of the theoretical model, the dynamic characteristics of the drill string are solved, and the result of the vibration displacement at the test point 1 is shown in Fig. 3.14. Figure 3.14a shows the vibration displacement of example, and Fig. 3.14b shows the enlarge figure of the results of experiment test and example calculation. Figures 3.13 and 3.14 indicate that the wellbore friction coefficient and the displacement at the test point present obvious randomness in the

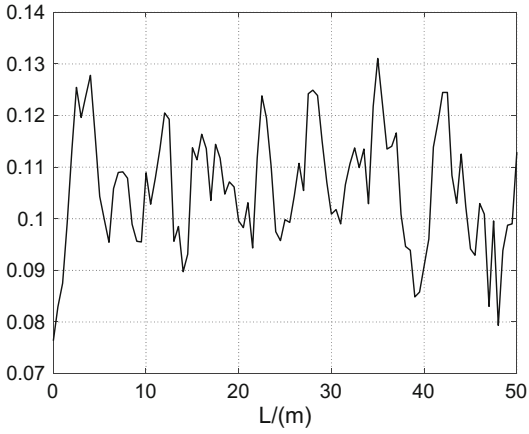


Fig. 3.13 Simulation of the bore friction coefficient

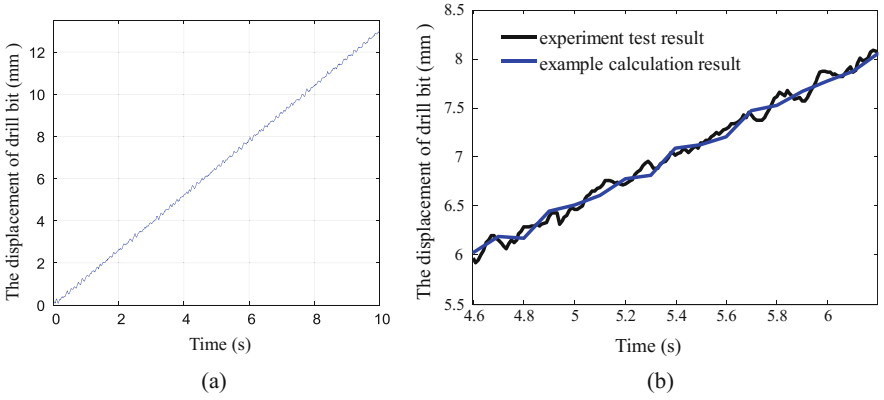


Fig. 3.14 Vibration displacement test values of the example and experimental test. (**a**) The vibration displacement of example. (**b**) The enlarge figure of comparison result

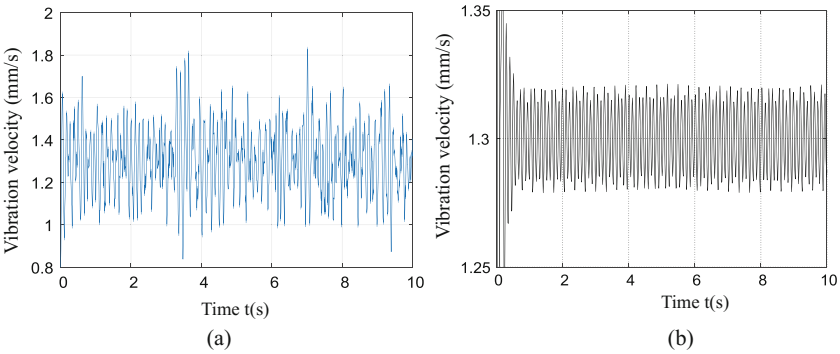
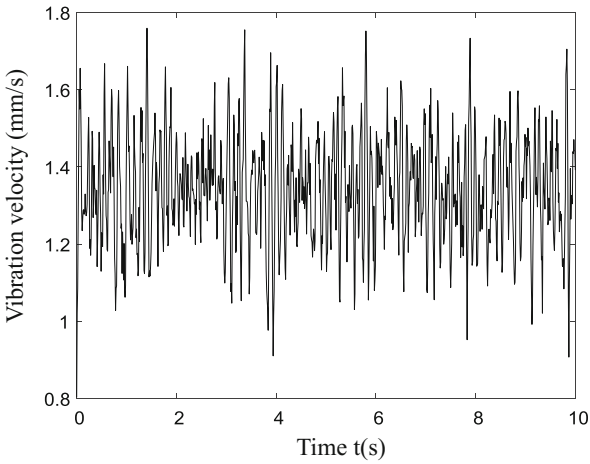


Fig. 3.15 Vibration velocity of test point 1. (**a**) The vibration velocity of random friction. (**b**) The vibration velocity of constant friction

Fig. 3.16 Vibration velocity of experiment test



drilling process. Besides, the vibration displacement in numerical example model is smaller in comparison to the drilling footage.

As the proportion relationship between the displacement of the test point and the footage, the comparison effect of the vibration displacement is not obvious. In order to compare the results of different methods, the vibration velocity at the test point is analyzed, including the wellbore random friction, the constant friction and the actual experiment test of the vibration velocity, as shown in Figs. 3.15 and 3.16.

It can be observed in the figure that the mean value of the vibration velocity is around 1.3 mm/s, but the vibration velocity, considering the friction randomness, shows the characteristic closer to the test result, which can reflect the actual motion features at the test point and verify the accuracy of the theoretical analysis. Meanwhile, it is necessary to pay more attention to the influence of the wellbore randomness on the dynamic characteristics of the drill string, especially in the

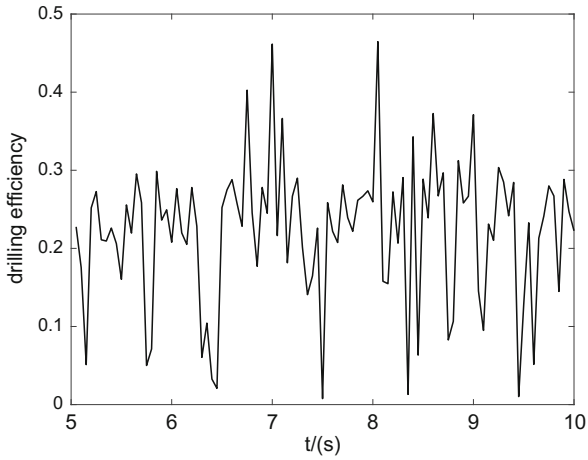


Fig. 3.17 Drilling efficiency

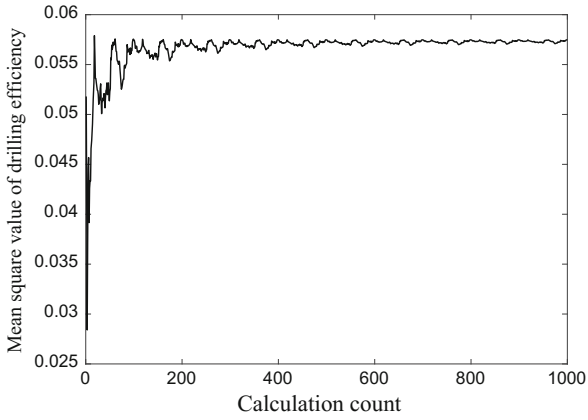


Fig. 3.18 Mean square value of drilling efficiency

horizontal drilling of deep or ultra-deep wells, or the exploitation of new oil and gas resources with larger target diameter.

The drilling efficiency is shown in Fig. 3.17. And the result show that due to vibration, the corresponding forces and velocities change during the drilling process. And because of the randomness of the friction force, the drilling efficiency generates a certain random fluctuation and the average value is about 0.25.

According to the efficiency value, different solution counts are installed to obtain the mean square value of drilling efficiency as shown in Figs. 3.18 and 3.19. With comparison to the results, although the efficiency fluctuation range is large, the average efficiency value converges to a certain value rapidly and steadily with the increase in the number of solution counts. Taking the numerical example, when the number of iterations is more than 200, the mean efficiency is rapidly concentrated near 0.057.

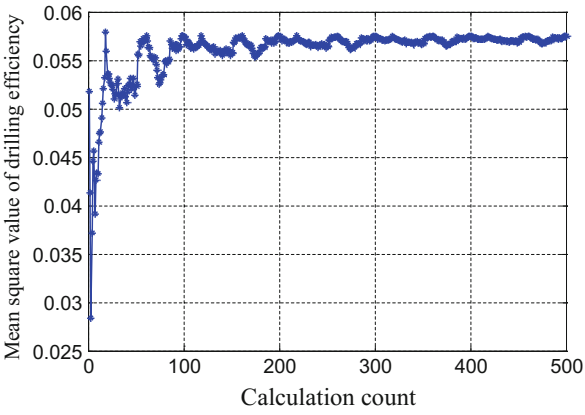


Fig. 3.19 Mean square value of drilling efficiency in experiment test

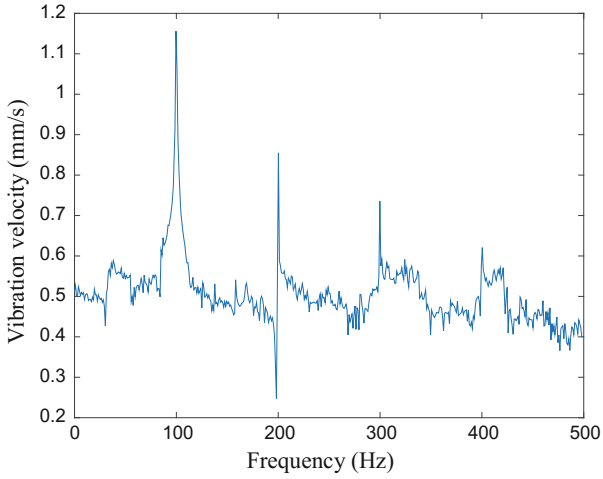


Fig. 3.20 Frequency spectrum analysis result of the vibration velocity of drill string

The accuracy and reliability of the method are verified by the comparison between the test results and the calculated results. According to the experiment test and the example calculation, the frequency spectrum of the drill string vibration velocity is obtained and the results are shown in Figs. 3.20 and 3.21. It can be seen from the figures that the frequency spectrum results show that the wellbore friction force has some influence on the frequency spectrum of the drill string vibration velocity; but in general, the dynamics input speed of the drilling platform has the greatest effect, which is at 100 rpm and its corresponding integer multiple.

Moreover, the phase diagram and the Poincare plot of the test point 1 can be obtained as shown in Figs. 3.22 and 3.23. The phase diagram shows the chaotic motion features of the system; moreover, which indicates that the attractors revolve around closed circles. Figure 3.22 illustrates that the system is not in a periodic

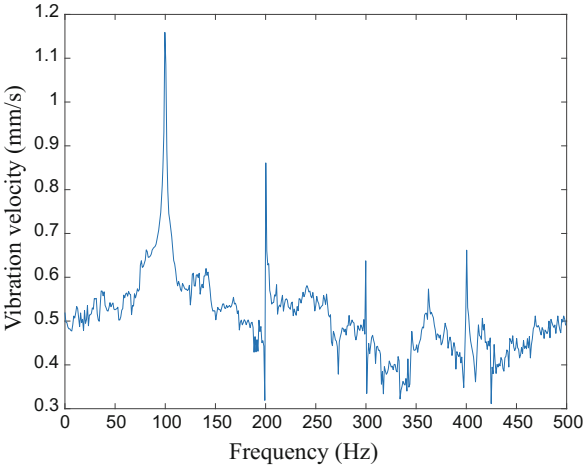


Fig. 3.21 Frequency spectrum analysis result of vibration velocity in experiment test

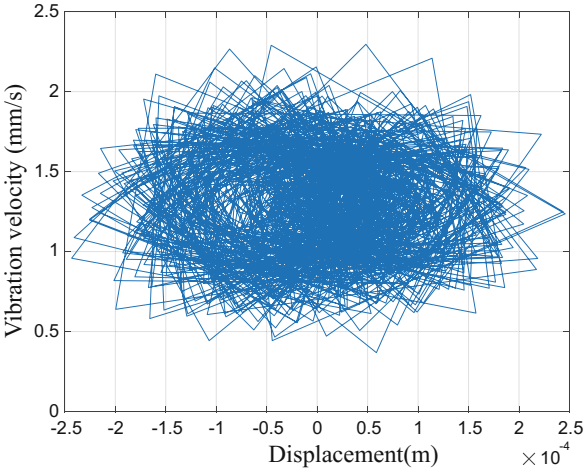


Fig. 3.22 Phase diagram of test point 1

or quasiperiodic vibration. Figure 3.23 shows the Poincare map is convergent to some irregular points, which indicates that there are some strange attractors. And the Poincare cross section is not a closed curve or fixed point but a messy assemblage, which thoroughly explains the chaotic vibration response of the system. Figures 3.22 and 3.23 show that due to the influence of the drilling parameters and the changing working conditions in downhole, the drill string produces random vibration response during the drilling process.

The key parameters, affecting the vibration characteristics of the drill string, also include the length of drill string, the radius of the wellbore friction circle, the ROP, and so on. With a combination of the actual situation of oil and gas production, the above solution methods and steps can be used to get the relevant results and complete the result analysis, where the relevant parameters are transformed.

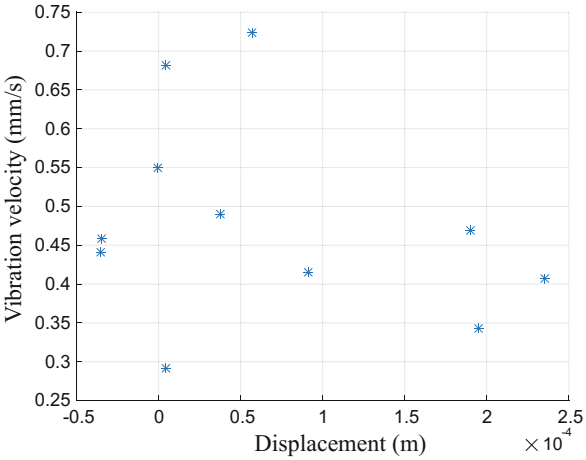


Fig. 3.23 Poincare plot of test point 1

3.2.4 Torsional Vibration on the Wear of Cutting Teeth

The Force Conditions Analysis of Drill String

Due to the complex working conditions of the PDC drill bit during drilling process, the following assumptions are made for the convenience of analysis and calculation: (1) the axis of the drill bit and the borehole always coincide; (2) the rotational speed of turntable and WOB always remain unchanged; and (3) the torque received on the turntable and the drill bit is constant.

According to the assumptions, the force on the drill string system under the action of torsional loads can be obtained, as shown in Fig. 3.24.

The drill string dynamics system is discretized to obtain a discrete system with limited degrees of freedom. For convenience of calculation, the matrix form is expressed as

$$[J]\ddot{\theta}(t) + [C]\dot{\theta}(t) + [K]\theta(t) = [T(t)] \tag{3.62}$$

where $[J]$, $[C]$, $[K]$, and $[T(t)]$ are the rotational inertia matrix, damping matrix, torsional rigidity matrix, and torque matrix of the drill string system, respectively. In addition, $\ddot{\theta}(t)$, $\dot{\theta}(t)$, and $\theta(t)$ are the angular acceleration vector, angular velocity vector, and angular displacement vector of the node, respectively.

According to the number of node degrees of freedom, the angular displacement vector of n nodes is

$$\theta(t) = \begin{bmatrix} \theta_1(t) \\ \theta_2(t) \\ \vdots \\ \theta_n(t) \end{bmatrix} \tag{3.63}$$

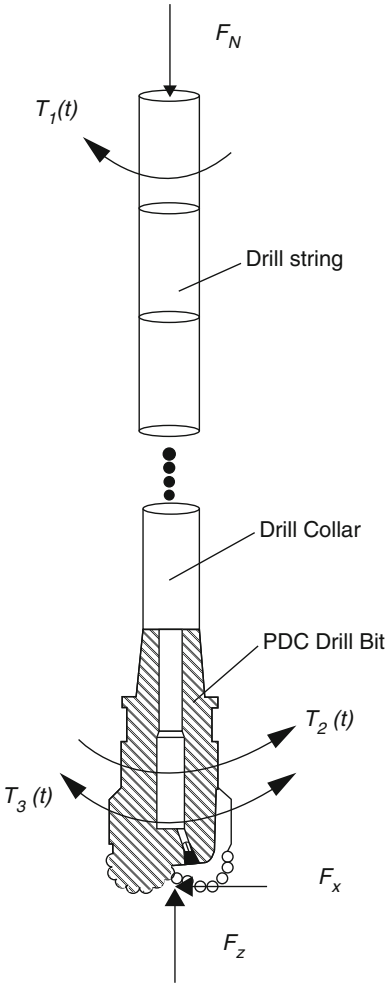


Fig. 3.24 Force condition of drill string

The angular velocity vector of n nodes is

$$\dot{\theta}(t) = \begin{bmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \vdots \\ \dot{\theta}_n(t) \end{bmatrix} \tag{3.64}$$

The angular acceleration vector of n nodes is

$$\ddot{\theta}(t) = \begin{bmatrix} \ddot{\theta}_1(t) \\ \ddot{\theta}_2(t) \\ \vdots \\ \ddot{\theta}_n(t) \end{bmatrix} \tag{3.65}$$

The torque matrix composed of n nodes is

$$[T(t)] = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n-1} \\ T_n \end{bmatrix} \tag{3.66}$$

The rotational inertia matrix composed of n nodes is

$$[J] = \begin{bmatrix} J_1 & 0 & 0 & \cdots & 0 \\ 0 & J_2 & 0 & \cdots & 0 \\ 0 & 0 & J_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & J_n \end{bmatrix} \tag{3.67}$$

The overall torsional rigidity matrix consisting of n nodes is

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & \cdots & 0 \\ -k_2 & k_2 + k_3 & -k_3 & \cdots & 0 \\ 0 & -k_3 & k_3 + k_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -k_n & k_n \end{bmatrix} \tag{3.68}$$

The overall damping matrix consisting of n nodes is

$$[C] = \begin{bmatrix} c_1 + c_2 & -c_2 & 0 & \cdots & 0 \\ -c_2 & c_2 + c_3 & -c_3 & \cdots & 0 \\ 0 & -c_3 & c_3 + c_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -c_n & c_n \end{bmatrix} \tag{3.69}$$

In the process of drilling, assuming that the torque on the turntable is T_1, the torque provided by the torsional load is T_2, which can be expressed as

$$T_{n-1}(t) = T_2 \tag{3.70}$$

The resistance torque produced by the rock to the drill string system is T_3, which is given by

$$T_n = T_3 \tag{3.71}$$

According to formula (3.62), the rotational angular displacement and the angular velocity of the PDC drill bit under the torsional load can be solved. According to the relationship between the linear velocity and the angular velocity, the circumferential linear velocity of the PDC drill bit under the action of the torsional load can be written as

$$V_{\mathrm{c}} = r\dot{\theta} \tag{3.72}$$

Since the drilling process of the PDC drill bit is a compound motion of circumferential rotation and axial movement, the axial linear velocity can be obtained from the relationship between the ROP and the axial linear velocity.

$$V_{\mathrm{a}} = \frac{1000R_J}{3600} \tag{3.73}$$

where R_J is the ROP of the PDC bit.

According to the vectorial resultant theorem of velocity, the absolute linear velocity of the PDC drill bit is obtained.

$$V = \sqrt{V_{\mathrm{c}}^2 + V_{\mathrm{a}}^2} \tag{3.74}$$

According to the relationship between linear velocity and rotation speed, the rotation speed of the PDC drill bit can be solved.

$$N = \frac{V}{2\pi r} \tag{3.75}$$

Geometry Equation of Cutter

During the drilling process, the essential problem of rock breaking is to cut rocks by the cutters of PDC drill bit. The geometry of PDC drill bit is the basis of PDC drill bit dynamics and cutting mechanics. Therefore, the geometry of PDC drill bit cutter has important significance for studying the PDC drill bit wear. Wherein, the cutter geometry includes the interrelationship between the cutting edge inclination angle, top rake, cutting depth, and cutting area of the PDC drill bit.

According to the definition of the cutting angle of the PDC drill bit, it is assumed that the coordinate of the point on the cutter is $w(r, \psi, \mathrm{h_c})$, where r is the radius of the cutter, ψ is the position angle of the cutter, and $\mathrm{h_c}$ is the point of the cutter axial distance. Based on the relationship between the angle of the cutters as shown in

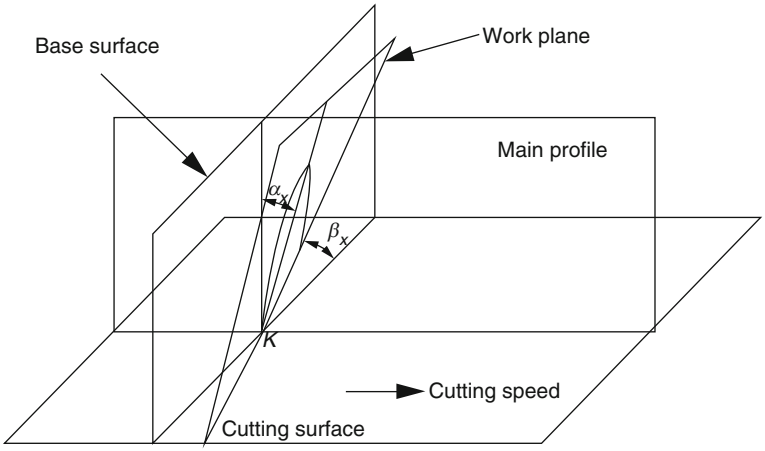


Fig. 3.25 Geometric model of PDC drill bit cutter

Fig. 3.25, the cosine of the normal direction of the cutter working plane can be expressed as [60]

$$\begin{cases} u = \cos\alpha\cos\beta\cos\psi + \sin\alpha\sin\psi \\ v = \cos\alpha\cos\beta \\ p = \sin\alpha\cos\psi - \cos\alpha\sin\beta\sin\psi \end{cases} \tag{3.76}$$

where ξ, α, and β are the normal angle, the top rake, and the roll angle of the cutter, respectively.

According to the geometrical equations of the cutters, the calculation model for each angle of the PDC cutters is established, and the expression of the cutting edge inclination angle is given by

$$\beta_k = \arctan \frac{u\left[\zeta_1\left(r + x_1\right) + \zeta_2 y_1\right] + w\left[\zeta_3\sqrt{\left(r + x_1\right)^2 + y_1^2}\right]}{v\sqrt{2\left[\zeta_1\left(r + x_1\right) + \zeta_2 y_1\right]^2}} \tag{3.77}$$

where x_1, y_1, ζ_1, ζ_2, and ζ_3 can be written as

$$\begin{cases} x_1 = r\sin\psi\cos\beta\cos\xi + r\cos\psi\left(\cos\alpha\sin\xi - \sin\alpha\sin\beta\cos\xi\right) \\ y_1 = -r\sin\psi\sin\beta - r\cos\psi\sin\alpha\cos\beta \\ \zeta_1 = r\left(\cos\psi\cos\beta\cos\xi + \sin\psi\sin\alpha\sin\beta\cos\psi\right) \\ \zeta_2 = r\left(\sin\psi\sin\alpha\cos\beta - \cos\psi\sin\beta\right) \\ \zeta_3 = -r\left(\cos\psi\cos\beta\sin\xi + \sin\psi\cos\alpha\cos\xi + \sin\psi\sin\alpha\sin\beta\sin\xi\right) \end{cases} \tag{3.78}$$

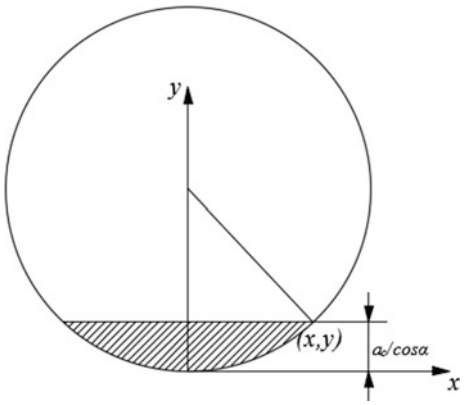


Fig. 3.26 Cutter of PDC drill bit

We assume that the coordinate of the point on the cutter is (x, y), as shown in Fig. 3.26. According to the geometric relation, the effective linear cutting edge of cutter can be expressed as

$$l_{\text{the effective cutting edge}} = \begin{cases} 2\left[r^2 - \left(r - \frac{a_c}{\cos\alpha}\right)^2\right]^{\frac{1}{2}} \\ 0 < y < r \end{cases} \tag{3.79}$$

Assuming that the cutting depth of the PDC cutter is a_c, the cutting arc length of the cutter can be obtained as follows:

$$L = 2r \arccos\left(1 - \frac{a_c}{r\cos\alpha}\right)\cos\beta \tag{3.80}$$

The cutting area of the cutter is given by

$$A_{\text{cut}} = \left(r^2 \arccos\left(1 - \frac{a_c}{r\cos\alpha}\right) - r\left(r - \frac{a_c}{\cos\alpha}\right)\sin\left(\arccos\left(\frac{r - \frac{a_c}{\cos\alpha}}{r}\right)\right)\right)\cos\beta \tag{3.81}$$

Cutter Wear Calculation Under Torque

The normal wear of the drill bit is a common case of PDC drill bit failure. According to the principle of nonlinear dynamic system, the PDC drill bit wear process can be divided into self-organizing stage, chaotic stage, and instability stage, corresponding to the three periods in tribology, wearing in, normal wearing, and sharp wear stage, respectively.

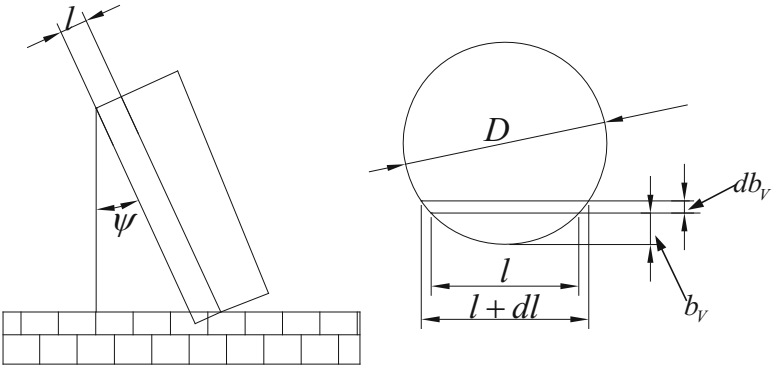


Fig. 3.27 Wear of PDC cutters

In actual work, the cutter wear rate of the PDC drill bit can be defined by the wear volume or height of the cutter in unit length or unit time. According to the previous definition, the cutter wear rate can be expressed as [16]

$$v_t = \frac{V_{\mathrm{V}}}{t} \tag{3.82}$$

or

$$v_l = \frac{V_{\mathrm{V}}}{l} \tag{3.83}$$

where v_t and v_l are the volume wear rates, V_{V} is the cutter wear volume, t is the wear time, and l is the drilling depth.

In order to study the cutter wear rate of the PDC drill bit, the following hypotheses are proposed: (1) the cutter wear volume is directly proportional to the frictional work; (2) the friction and wear of the cutter side are ignored; and (3) the wear amount of the cutter is very small.

As shown in Fig. 3.27, the wear volume of the PDC cutter is given by

$$dV_{\mathrm{V}} = \frac{[l + (l + dl)\, b_{\mathrm{V}}]}{2} [(a_{\mathrm{c}} + da_{\mathrm{c}}) - a_{\mathrm{c}}] = l b_{\mathrm{V}} da_{\mathrm{c}} - \frac{b_{\mathrm{V}}}{2 \times dl\ da_{\mathrm{c}}} \tag{3.84}$$

where dV_{V} is the differential volume of cutter wear; a and b are the width of the interface between the cutter and the rock and the thickness of the wearing surface, respectively; and dh and da are the differential height and the differential width of the cutter, respectively.

Cutter wear is so small that da and dh can be ignored. Since wear is generated under the same conditions, b can be calculated as a constant.

Frictional work is proportional to the wear volume of the PDC drill bit cutter, so normal wear of the PDC drill bit produced by frictional work is expressed as

$$dV_{\mathrm{V}} = \omega_{\mathrm{V}} d\omega_{\mathrm{V}} \tag{3.85}$$

where $d\omega_{\mathrm{V}}$ is the differential work done by friction.

Assumptions show that the friction at length $d\delta$ is given by

$$df = \mu p \, d\delta \tag{3.86}$$

The distance the differential length takes is written as

$$ds = \delta d\psi \tag{3.87}$$

The differential work is obtained as follows:

$$d\omega = df \, ds = \mu p \xi \, d\xi \, d\psi \tag{3.88}$$

where μ, p, and δ are the friction coefficient between the rock and the drill bit cutter, the linear density of the pressure, and the distance from the drill bit center to the wear part, the eccentric distance of the cutter, respectively; and $d\delta$ and $d\psi$ are the differential value of the cutter eccentric distance and the differential angle of the cutter rotation, respectively.

The work done by one rotation of the cutter is expressed as

$$W = \mu p \int_{\delta}^{\delta+a} \xi \, d\xi \int_{0}^{2\pi} d\psi = \pi l \, (l + 2\delta) \, \mu p \tag{3.89}$$

The cutter wear volume is given by

$$V_{\mathrm{V}} = \omega_{\mathrm{V}} W = \omega \pi l \, (l + 2\delta) \, \mu p N t \tag{3.90}$$

Substituting formula (3.90) into formula (3.82), we get

$$v_t = \frac{V_{\mathrm{V}}}{t} = \omega \pi l \, (l + 2\delta) \, \mu p N \tag{3.91}$$

The pressure distribution density p is defined as

$$p = \frac{F}{L} \tag{3.92}$$

where L is the contact length between the cutter and the rock and F is the pressure on the cutter.

Considering the cutter contact arc length, the state of wear, etc., the pressure on the cutter of the PDC drill bit can be expressed as

$$F = \eta\beta F_{\mathrm{d}} \tag{3.93}$$

where the arc length coefficient is defined as

$$\beta = \frac{L}{L_{\mathrm{d}}} \tag{3.94}$$

where F_{d} and L_{d} are the axial pressures on the standard cutter and the equivalent arc lengths of the cutter cutting rock, respectively.

Cutter wear increasing coefficient is written as

$$\eta = \exp\left(\frac{a_{\mathrm{c}}}{D\cos\phi}\right) \tag{3.95}$$

where a_{c}, D, and ϕ are the wear height, diameter, and caster angle of the cutter, respectively.

Although the old cutter and the new cutter are essentially the same in their stress, both the wear increase coefficient a and the arc length coefficient b affect the force condition of the old cutting teeth. Therefore, the model is a comprehensive consideration of the PDC bit cutting teeth by cutting teeth diameter, wear, and other effects.

Although the force of the old cutter and the new cutter is the same, the wear increasing coefficient μ and the arc length coefficient f both affect the force condition of the old cutter. Therefore, the model comprehensively considers the diameter and wear of the PDC drill bit cutter.

Substituting formulae (3.95) and (3.93) into formula (3.92), we get

$$p = \exp\left(\frac{a_{\mathrm{c}}}{D\cos\phi}\right)\frac{F_{\mathrm{d}}}{L_{\mathrm{d}}} \tag{3.96}$$

Substituting formula into formula (3.82), we get

$$v_t = \pi l\,(l+2\delta)\,\mathrm{C_p}\mu\tau N\exp\left(\frac{a_{\mathrm{c}}}{D\cos^2\phi}\right)\frac{F_{\mathrm{d}}}{L} \tag{3.97}$$

Assuming the wear height $a_{\mathrm{c}} = 1\,\mathrm{mm}$, the relationship between the linear velocity at the unit cutting depth and the volume wear rate can be expressed as

$$v_t = \pi l\,(l+2\delta)\,\mathrm{C_p}\mu\tau\frac{V}{D\pi}\exp\left(\frac{1}{D\cos^2\phi}\right)\frac{F_{\mathrm{d}}}{L} \tag{3.98}$$

Calculation Analysis

According to the previous calculation method, combined with the drill bit parameters, as shown in Table 3.6, the angular velocity and angular displacement of the drill string under the action of the torsional load can be obtained, as shown in Figs. 3.28 and 3.29, respectively.

As can be seen from Fig. 3.28, the rotational angular velocity of the drill string under torsional load is fluctuating. Torsional loading of high-frequency reciprocating impact drill string results in changes in the force of the drill string, so the rotational angular velocity changes.

It can be seen from Fig. 3.29 that under the effect of the torsional load, the rotational angular velocity fluctuates, so the rotational angular displacement also fluctuates.

In order to obtain the relationship between the top rake, the effective cutting edge length, cutting arc length, and cutting area, the specific parameters used, as shown in Table 3.6.

Substituting the data in Table 3.6 into the previous formula, the relationship between the effective cutting edge length, cutting arc length, cutting area, and top rake can be obtained, and the results are shown in Table 3.7.

Table 3.6 Parameters of drill bit

Parameter	Value
The roll angle β (°)	10
Cutting depth a_c (mm)	1
Cutter position angle ψ (°)	10
Normal angle of the cutter (°)	10
PDC drill bit size	8.5″
Number of blades	6
Compact radius r (mm)	8

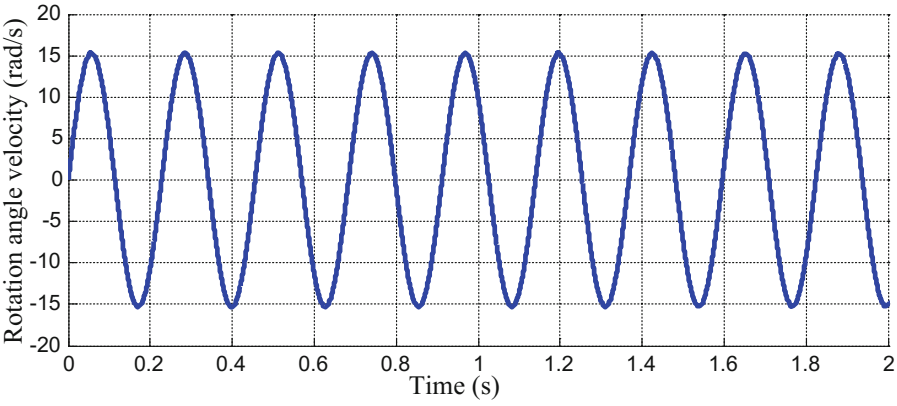


Fig. 3.28 Angular velocity of drill string

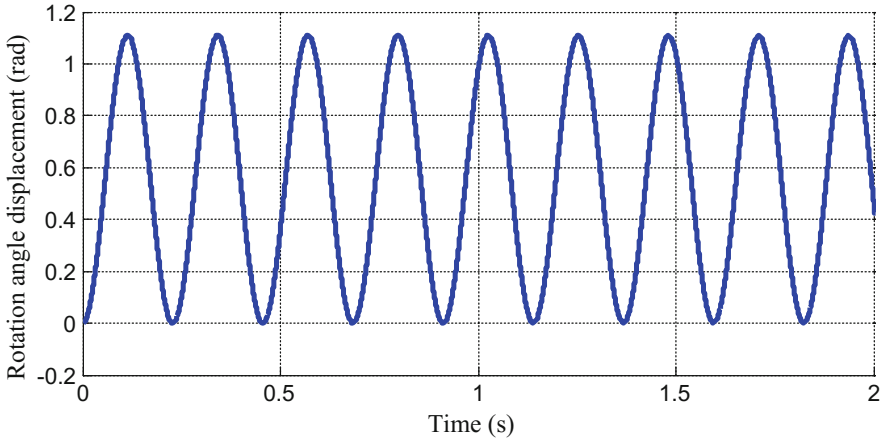


Fig. 3.29 Rotational angular displacement

Table 3.7 Parameters of drill bit cutter

Top rake	Effective cutting edge length	Cutting arc length	Cutting area
2	7.7482	7.9606	5.1376
4	7.7548	7.9532	5.0921
6	7.7658	7.9408	5.0161
8	7.7814	7.9234	4.9097
10	7.8015	7.9011	4.7726
12	7.8262	7.8737	4.6050
14	7.8556	7.8414	4.4065

The results obtained in Table 3.7 are plotted as shown in Figs. 3.30, 3.31, and 3.32.

Figure 3.30 shows the relationship between the top rake and the effective cutting edge length. As can be seen from the figure, as the top rake increases, the effective cutting edge length also increases, which means that the top rake is proportional to the effective cutting edge length.

Figure 3.31 shows the relationship between the top rake and the cutting arc length. It can be seen from the figure that as top rake increases, the cutting arc length decreases, which means that top rake is inversely proportional to the cutting arc length.

Figure 3.32 shows the relationship between the top rake and the cutting area. As can be seen from the figure, with the increase of the top rake, the cutting area decreases, which means that the top rake is inversely proportional to the cutting area.

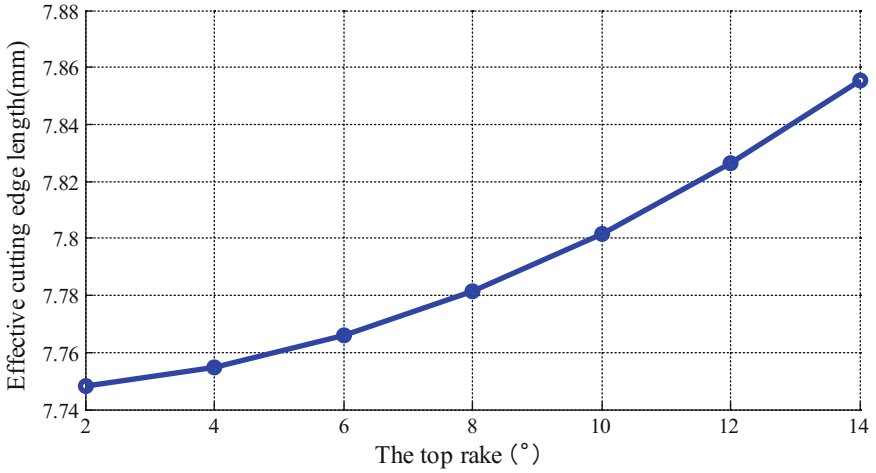


Fig. 3.30 Relationship between the top rake and effective cutting edge length

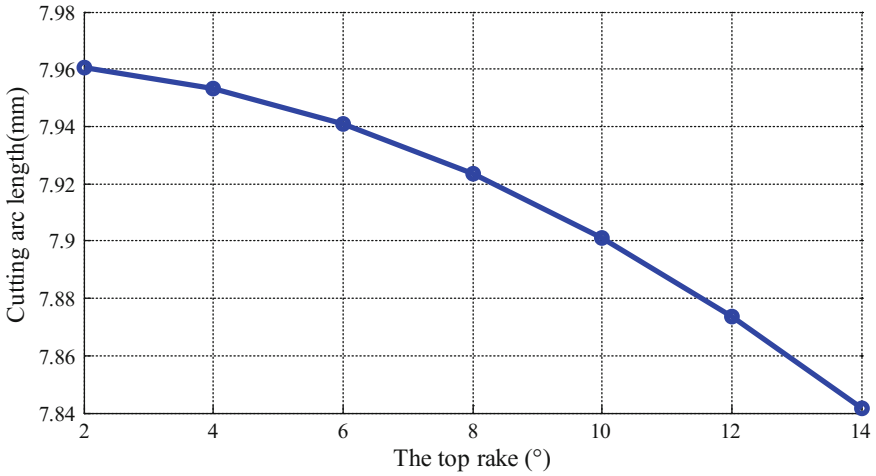


Fig. 3.31 Relationship between the top rake and cutting arc length

As can be seen from Figs. 3.30, 3.31, and 3.32, the top rake has different effects on the effective cutting edge length, cutting arc length, and cutting area. With the increase of the top rake, the effective cutting edge length increases, while the cutting arc length and cutting area decrease. This is not conducive to the improvement of the ROP and rock-breaking efficiency, and it is not possible to break rock efficiently (Table 3.8).

The results of substituting the parameters in Table 3.9 into the previous formula are shown in Fig. 3.33.

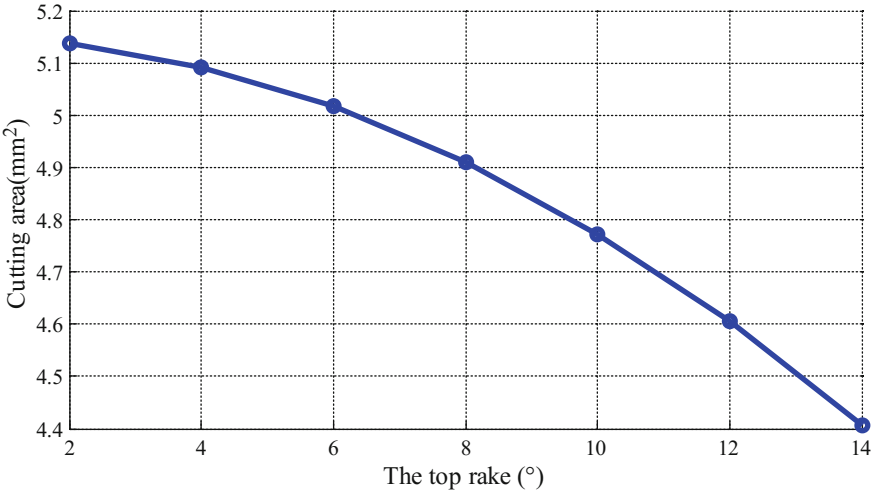


Fig. 3.32 Relationship between the top rake and cutting area

Table 3.8 Cutter force parameters

Parameter	Value
Axial pressure on cutter F_d (N)	400
Cutter wear strength τ (N/mm^2)	70
Hydraulic coefficient C_p	0.14
Friction coefficient between cutter and rock μ	0.4
Caster angle $\varnothing$ (°)	10

Table 3.9 Drilling parameters

Service condition of torque load	ROP (m/h)
√	6.5
×	4.5

Figure 3.33 shows that there is a direct relationship between the volume wear rate of the cutter (v_t) and the distance from wear part to drill bit center (δ). With the increase of δ, the v_t also increases, and it is a linear distribution. The phenomenon shows that the cutters in the center of the blade have small cutting volume, and the less wear, while the outer cutter in cutting blades edges have large cutting volume and heavy wear.

The results of substituting the parameters in Table 3.9 into the previous calculation formula are shown in Fig. 3.34.

From Fig. 3.34, it can be seen that the cutter volume wear rate under the action of torsional load is small, and as the distance increases, the greater the difference between the cutter wear with torsional load or not.

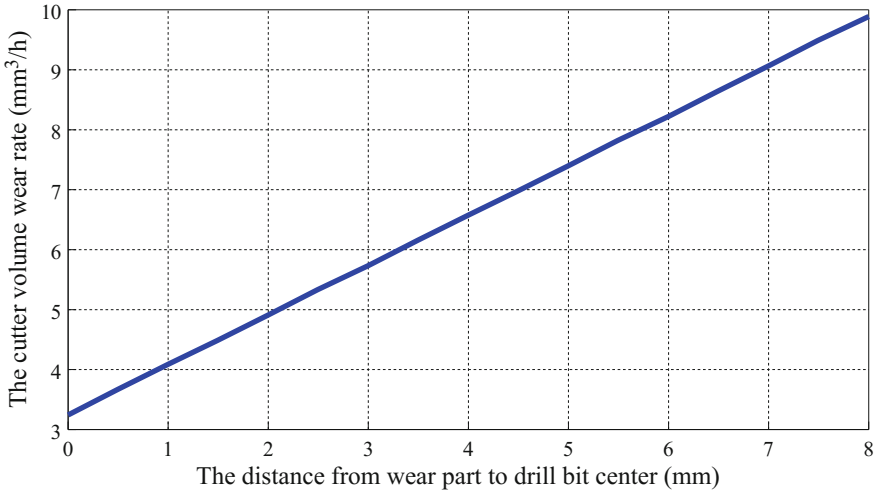


Fig. 3.33 Distance between wear part and drill bit center

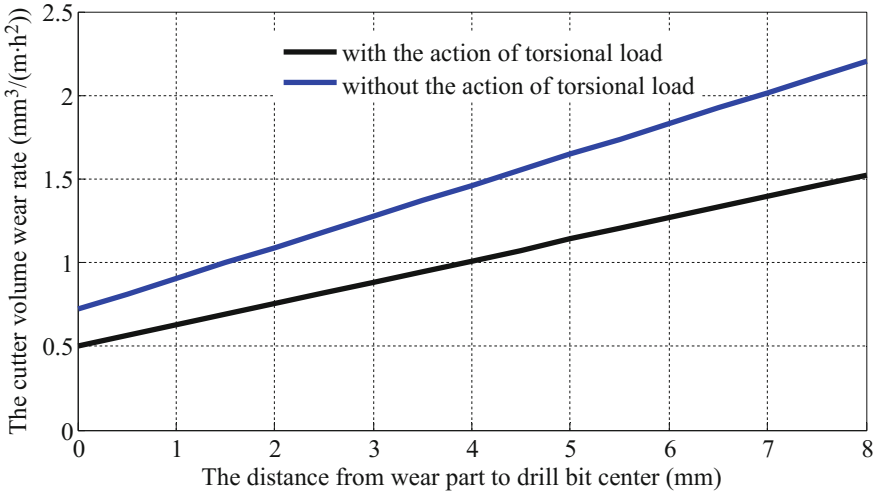


Fig. 3.34 Results comparison of cutter wear with torture load or not

3.3 Concluding Remarks

Aiming at the existing exploitation conditions of new oil and gas resources, there is great importance to conduct the drill string dynamics study, especially for shale gas, coal-bed methane, large displacement horizontal well, and so on.

With the consideration of the friction randomness of drill string, drill string dynamic characteristics can be evaluated accurately. And with the combination

of the actual downhole conditions, the wellbore friction randomness is described and the steps and methods of constructing the wellbore random field are given. Based on this, the dynamic solution models of the horizontal drill string are established. According to the experiment test, the basic input parameters are determined, and a numerical example is carried out to calculate the wellbore friction coefficient. The vibration displacement, the vibration velocity, and drilling efficiency are solved and analyzed, and the frequency spectrum analysis of the vibration velocity is completed. Besides, the phase diagram and Poincare plot of the test point are obtained. The established method can provide a reference for the drill string dynamic analysis on the basis of the wellbore friction randomness and the quantitative evaluation of the influence of the key parameters on the drill string dynamics.

For the actual condition of drilling process, the mechanical model and vibration equations are established respectively with considering the radial inertia effect or not. Within the frequency range of the drill string's dynamic design, the conclusions can be drawn. The drill string's dynamic stiffness presents cyclical transformation of amplitude increase with increase in frequency. Therefore, the right drill column length should be selected comprehensively according to the requirements of resisting vertical deformation and vibration. As the drill column length and cross-sectional area increase, the dynamic stiffness of the drill string also increases; when inner diameter is low or outer diameter sets high scale at the same frequency, the absolute value of the dynamic stiffness is greater correspondingly. The bigger the damping coefficient, the greater is the force to drill column from the bottom of the well; then, the amplitude of dynamic stiffness is smaller, which conforms to the actual condition. The influence of bottom-well's dynamic stiffness from the Poisson's ratio and the radial inertia effect is obvious gradually with increase in frequency. Therefore, the influence on the drill string vertical vibration characteristics from Poisson's ratio cannot be ignored. Especially, when the vibration frequency is bigger, the influence of lateral dimension effect should be considered while designing the drill string; otherwise, the results will show deviations.

The torsional vibration model is established in the chapter, along with the changing relationship of vibration angular displacement and vibration angular velocity with the change of time. According to the analysis result, for the vibration of drill string, which is under the torsional load, angular velocity and angular displacement are varied wavelike. And on the basis of theoretical research, the cutting teeth wear model of the PDC drill bit is given to analyze the cutting teeth wear pattern. The cutter volume wear rate under the action of torsional load is smaller. And as the distance increases, the difference also increases between the cutter wear with torsional load or not.

Drill string dynamics is the theoretical basis for the safe production of down-hole tools, the improvement of rock-breaking efficiency and the development of new down-hole tools in oil and gas production process. Simultaneously, through the tool failure, rock-breaking, and other result parameters, we can contradict that the

corresponding drill string dynamics calculation method is reasonable. And when the result parameters are constant with the reversed result, the dynamics calculation method can meet the actual needs and effectively promote the development of related technology.

References

1. Gao, D., & Gao, B. (1997). Analysis of buckling and friction of horizontal well string. In Symposium on *Fundamental Theory of Extended Reach Wells of Petroleum Engineering Society of China Petroleum Institute*.
2. He, Z., Jianhong, F., Taihe, S., et al. (2001). Torque resistance torque model for large displacement wells. *Natural Gas Industry, 21*(5), 52–54.
3. Huizeng, Z., Zhichuan, G., Yi, K., et al. (2015). Lateral vibration effects on drill string friction in horizontal wellbore. *Petroleum Drilling Technology, 43*(3), 61–64.
4. Valery, G., & Natalya, S. (2016). Influence of friction on buckling of a drill string in the circular channel of a bore hole. *Petroleum Science, 13*(4), 698–711.
5. Jaculli, M. A., Mendes, J. R. P., & Miura, K. (2017). Dynamic buckling with friction inside directional wells. *Journal of Petroleum Science & Engineering, 153*(5), 145–156.
6. Elgibaly, A. A., Farhat, M. S., Trant, E. W., et al. (2016). A study of friction factor model for directional wells. *Egyptian Journal of Petroleum, 26*(2), 489–504.
7. Ritto, T. G., Escalante, M. R., Sampaio, R., & Rosales, M. B. (2013). Drill-string horizontal dynamics with uncertainty on the frictional force. *Journal of Sound and Vibration., 332*, 145–153.
8. Zebing, W., Dekun, M., & Yuchun, Z. (2000). Simulation analysis of drill string longitudinal vibration. *Acta Petrolei Sinica, 21*(3), 73–76.
9. Yuchun, Y., Dekun, M., Qingyou, L., & zebing, W. (2001). Dynamic behavior simulation of drill string-bit-rock system. *Acta Petrolei Sinica, 22*(3), 81–85.
10. Ghasemzadeh, H., & Alibeikloo, M. (2011). Pile-soil-pile interaction in pile groups with batter piles under dynamic loads[J]. *Soil Dynamics and Earthquake Engineering, 31*(8), 1159–1170.
11. Tian, J., Yang, Y., & Yang, L. (2017). Vibration characteristics analysis and experimental study of horizontal drill string with wellbore random friction force. *Archive of Applied Mechanics, 87*(2), 1–13.
12. Yang, J., Zhang, D., & Lu, Z. (2004). Stochastic analysis of saturated–unsaturated flow in heterogeneous media by combining Karhunen-Loeve expansion and perturbation method. *Journal of Hydrology, 294*(1), 18–38.
13. Yin, L., Yang, L., & Yuanchang, M. (2008). Analysis of structural system reliability based on stochastic response surface method. *Journal of Guangxi University (Nat Sci Ed), 33*(4), 366–369.
14. Depouhon, A., & Detournay, E. (2014). Instability regimes and self-excited vibrations in deep drilling systems. *Journal of Sound and Vibration, 333*(7), 2019–2039.
15. Franca, L. F. P. (2010). Drilling action of roller-cone bits: modeling and experimental validation. *Journal of Energy Resources Technology, 132*(4), 043101.
16. Guangzhi, L. (2009). Diamond, PDC bit and technology. *Earth Science: Journal of China University of Geosciences, 34*(3), 539–539.

Chapter 4
Nonlinear Modeling Application to Micro-/Nanorobotics

Ali Ghanbari and Mohsen Bahrami

4.1 Introduction

Micro-/nanorobots have the potential to revolutionize medicine by specific applications, such as targeted drug delivery, biopsy, hyperthermia, brachytherapy, scaffolding, in vivo ablation, sensing, marking, and stem cell therapy [1, 2]. Application of microrobots can move us to the stage that monitoring diseases, highly localized drug delivery, minimally invasive surgery, and novel therapies such as stem cell therapy are done using the tools inside the human body. The tiny machines can be inserted into the human body through the natural conduits of the body. The application of microrobots in medicine requires a multidisciplinary delicate investigation. Microrobot propulsion system, actuation and control system, and microrobot motion control are the areas that should be addressed.

Since size is small and velocity is low, microrobots have a very low Reynolds (*Re*) number. *Re* number is the ratio between inertial forces and viscous forces, and a low *Re* number indicates the dominance of viscous forces. Hence, swimming methodologies at microscale are different from those at the macroscale. Several methodologies have been proposed and demonstrated for motile microrobots in viscous fluid environments. These microswimming robots are usually biomimetic, employing techniques inspired by microorganisms and bacteria, which use cilia and flagella to propel themselves [3–7]. However, researchers have also used the direct pulling of a permanent magnet or a soft magnet in the fluid using external magnetic fields with a magnetic field gradient at the location of the microrobot

A. Ghanbari
Mechanical Engineering Department, Tafresh University, Tafresh, Iran

M. Bahrami (✉)
Mechanical Engineering Department, Amirkabir University of Technology, Tehran, Iran
e-mail: mbahrami@aut.ac.ir

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_4

[8]. Although motion is linear at Stokes flow, hydrodynamics of flagella and cilia involve nonlinear models that should be address for precise actuation and control of micro-/nanorobots. Nonlinear modeling is of great significance especially when the artificial filaments are fabricated from soft materials to mimic natural flagella and cilia and provide enhanced propulsion [9].

Another great challenge in developing an autonomous microrobotic system is to provide power and control to the microrobot. Since untethered microrobots can be used as implants and have a higher maneuverability, the control system should benefit from a wireless actuation mechanism. Magnetic actuation can transfer a reasonable amount of power wirelessly. There are different systems for generating magnetic field and gradients:

- Permanent magnets
- Helmholtz coils, Maxwell coils, or a combination thereof
- Magnetic resonance imaging (MRI) systems
- Customized sets of electromagnetic coils

Permanent magnets do not present the required flexibility for control of microrobots and it would be difficult to generate gradient fields using permanent magnets. Helmholtz and Maxwell coils, which use air-core electromagnetic coils, surround the workspace entirely causing some constraints for biomedical applications. Indeed, they generate a weaker magnetic field when compared to the electromagnetic coils with a soft-magnet core. Magnetic resonance imaging (MRI) systems have also been used to manipulate the micro-/nanorobots [10, 11].

Customized sets of electromagnetic coils have widely been used for actuation of micro-/nanorobots [12, 13]. These devices can be utilized in any of methodologies for propulsion at microscale including gradient-based pulling, flagellar motion, and finally ciliary propulsion. The electromagnetic coils generate the required power for actuation of magnetic micro-/nanorobots and are safe in biomedical applications.

Hydrodynamic model of swimming microrobots which use flagellar, ciliary, and gradient-pulling methodologies leads to coupled nonlinear systems of equations describing the fluidic–elastic–actuation problem. Also, magnetic actuation predominantly involves forces and moments that are nonlinear functions of spatial coordinates [14, 15].

Since propulsion and actuation are main challenges in micro-/nanorobotics besides sensing, modeling of these functions provides an insightful realization of these systems and their implementation at micro-/nanoscale. Here, we discuss the application of nonlinear modeling in propulsion and magnetic actuation of micro-/nanorobots. We provide the hydrodynamic modeling and magnetic propulsion of a ciliary microrobot. We also give a nonlinear model of magnetic stimulation using a set of electromagnetic coils that can be used in actuation and control of micro-/nanorobots. We solve the models for numerical examples and investigate the role of various design parameters.

4.2 Hydrodynamic Modeling of a Ciliary Microrobot

4.2.1 Stokes Flow

Hydrodynamics of an incompressible fluid flow with a constant density ρ and a constant dynamic viscosity μ is governed by the Navier–Stokes equations:

$$\begin{aligned} &\rho\left(\tfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u}.\nabla\mathbf{u}\right) = \rho\mathbf{g} + \mu\nabla^2\mathbf{u} - \nabla p \\ &\nabla \cdot \mathbf{u} = 0 \end{aligned} \tag{4.1}$$

where $\mathbf{u}$ is the fluid velocity, $\mathbf{g}$ is the gravitational acceleration, and p is the pressure. Stokes flow is a type of fluid flow in which the inertia forces are small against the viscous forces and their effects can be ignored. For a fluid with a constant kinematic viscosity ν, Reynolds number is defined as $\mathrm{Re} = \frac{uD}{\nu}$, which is a dimensionless ratio between the inertia and viscous forces. Reynolds number is very small for a Stokes flow; therefore, the inertia forces can be neglected in the Navier–Stokes equations, and Stokes equations are obtained. For an incompressible Newtonian fluid, the Stokes equations are derived as:

$$\begin{aligned} &\nabla p - \mu\nabla^2\mathbf{u} = \mathbf{0} \\ &\nabla \cdot \mathbf{u} = 0 \end{aligned} \tag{4.2}$$

Since flagella and cilia have a small size and motion is slow for artificial microswimmers, Reynolds number is very small, on the order of 10^{-4}, and the fluid by flagella and cilia is described using Eq. (4.2).

Taylor modeled flagella motion by forming an envelope curve of the flagella tip generating a wave [16]. He obtained the flagellum mathematical model and established a relationship between the speed of the microorganism movement and the velocity of its wave propagation.

Gray and Hancock suggested that since cilia and flagella are slender, i.e., their radius relative to the length and the radius of curvature are very small, the local drag forces (exerted by the surrounding fluid) can be related to the local velocities [17]. In the Gray–Hancock model, the components of the drag force are considered to be proportional to the velocity component in the same direction with a constant coefficient, called resistive coefficient [17]. A drag force can be decomposed into three directions of tangential, normal, and binormal. Although the model is simple to apply, the resistive coefficients have not been specified for complex shapes. The method also does not take into account the effect of flow induced by the cell body or neighboring flagella or cilia [18, 19].

Slender body theory (SBT) is an alternative method to analyze the hydrodynamic flow around, for example, a cylindrical filament with a very small radius compared to its length. This theory, developed by many authors [20–22], relates the flow distribution at a cross section along the slender body not only to the resistive forces (which are the effect of "near field") but also to the hydrodynamic interactions of the

"far field" [19]. Therefore, this method is more precise than the resistive force theory and is particularly for the case where the flagella or cilia are attached to a body, the microorganisms move near the walls, or the microorganism has several flagella or cilia [19, 23]. The SBT model takes into account the hydrodynamic interactions in any of these cases.

4.2.2 Cilia Hydrodynamics

For a cilium with a radius c and length L moving in the fluid, the drag forces in tangential and normal directions are given by RFT as:

$$\phi_{\mathrm{T}} = -C_{\mathrm{T}} V_{\mathrm{T}} \tag{4.3}$$

$$\phi_{\mathrm{N}} = -C_{\mathrm{N}} V_{\mathrm{N}} \tag{4.4}$$

where $V(s,t)$ is the velocity at a cross section s along the cilium at the time t and has two components in two dimensions in the tangential and normal directions as V_{T} and V_{N}. ϕ_{T} and ϕ_{N} are the tangential and normal drag forces per unit length applied by the surrounding fluid with a viscosity μ. Tangential and normal resistance coefficients C_{T} and C_{N} that relate drag forces to the velocity in that direction are defined as:

$$C_{\mathrm{T}} = \frac{8\pi\mu}{-2 + 4\ln(2a/c)} \tag{4.5}$$

$$C_{\mathrm{N}} = \frac{8\pi\mu}{1 + 2\ln(2a/c)} \tag{4.6}$$

where a has an arbitrary value so that $a/L \ll 1$ and $c/a \ll 1$. Brennen and Winet [24] showed these expressions are more accurate for the tangential and vertical resistance coefficients than those calculated by Gary and Hankook [17].

Gueron and Liron utilized SBT to model the hydrodynamics of cilia considering the effect of the distant segment, the cell body, neighboring cilia, and external flows on the drag force of a given cross section along the cilium [19, 23]. These effects are captured by pseudo-drag forces g_{T} and g_{N}:

$$\phi_{\mathrm{T}} = -C_{\mathrm{T}} V_{\mathrm{T}} + g_{\mathrm{T}} \tag{4.7}$$

$$\phi_{\mathrm{N}} = -C_{\mathrm{N}} V_{\mathrm{N}} + g_{\mathrm{N}} \tag{4.8}$$

where g_T and g_N are given by:

$$g_T = C_T G_T \tag{4.9}$$

$$g_N = C_N G_N \tag{4.10}$$

where $\mathbf{G} = (G_T, G_N)$ is the velocity induced at a cross section by other sources excluding the local near segment. A full description of the method can be found in [19]. To solve the Stokes flow, a distribution of Stokeslets and doublets is considered at the cilium centerline along its length. Then, these Stokeslets and doublets are integrated over the far field and neighboring cilia to account for their contribution into the total drag force. The external flow velocity is directly included in $\mathbf{G}$ [19].

A low Reynolds number simplifies the mathematical analysis of the corresponding hydrodynamics since the fluid mechanics is linear at this regime. However, even at Stokes flow modeling, the cilia motion involves a system of interconnected nonlinear equations which includes the actuation force, cilia elasticity, and the external fluid dynamics.

Consider the cilium as an inextensible filament shown in Fig. 4.1a. Suppose (x, y) and (T, N) are global and local coordinates (as shown in Fig. 4.1a). It is possible to write the velocity of the cilium at a cross section s in terms of local coordinates. By differentiating this term with respect to the spatial variable s along the cilium and evaluating the differentiation of the tangential and normal unit vectors, the following formulations are obtained [19]:

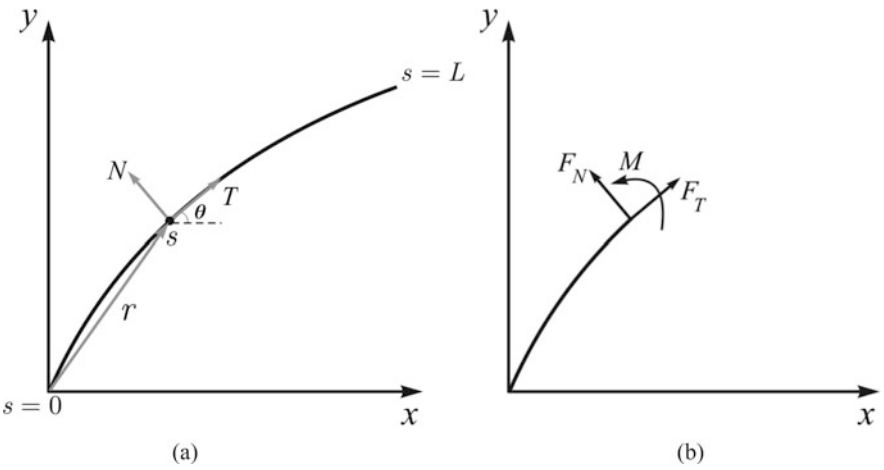


Fig. 4.1 (**a**) Global and local coordinates shown for the cilium and location vector r of a point s along the cilium. (**b**) Internal forces and moment at a cross section s

$$V_{N_s} = \theta_t - V_T \theta_s \tag{4.11}$$

$$V_{T_s} = V_N \theta_s \tag{4.12}$$

where indices s and t represent partial differentiation with respect to the spatial variable and time. If we cut the cilium at a point s, then we have two internal tangential and normal forces $F_{\mathrm{T}}(s,t)$ and $F_{\mathrm{N}}(s,t)$, respectively, and one internal moment $M(s,t)$ to maintain the movement of this point (Fig. 4.1b). These should satisfy the forces and moments balance [19]:

$$\phi_{\mathrm{T}} = F_{\mathrm{T}_s} - F_{\mathrm{N}} \theta_s \tag{4.13}$$

$$\phi_{\mathrm{N}} = F_{\mathrm{N}_s} + F_{\mathrm{T}} \theta_s \tag{4.14}$$

$$M_s = F_{\mathrm{N}} \tag{4.15}$$

The internal moment is due to the cilium elasticity as well as the applied actuation torque, $T_{\mathrm{A}}(s,t)$. If cilium is considered as a beam, the bending moment due to elasticity is related to the beam curvature:

$$M = K_{\mathrm{b}} \kappa \tag{4.16}$$

where $\kappa = \frac{\partial \theta}{\partial s}$ and K_{b} are the cilium bending stiffness. Therefore, using Eq. (4.15), we obtain the equation for the normal force as:

$$F_{\mathrm{N}} = K_{\mathrm{b}} \theta_{ss} + T_{\mathrm{A}} \tag{4.17}$$

If we eliminate the velocity components in Eqs. (4.13)–(4.17) using Eqs. (4.11), (4.12) and (4.7), (4.8), we get the following equations [23]:

$$F_{\mathrm{N}_{ss}} + \left(1 + \frac{C_{\mathrm{N}}}{C_{\mathrm{T}}}\right) F_{\mathrm{N}_s} \theta_s + F_{\mathrm{T}} \theta_{ss} = -C_{\mathrm{N}} \theta_t + \frac{C_{\mathrm{N}}}{C_{\mathrm{T}}} F_{\mathrm{N}} (\theta_s)^2 + \frac{C_{\mathrm{N}}}{C_{\mathrm{T}}} g_{\mathrm{T}} \theta_s + g_{\mathrm{N}_s} \tag{4.18}$$

$$F_{\mathrm{T}_{ss}} = \left(1 + \frac{C_{\mathrm{T}}}{C_{\mathrm{N}}}\right) F_{\mathrm{N}_s} \theta_s + \frac{C_{\mathrm{T}}}{C_{\mathrm{N}}} F_{\mathrm{T}} (\theta_s)^2 + F_{\mathrm{N}} \theta_{ss} - \frac{C_{\mathrm{T}}}{C_{\mathrm{N}}} g_{\mathrm{N}} \theta_s + g_{\mathrm{T}_s} \tag{4.19}$$

The cilium motion is expressed using Eqs. (4.17)–(4.19). If an actuation torque is defined, then this coupled system of nonlinear partial differential equations should be solved along with boundary and initial conditions to find the cilium internal forces, F_{T} and F_{N}, and the cilium shape, $\theta(s,t)$, during its motion.

The cilium is considered to be stiff at the base [23] and erect at its end, then:

$$\theta_s\,(0,t) = 0 \tag{4.20}$$

$$\theta_s\,(L,t) = 0 \tag{4.21}$$

Since the cilium has a limited motion near its base, drag forces are considered to be zero at $s = 0$. Hence, from Eq. (4.13), (4.14), and (4.20), we get:

$$F_{\mathrm{T}_s}\,(0,t) = F_{\mathrm{N}_s}\,(0,t) = 0 \tag{4.22}$$

and using Eqs. (4.22) and (4.17):

$$\theta_{sss}\,(0,t) = -T_{\mathrm{A}_s}\,(0,t) \tag{4.23}$$

The cilium is anchored at $s = 0$ and is free at $s = L$. Hence, there is no internal force or moment at the free end:

$$F_{\mathrm{T}}\,(L,t) = F_{\mathrm{N}}\,(L,t) = 0 \tag{4.24}$$

$$\theta_{ss}\,(L,t) = -T_{\mathrm{A}}\,(L,t) \tag{4.25}$$

4.2.3 Equations of Motion

Inertia is neglected for a microrobot swimming in a viscous fluid with a low Reynolds number. Hence, since the rate of change of linear and angular momentums of a microswimmer is negligible, the applied forces and moments are at static equilibrium. Thus, for the microrobot shown in Fig. 4.2, we write the force balance in x- and y-directions as:

$$\sum \mathbf{F}_{x,\mathrm{cilia}} + \sum \mathbf{F}_{x,\mathrm{body}} = 0 \tag{4.26}$$

$$\sum \mathbf{F}_{y,\mathrm{cilia}} + \sum \mathbf{F}_{y,\mathrm{body}} = 0 \tag{4.27}$$

and the moment balance perpendicular to the plane:

$$\sum \mathbf{M}_{\mathrm{cilia}} + \sum \mathbf{M}_{\mathrm{body}} = 0 \tag{4.28}$$

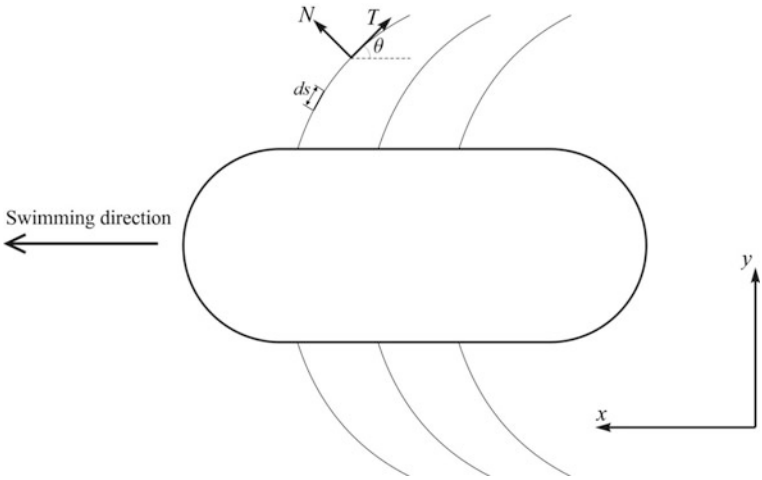


Fig. 4.2 Swimming microrobot model with global and local coordinates

Only hydrodynamic forces are applied to the microswimmer and need to be considered in Eqs. (4.26) and (4.27). For an element ds of the cilium length, these forces are written in x- and y-directions:

$$dF_{x,\text{cilium}} = \phi_{\text{T}} ds \cos\theta + \phi_{\text{N}} ds \sin\theta \tag{4.29}$$

$$dF_{y,\text{cilium}} = \phi_{\text{N}} ds \cos\theta + \phi_{\text{T}} ds \sin\theta \tag{4.30}$$

where ϕ_{T} and ϕ_{N} are tangential and normal drag forces and are obtained from:

$$\phi_{\text{T}} = -C_{\text{T}} \left(V_{\text{T}} - U\cos\theta\right) \tag{4.31}$$

$$\phi_{\text{N}} = -C_{\text{N}} \left(V_{\text{N}} + U\sin\theta\right) \tag{4.32}$$

where U is the velocity of the microrobot body.

Two rows of cilia are symmetrically aligned around the microrobot centerline on both sides of the body (Fig. 4.2). For a planar motion of the microrobot, no force is applied to the body in y-direction. Also, the vertical component of the hydrodynamic force is identical for the cilia on both sides, but, with an opposite direction. Therefore, they cancel each other and Eq. (4.27) is satisfied. The resulting moment of cilia force, $F_{x,\text{ cilia}}$, for the row of cilia on one side equals the one for the row of cilia on the other side, but with a negative direction. Hence, they cancel each other, and since there is no other moment exerted on the microrobot, then Eq. (4.28) is also satisfied.

Replacing Eqs. (4.31) and (4.32) into (4.29), we find the force on the element ds in x-direction:

$$dF_{x,\text{cilium}} = \left(C_\text{T} V_\text{T} \cos\theta - C_\text{T} U \cos^2\theta - C_\text{N} V_\text{N} \sin\theta - C_\text{N} U \sin^2\theta\right) ds \quad (4.33)$$

Integrating Eq. (4.33) along the length of one cilium and multiplying by the total number of cilia, n, we obtain the total force of cilia in x-direction:

$$F_{x,\text{cilia}} = n \int_0^L dF_{x,\text{cilium}} \quad (4.34)$$

For objects moving in the fluid with a small Reynolds number, the hydrodynamic drag force is proportional to their velocity. For the specific microrobot body shape considered as an ellipsoid (Fig. 4.2), the drag force is in the parallel direction to its longitudinal centerline with a magnitude [25]:

$$F_{x,\text{body}} = 8.5 \mu R U \quad (4.35)$$

where R is the radius of the head hemisphere. Since there is no other force than the cilia hydrodynamic force given by Eq. (4.34) and the body drag force given by (4.35), we balance these forces and obtain the propulsive velocity.

4.2.4 Numerical Example

To solve the nonlinear equations of cilia numerically, we design the microrobot parameters and evaluate its dynamic properties. Kim et al. fabricated a ciliary microrobot using three-dimensional laser lithography [26]. Figure 4.3a shows the microrobot model with its dimensions. Laser lithography enables microfabrication of 3D structures with a precise control over their size and a high resolution. In this technique, two laser beams are concentrated to form a single ellipsoidal spot, which is used as a building unit. The movement of a piezoelectric stage is controlled precisely to follow a pre-programmed path to partially expose the photoresist. A full 3D structure is produced after removing the unexposed photoresist in a developer. Scanning electron microscopy (SEM) image of the fabricated ciliary microrobot is depicted in Fig. 4.3b. A Nickel layer has been sputtered over the cilia to provide magnetic properties for actuation. To solve a numerical example of the nonlinear model developed in Sect. 4.2.2 and 4.2.3, we choose properties and dimension of the microrobot body and cilia as in Table 4.1. We have selected the microrobot dimensions so that they are producible using 3D laser lithography fabrication.

Artificial cilia should be sufficiently flexible to bend properly during the recovery stroke. Kim et al. fabricated the artificial cilia from IP-dip with a modulus of elasticity of 4 GPa, which is close to the natural cilia Young modulus [23]. However, natural cilia have a typical diameter of 25 nm, six times less than what

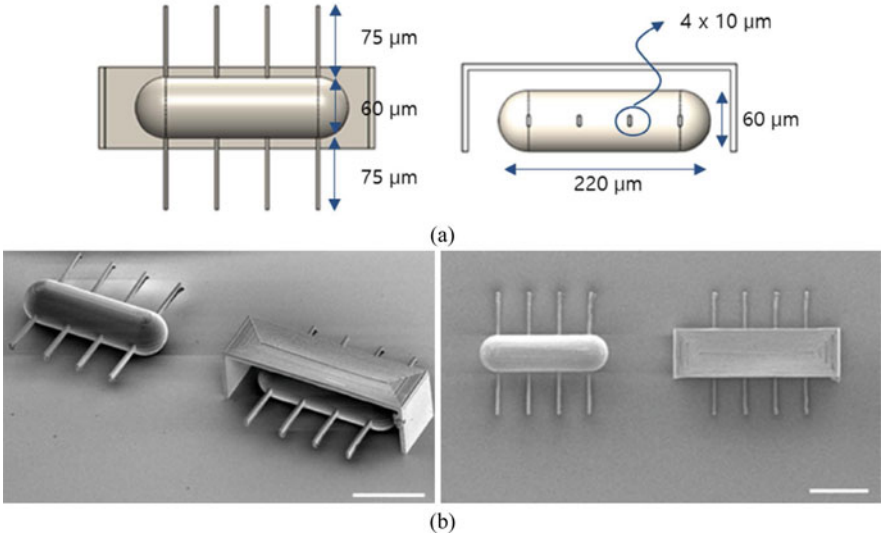


Fig. 4.3 (**a**) Model of a ciliary microrobot with design parameters [26]. (**b**) The ciliary microrobot fabricated using laser lithography method. Scale bar is 100 μm. Reproduced with permission from [26]

Table 4.1 Ciliary microrobot design parameters

Design parameter	Value
Cilium length, L_{cilia}	75 μm
Cilium radius, c_{cilia}	75 nm
Cilium Young's modulus, E	4 GPa
Body radius, R_{body}	150 μm
Body length, L_{body}	220 μm
Number of cilia, n	8
Water viscosity, μ_{water}	0.001 kg/(m·s)

we consider here (150 nm). Hence, flexural stiffness of the artificial cilia, obtained by multiplying the modulus of elasticity and the moment of inertia, becomes larger compared to that of the natural cilia. We assume the microrobot moves in a fluid with the viscosity of water.

For actuation of cilia, we assume $T_{\text{A}}(s,t)$ to have a function like the model described for an internal engine of natural cilia [23]. This model generates a beating pattern similar to what is seen in *Paramecium*. The internal engine of natural cilia has a $9 + 2$ structure with 9 peripheral pairs of microtubules connected to a central pair of microtubules by radial spokes [27]. A relative sliding between microtubules leads to a surprising active bending of cilia. To understand the behavior of ciliated microorganisms, their movement has been observed by many researchers and their various beating patterns have been identified [28, 29]. Based on these

observed patterns, Gueron and Levit-Gurevich proposed a model that approximates the internal motor of cilia [23]. This model works for effective and recovery strokes and is given by:

$$T_{\mathrm{A}_{\mathrm{eff/rec}}} = (\pm 1) \cdot \left\{ \frac{C_{\mathrm{N}} L^2}{T_{\mathrm{A}_0}} \omega_{\mathrm{eff/rec}} \cdot \frac{\left(s^2 - 1\right)}{2} \cdot \left[1 + h_{\mathrm{eff/rec}}\left(s, t\right)\right] + B_{\mathrm{eff/rec}} \cdot \kappa\left(s, t\right) \right\} \tag{4.36}$$

where T_{A_0} is the typical magnitude of shear force within the cilia [23], ω_{eff} and ω_{rec} are the average angular velocities during effective and recovery stroke, respectively, B is the model constant that varies for effective and recovery strokes as $B_{\mathrm{eff}} = 0$ and $B_{\mathrm{rec}} = 2$, and $\kappa = \theta_s$ is the cilium curvature. The parameter h is given by two different equations for effective and recovery strokes:

$$h_{\mathrm{eff}}\left(s, t\right) = C_1 + C_2 \cdot \left(\theta - \frac{\pi}{2}\right)^2 \tag{4.37}$$

$$h_{\mathrm{rec}}\left(s, t\right) = \begin{cases} 1 + C_1 + C_2 \cdot \left(\theta\left(0, t\right) - \frac{\pi}{2}\right)^2 & 0 \le s \le 0.1 \\ C_1 + C_2 \cdot \left(\theta\left(0, t\right) - \frac{\pi}{2}\right)^2 & 0.1 \le s \le 1 \end{cases} \tag{4.38}$$

where $A_1 = 0.26$ and $A_2 = -0.17$, $A_1 = 1$ and $A_2 = -2$ are constants for effective and recovery stroke, respectively.

To solve the system of nonlinear partial differential equations of cilia (Sect. 4.2.2), we divide the cilium length into 50 elements ($ds = L/50$). The time step is chosen 0.1 ms ($dt = 0.1$ ms). For convenience, a nondimensional form of equations has been used in the numerical solution. The internal motor nondimensional frequency is taken as $\omega_{\mathrm{eff}} = 393$ and $\omega_{\mathrm{rec}} = 82$ for effective and recovery strokes, respectively [23].

Assuming cilia shape at $t = 0$ (initial condition) and the actuation force, we calculate F_{N} from Eq. (4.17). Then, using the values of g_{T} and g_{N} at a previous time step, Eq. (4.19) is solved to find F_{T}. Now having F_{T} and F_{N}, we calculate ϕ_{T} and ϕ_{N} from Eqs. (4.13) and (4.14). Then, new values for g_{T} and g_{N} are obtained for the current time step. This process is iterated until the relative error of the value for F_{T} becomes less than a given amount [23]. Then, Eq. (4.18), which is a fourth-order nonlinear PDE with respect to θ, is solved to get the cilia shape over time.

A beating cycle is defined from the beginning of the effective stroke to its end and then returning to the initial position through the recovery stroke. During the effective stroke, cilia are straight and beat on the fluid. However, they are firmly bent during the recovery stroke to have minimal resistance to the fluid. Effective stroke commences at $\theta = 160°$ and ends at $\theta = 20°$. In addition to boundary conditions described in Sect. 4.2.2, we select the initial condition as $\theta(s, 0) = 160°$. We can solve the equations for one beating cycle and determine the propulsive force from Eq. (4.34). Then, we write the force balance and obtain the propulsive velocity of

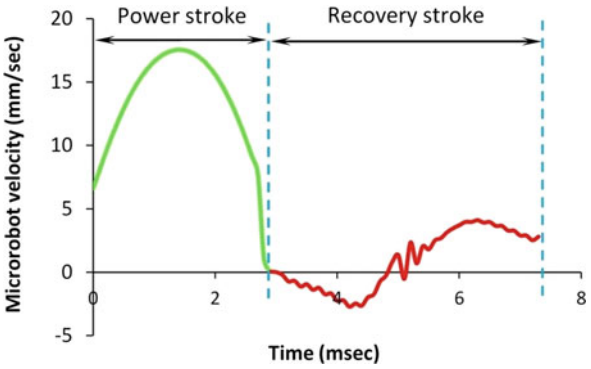


Fig. 4.4 The propulsive velocity of the microrobot for one beating cycle of cilia as a function of time

the microswimmer from Eq. (4.35). The resistance coefficients for cilia in tangential and normal directions are $C_T = 0.00143$ and $C_N = 0.00233$, respectively.

The microswimmer propulsive velocity is shown in Fig. 4.4 as a function of time. One full cycle of cilia beating takes 7.3 ms. The propulsive force is proportional to the propulsive velocity through the coefficient of body drag (Eq. 4.35). The microswimmer velocity increases when cilia start beating at $\theta = 160°$ to a maximum value at $\theta = 90°$, then decreases. Figure 4.4 suggests that cilia motion during the recovery stroke does not contribute to or withhold from the microrobot propulsion significantly. The maximum velocity during effective stroke is seven times higher than the utmost velocity during the recovery stroke. The net displacement of the microswimmer can be evaluated from the area under the velocity–time curve (Fig. 4.4). Displacement during effective and recovery stroke is 38.0 and 2.2 μm, respectively. Hence, this results in a net displacement of 40.2 μm for the microswimmer at a mean speed of 5.66 mm/s.

4.2.5 Design Parameters

In this section, we characterize effect of each of the design parameters on the microrobot motion. Figure 4.5a shows the microrobot velocity for various radiuses of cilia during the effective stroke. Other parameters have the same values as in Table 4.1. The microrobot velocity during effective stroke increases with an increase in the radius of cilia. This rise in velocity results from increasing tangential and normal resistance coefficients for larger sizes of the cilium radius.

On the other hand, when getting thicker cilia become stiffer and exhibit a higher resistance in bending. Thus, they bend less with a certain amount of moment and provoke higher resistance to the fluid. Therefore, the mean speed of microrobot decreases after a maximum value of about 70 nm of the cilium radius (Fig. 4.5b). With a given driving moment, a relatively small bending stiffness causes a considerable curvature of cilia during recovery stroke. Hence, cilia cannot have a

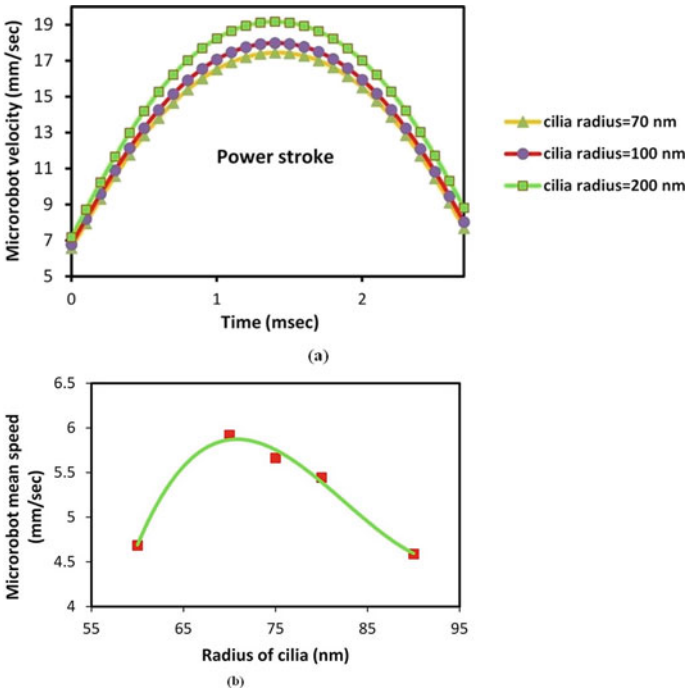


Fig. 4.5 (**a**) Velocity of the microrobot during effective stroke for different radii of cilia as a function of time. (**b**) The mean speed of the microrobot for various sizes of cilia radius

large radius. By testing different radiuses of cilia, when cilia get thicker than 150 nm in radius, the motor is not able to bend them. It is also noteworthy that a low modulus of elasticity, for instance on the order of MPa, does not help considerably, instead, cilia dimension affects its curvature significantly.

The microrobot velocity is shown for different cilia lengths in Fig. 4.6a. Since a longer cilium produces higher propulsive force during the effective stroke, the propulsive velocity increases with an increase in the length. Figure 4.6a depicts a ciliary microswimmer with a cilium length of 50, 70, and 100 μm achieves a maximum speed of 9.46, 15.84, and 25.29 mm/s, respectively.

There is an optimum length of cilia for generating propulsive force and afterward, longer cilia contribute to higher resistive force during recovery stroke and cause a decreasing average speed for the microrobot. The maximum average speed that is attainable by the microrobot is 7.78 mm/s for a 94.6 μm length of cilia (Fig. 4.6b).

A larger number of artificial cilia contribute to a higher propulsive force. Figure 4.7a, b shows how the microrobot maximum velocity and maximum mean speed change with an increase in cilia number. To a number of 50 cilia, the average speed rises sharply, and beyond that multiplying cilia number no longer affects the average speed significantly.

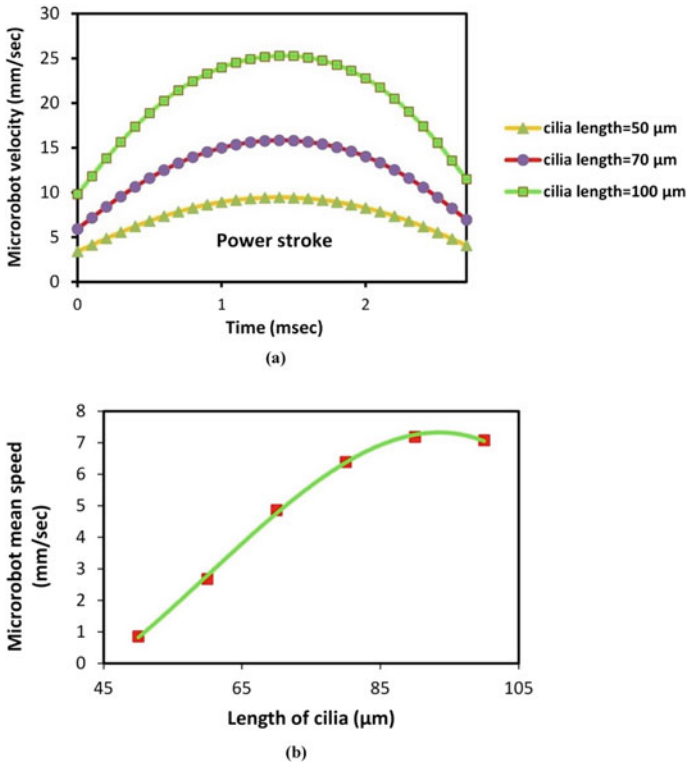


Fig. 4.6 (**a**) Velocity of the microrobot during effective stroke for different cilia lengths as a function of time. (**b**) The mean speed of the microrobot for various lengths of cilia

The viscosity of the surrounding fluid affects the microrobot movement. A more viscous fluid leads to higher resistive coefficients and drag forces and consequently a higher propulsive force. On the other hand, the resistance against the body also increases with increasing viscosity of the fluid. Figure 4.8 shows the velocity of microrobot in a fluid with a viscosity of water, and as well twice and four times this viscosity. The curve depicts that the positive and negative roles of increasing viscosity to the motion of the microswimmer are in equilibrium at first; however, as cilia incline from right angle the velocity suddenly drops.

Figure 4.9 shows how the microrobot mean speed varies with changes in the radius of the body. Although Eq. (4.35) is linear with respect to the body radius, since we have considered the effect of body's speed on the cilia velocity, the propulsive force depends on the body velocity and consequently on its radius, and the relationship between the velocity and radius of the body is not linear.

We have estimated the ciliary microrobot velocity on the order of few millimeters per second, which is higher than the maximum speed, 340 μm/s, of the ciliary microrobot developed by Kim et al. in [26]. Thinner cilia that we have designed

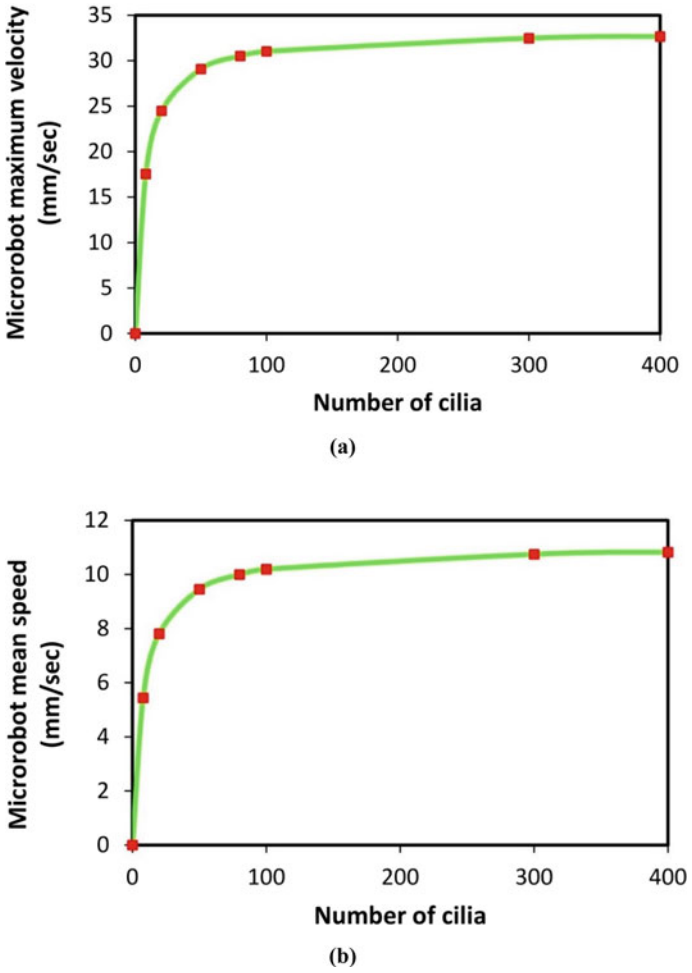


Fig. 4.7 (**a**) Maximum velocity of the microrobot during effective stroke based on the number of cilia. (**b**) The microrobot mean speed for different cilia numbers

in our model contribute to this difference between velocities. Also, we have not considered the surface friction on the microrobot, which can be an adverse factor against the microrobot propulsion. However, the main reason for the difference in propulsion is the way we actuate cilia. Although, with any actuation of cilia, the ciliary microrobot might have an amount of propulsion, the optimum velocity is obtained when cilia are stimulated to follow a biomimicking pattern during effective and recovery strokes, like we have considered in our design.

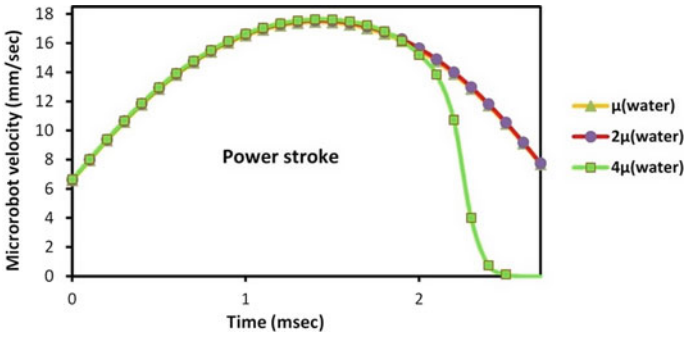
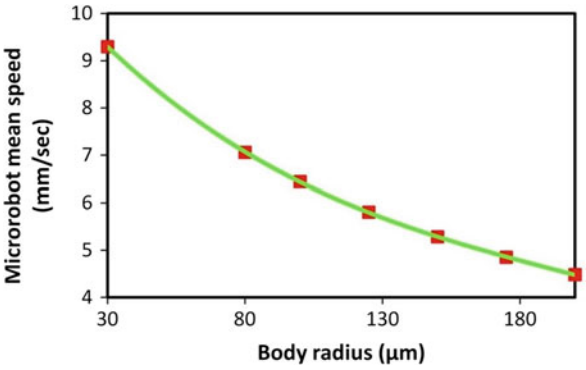


Fig. 4.8 The velocity of the microrobot in a fluid with a viscosity of μ_{water}, $2\mu_{\text{water}}$, and $4\mu_{\text{water}}$ during the effective stroke

Fig. 4.9 The mean speed of the microrobot as a function of radius of the body



4.3 Magnetic Actuation of Cilia

Magnetic actuation of artificial cilia transfers an enormous amount of power wirelessly. In addition, the availability of magnetic field generators such as MRI devices facilitates the application of magnetic powering and control. Magnetic stimulation has shown promise for application in micro-/nanorobotics [4, 13, 30–32]. However, modeling magnetic fields involve nonlinear methods that need to be addressed.

Artificial cilia, with a typical length of few micrometers and a diameter of few nanometers to one micrometer, are usually composite polymeric nanofilaments that have magnetic particles in their context. By molding polymeric materials in which the iron oxide nanoparticles are dispersed, it is possible to fabricate arrays of nanofilaments, and then dissolve and remove the mold [33, 34]. Another technique is to fabricate nanorods using a 3D lithography method, such as laser lithography, and coat them with a magnetic nanolayer [26].

To fabricate magnetic artificial cilia, we can use spherical magnetic nanoparticles dispersed in a polymeric matrix. We assume artificial cilia are partially magnetized

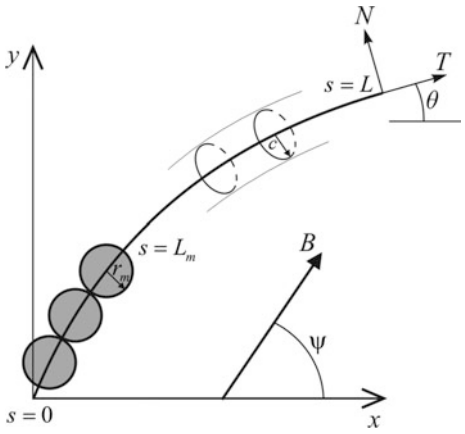


Fig. 4.10 Schematic representation of a cilium with magnetic nanoparticles. *xy* and *TN* represent the global and local coordinate systems, respectively. The cilium has magnetic particles attached together from s=0 to $s = L_{\mathrm{m}}$. B is the applied magnetic field with an angle of ψ. Reprinted with permission from [5]

using an arrangement of magnetic particles along a section of their centerline (Fig. 4.10). The magnetic nanoparticles are aligned coherently from the basal end of cilia to a fraction of the cilia length (Fig. 4.10). To solve the hydrodynamics of cilia motion, we need to find the model for magnetic actuation torque to use in Eq. (4.17).

Particles are superparamagnetic and when exposed to an external magnetic field achieve a magnetic dipole moment [35]:

$$\mathbf{m} = \frac{4\pi\ {r_{\mathrm{m}}}^3}{3\mu_0}\boldsymbol{\chi} \cdot \mathbf{B} \tag{4.39}$$

where $\mu_0 = 4\pi\ \times 10^{-7}\ \mathrm{H\ m^{-1}}$ is the magnetic permeability of free space and $\boldsymbol{\chi}$ is the magnetic susceptibility.

We consider the magnetic field without any spatial gradient; hence, a pure magnetic torque and no magnetic force is applied to the cilia. This torque actuates the artificial cilia and bends them during recovery stroke. Applying a magnetic field, paramagnetic particles align themselves to the magnetic field with their maximal magnetization directions along the cilium length. To account for the shape anisotropy of particle magnetization, two tangential and normal susceptibilities, χ_{T} and χ_{N}, are assumed for cilia.

The adjacent magnetic dipoles interact with the magnetic field of a particle. A single particle in the chain with a dipole moment of $\mathbf{m} = m_{\mathrm{T}}\hat{T} + m_{\mathrm{N}}\hat{N}$ generates a magnetic field, which is perceived by neighboring particles. At a distance r, the field is given by:

$$\mathbf{B}_{\mathrm{dip}} = \mu_0 \frac{\left(2m_{\mathrm{T}}\hat{T} - m_{\mathrm{N}}\hat{N}\right)}{4\pi\ r^3} \tag{4.40}$$

Since magnetic field drops rapidly with increasing distance, the interaction of a particle magnetic dipole is solely considered on its two adjacent particles. By taking

into account this field and the magnetic field, we reach the following expressions for tangential and normal magnetizations [35]:

$$m_{\mathrm{T}} = \frac{\frac{4}{3}\pi\ {r_{\mathrm{m}}}^3 \chi_{\mathrm{T}} B_{\mathrm{T}}}{\mu_0\,(1 - \chi_{\mathrm{T}}/6)} \tag{4.41}$$

$$m_{\mathrm{N}} = \frac{\frac{4}{3}\pi\ {r_{\mathrm{m}}}^3 \chi_{\mathrm{N}} B_{\mathrm{N}}}{\mu_0\,(1 + \chi_{\mathrm{N}}/12)} \tag{4.42}$$

where B_{T} and B_{N} are the tangential and normal components of the applied magnetic field. The cross product between particle magnetization and the magnetic field gives the magnetic torque per unit length of the cilium:

$$T_{\mathrm{A}_{\mathrm{magnetic}}} = \frac{\pi\ {r_{\mathrm{m}}}^2 |\mathbf{B}|^2}{3\mu_0} \left(\frac{\chi_{\mathrm{T}} - \chi_{\mathrm{N}} + \chi_{\mathrm{T}}\chi_{\mathrm{N}}/4}{(1 - \chi_{\mathrm{T}}/6)\,(1 + \chi_{\mathrm{T}}/12)} \right) \sin\left(2\,(\psi - \alpha)\right) \tag{4.43}$$

where ψ is the angle that the magnetic field makes with the x-axis. We rewrite Eq. (4.43) in a new form as:

$$T_{\mathrm{A}_{\mathrm{magnetic}}} = T_{\mathrm{A}_0} \sin\left(2\,(\psi - \theta)\right) \tag{4.44}$$

where T_{A_0} is given by:

$$T_{\mathrm{A}_0} = \frac{\pi\ {r_{\mathrm{m}}}^2 |\mathbf{B}|^2}{3\mu_0} \left(\frac{\chi_{\mathrm{T}} - \chi_{\mathrm{N}} + \chi_{\mathrm{T}}\chi_{\mathrm{T}}/4}{(1 - \chi_{\mathrm{T}}/6)\,(1 + \chi_{\mathrm{T}}/12)} \right) \tag{4.45}$$

We design kinematics of the magnetic field, i.e., magnitude and direction, so that (1) cilia motion has two effective and recovery stages; (2) cilia are straight and beat on the fluid during the effective stroke; and (3) cilia bend during recovery stroke and remain parallel to the body surface.

The initial condition for cilia is $\theta(s, 0) = 150°$. During the effective stroke, a magnetic field is designed to have a magnitude of $|\mathbf{B}| = 70\,\mathrm{mT}$ and an angle of $\alpha(0, t) - \pi/4$. The magnetic field direction aims to maximize the sine function in Eq. (4.44), and, therefore, the exerted magnetic torque. Using this scheme, the magnetic field rotates from 105° to about 0° throughout the effective stroke, which lasts 7.5 ms.

We apply two different magnetic fields in horizontal and vertical directions during recovery stroke. A 90 mT constant magnetic field in x-direction, $\mathbf{B}_x = 90\,\mathrm{mT}$, and a magnetic field in y-direction with a sine function, $\mathbf{B}_y = 0.7 \sin(10t)$, are applied on cilia for a period of 2.5 ms from the beginning of the recovery stroke. Then, the magnetic fields are turned off until the end of the recovery stroke, which lasts 13.2 ms. The time interval for a full beating cycle is 20.7 ms.

We use partial magnetization of cilia to induce a large bending during the recovery stroke. From basal end ($s = 0$), we magnetize cilia up to $L_m = 0.4L$. This reduces the energy threshold required to provoke bending in cilia and make cilia prone to a large bending.

A nonlinear coupled magnetic–fluidic–elastic problem is formed by combining cilia hydrodynamics, described in Sect. 4.2.2, and the magnetic actuation torque given by Eq. (4.43). This nonlinear system should be solved to find the microswimmer motion due to a magnetic stimulation of cilia.

4.3.1 Numerical Example

We solve a numerical example of magnetic actuation of the ciliary microrobot with parameters specified in Table 4.2. Figure 4.11 shows cilia beating sequences in effective and recovery strokes, which mimics natural cilia beating. Solving nonlinear equations of motion, we could design kinematics of a magnetic field that generates a biomimicking beating pattern for artificial cilia. The designed magnetic moment has approximately a maximum value throughout the effective stroke. In the first steps of the recovery stroke when a large bending is required, a magnetic moment is applied to the cilia. Thereafter, for the rest of the recovery stroke, the magnetic moment is

Table 4.2 Parameters of the ciliary microrobot with magnetic cilia

Design parameter	Value
Cilium length, L_{cilia}	75 μm
Cilium radius, c_{cilia}	75 nm
Magnetic particles radius, r_m	70 nm
Cilium Young's modulus, E	4 GPa
Body diameter, R_{body}	150 μm
Body length, L_{body}	220 μm
Number of cilia, n	8
Tangential susceptibility, χ_T	4.5
Normal susceptibility, χ_N	1
Water viscosity, μ_{water}	0.001 kg/(m·s)

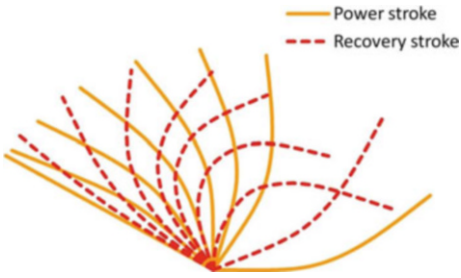


Fig. 4.11 Beating cycle of cilia subjected to a magnetic actuation in effective and recovery strokes

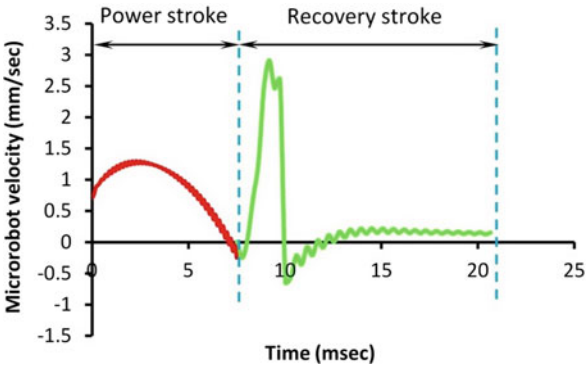


Fig. 4.12 Microrobot velocity with magnetic actuated cilia for a beating cycle

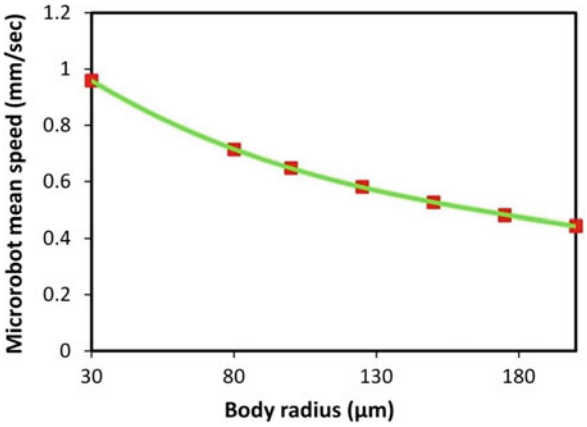


Fig. 4.13 Microrobot mean speed with magnetically actuated cilia as a function of the radius of the microrobot body

not exerted until the end where cilia open up due to their elasticity to commence the next cycle.

We consider the motion of the ciliary microswimmer, discussed in Sect. 4.2.3, possessing magnetic artificial cilia under a magnetic field with the above kinematics. Figure 4.12 shows the microrobot velocity for one cycle of cilia beating. The microrobot velocity increases when cilia start beating to the upright position; thereafter, the microrobot velocity decreases. By further continuing to the recovery stroke, the swimmer velocity increases in the reverse direction until actuation is stopped, where we see a discontinuity in the microrobot velocity. Then, velocity remains close to zero till the cycle terminates.

The mean speed is obtained by dividing the area under the velocity–time curve in Fig. 4.12 by the cycle time. Figure 4.13 shows the microrobot average speed for different radii of the microrobot body.

4.4 Magnetic Actuation Modeling

In the previous section, we observed the ciliary microrobot uses magnetic actuation strategy for propulsion. Here, we establish a nonlinear model of magnetic fields and gradients of a set of electromagnetic coils that can be used for the actuation of microrobots.

A microrobot placed in a pure magnetic field, with no gradient, bears a magnetic torque only:

$$\mathbf{T}_{\mathrm{m}} = \mathbf{m} \times \mathbf{B} \tag{4.46}$$

where $\mathbf{m}$ is the magnetization vector of the microrobot and $\mathbf{B}$ is the magnetic flux density. However, applying magnetic fields with a spatial gradient exerts a magnetic force on the microrobot:

$$\mathbf{F}_{\mathrm{m}} = (\mathbf{m} \cdot \nabla)\, \mathbf{B} \tag{4.47}$$

We can write Eqs. (4.46) and (4.47) in a matrix form:

$$\mathbf{T}_{\mathrm{m}} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ m_x & m_y & m_z \\ B_x & B_y & B_z \end{vmatrix} \tag{4.48}$$

$$\mathbf{F}_{\mathrm{m}} = \begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_y}{\partial x} & \frac{\partial B_z}{\partial x} \\ \frac{\partial B_x}{\partial y} & \frac{\partial B_y}{\partial y} & \frac{\partial B_z}{\partial y} \\ \frac{\partial B_x}{\partial z} & \frac{\partial B_y}{\partial z} & \frac{\partial B_z}{\partial z} \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{4.49}$$

where $\frac{\partial B_k}{\partial n}, k = x, y, z, n = x, y, z$ is the partial derivative of the magnetic field component in k-direction with respect to the spatial variable, n. The components of magnetization in x-, y- and z-directions are indicated by m_x, m_y, and m_z, respectively. The $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the unit vectors along the x-, y- and z-axes, respectively.

An electromagnetic coil can be assumed as a stack of N loops. For one loop carrying current I (Fig. 4.14), terms in Eqs. (4.48) and (4.49) can be evaluated at a point $P(x, y, z)$ outside the loop and sufficiently far [36, 37]:

$$B_x = \frac{\mu_0 I}{\pi} \frac{xz}{2\alpha^2 \beta \rho^2} \left[\left(a^2 + r^2\right) E\left(\kappa^2\right) - \alpha^2 K\left(\kappa^2\right) \right] \tag{4.50}$$

$$B_y = \frac{\mu_0 I}{\pi} \frac{yz}{2\alpha^2 \beta \rho^2} \left[\left(a^2 + r^2\right) E\left(\kappa^2\right) - \alpha^2 K\left(\kappa^2\right) \right] \tag{4.51}$$

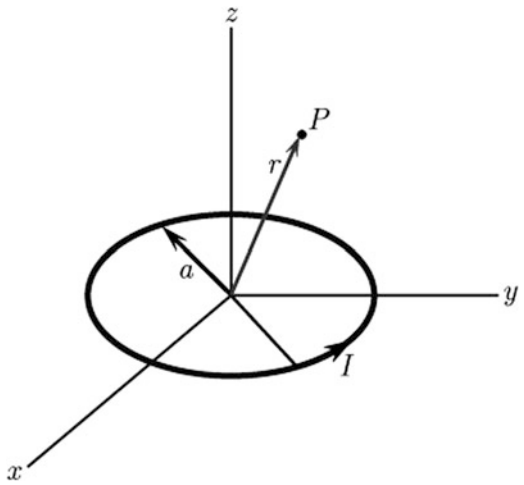


Fig. 4.14 A circular loop of radius a carrying current I. The magnetic field is evaluated at point $P(x, y, z)$

$$B_z = \frac{\mu_0 I}{\pi} \frac{1}{2\alpha^2 \beta} \left[\left(a^2 - r^2 \right) E\left(\kappa^2 \right) + \alpha^2 K\left(\kappa^2 \right) \right] \tag{4.52}$$

where

$$\rho = \sqrt{x^2 + y^2} \tag{4.53}$$

$$r = \sqrt{x^2 + y^2 + z^2} \tag{4.54}$$

$$\alpha = \sqrt{a^2 + r^2 - 2a\rho} \tag{4.55}$$

$$\beta = \sqrt{a^2 + r^2 + 2a\rho} \tag{4.56}$$

$$\kappa^2 = 1 - \frac{\alpha^2}{\beta^2} \tag{4.57}$$

and

$$\gamma = x^2 - y^2 \tag{4.58}$$

The derivatives of the magnetic field are given by:

$$\begin{aligned}\frac{\partial B_x}{\partial x} = \frac{\mu_0 I}{\pi}\frac{z}{2\alpha^4\beta^3\rho^4}\left\{\left[a^4\left(-\gamma\left(3z^2+a^2\right)+\rho^2\left(8x^2-y^2\right)\right)\right.\right.\\ -a^2\left(\rho^4\left(5x^2+y^2\right)-2\rho^2z^2\left(2x^2+y^2\right)+3z^4\gamma\right)-r^4\left(2x^4+\gamma\left(y^2+z^2\right)\right)\left.\right]E\left(\kappa^2\right)\\ \left.+\alpha^2\left[a^2\left(\gamma\left(a^2+2z^2\right)-\rho^2\left(3x^2-2y^2\right)\right)+r^2\left(2x^4+\gamma\left(y^2+z^2\right)\right)\right]K\left(\kappa^2\right)\right\}\end{aligned} \tag{4.59}$$

$$\begin{aligned}\frac{\partial B_x}{\partial y} = \frac{\mu_0 I}{\pi}\frac{xyz}{2\alpha^4\beta^3\rho^4}\left\{\left[3a^4\left(3\rho^2-2z^2\right)-r^4\left(2r^2+\rho^2\right)\right.\right.\\ \left.-2a^6-2a^2\left(2\rho^4-\rho^2z^2+3z^4\right)\right]E\left(\kappa^2\right)\\ \left.+\alpha^2\left[r^2\left(2r^2+\rho^2\right)-a^2\left(5\rho^2-4z^2\right)+2a^4\right]K\left(\kappa^2\right)\right\}\end{aligned} \tag{4.60}$$

$$\begin{aligned}\frac{\partial B_x}{\partial z} = \frac{\mu_0 I}{\pi}\frac{x}{2\alpha^4\beta^3\rho^4}\left\{\left[\left(\rho^2-a^2\right)^2\left(\rho^2+a^2\right)\right.\right.\\ \left.+2z^2\left(a^4-6a^2\rho^2+\rho^4\right)+z^4\left(a^2+\rho^2\right)\right]E\left(\kappa^2\right)\\ \left.-\alpha^2\left[\left(\rho^2-a^2\right)^2+z^2\left(\rho^2+a^2\right)\right]K\left(\kappa^2\right)\right\}\end{aligned} \tag{4.61}$$

$$\frac{\partial B_y}{\partial x} = \frac{\partial B_x}{\partial y} \tag{4.62}$$

$$\begin{aligned}\frac{\partial B_y}{\partial y} = \frac{\mu_0 I}{\pi}\frac{z}{2\alpha^4\beta^3\rho^4}\left\{\left[a^4\left(\gamma\left(3z^2+a^2\right)+\rho^2\left(8y^2-x^2\right)\right)\right.\right.\\ -a^2\left(\rho^4\left(5y^2+x^2\right)-2\rho^2z^2\left(2y^2+x^2\right)+3z^4\gamma\right)-r^4\left(2y^4-\gamma\left(x^2+z^2\right)\right)\left.\right]E\left(\kappa^2\right)\\ \left.+\alpha^2\left[a^2\left(-\gamma\left(a^2+2z^2\right)-\rho^2\left(3y^2-2x^2\right)\right)+r^2\left(2y^4-\gamma\left(x^2+z^2\right)\right)\right]K\left(\kappa^2\right)\right\}\end{aligned} \tag{4.63}$$

$$\frac{\partial B_y}{\partial z} = \frac{y}{x}\frac{\partial B_x}{\partial z} \tag{4.64}$$

$$\frac{\partial B_z}{\partial x} = \frac{\partial B_x}{\partial z} \tag{4.65}$$

$$\frac{\partial B_z}{\partial y} = \frac{\partial B_y}{\partial z} \tag{4.66}$$

and

$$\frac{\partial B_z}{\partial z} = \frac{\mu_0 I}{\pi}\frac{z}{2\alpha^4\beta^3}\left\{\left[6a^2\left(\rho^2-z^2\right)-7a^4+r^4\right]E\left(\kappa^2\right)+\alpha^2\left[a^2-r^2\right]K\left(\kappa^2\right)\right\} \tag{4.67}$$

K and E are complete elliptic integrals of the first and second kind, respectively. From Eqs. (4.50) to (4.67), we deduce that the magnetic field and gradients in the vicinity of an electromagnetic coil are proportional to the current passing through the coil:

$$\mathbf{B}\,(\mathbf{P}) = B\,(\mathbf{P})\,I \tag{4.68}$$

$$\frac{\mathbf{dB}}{\mathbf{dP}} = B_{\mathrm{p}}\,(\mathbf{P})\,I \tag{4.69}$$

where **P** is the position at which the magnetic field and gradient areevaluated.

Matrices in (4.68) and (4.69) for a set of electromagnetic coils are obtained by summation of the magnetic field and gradient of each coil:

$$B\,(\mathbf{P}) = \sum_{i=1}^{M} B_i\,(\mathbf{P})\,I_i \tag{4.70}$$

$$B_{\mathrm{p}}\,(\mathbf{P}) = \sum_{i=1}^{M} B_{\mathrm{p}i}\,(\mathbf{P})\,I_i \tag{4.71}$$

where M is the number of coils in the set.

We evaluate the magnetic field and gradients of a set of six electromagnetic coils which have been positioned in pairs along x-, y- and z-directions on both sides of a workspace (Fig. 4.15). A 5 mm × 5 mm × 5 mm workspace has been considered in the center. We assume two coils in each direction to be at 20 mm apart from each other. Each coil has a radius of 6 mm and 100 turns. The magnetic field is shown in different xy planes along the z-axis (Fig. 4.16). Magnetic gradients are also shown in Fig. 4.17.

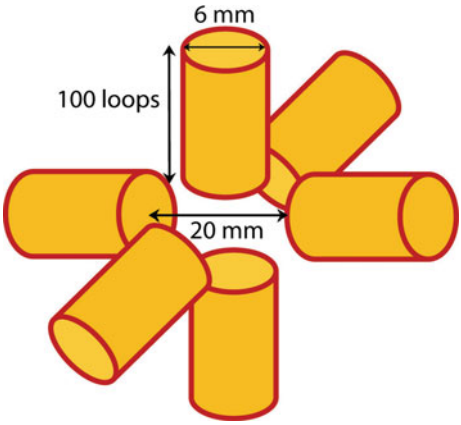


Fig. 4.15 A set of six electromagnetic coils placed along three perpendicular axes surrounding a workspace. Each coil has 100 turns and a radius of 6 mm. Two coils in each direction are at a 20 mm distance

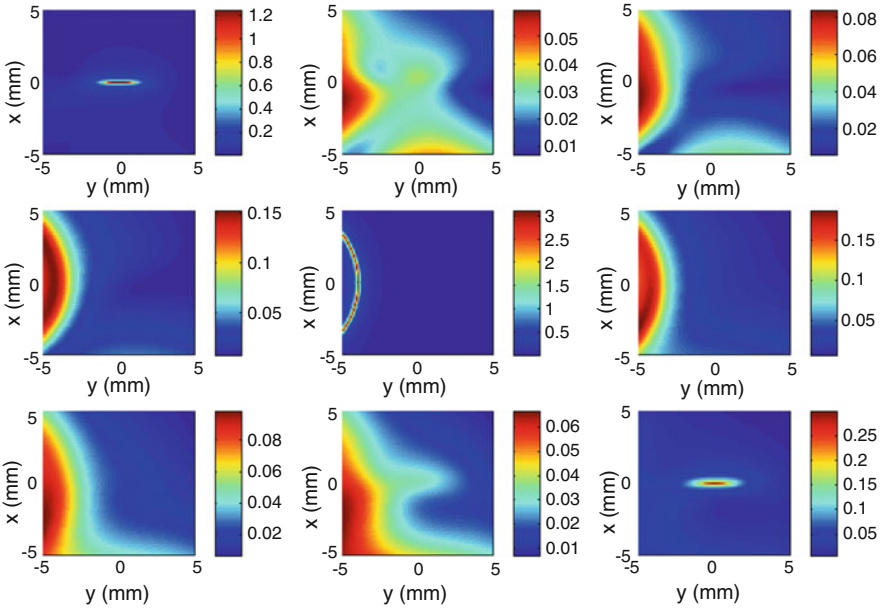


Fig. 4.16 Magnetic field strength (Tesla) in *xy* planes of a 5mm×5mm×5mm workspace along the *z*-axis at (**a**) $z = -4$, (**b**) $z = -3$, (**c**) $z = -2$, (**d**) $z = -1$, (**e**) $z = 0$, (**f**) $z = 1$, (**g**) $z = 2$, (**h**) $z = 3$, and (**i**) $z = 4$ mm, shown sequentially, for a set of coils current of $I = [1\ \ 2\ \ 4\ \ 4\ \ 2\ \ 1]$

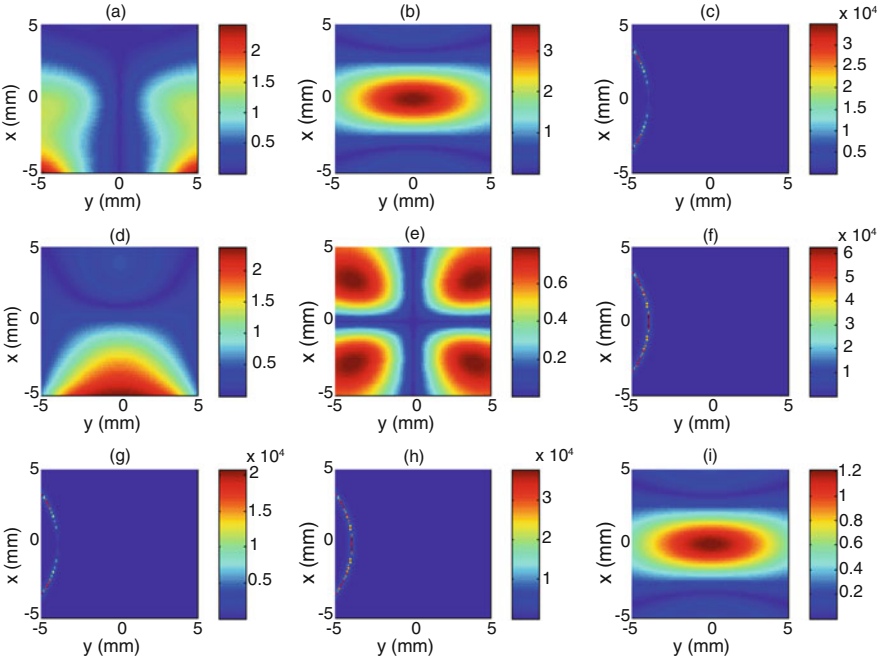


Fig. 4.17 Magnetic field gradient components (Tesla/m), $\frac{\partial B_x}{\partial x}$, $\frac{\partial B_x}{\partial y}$, $\frac{\partial B_x}{\partial z}$, $\frac{\partial B_y}{\partial x}$, $\frac{\partial B_y}{\partial y}$, $\frac{\partial B_y}{\partial z}$ $\frac{\partial B_y}{\partial z}$, $\frac{\partial B_z}{\partial x}$, $\frac{\partial B_z}{\partial y}$, and $\frac{\partial B_z}{\partial z}$ shown sequentially from (**a**) to (**i**), in $z = 0$ plane for a set of coils current of $I = [1\ \ 2\ \ 4\ \ 4\ \ 2\ \ 1]$

4.5 Conclusion

Implementation, control, and application of micro-/nanorobotic systems for biomedical applications can be enhanced using a full understanding of the system dynamics. However, dynamic modeling at micro-/nanoscale usually leads to nonlinear equations of motion. In this chapter, we described nonlinear modeling application in microrobotics. Hydrodynamic modeling of a ciliary swimming microrobot and magnetic actuation of cilia for propulsion of the robot provided examples of nonlinear modeling, which resulted in a nonlinear boundary value problem. We solved these nonlinear coupled differential equations under the boundary and initial conditions to obtain the microrobot propulsion velocity. In the last part in Sect. 4.4, an analytical model was developed to determine nonlinear magnetic forces and torques. This model can be used for a proper dynamic design and control of various micro-/nanosystems actuated by magnetic fields and gradients of a set of electromagnetic coils.

However, there are methods to consider nonlinearities as unknowns in the system dynamics, and solve the equations in order to control the micro-/nanosystems without any information of these undetermined parameters [36]. Nevertheless, nonlinear modeling enhances the accuracy of dynamic modeling and control of the micro-/nanorobots to achieve a desired performance.

References

1. Nelson, B. J., Kaliakatsos, I. K., & Abbott, J. J. (2010). Microrobots for minimally invasive medicine. *Annual Review of Biomedical Engineering, 12*, 55–85.
2. Sitti, M., Giltinan, J., & Yim, S. (2015). Biomedical applications of untethered mobile milli/microrobots. *Proceedings of the IEEE, 103*, 205–224.
3. Ceylan, H., Giltinan, J., Kozielskia, K., & Sitti, M. (2017). Mobile microrobots for bioengineering applications. *Lab on a Chip, 17*, 1705–1724.
4. Chen, X. Z., Hoop, M., Mushtaq, F., Siringil, E., Hu, C., Nelson, B. J., & Pané, S. (2017). Recent developments in magnetically driven micro- and nanorobots. *Applied Materials Today, 9*, 37–48.
5. Ghanbari, A., Bahrami, M., & Nobari, M. R. H. (2011). Methodology for artificial microswimming using magnetic actuation. *Physical Review E, 83*, 046301.
6. Ghanbari, A., & Bahrami, M. (2011). A novel swimming microrobot based on artificial cilia for biomedical applications. *Journal of Intelligent and Robotic Systems, 63*, 399–416.
7. Zhang, L., Abbott, J. J., Dong, L., Kratochvil, B. E., Bell, D., & Nelson, B. J. (2009). Artificial bacterial flagella: fabrication and magnetic control. *Applied Physics Letters, 94*, 064107.
8. Abbott, J. J., Peyer, K. E., Lagomarsino, M. C., Zhang, L., Dong, L., Kaliakatsos, I. K., & Nelson, B. J. (2009). How should microrobots swim? *International Journal of Robotics Research, 28*, 1434–1447.
9. Huang, H., Chao, Q., Sakar, M. S., & Nelson, B. J. (2017). Optimization of tail geometry for the propulsion of soft microrobots. *IEEE Robotics and Automation Letters, 2*, 727–732.
10. Martel, S., Mohammadi, M., Felfoul, O., Lu, Z., & Pouponneau, P. (2009). Flagellated magnetotactic bacteria as controlled MRI-trackable propulsion and steering systems for medical nanorobots operating in the human microvasculature. *International Journal of Robotics*

Research, 28, 571–582.
11. Martel, S., Felfoul, O., Mathieu, J., Chanu, A., Tamaz, S., Mohammadi, M., Mankiewicz, M., & Tabatabaei, N. (2009). MRI-based medical nanorobotic platform for the control of magnetic nanoparticles and flagellated bacteria for target interventions in human capillaries. *International Journal of Robotics Research, 28*, 1169–1182.
12. Kummer, M. P., Abbott, J. J., Kratochvil, B. E., Borer, R., Sengul, A., & Nelson, B. J. (2010). Octomag: An electromagnetic system for 5-DOF wireless micromanipulation. *IEEE Transactions on Robotics, 26*, 1006–1017.
13. Schuerle, S., Erni, S., Flink, M., Kratochvil, B. E., & Nelson, B. J. (2013). Three-dimensional magnetic manipulation of micro-and nanostructures for applications in life sciences. *IEEE Transactions on Magnetics, 49*, 321–330.
14. Zhang, Z., & Menq, C. H. (2011). Design and modeling of a 3-D magnetic actuator for magnetic microbead manipulation. *IEEE/ASME Transactions on Mechatronics, 16*, 421–430.
15. Grady, M. S., Howard, M. A., III, Molloy, J. A., Ritter, R. C., Quate, E. G., & Gillies, G. T. (1990). Nonlinear magnetic stereotaxis: three dimensional, *in vivo* remote magnetic manipulation of a small object in canine brain. *Medical Physics, 17*, 405–415.
16. Taylor, G. I. (1951). Analysis of the swimming of microscopic organisms. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, A209*, 447–461.
17. Gray, J., & Hancock, G. (1955). The propulsion of sea-urchin spermatozoa. *Journal of Experimental Biology, 32*, 802–814.
18. Brokaw, C. J. (1970). Bending moments in free-swimming flagella. *Journal of Experimental Biology, 53*, 445–464.
19. Gueron, S., & Liron, N. (1992). Ciliary motion modeling, and dynamic multicilia interactions. *Biophysical Journal, 63*, 1045–1058.
20. Childress, S. (1981). *Mechanics of swimming and flying*. New York: Cambridge University Press.
21. Lighthill, J. L. (1975) Mathematical biofluiddynamics. In *Regional Conference Series in Applied Mathematics, SIAM* (pp. 45–62).
22. Johnson, R. E., & Brokaw, C. J. (1979). Flagellar hydrodynamics: a comparison between resistive-force theory and slender-body theory. *Biophysical Journal, 25*, 113–127.
23. Gueron, S., & Levit-Gurevich, K. (1998). Computation of the internal forces in cilia: application to ciliary motion, the effects of viscosity, and ciliainteractions. *Biophysical Journal, 74*, 1658–1676.
24. Brennen, C., & Winet, H. (1977). Fluid mechanics of propulsion by cilia and flagella. *Annual Review of Fluid Mechanics, 9*, 339–398.
25. Feng, J., Joseph, D. D., Glowinski, R., & Pan, T. W. (1995). A three-dimensional computation of the force and torque on an ellipsoid settling slowly through a viscoelastic fluid. *Journal of Fluid Mechanics, 283*, 1–16.
26. Kim, S., Lee, S., Lee, J., Nelson, B. J., Zhang, L., & Choi, H. (2016). Fabrication and manipulation of ciliary microrobots with non-reciprocal magnetic actuation. *Scientific Reports, 6*, 30713.
27. Gibbons, I. R. (1981). Cilia and flagella of eukaryotes. *Journal of Cell Biology, 91*, 107s–124s.
28. Sleigh, M. A. (1962). *The biology of cilia and flagella*. Oxford: Pergamon Press.
29. Sleigh, M. A. (1968) Patterns of ciliary beating. In *Aspects of cell motility (22nd Symposium of the Society for Experimental Biology*) (pp. 131–150).
30. Peyer, K. E., Zhang, L., & Nelson, B. J. (2013). Bio-inspired magnetic swimming microrobots for biomedical applications. *Nanoscale, 5*, 1259–1272.
31. Kim, S., Qiu, F., Kim, S., Ghanbari, A., Moon, C., Zhang, L., Nelson, B. J., & Choi, H. (2013). Fabrication and characterization of magnetic microrobots for three-dimensional cell culture and targeted transportation. *Advanced Materials, 25*, 5863–5868.
32. Ghanbari, A., Chang, P. H., Nelson, B. J., & Choi, H. (2014). Electromagnetic steering of a magnetic cylindrical microrobot using optical feedback closed-loop control. *International Journal of Optomechatronics, 8*, 129–145.

33. Evans, B. A., Shields, A. R., Lloyd Carroll, R., Washburn, S., Falvo, M. R., & Superfine, R. (2007). Magnetically actuated nanorod arrays as biomimetic cilia. *Nano Letters, 7*, 1428–1434.
34. Goubault, C. (2003). Flexible magnetic filaments as micromechanical sensors. *Physical Review Letters, 91*, 260802.
35. Roper, M., Dreyfus, R., Baudry, J., Fermigier, M., Bibette, J., & Stone, H. A. (2006). On the dynamics of magnetically driven elastic filaments. *Journal of Fluid Mechanics, 554*, 167–190.
36. Ghanbari, A., Chang, P. H., Nelson, B. J., & Choi, H. (2014). Magnetic actuation of a cylindrical microrobot using time-delay-estimation closed-loop control: modeling and experiments. *Smart Materials and Structures, 23*, 035013.
37. Simpson, J., Lane, J., Immer, C., & Youngquist, R. (2001) Simple analytic expressions for the magnetic field of a circular current loop. *NASA Technical Reports* [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20010038494.pdf.

Chapter 5
The Nonlinear Pattern of Sea Levels: A Case Study of North America

Alberto Boretti

5.1 Introduction

Same as other climate-related parameters, the sea levels oscillate with well-known periodicities in the 60-year range [1, 2]. More than 60 years of continuous recording from the same tide gauge, without any major perturbation, is needed to compute a reliable slope by the linear fitting. More than 90 years are needed to compute a reliable acceleration by the parabolic fitting.

There are 20 long-term-trend (LTT) tide stations along the West Coast of North America and 33 along the East Coast.

The measured monthly average mean sea levels (MSL) relative to the tide gauge instrument are given by the Permanent Service for Mean Sea Level (PSMSL), www.psmsl.org. Analyses of these data are offered by PSMSL, sealevel.info, www.sealevel.info, National Oceanic and Atmospheric Administration (NOAA), tidesandcurrents.noaa.gov/sltrends/, Système d'Observation du Niveau des Eaux Littorales (SONEL), www.sonel.org (Fig. 5.1).

5.2 Method

Two regressions are usually applied to the measured relative sea levels of a tide gauge record to compute the relative sea level rate of rise and acceleration. A linear regression:

A. Boretti (✉)
Department for Management of Science and Technology Development, Faculty of Applied Sciences, Ton Duc Thang University, Ho Chi Minh City, Vietnam
e-mail: albertoboretti@tdtu.edu.vn

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_5

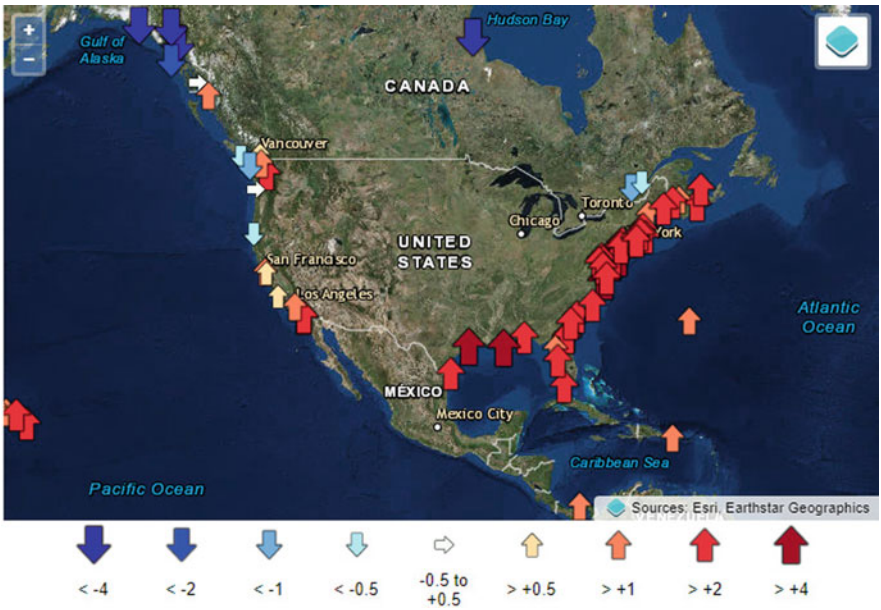


Fig. 5.1 Locations of the tide gauges with more than 80 years of data in the PSMSL database. Image reproduced modified after www.psmsl.org

$$y(x) = A + B \cdot x$$

returns the sea level rate of rise $\boldsymbol{u}$ as the slope B. A quadratic regression

$$y(x) = A' + B' \cdot x + C \cdot x^2$$

returns the acceleration $\boldsymbol{a}$ taken as 2·C.

Sea level rise forecasts to 2050 and 2100 may then be constructed by using the values of the conventional present sea level velocity $\boldsymbol{u}$ and acceleration $\boldsymbol{a}$.

$$\Delta y_1 = \boldsymbol{u} \cdot \Delta x$$

$$\Delta y_2 = \boldsymbol{u} \cdot \Delta x + \frac{\boldsymbol{a}}{2} \cdot n \cdot \Delta x$$

$$\Delta y_3 = \boldsymbol{u} \cdot \Delta x + \frac{\boldsymbol{a}}{2} \cdot n \cdot \Delta x + \frac{\boldsymbol{a}}{2} \cdot \Delta x \cdot \Delta x$$

where Δy is the sea level rise forecasted after a time Δx, and n is the length of the record used to compute $\boldsymbol{u}$ and $\boldsymbol{a}$. Δy_1 is a linear forecast, Δy_3 is a constant acceleration nonlinear forecast, and Δy_2 is an intermediate forecast.

5.3 Detailed Results

Here are the analyses of the relative rates of rise and accelerations of the sea level in the 20 long-term-trend (LTT) tide stations of the West Coast of North America and the longest 20 of the 33 LTT tide stations of the East Coast of North America.

5.3.1 Ketchikan, AK, USA

- The MSL trend at Ketchikan, AK, USA (Fig. 5.2) is −0.33 mm/year with a 95% confidence interval of ±0.22 mm/year, based on MSL data from 1919/1 to 2017/12.
- The acceleration is -0.01808 ± 0.01746 mm/year^2.

5.3.2 Sitka, AK, USA

- The MSL trend at Sitka, AK, USA (Fig. 5.3) is −2.33 mm/year with a 95% confidence interval of ±0.27 mm/year, based on MSL data from 1924/5 to 2017/12.
- The acceleration is -0.01631 ± 0.02273 mm/year^2.

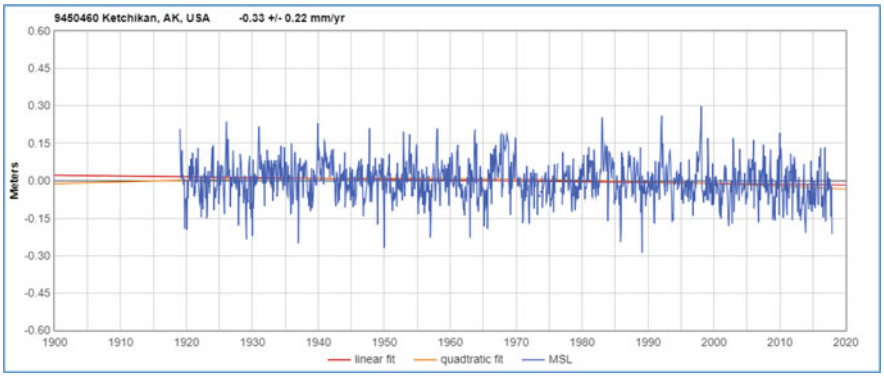


Fig. 5.2 MSL data for Ketchikan, AK, USA. Image reproduced modified after www.sealevel.info

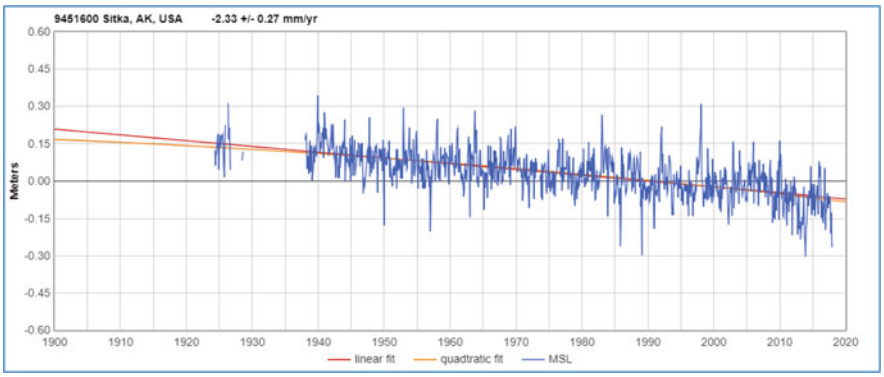


Fig. 5.3 MSL data for Sitka, AK, USA. Image reproduced modified after www.sealevel.info

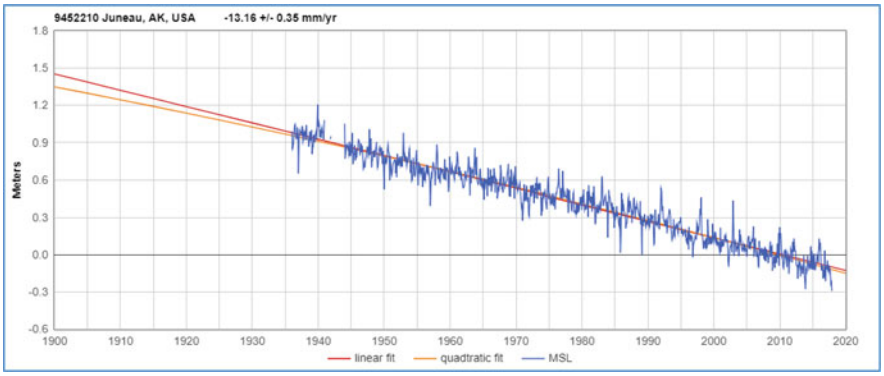


Fig. 5.4 MSL data for Juneau, AK, USA. Image reproduced modified after www.sealevel.info

5.3.3 Juneau, AK, USA

- The MSL trend at Juneau, AK, USA (Fig. 5.4) is −13.16 mm/year with a 95% confidence interval of ±0.35 mm/year, based on MSL data from 1936/1 to 2017/12.
- The acceleration is -0.0378 ± 0.0322 mm/year^2.

5.3.4 Unalaska, AK, USA

- The MSL trend at Unalaska, AK, USA (Fig. 5.5) is −4.14 mm/year with a 95% confidence interval of ±0.38 mm/year, based on MSL data from 1934/1 to 2017/12.
- The acceleration is -0.01453 ± 0.03414 mm/year^2.

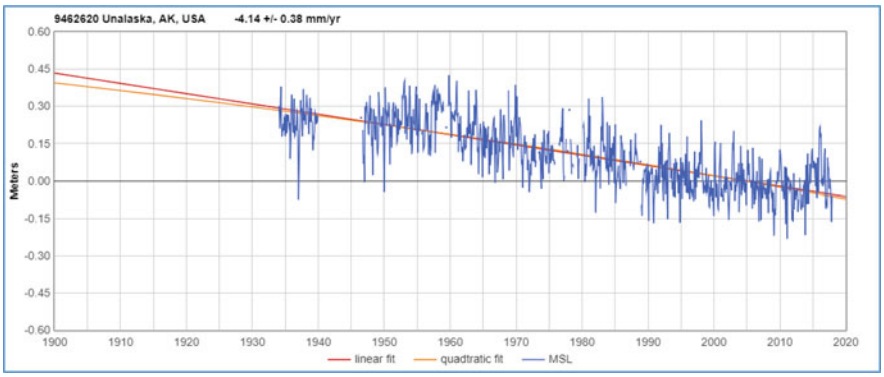


Fig. 5.5 MSL data for Unalaska, AK, USA. Image reproduced modified after www.sealevel.info

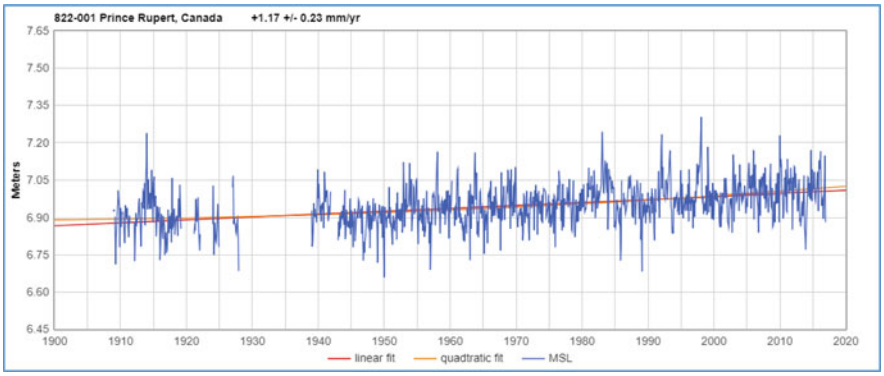


Fig. 5.6 MSL data for Prince Rupert, Canada. Image reproduced modified after www.sealevel.info

5.3.5 Prince Rupert, Canada

- The MSL trend at Prince Rupert, Canada (Fig. 5.6) is +1.17 mm/year with a 95% confidence interval of ±0.23 mm/year, based on MSL data from 1909/1 to 2016/12.
- The acceleration is 0.01484 ± 0.01463 mm/year^2.

5.3.6 Point Atkinson, Canada

- The MSL trend at Point Atkinson, Canada (Fig. 5.7) is +0.95 mm/year with a 95% confidence interval of ±0.24 mm/year, based on MSL data from 1914/5 to 2016/12.
- The acceleration is -0.00531 ± 0.01572 mm/year^2.

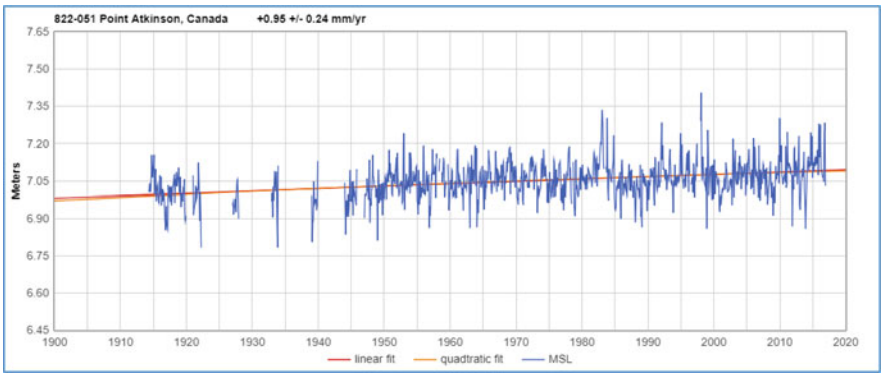


Fig. 5.7 MSL data for Point Atkinson, Canada. Image reproduced modified after www.sealevel.info

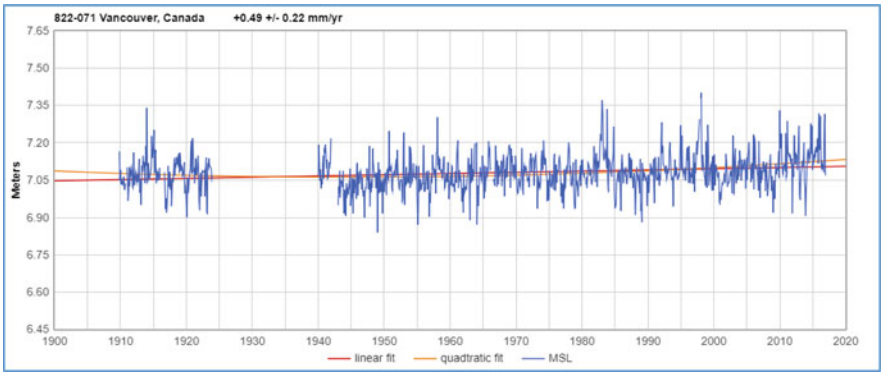


Fig. 5.8 MSL data for Vancouver, Canada. Image reproduced modified after www.sealevel.info

5.3.7 Vancouver, Canada

- The MSL trend at Vancouver, Canada (Fig. 5.8) is +0.49 mm/year with a 95% confidence interval of ±0.22 mm/year, based on MSL data from 1909/11 to 2016/12.
- The acceleration is 0.0251 ± 0.0141 mm/year^2.

5.3.8 Victoria, Canada

- The MSL trend at Victoria, Canada (Fig. 5.9) is +0.73 mm/year with a 95% confidence interval of ±0.19 mm/year, based on MSL data from 1909/3 to 2016/12.
- The acceleration is 0.00663 ± 0.01361 mm/year^2.

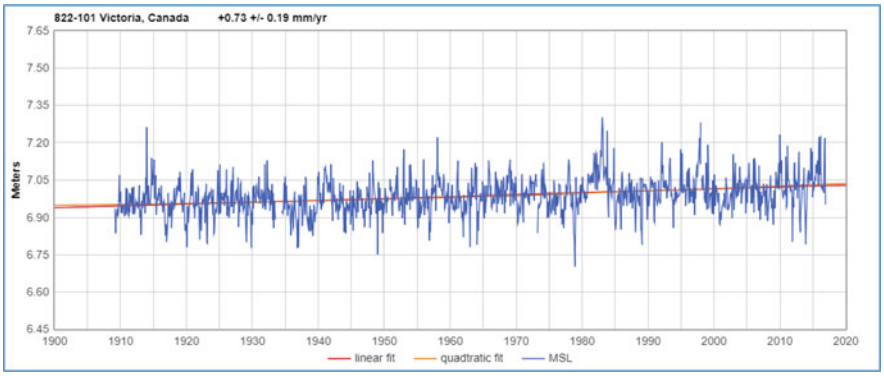


Fig. 5.9 MSL data for Victoria, Canada. Image reproduced modified after www.sealevel.info

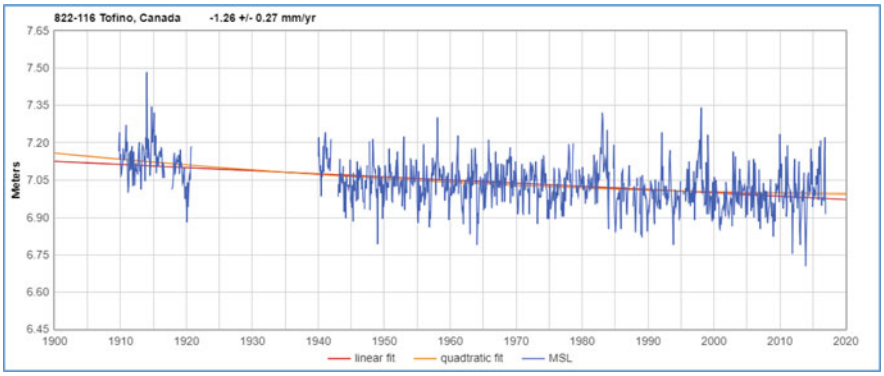


Fig. 5.10 MSL data for Tofino, Canada. Image reproduced modified after www.sealevel.info

5.3.9 Tofino, Canada

- The MSL trend at Tofino, Canada (Fig. 5.10) is −1.26 mm/year with a 95% confidence interval of ±0.27 mm/year, based on MSL data from 1909/10 to 2016/12.
- The acceleration is 0.01998 ± 0.01699 mm/year^2.

5.3.10 Friday Harbor, WA, USA

- The MSL trend at Friday Harbor, WA, USA (Fig. 5.11) is +1.20 mm/year with a 95% confidence interval of ±0.27 mm/year, based on MSL data from 1934/1 to 2017/12.
- The acceleration is 0.01018 ± 0.02539 mm/year^2.

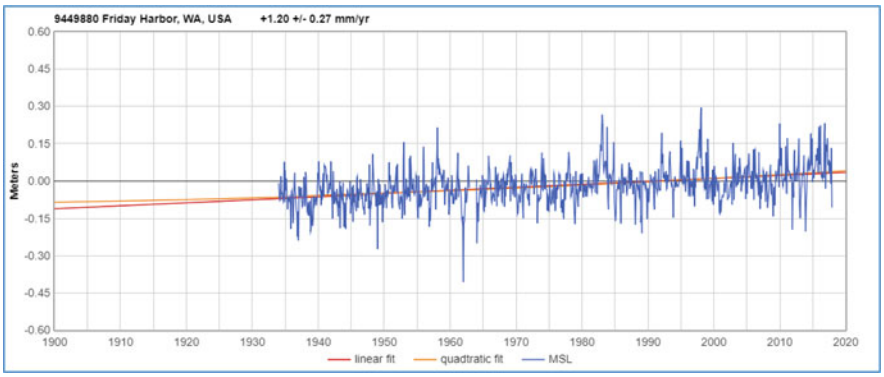


Fig. 5.11 MSL data for Friday Harbor, WA, USA. Image reproduced modified after www.sealevel.info

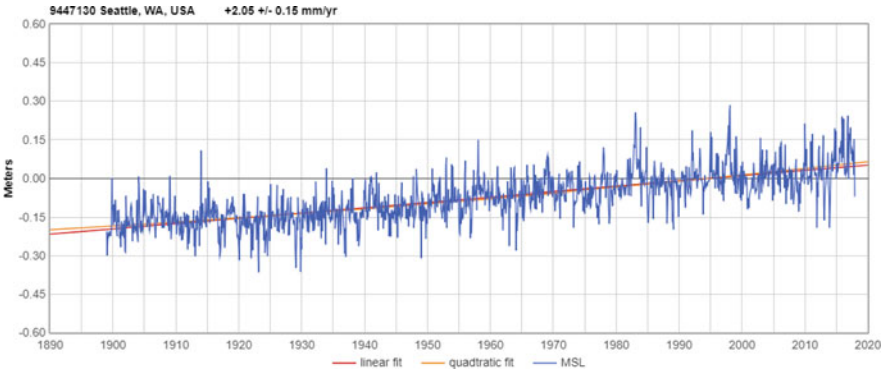


Fig. 5.12 MSL data for Seattle, WA, USA. Image reproduced modified after www.sealevel.info

5.3.11 Seattle, WA, USA

- The MSL trend at Seattle, WA, USA (Fig. 5.12) is +2.05 mm/year with a 95% confidence interval of ±0.15 mm/year, based on MSL data from 1899/1 to 2017/12.
- The acceleration is 0.00987 ± 0.00986 mm/year^2.

5.3.12 Neah Bay, WA, USA

- The MSL trend at Neah Bay, WA, USA (Fig. 5.13) is −1.69 mm/year with a 95% confidence interval of ±0.30 mm/year, based on MSL data from 1934/8 to 2017/12.
- The acceleration is −0.01619 ± 0.02760 mm/year^2.

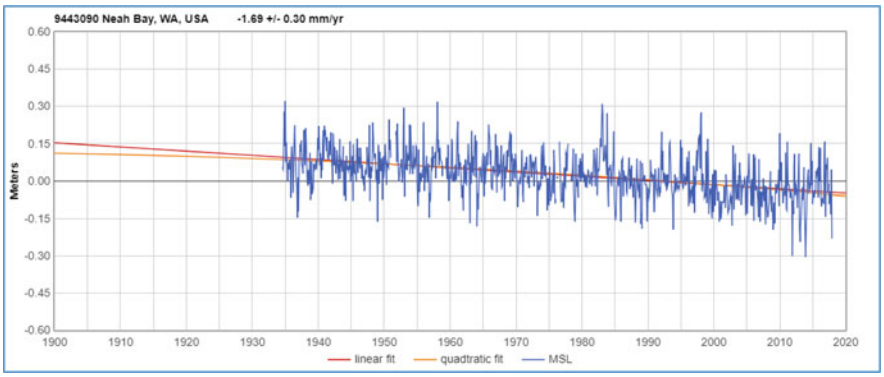


Fig. 5.13 MSL data for Neah Bay, WA, USA. Image reproduced modified after www.sealevel.info

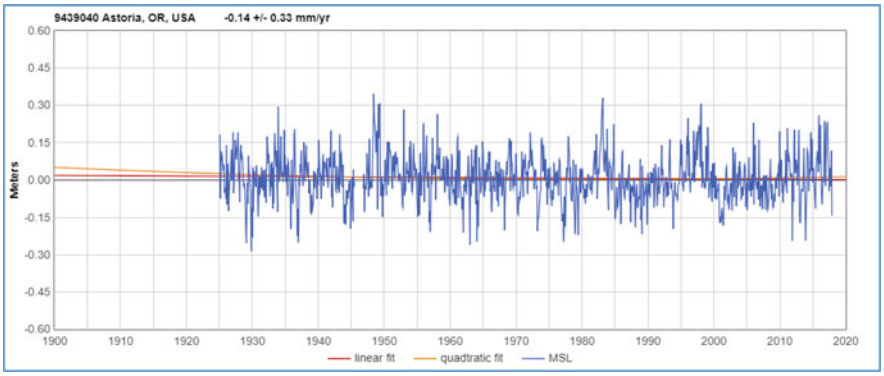


Fig. 5.14 MSL data for Astoria, OR, USA. Image reproduced modified after www.sealevel.info

5.3.13 Astoria, OR, USA

- The MSL trend at Astoria, OR, USA (Fig. 5.14) is −0.14 mm/year with a 95% confidence interval of ±0.33 mm/year, based on MSL data from 1925/2 to 2017/12.
- The acceleration is 0.01480 ± 0.02760 mm/year^2.

5.3.14 Crescent City, CA, USA

- The MSL trend at Crescent City, CA, USA (Fig. 5.15) is −0.78 mm/year with a 95% confidence interval of ±0.30 mm/year, based on MSL data from 1933/1 to 2017/12.
- The acceleration is -0.00857 ± 0.02672 mm/year^2.

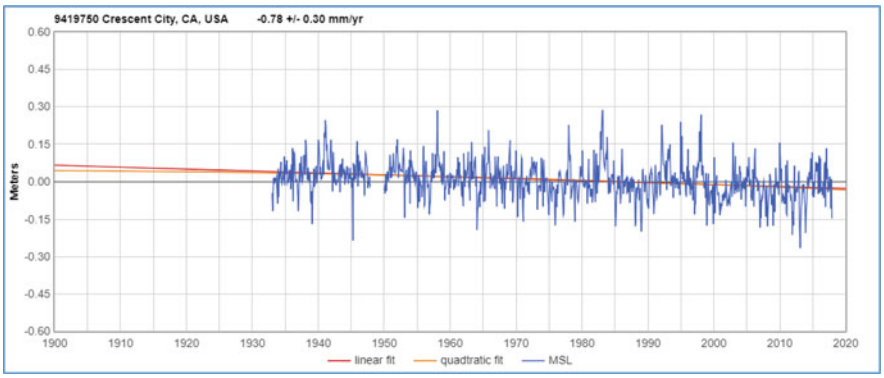


Fig. 5.15 MSL data for Crescent City, CA, USA. Image reproduced modified after www.sealevel.info

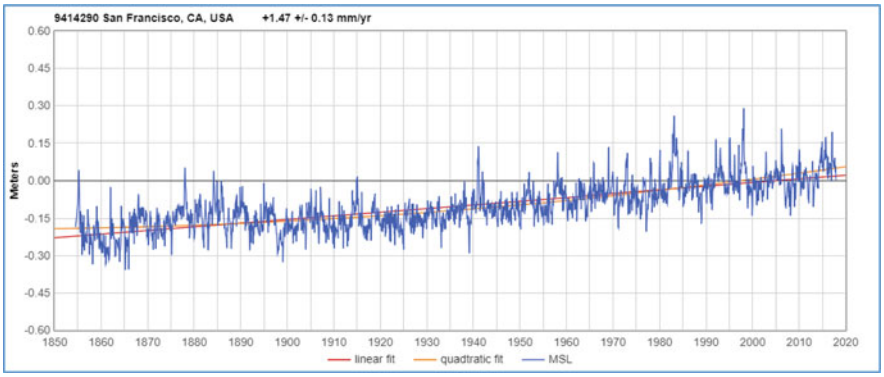


Fig. 5.16 MSL data for San Francisco, CA, USA. Image reproduced modified after www.sealevel.info

5.3.15 San Francisco, CA, USA

- The MSL trend at San Francisco, CA, USA (Fig. 5.16) is +1.47 mm/year with a 95% confidence interval of ±0.13 mm/year, based on MSL data from 1854/7 to 2017/12.
- The acceleration is 0.01406 ± 0.00619 mm/year^2.

5.3.16 Santa Monica, CA, USA

- The MSL trend at Santa Monica, CA, USA (Fig. 5.17) is +1.52 mm/year with a 95% confidence interval of ±0.33 mm/year, based on MSL data from 1933/1 to 2017/12.
- The acceleration is -0.00718 ± 0.03198 mm/year^2.

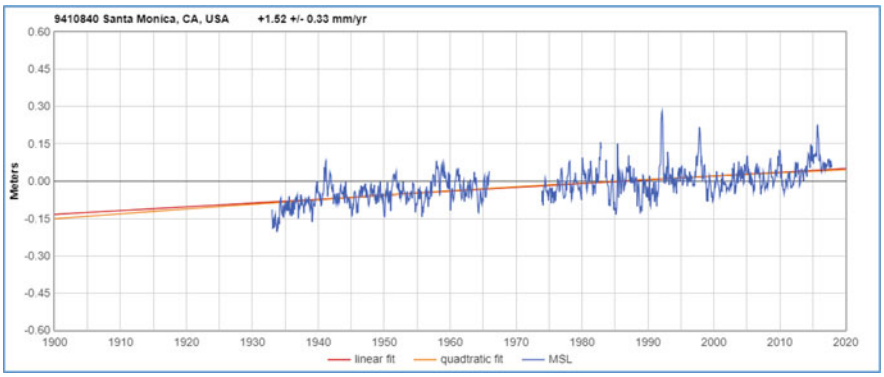


Fig. 5.17 MSL data for Santa Monica, CA, USA. Image reproduced modified after www.sealevel.info

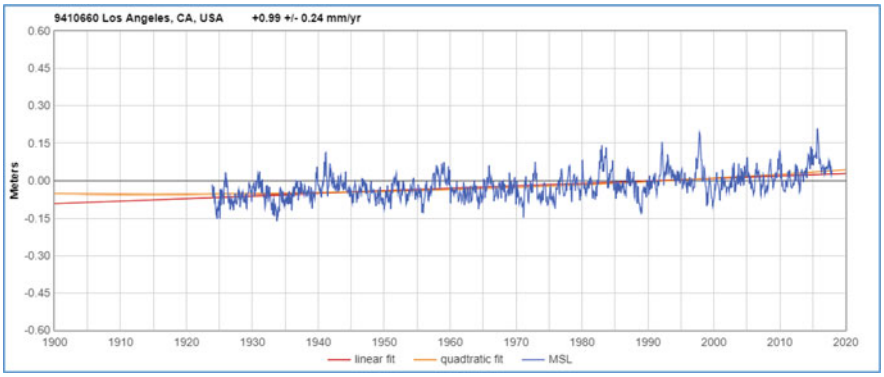


Fig. 5.18 MSL data for Los Angeles, CA, USA. Image reproduced modified after www.sealevel.info

5.3.17 Los Angeles, CA, USA

- The MSL trend at Los Angeles, CA, USA (Fig. 5.18) is +0.99 mm/year with a 95% confidence interval of ±0.24 mm/year, based on MSL data from 1923/12 to 2017/12.
- The acceleration is 0.01773 ± 0.01924 mm/year^2.

5.3.18 La Jolla, CA, USA

- The MSL trend at La Jolla, CA, USA (Fig. 5.19) is +2.15 mm/year with a 95% confidence interval of ±0.26 mm/year, based on MSL data from 1924/11 to 2017/12.
- The acceleration is 0.01121 ± 0.02154 mm/year^2.

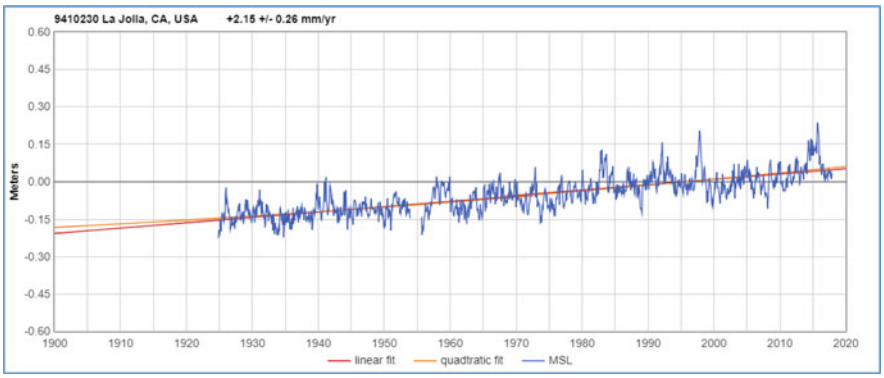


Fig. 5.19 MSL data for La Jolla, CA, USA. Image reproduced modified after www.sealevel.info

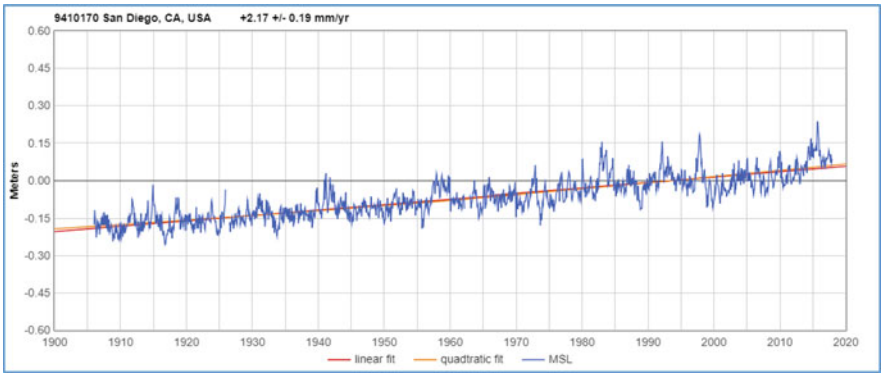


Fig. 5.20 MSL data for San Diego, CA, USA. Image reproduced modified after www.sealevel.info

5.3.19 San Diego, CA, USA

- The MSL trend at San Diego, CA, USA (Fig. 5.20) is +2.17 mm/year with a 95% confidence interval of ±0.19 mm/year, based on MSL data from 1906/1 to 2017/12.
- The acceleration is 0.00813 ± 0.01279 mm/year^2.

5.3.20 Balboa, Panama

- The MSL trend at Balboa, Panama (Fig. 5.21) is +1.44 mm/year with a 95% confidence interval of ±0.21 mm/year, based on MSL data from 1908/1 to 2017/12.
- The acceleration is -0.00548 ± 0.01504 mm/year^2.

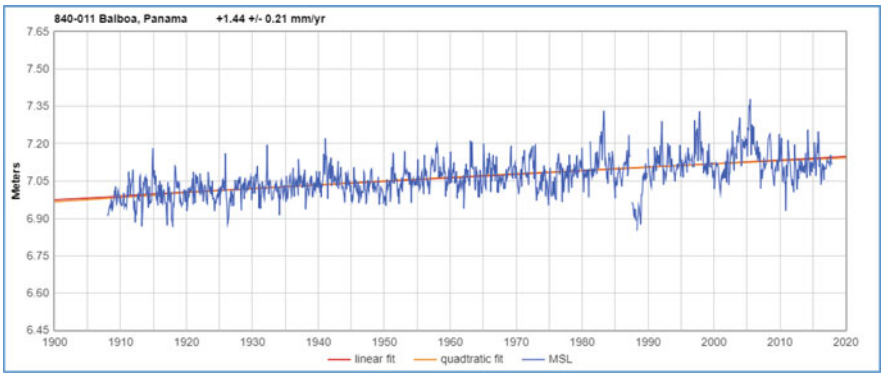


Fig. 5.21 MSL data for Balboa, Panama. Image reproduced modified after www.sealevel.info

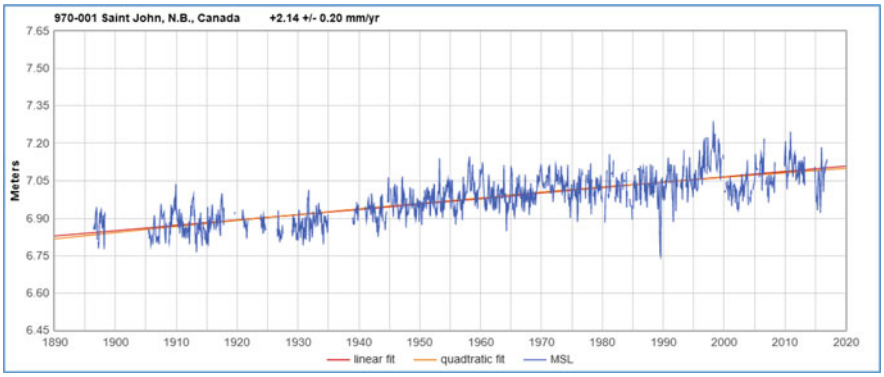


Fig. 5.22 MSL data for Saint John, N.B., Canada. Image reproduced modified after www.sealevel.info

5.3.21 Saint John, N.B., Canada

- The MSL trend at Saint John, N.B., Canada (Fig. 5.22) is +2.14 mm/year with a 95% confidence interval of ±0.20 mm/year, based on MSL data from 1896/6 to 2016/12.
- The acceleration is -0.00631 ± 0.01237 mm/year^2.

5.3.22 Halifax, Canada

- The MSL trend at Halifax, Canada (Fig. 5.23) is +3.18 mm/year with a 95% confidence interval of ±0.13 mm/year, based on MSL data from 1895/11 to 2014/7.
- The acceleration is -0.001387 ± 0.008732 mm/year^2.

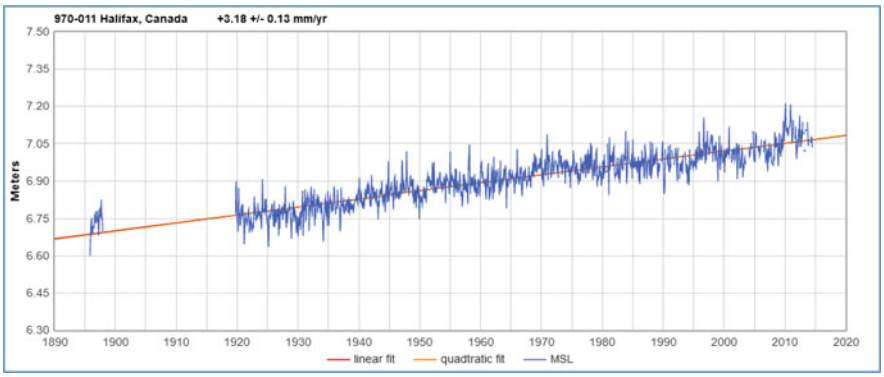


Fig. 5.23 MSL data for Halifax, Canada. Image reproduced modified after www.sealevel.info

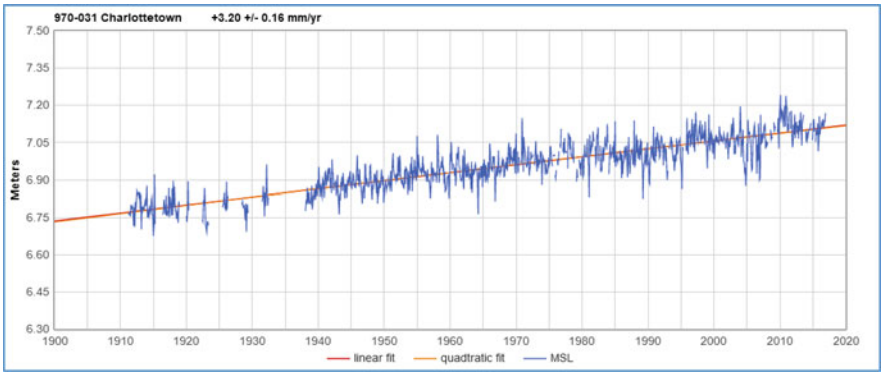


Fig. 5.24 MSL data for Charlottetown. Image reproduced modified after www.sealevel.info

5.3.23 Charlottetown, Canada

- The MSL trend at Charlottetown, Canada (Fig. 5.24) is +3.20 mm/year with a 95% confidence interval of ±0.16 mm/year, based on MSL data from 1911/4 to 2016/12.
- The acceleration is -0.00230 ± 0.01056 mm/year^2.

5.3.24 Quebec, Canada

- The MSL trend at Quebec, Canada (Fig. 5.25) is −0.26 mm/year with a 95% confidence interval of ±0.45 mm/year, based on MSL data from 1910/1 to 2012/10.
- The acceleration is -0.0367 ± 0.0314 mm/year^2.

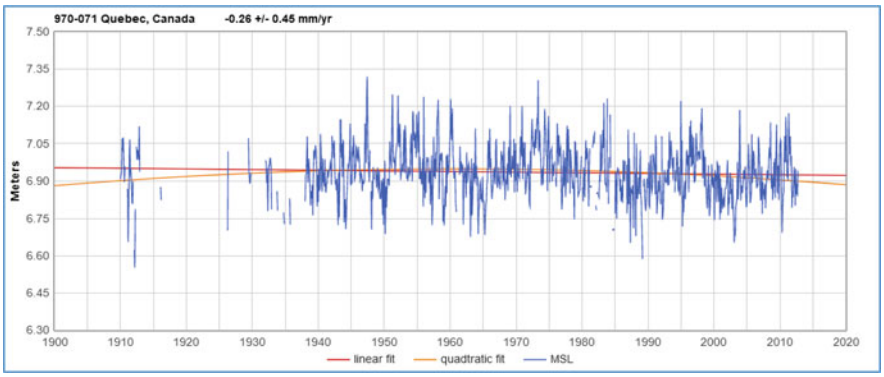


Fig. 5.25 MSL data for Quebec. Image reproduced modified after www.sealevel.info

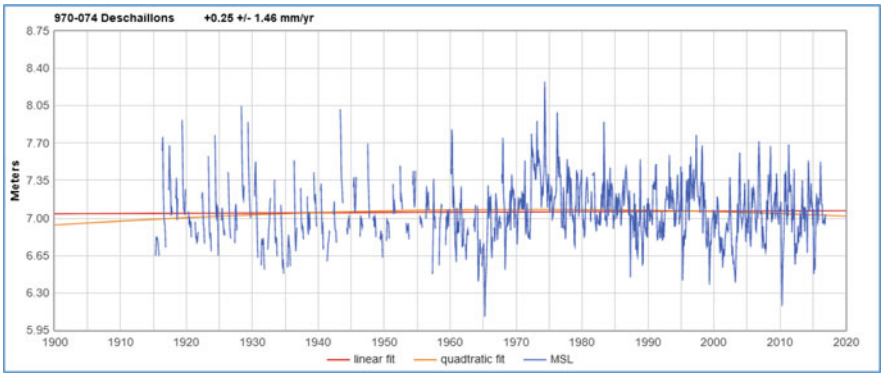


Fig. 5.26 MSL data for Deschaillons. Image reproduced modified after www.sealevel.info

5.3.25 Deschaillons, Canada

- The MSL trend at Deschaillons, Canada (Fig. 5.26) is +0.25 mm/year with a 95% confidence interval of ±1.46 mm/year, based on MSL data from 1915/5 to 2016/12.
- The acceleration is -0.0556 ± 0.1069 mm/year^2.

5.3.26 Trois-Rivieres, Canada

- The MSL trend at Trois-Rivieres, Canada (Fig. 5.27) is −1.43 mm/year with a 95% confidence interval of ±1.84 mm/year, based on MSL data from 1899/10 to 2016/12.
- The acceleration is -0.0440 ± 0.1225 mm/year^2.

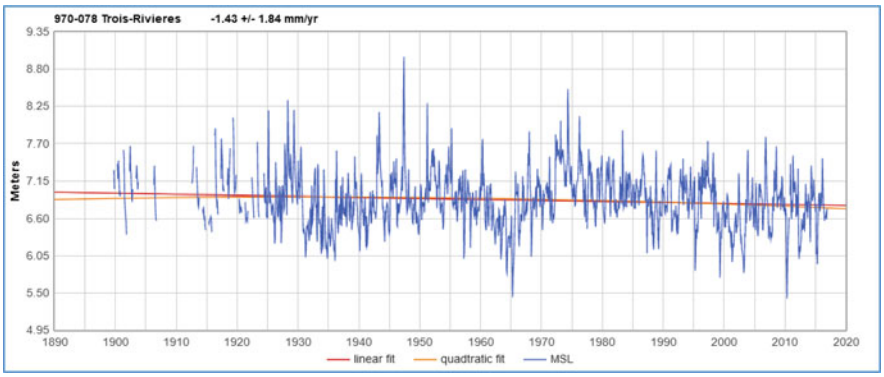


Fig. 5.27 MSL data for Trois-Rivieres. Image reproduced modified after www.sealevel.info

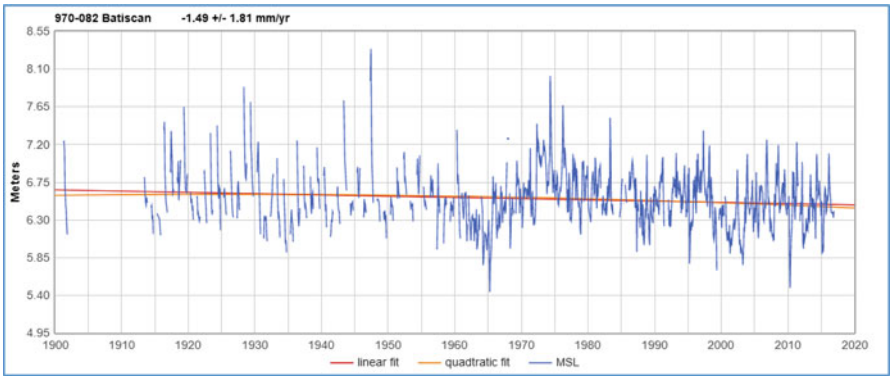


Fig. 5.28 MSL data for Batiscan. Image reproduced modified after www.sealevel.info

5.3.27 Batiscan, Canada

- The MSL trend at Batiscan, Canada (Fig. 5.28) is -1.49 mm/year with a 95% confidence interval of ± 1.81 mm/year, based on MSL data from 1901/5 to 2016/12.
- The acceleration is -0.0376 ± 0.1248 mm/year^2.

5.3.28 Neuville, Canada

- The MSL trend at Neuville, Canada (Fig. 5.29) is $+0.05$ mm/year with a 95% confidence interval of ± 0.70 mm/year, based on MSL data from 1914/6 to 2016/12.
- The acceleration is -0.0266 ± 0.0520 mm/year^2.

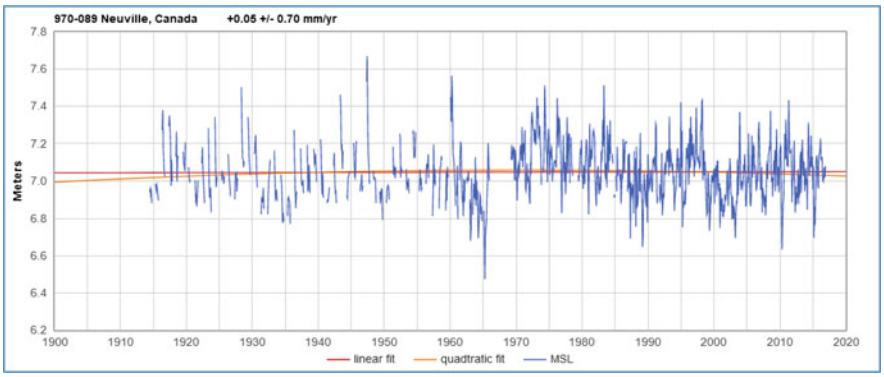


Fig. 5.29 MSL data for Neuville, Canada. Image reproduced modified after www.sealevel.info

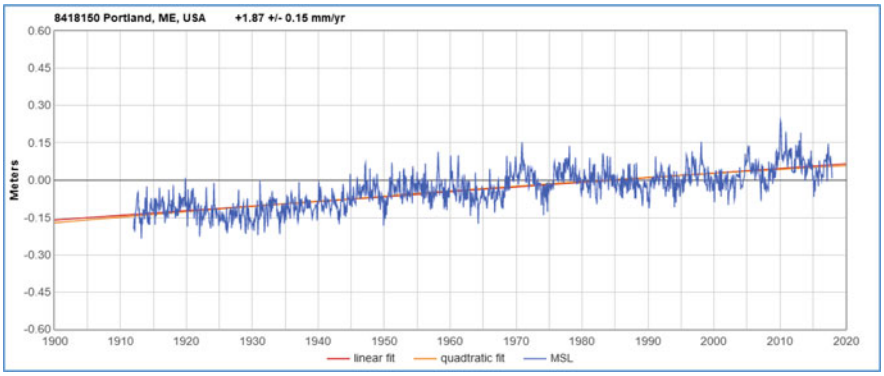


Fig. 5.30 MSL data for Portland, ME, USA. Image reproduced modified after www.sealevel.info

5.3.29 Portland, ME, USA

- The MSL trend at Portland, ME, USA (Fig. 5.30) is +1.87 mm/year with a 95% confidence interval of ±0.15 mm/year, based on MSL data from 1912/1 to 2017/12.
- The acceleration is -0.00694 ± 0.01079 mm/year^2.

5.3.30 The Battery, NY, USA

- The MSL trend at The Battery, NY, USA (Fig. 5.31) is +2.85 mm/year with a 95% confidence interval of ±0.09 mm/year, based on MSL data from 1856/1 to 2018/3.
- The acceleration is 0.00849 ± 0.00388 mm/year^2.

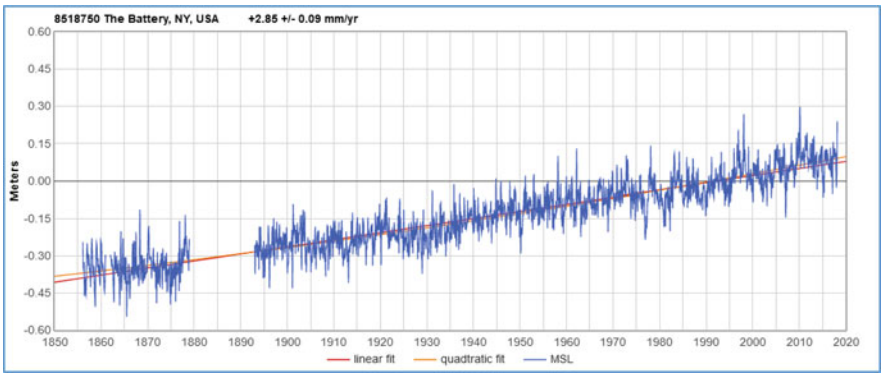


Fig. 5.31 MSL data for The Battery, NY, USA. Image reproduced modified after www.sealevel.info

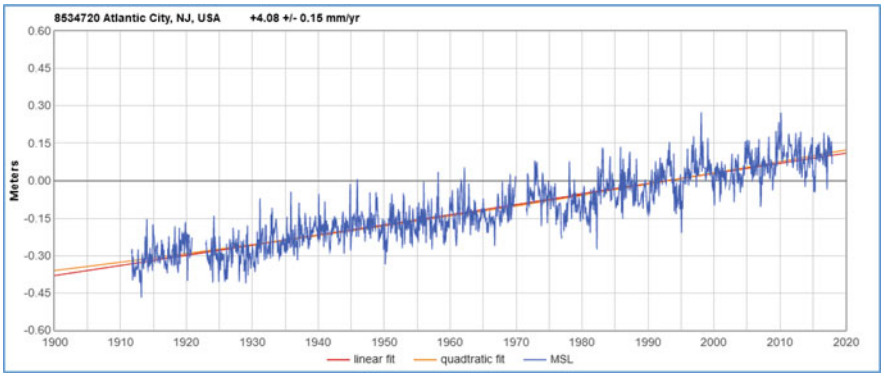


Fig. 5.32 MSL data for Atlantic City, NJ, USA. Image reproduced modified after www.sealevel.info

5.3.31 Atlantic City, NJ, USA

- The MSL trend at Atlantic City, NJ, USA (Fig. 5.32) is +4.08 mm/year with a 95% confidence interval of ±0.15 mm/year, based on MSL data from 1911/9 to 2017/12.
- The acceleration is 0.01225 ± 0.01122 mm/year^2.

5.3.32 Philadelphia, PA, USA

- The MSL trend at Philadelphia, PA, USA (Fig. 5.33) is +2.94 mm/year with a 95% confidence interval of ±0.19 mm/year, based on MSL data from 1900/7 to 2017/12.
- The acceleration is 0.01607 ± 0.01221 mm/year^2.

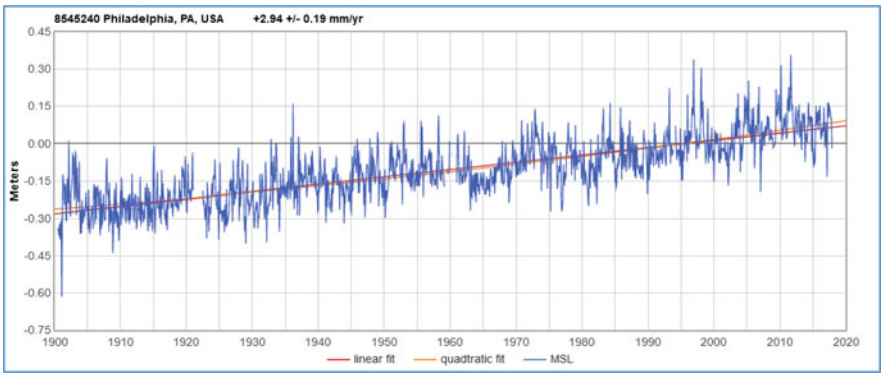


Fig. 5.33 MSL data for Philadelphia, PA, USA. Image reproduced modified after www.sealevel.info

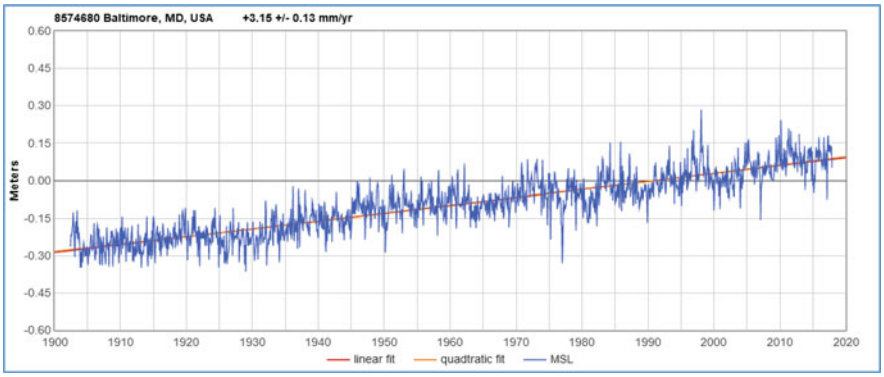


Fig. 5.34 MSL data for Baltimore, MD, USA. Image reproduced modified after www.sealevel.info

5.3.33 Baltimore, MD, USA

- The MSL trend at Baltimore, MD, USA (Fig. 5.34) is +3.15 mm/year with a 95% confidence interval of ±0.13 mm/year, based on MSL data from 1902/6 to 2017/12.
- The acceleration is 0.00382 ± 0.00852 mm/year^2.

5.3.34 Fernandina Beach, FL, USA

- The MSL trend at Fernandina Beach, FL, USA (Fig. 5.35) is +2.11 mm/year with a 95% confidence interval of ±0.18 mm/year, based on MSL data from 1897/6 to 2017/12.
- The acceleration is 0.01600 ± 0.01108 mm/year^2.

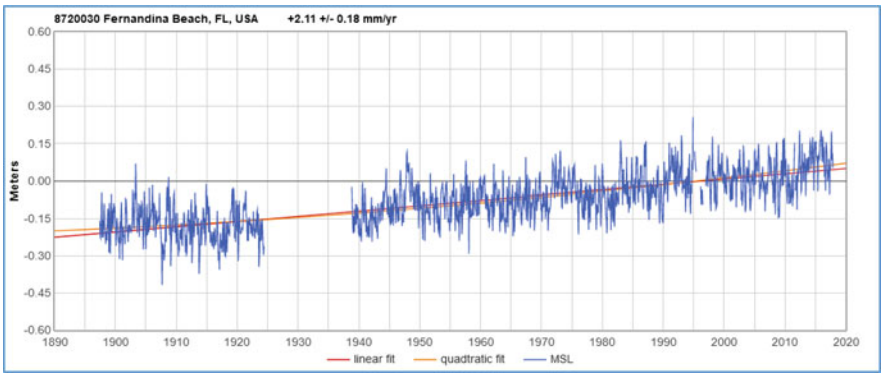


Fig. 5.35 MSL data for Fernandina Beach, FL, USA. Image reproduced modified after www.sealevel.info

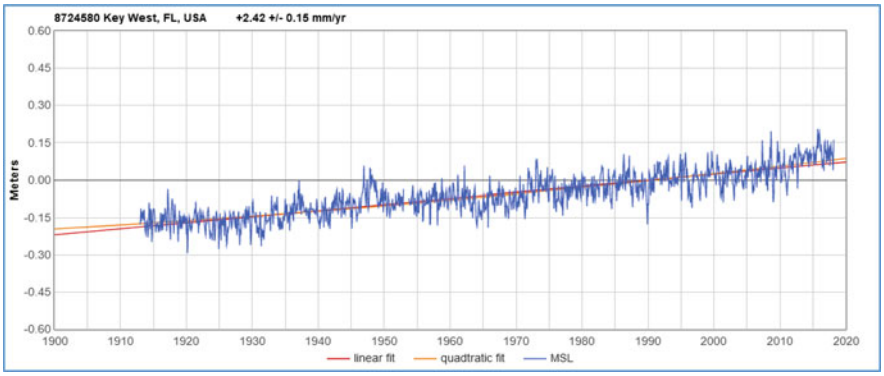


Fig. 5.36 MSL data for Key West, FL, USA. Image reproduced modified after www.sealevel.info

5.3.35 Key West, FL, USA

- The MSL trend at Key West, FL, USA (Fig. 5.36) is +2.42 mm/year with a 95% confidence interval of ±0.15 mm/year, based on MSL data from 1913/1 to 2018/3.
- The acceleration is 0.01412 ± 0.01064 mm/year^2.

5.3.36 Mayport, FL, USA

- The MSL trend at Mayport, FL, USA (Fig. 5.37) is +2.58 mm/year with a 95% confidence interval of ±0.27 mm/year, based on MSL data from 1928/5 to 2016/11.
- The acceleration is 0.01524 ± 0.02314 mm/year^2.

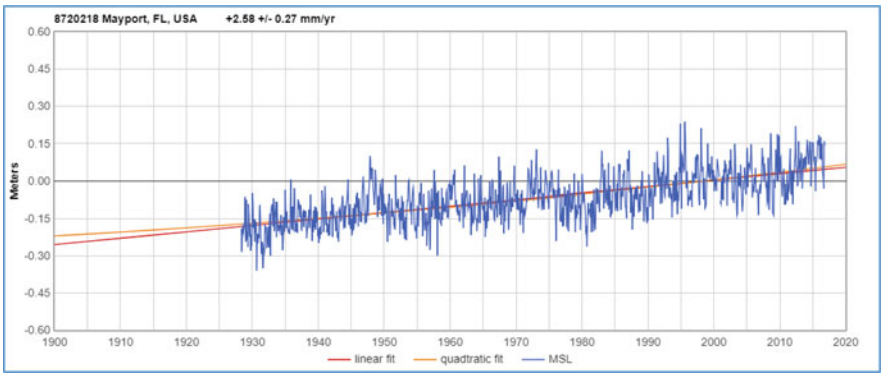


Fig. 5.37 MSL data for Mayport, FL, USA. Image reproduced modified after www.sealevel.info

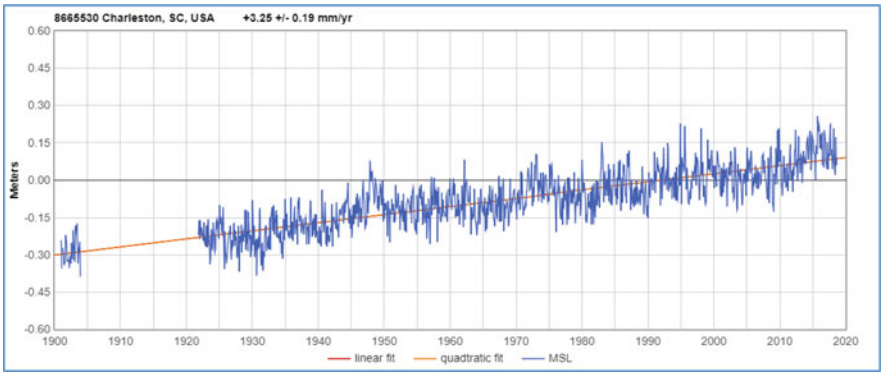


Fig. 5.38 MSL data for Charleston, SC, USA. Image reproduced modified after www.sealevel.info

5.3.37 Charleston, SC, USA

- The MSL trend at Charleston, SC, USA (Fig. 5.38) is +3.25 mm/year with a 95% confidence interval of ±0.19 mm/year, based on MSL data from 1901/1 to 2018/9.
- The acceleration is -0.000150 ± 0.012568 mm/year^2.

5.3.38 Washington, DC, USA

- The MSL trend at Washington, DC, USA (Fig. 5.39) is +3.22 mm/year with a 95% confidence interval of ±0.28 mm/year, based on MSL data from 1924/12 to 2017/12.
- The acceleration is -0.000525 ± 0.023634 mm/year^2.

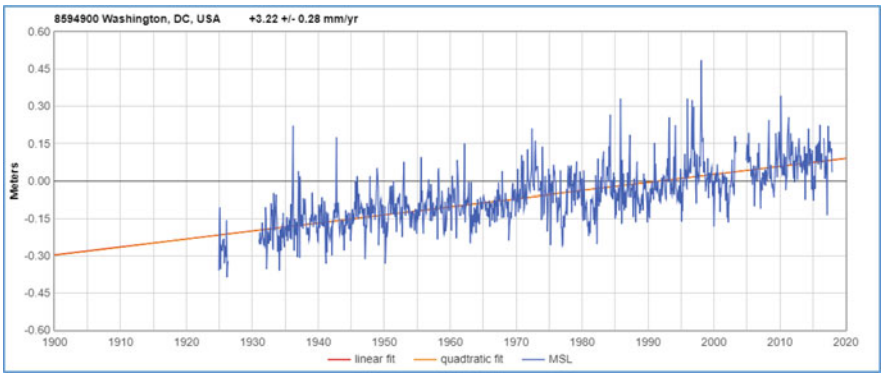


Fig. 5.39 MSL data for Washington, DC, USA. Image reproduced modified after www.sealevel.info

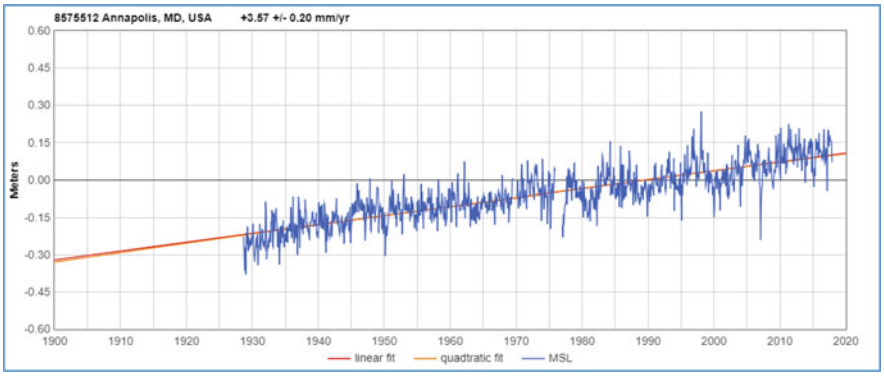


Fig. 5.40 MSL data for Annapolis, MD, USA. Image reproduced modified after www.sealevel.info

5.3.39 Annapolis, MD, USA

- The MSL trend at Annapolis, MD, USA (Fig. 5.40) is +3.57 mm/year with a 95% confidence interval of ±0.20 mm/year, based on MSL data from 1928/9 to 2017/12.
- The acceleration is -0.00356 ± 0.01744 mm/year^2.

5.3.40 Eastport, ME, USA

- The MSL trend at Eastport, ME, USA (Fig. 5.41) is +2.13 mm/year with a 95% confidence interval of ±0.18 mm/year, based on MSL data from 1929/10 to 2017/11.
- The acceleration is -0.01719 ± 0.01551 mm/year^2.

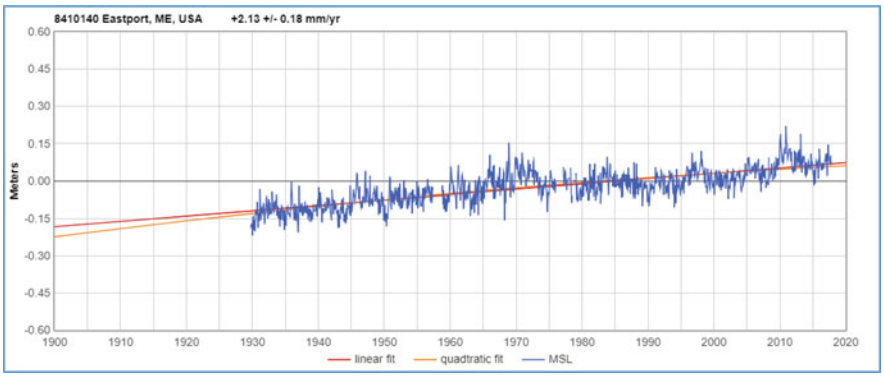


Fig. 5.41 MSL data for Eastport, ME, USA. Image reproduced modified after www.sealevel.info

5.4 Summary Tables

Table 5.1 presents a summary of the sea level results for the LTT stations of West North America.

The average relative rate of rise is −0.38 mm/year, the average acceleration is +0.00120 mm/year^2.

Table 5.2 presents a summary of the sea level results for the LTT stations of East North America.

The average relative rate of rise is +2.22 mm/year, the average acceleration is +0.00280 mm/year^2.

5.5 Summary and Conclusions

Nonlinear patterns in tide gauge signals are detected by polynomial fittings. As the second-order coefficients of parabolic fittings are generally small, I may conclude that the nonlinearities in the rate of rise of sea levels are small.

The nonlinearities reflect changes in the sea component to the relative sea level signal, originated from the increased volume of the water ocean by mass addition and thermal expansion, and in the land component, originated from the subsidence or uplift of the land.

Recent subsidence and global isostatic adjustments (GIA) explains the different patterns of sea level rise along the East and West Coast of the United States. The subsidence increases moving southwards, along both the East and the West Coasts. Different phasing of the multi-decadal oscillations along the East and West coast explains the remaining features of the Pacific and Atlantic patterns.

Table 5.1 Summary of sea level rise along the West Coast of North America

Tide gauge	Start	End	Range	u (mm/year)	a (mm/year^2)	Δy 2050–2018			Δy 2100–2018		
						mm	mm	mm	mm	mm	mm
Ketchikan, AK, USA	1919.04	2017.96	98.92	−0.33	−0.0181	−11	−20	−48	−27	−88	−161
Sitka, AK, USA	1924.37	2017.96	93.59	−2.33	−0.0163	−75	−83	−107	−191	−246	−308
Juneau, AK, USA	1936.04	2017.96	81.92	−13.16	−0.0378	−421	−440	−490	−1079	−1206	−1333
Unalaska, AK, USA	1934.04	2017.96	83.92	−4.14	−0.0145	−132	−140	−159	−339	−388	−438
Prince Rupert, Canada	1909.04	2016.96	107.92	1.17	0.0148	37	45	71	96	146	211
Point Atkinson, Canada	1914.37	2016.96	102.59	0.95	−0.0053	30	28	19	78	60	38
Vancouver, Canada	1909.87	2016.96	107.09	0.49	0.0251	16	29	72	40	125	235
Victoria, Canada	1909.2	2016.96	107.76	0.73	0.0066	23	27	38	60	82	111
Tofino, Canada	1909.79	2016.96	107.17	−1.26	0.02	−40	−30	4	−103	−36	52
Friday Harbor, WA, USA	1934.04	2017.96	83.92	1.2	0.0102	38	44	57	98	133	168
Seattle, WA, USA	1899.04	2017.96	118.92	2.05	0.0099	66	71	90	168	201	250
Neah Bay, WA, USA	1934.62	2017.96	83.34	−1.69	−0.0162	−54	−62	−84	−139	−193	−248
Astoria, OR, USA	1925.12	2017.96	92.84	−0.14	0.0148	−4	3	25	−11	38	95
Crescent City, CA, USA	1933.04	2017.96	84.92	−0.78	−0.0086	−25	−29	−41	−64	−93	−123
San Francisco, CA, USA	1854.54	2017.96	163.42	1.47	0.0141	47	54	91	121	168	262
Santa Monica, CA, USA	1933.04	2017.96	84.92	1.52	−0.0072	49	45	35	125	100	75
Los Angeles, CA, USA	1923.96	2017.96	94	0.99	0.0177	32	41	67	81	141	209
La Jolla, CA, USA	1924.87	2017.96	93.09	2.15	0.0112	69	75	91	176	214	257
San Diego, CA, USA	1906.04	2017.96	111.92	2.17	0.0081	69	74	88	178	205	242
Balboa, Panama	1908.04	2017.96	109.92	1.44	−0.0055	46	43	34	118	100	75
Averages				−0.38	0.0012	−12	−11	−7	−31	−27	−17

Table 5.2 Summary of sea level rise along the East Coast of North America

Station	Start	End	Range	u (mm/year)	a (mm/year^2)	Δy 2050–2018			Δy 2100–2018		
						mm	mm	mm	mm	mm	mm
Saint John, N.B., Canada	1896.46	2016.96	120.5	2.14	−0.0063	68	65	53	175	154	123
Halifax, Canada	1895.87	2014.54	118.67	3.18	−0.0014	102	101	98	261	256	249
Charlottetown, Canada	1911.29	2016.96	105.67	3.2	−0.0023	102	101	97	262	255	245
Pointe-Au-Pere, Canada	1900.04	1983.96	83.92	−0.37	0.0829	−12	31	142	−30	248	534
Quebec, Canada	1910.04	2012.79	102.75	−0.24	−0.035	−8	−26	−83	−20	−137	−285
Deschaillons, Canada	1915.37	2016.96	101.59	0.25	−0.0556	8	−20	−111	21	−166	−398
Trois-Rivieres, Canada	1899.79	2016.96	117.17	−1.43	−0.044	−46	−68	−151	−117	−265	−477
Batiscan, Canada	1901.37	2016.96	115.59	−1.49	−0.0376	−48	−67	−136	−122	−249	−427
Neuville, Canada	1914.46	2015.96	101.5	0.05	−0.0266	2	−12	−55	4	−85	−196
Harrington Harbor, Canada	1910.62	1989.62	79	−0.63	−0.0216	−20	−31	−59	−52	−124	−194
St John's, Nfld., Canada	1935.62	2015.96	80.34	2.06	0.0016	66	67	69	169	174	180
Eastport, ME, USA	1929.79	2017.88	88.09	2.13	−0.0172	68	59	35	175	117	55
Portland, ME, USA	1912.04	2017.96	105.92	1.87	−0.0069	60	56	45	153	130	100
Boston, MA, USA	1921.04	2017.96	96.92	2.82	−0.003	90	89	84	231	221	209
Woods Hole, MA, USA	1932.62	2017.96	85.34	2.86	0.0172	92	100	124	235	292	353
Newport, RI, USA	1930.79	2017.96	87.17	2.75	0.0113	88	94	110	226	263	304
Kings Pt/Willets Pt, NY, USA	1931.62	2016.87	85.25	2.5	0.0007	80	80	81	205	207	210
The Battery, NY, USA	1856.04	2018.21	162.17	2.85	0.0085	91	96	118	234	262	319
Sandy Hook, NJ, USA	1932.87	2017.96	85.09	4.07	0.0063	130	133	142	334	355	377
Atlantic City, NJ, USA	1911.71	2017.96	106.25	4.08	0.0123	131	137	158	335	376	429
Philadelphia, PA, USA	1900.54	2017.96	117.42	2.94	0.0161	94	102	133	241	295	373
Lewes, DE, USA	1919.12	2017.96	98.84	3.44	0.0202	110	120	152	282	350	432
Baltimore, MD, USA	1902.46	2017.96	115.5	3.15	0.0038	101	103	110	258	271	289
Annapolis, MD, USA	1928.71	2017.96	89.25	3.57	−0.0036	114	112	107	293	281	267

(continued)

Table 5.2 (contiunued)

Station	Start	End	Range	u (mm/year)	a (mm/year^2)	Δy 2050–2018			Δy 2100–2018		
						mm	mm	mm	mm	mm	mm
Solomon's Island (Biol. Lab.), MD, USA	1937.96	2017.96	80	3.78	0.0544	121	149	218	310	493	671
Washington, DC, USA	1924.96	2017.96	93	3.22	−0.0005	103	103	102	264	262	260
Sewells Point, VA, USA	1927.62	2017.96	90.34	4.62	0.0182	148	157	183	379	440	507
Wilmington, NC, USA	1935.37	2017.96	82.59	2.29	0.0302	73	89	129	188	289	392
Charleston, SC, USA	1921.79	2017.96	96.17	3.27	−0.0039	105	103	97	268	255	240
Fort Pulaski, GA, USA	1935.04	2017.96	82.92	3.22	0.0281	103	117	155	264	359	454
Fernandina Beach, FL, USA	1897.46	2017.96	120.5	2.11	0.016	68	76	107	173	227	306
Mayport, FL, USA	1928.37	2016.87	88.5	2.58	0.0152	83	90	112	212	263	318
Key West, FL, USA	1913.04	2018.21	105.17	2.42	0.0141	77	85	108	198	246	307
Averages				2.22	0.0028	71	72	75	182	191	198

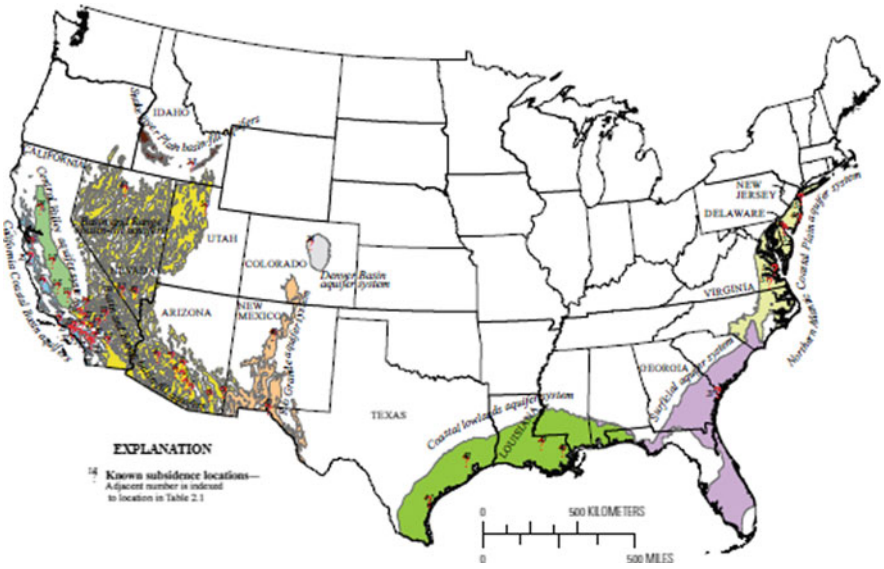


Fig. 5.42 Areas of land subsidence, owing primarily or secondarily to groundwater or oil and gas abstractions, in the 48 conterminous USA, and associated aquifer systems. Image reproduced modified from Galloway, Bawden, Leake, & Honegger [4]

Land subsidence is a problem especially relevant in the United States, where more than 45,000 km^2 of land [3] have been directly affected by subsidence (Fig. 5.42).

The principal causes are aquifer-system compaction, hydro-compaction, natural compaction, underground mining, drainage of organic soils, sinkholes, and thawing permafrost [5].

Nearly the entire East *Coast* of the United States, from Massachusetts and parts of Maine to Florida, is known to be affected by subsidence [6–10].

Subsidence is much stronger along the East Coast of the United States and significant only in Southern California along the West Coast, and it increased in intensity since the mid-1900s.

The data of absolute (geodetic) position of antennas nearby the tide gauges from satellite may permit to better attribute the relative sea level rise of a specific tide gauge to the growth of the water volume or the sinking of the land [11, 12]. However, the GPS time series only covers a few years of data, they are only recent, do not help when there are sudden land motions, such as earthquakes.

The negligible acceleration result is consistent with other global and regional estimations from LTT stations such as Houston and Dean [13], Boretti [14, 15], Parker [16], or Parker and Ollier [17, 18].

Sea level acceleration is small, but larger along the East coast, because of the recent subsidence and the recent upward phase of the multi-decadal oscillations that are not phased with those of the West Coast.

The proposed simple formulae for sea level rise to 2050 and 2100 based on the conventional present sea level velocity $\boldsymbol{u}$ and acceleration $\boldsymbol{a}$ return the small values of Tables 5.1 and 5.2, on average negative for the West Coast, −12 to −7 and −31 to −17 mm, respectively, and on average positive for the East Coast, +71 to +75 and +182 to +198 mm, respectively.

Acknowledgment The author received no funding and declares no competing interests.

References

1. Chambers, D., Merrifield, M. A., & Nerem, R. S. (2012). Is there a 60-year oscillation in global mean sea level? *Geophysical Research Letters, 39*, 18.
2. Schlesinger, M., & Ramankutty, N. (1994). An oscillation in the global climate system of period 65-70 years. *Nature, 367*, 723–726.
3. Galloway, D. L., Jones, D. R. & Ingebritsen, S. E. (1999). *Land subsidence in the United States* (Vol. 1182). United States Geological Survey. Retrieved from https://pubs.usgs.gov/circ/circ1182/.
4. Galloway, D. L., Bawden, G. W., Leake, S. A., & Honegger D. G. (2008). Land subsidence hazards. In R. L. Baum, D. L. Galloway, & E. L. Harp (Eds.), *Landslide and land subsidence hazards to pipelines* (chapter 2). U.S. Geological Survey Open-File Report 2008-1164. Retrieved from http://pubs.usgs.gov/of/2008/1164/.
5. National Research Council. (1991). *Mitigating losses from land subsidence in the United States* (58p). Washington, DC: National Academy Press.
6. Davis, G. H. (1987). Land subsidence and sea level rise on the Atlantic Coastal Plain of the United States. *Environmental Geology and Water Sciences, 10*(2), 67–80.
7. Johnson, D. W. (1917). Is the Atlantic coast sinking? *Geographical Review, 3*(2), 135–139.
8. Karegar, M. A., Dixon, T. H., & Engelhart, S. E. (2016). Subsidence along the Atlantic Coast of North America: Insights from GPS and late Holocene relative sea level data. *Geophysical Research Letters, 43*(7), 3126–3133.
9. United States Geological Survey. (2000). *Land subsidence in the United States*. United States Geological Survey Fact Sheet-087-00. Retrieved from https://water.usgs.gov/ogw/gwrp/fs2001/test1/.
10. Galloway, D. L., & Sneed, M. (2013). Analysis and simulation of regional subsidence accompanying groundwater abstraction and compaction of susceptible aquifer systems in the USA. *Boletín de la Sociedad Geológica Mexicana, 65*(1), 123–136.
11. Blewitt, G., Kreemer, C., Hammond, W. C., & Gazeaux, J. (2016). MIDAS robust trend estimator for accurate GNSS station velocities without step detection. *Journal of Geophysical Research, 121*. https://doi.org/10.1002/2015JB012552.
12. Wöppelmann, G., & Marcos, M. (2016). Vertical land motion as a key to understanding sea level change and variability. *Reviews of Geophysics, 54*(1), 64–92.
13. Houston, J. R., & Dean, R. G. (2011). Sea-level acceleration based on U.S. tide gauges and extensions of previous global-gauge analyses. *Journal of Coastal Research, 27*, 409–417.
14. Boretti, A. (2012). Short term comparison of climate model predictions and satellite altimeter measurements of sea levels. *Coastal Engineering, 60*, 319–322.
15. Boretti, A. (2012). Is there any support in the long term tide gauge data to the claims that parts of Sydney will be swamped by rising sea levels? *Coastal Engineering, 64*, 161–167.
16. Parker, A. (2013). Sea level trends at locations of the United States with more than 100 years of recording. *Natural Hazards, 65*(1), 1011–1021.
17. Parker, A., & Ollier, C. D. (2017). California sea level rise: Evidence based forecasts vs. model predictions. *Ocean & Coastal Management, 149*, 198–209.
18. Parker, A., & Ollier, C. D. (2017). Short-term tide gauge records from one location are inadequate to infer global sea-level acceleration. *Earth Systems and Environment, 1*(2), 17.

Part II
Analytical System Applications

Chapter 6
Illustrated Guidelines for Modelling and Dynamic Simulation of Linear and Non-linear Deterministic Engineering Systems

Pavel M. Trivailo, Hamid Khayyam, and Reza N. Jazar

6.1 Introduction

In science, a *model* is a representation of an idea, an object, a system or a process that is used to describe, explain and analyse phenomena. In many practical situations, scientific model is a simplified and idealized replica of an engineering system (ES).

Scientific *modelling* is a scientific activity, the aim of which is to make an engineering system or its feature to understand, define, quantify, visualize or simulate by referencing it to existing and usually commonly accepted knowledge.

A *simulation* is the implementation of a model. A steady state simulation provides information about the system at a specific instant in time (usually at equilibrium, if such a state exists).

A *dynamic simulation* provides information over time. A simulation brings a model to life and shows how a particular object or phenomenon will behave. Such a simulation can be useful for testing, analysis or training in those cases, where real-world systems or concepts can be represented by models.

Modelling and simulations are extremely useful in engineering, and therefore are widely used. Some of the benefits include the following:

- In engineering, it is often either impossible or impractical to create experimental conditions in which scientists can directly measure outcomes.
- Using simulations is generally cheaper, safer and sometimes more ethical than conducting real-world experiments.

P. M. Trivailo (✉) · H. Khayyam
School of Engineering, RMIT University, Melbourne, VIC, Australia
e-mail: pavel.trivailo@rmit.edu.au

R. N. Jazar
Xiamen University of Technology, Xiamen, China

School of Engineering, RMIT University, Bundoora, VIC, Australia

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_6

(a) (b)

Fig. 6.1 The Federation Bell Installation in Birrarung Marr, Melbourne: (**a**) day view, (**b**) night view. Courtesy: (**a**) Neil McLachlan, (**b**) Trevor Mien

- Simulations can often be conducted faster than real time.
- Modelling and simulations enable designers to optimize and control the engineering systems.

In order to illustrate efficiency of the modelling and simulations in engineering, we present below three examples from RMIT University research practice.

The first example of the successful application of the modelling and simulation, preceding manufacture of the real system, is a design of the world-first set "harmonic" and "poly-tone" bells, developed by RMIT University research team, which included Dr. Neil McLachlan and Dr. Nigjeh Keramati (Chief Investigators), Prof. Joe Thomas (Research Consultant) and Prof. Pavel M. Trivailo (Project Research Supervisor). Using computer simulations, it was possible to develop unique bell shapes, enabling for their first natural frequencies to be in the exact ratio up to tenth harmonic, i.e. in the ratio 1:2:3:4:5:6:7:8:9:10 [1]. This project received support worth $2 M and came from Arts Victoria and the Melbourne International Festival for the Arts. After completion of successful modelling of the harmonic bells, 39 bells of up to 1.2 tonnes mass were manufactured and Bell Installation was built at the Federation Square in Birrarung Marr, Melbourne, which was opened by Sir Gustav Nostle on 26th January 2002 (Figs. 6.1a and 6.1b).

The second example of the successful application of the modelling and simulation, preceding manufacture of the real system, is related to the aerospace engineering. The World first Harmonic Bells on the Federation Square points to the sky—arena for another world-first achievement: on 25th September 2009, the longest ever launched tethered system of 31.7 km was successfully deployed. Being the tallest vertical structure ever made by mankind (as tall as 100 Eiffel towers or nearly 4 times Mount Everest), it is in the 2009 Guinness Book of world records [2]. The whole system was designed by the Delta-Utech space consulting company in the Netherlands, and led by the European Space Agency (ESA), with participation of

Fig. 6.2 Illustration of the YES-2 Concept, involving return of a small capsule from space to Earth, using a 30 km 5 kg tether rather than rocket. *Courtesy:* European Space Agency (ESA), S. Corvaja

hundreds of students [3]. However, optimal control design of the tether deployment in this YES-2 space mission, was performed by RMIT Team, including Dr. Paul Williams and Prof. Pavel M. Trivailo (Research Supervisor). The tether control design by RMIT was critical for the successful deployment of the longest ever deployed tethered system during this ESA space mission [4]. Preparation of the relevant launch of Foton-M3, carrying YES-2, is shown in Fig. 6.2.

Contribution of RMIT University to the simulation and design of the world-first taped tether mission, launched by JAXA has been officially acknowledged by Prof Hironori Fujii, the Leader of the S520-25 Space Rocket Project, who reported on the successful deployment of a 132-m taped tether, launched and deployed on 30th September 2010 by the Japan Aerospace Exploration Agency (JAXA) [5].

6.2 Modelling and Simulation of Linear Systems: General Discussion and Examples of the Assumptions, Fundamental Laws and Implementations

Modelling and simulation of engineering systems typically involve the following:

- Mathematical formulation of the task (representation of the real system as a model, scaling, simplifications, assumptions, etc.)

- Application of the proper physical principles/fundamental laws
- Selection of efficient solution paradigm (analytical, numerical)
- Selection of efficient methods, tools/computer packages/environments
- Simulation of the system or process
- Optimal design (where appropriate)
- Analysis of the results, including visualization/animation
- Validation of the results, model, assumptions, etc.
- Communication of the results

6.2.1 Multifunctional Significance of the Simulation Models

One of the most popular engineering models, mass-spring-damper system, is shown in Fig. 6.3.

It is well known that the differential equation of motion of the system and the natural frequency of its undamped counterpart are given by the following relationships:

$$m\ddot{q}(t) + c\dot{q}(t) + kq(t) = F(t) \tag{6.1}$$

$$\text{if } c = 0, \quad \text{then } \omega_1 = \sqrt{\frac{k}{m}} \text{ (rad/s)} \tag{6.2}$$

Using this model, it can be shown, that it may be relevant to a wide range of cases and systems, not only physical trolley, connected to the wall. In order to illustrate the multifunctional significance of this particular model, let us consider a basic model of the wing of the aircraft, which can be represented as a clamped elastic beam of length l with constant bending stiffness EI and attached concentrated mass m at its tip (see Fig. 6.4a).

If the mass-spring system (shown in Fig. 6.4b), which we introduce to replicate the system in Fig. 6.4a, is assumed to be linear, than the static *translational* displacement Δ of the mass m under the application of the static force F is given with the linear relationship:

$$F = k\Delta \text{ (for mass-spring model)} \tag{6.3}$$

For the wing model, also assumed to be a linear system, the relationship for the *lateral* displacement of the beam's tip and the lateral force F at the tip is also given with the linear relationship:

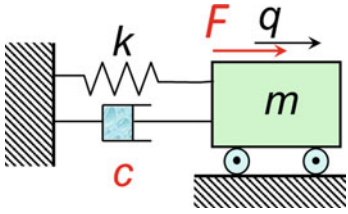


Fig. 6.3 Mass-spring-damper system

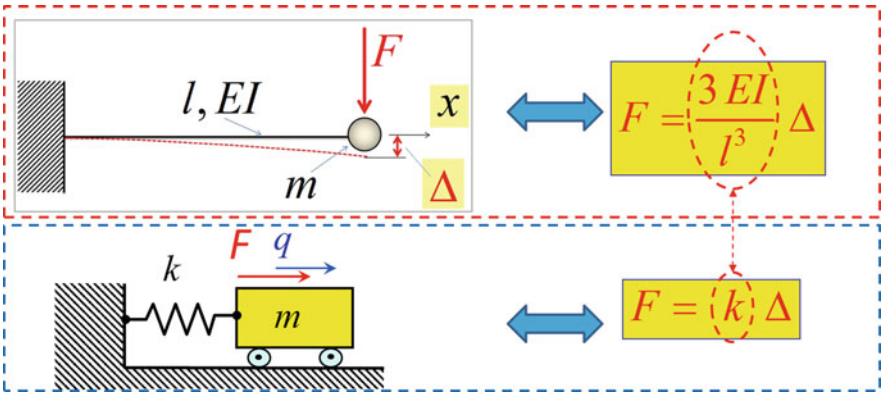


Fig. 6.4 Correspondence between the cantilevered beam system and the mass-spring system

$$F = \frac{3EI}{l^3}\Delta \quad \text{(for wing model)} \tag{6.4}$$

Comparison of Eqs. (6.3) and (6.4) allows us to identify that, for the wing, the equivalent of the stiffness k in the mass spring model is equal to $3EI/l^3$, therefore, the mass-spring model's Eq. (6.2) can now be also used for calculation of the first (fundamental) natural frequency of the wing's model:

$$\omega_1 = \sqrt{\frac{3EI}{ml^3}} \ \text{(rad/s)} \tag{6.5}$$

6.2.2 *Classical Analytical Modelling: Static Beam Deflection Example*

The classical way of modelling engineering systems is to derive their differential equations, which can be *ordinary* or *partial* differential equations. Their examples are presented below.

In case of the classical beam, the Bending Moment equation can be written as follows:

$$M = -EI\frac{d^2y}{dx^2} \tag{6.6}$$

This enables formulation of the relationship between the beam's deflection y and the applied distributed load q, known as the Euler–Bernoulli equation:

$$\frac{d^2}{dx^2}\left(EI\frac{d^2y}{dx^2}\right) = q \tag{6.7}$$

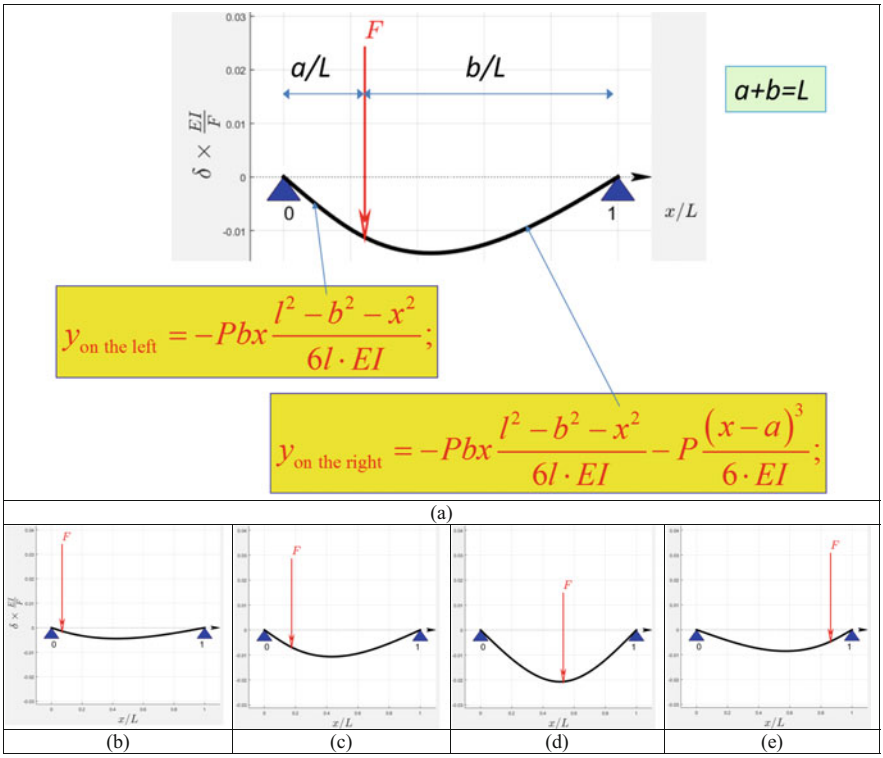


Fig. 6.5 Lateral deflections of the Euler-Bernoulli beam, loaded with the concentrated force F: (**a**) notations and analytical solutions; (**b–e**) beam's shape variation with variation of the position x/L of the applied force F

The last Eq. (6.7) is an *ordinary* differential equation and can be solved analytically for the case of the hinged-hinged beam, loaded with the concentrated force F, applied at the distance a from the left hinge. The exact analytical corresponding solution is given with two different equations for the left and right segments with respect to the force F and is presented in Fig. 6.5a.

Results of the simulations of the variation of the beam's shape with variation of the position of the force F along the beam can be conveniently plotted using MATLAB® and are shown in Fig. 6.5b–e.

6.2.3 *Classical Analytical Modelling: Temperature Distribution in the Rectangular Plate Example*

Let us consider a rectangular plate, which is bounded by $x = 0$, $x = a$, $y = 0$, and $y = b$ with the following boundary conditions:

$$T(x,0)=0,\quad T(0,y)=0,\quad T(a,y)=0,\quad \text{and}\quad T(x,b)=T_1 \tag{6.8}$$

The steady-state temperature distribution is governed by the Laplace Equation:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0 \tag{6.9}$$

This is a *partial* differential equation and can be solved analytically for a few simple cases, including rectangular plate. Solution is given with the following expression:

$$T(x,y)=\frac{4T_1}{\pi}\sum_{n=1,3,5,\ldots}^{\infty}\frac{\sin\left(\frac{\pi nx}{a}\right)\sinh\left(\frac{\pi ny}{a}\right)}{n\ \sinh\left(\frac{\pi nb}{a}\right)} \tag{6.10}$$

This solution can be presented in terms of the 2D plot as a 3D surface. Plot for the temperature distribution for $T_1 = 100$ °C and $a = b = 1$ m is shown in Fig. 6.6.

It is amazing, that the modern computer environments, like MATLAB®, enable to show results of the simulations, corresponding to the exact analytical solution cases, with so small effort! As a confirmation of this statement, we present in Fig. 6.6c MATLAB® script, corresponding to the case, shown in Fig. 6.6b.

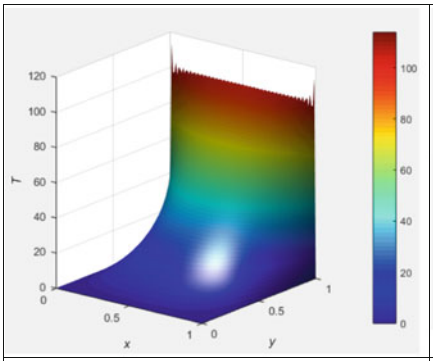
Unfortunately, in many practical cases, analytical solutions are not readily available, and one of the alternatives in these cases is to *solve numerically* associated differential equations. In the following sections, we will present examples of this strategy for various cases associated with *first* and *second order ordinary differential equations*.

6.2.4 *Numerical Simulation of the Problems, Described with the First Order Ordinary Differential Equations*

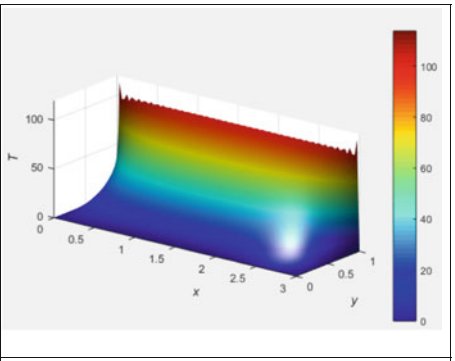
Series Resistance-Inductance Electrical Circuit Example-1

When the electromotive force is removed from a circuit containing inductance and resistance but no capacitors, the rate of decrease of current is proportional to the current.

Consider the electrical engineering system, shown in Fig. 6.7. When the electromotive force (emf) is removed from a circuit containing inductance and resistance but no capacitors, the rate of decrease of current is proportional to the current. If the initial current is 30 Amps but decays to 11 Amps after 0.01 s, find an expression for the current as a function of time.



(a)



(b)

```
%% Temperature Distribution, governed by Laplace's Eq.
% ----- Programmed by P.M.Trivailo ----------
T1=100; N=121; a=3; b=1;
[x,y]=meshgrid([0:0.01:1]*a,[0:0.01:1]*b);
T=zeros(length(x),length(y));
for n=1:2:N
   T=T + (4*T1/pi).*sin(pi*n*x/a).*sinh(pi*n*y/a)./(n*sinh(pi*n*b/a));
end
surface(x,y,T);
set(gca,'DataAspectRatio',[1 1 T1]);
colorbar; colormap JET;
grid on; xlabel('¥it x'); ylabel('¥it y'); zlabel('¥it T');
camlight HEADLIGHT; lighting PHONG; shading INTERP;
view([38, 18]); rotate3d;
position=[22 53 560 420]; dx=100; dy=50;
set(gcf,'position',position);
```

(c)

Fig. 6.6 Temperature distribution in the rectangular plate: (**a**) $a = 1, b = 1$ case; (**b**) $a = 3, b = 1$ case; (**c**) MATLAB® script for $a = 3, b = 1$ case

If $I(t)$ denotes current and t denotes time, the statement "the rate of decrease of current is proportional to the current" translates to the ordinary differential equation of the first order:

$$\frac{dI}{dt} = -kI \tag{6.11}$$

where k denotes the constant of proportionality.

This example is selected for illustration purposes, as it has an *exact* analytical solution, which we can use as a verification benchmark for the *approximated numerical* solution to be presented immediately after the exact solution.

For deriving an analytical solution, we notice that the differential equation (6.11) is separable, this allows analytical solutions with the process presented below:

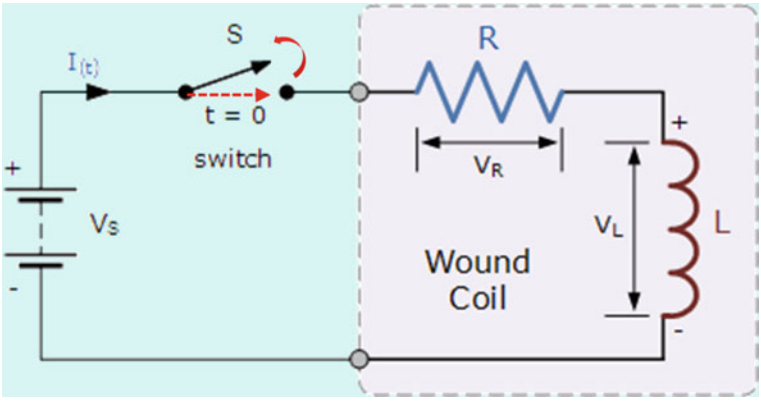


Fig. 6.7 Series resistance-inductance DC electrical circuit

$$\frac{dI}{dt} = -kI; \quad \frac{dI}{I} = -k\,dt; \quad \int \frac{dI}{I} = -k \int dt$$
$$\ln|I| = -kt + \text{Constant}_1$$
$$I = (\text{Constant}_2) \times \text{e}^{-kt}; \left(\text{the e}^{\text{Constant1}} \text{ and the } \pm \text{ all gets absorbed in the Constant}_2\right) \tag{6.12}$$

Another condition in the task, saying that "if the current decays to 11 amps after 0.01 seconds", enables us to calculate value of k:

$$\begin{aligned} &I|_{t=0.01} = 11; \quad \Rightarrow \quad 11 = 30 \times \text{e}^{-0.01k}; \quad \Rightarrow \quad \text{e}^{-0.01k} = \frac{11}{30}; \quad \Rightarrow \\ &-0.01k = \ln\left(\frac{11}{30}\right); \quad \Rightarrow \quad k = -100 \times \ln\left(\frac{11}{30}\right) = 100.3302 \end{aligned} \tag{6.13}$$

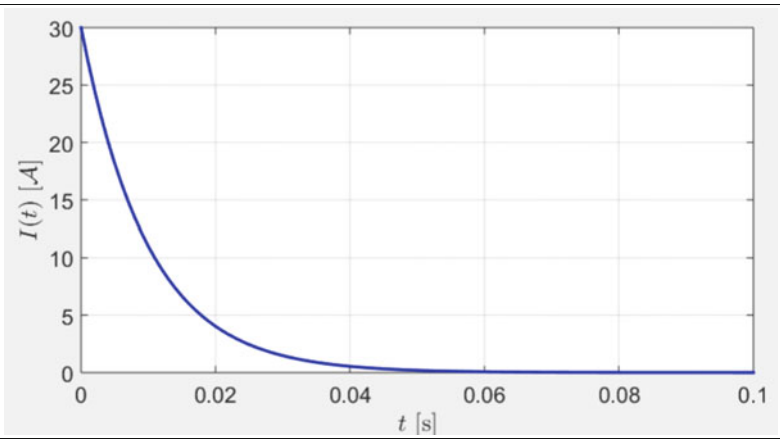
Therefore, the analytical exact solution for this particular electrical circuit can be written in its final, easy to plot, format:

$$I(t) = 30 \times \text{e}^{-100.3302t} \tag{6.14}$$

Plotted results together with the corresponding MATLAB® script are presented in Fig. 6.8.

The same task can be solved numerically, using MATLAB®. For this purpose, the *following steps can be recommended* for programming:

1. Input of the data for the particular case;
2. Programming of the differential equation (we use as an anonymous function method in the illustration example) (Fig. 6.9);
3. Solving differential equation, using one of the MATLAB® "ode" procedures (we use ode45, based on an explicit Runge-Kutta (4,5) formula [6]).



(a)

```
%%
% Designed by Prof P.M.Trivailo (C)2019
% Series Resistance-inductance Electrical Circuit
clc; clear; close('all');
t=[0:0.01:1]*0.1;
k=-100*log(11/30);
I=30*exp(-k*t);
figure; plot(t,I,'b','LineWidth',2);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$I(t)$ [$\cal{A}$]','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
set(gca,'FontSize',16);
```

(b)

Fig. 6.8 (**a**) Simulation results (exact solution) for the series resistance-inductance DC electrical circuit; (**b**) corresponding MATLAB® script

Numerical results of the simulation are shown in Fig. 6.10 with continuous line. This plot, for reference and comparison purposes, also presents an exact solution, shown in dashed line. The plot shows remarkably close correspondence between *exact analytical* and *approximate numerical* solutions. This explains, why numerical methods are so popular in modern engineering: offering high accuracy, they enable solution of much wider range of more complex problems than accessible with the analytical methods, typically offering solutions for very simplified cases of the very limited range of formulations. Later on we will also demonstrate that numerical methods are very efficient for linear and also non-linear formulations associated with any order of the ordinary differential equations.

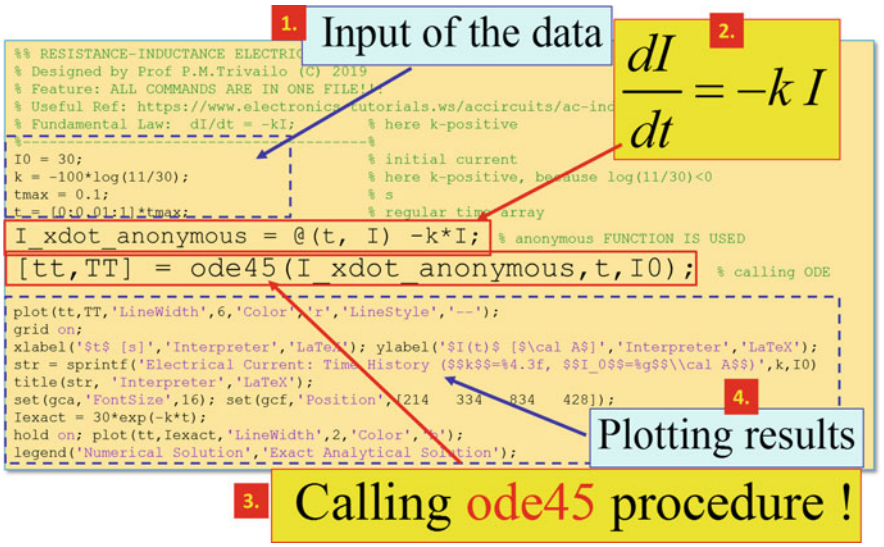


Fig. 6.9 Annotated MATLAB® script for solving electric circuit problem, with main conceptual steps shown

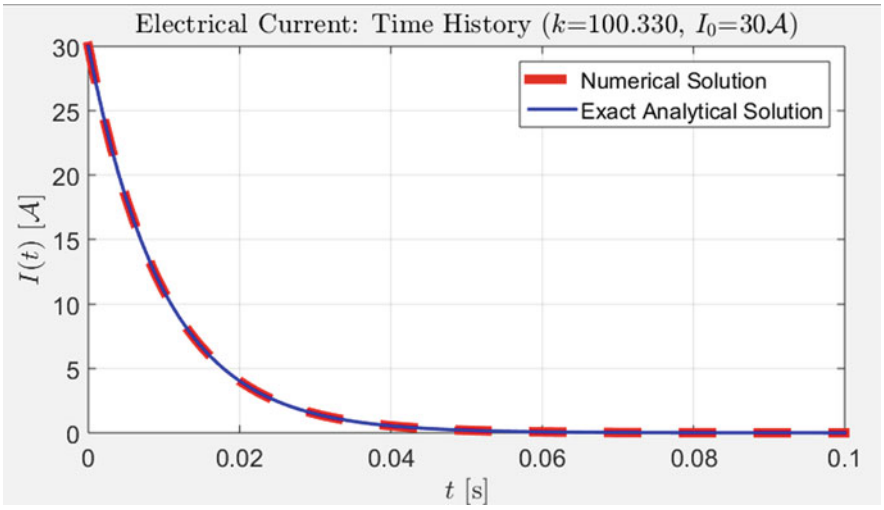


Fig. 6.10 Simulation results for the series resistance-inductance DC electrical circuit: numerical solution is shown with continuous blue line, exact analytical solution is also superimposed and shown with the dashed red line

Series Resistance-Inductance Electrical Circuit Example-2

The LR series circuit is connected across a constant voltage source (the battery) and a switch S. Assume that the switch, S, is open until it is closed at a time $t = 0$, and

then remains permanently closed producing a "step response" type voltage input. Find an expression for the current I as a function of time t.

Modelling of the system requires us to use Ohm's law, Lenz's law, Kirchhoff's voltage law and Faraday's law of induction.

The current, I begins to flow through the circuit, but does not rise instantly to its maximum value of $I_{\max}$, as determined by the ratio of V/R (*Ohm's Law*).

This limiting factor is due to the presence of the self-induced emf within the inductor as a result of the growth of magnetic flux (*Lenz's Law*).

After a time the voltage source neutralizes the effect of the self-induced emf, the current flow becomes constant and the induced current and field are reduced to zero.

We can use *Kirchhoff's Voltage Law*, to define the individual voltage drops that exist around the circuit and then hopefully use it to give us an expression for the flow of current. Kirchhoff's Voltage Law gives:

$$V(t) - (V_R + V_L) = 0 \tag{6.15}$$

The voltage drop across the resistor, R is IR (*Ohm's Law*):

$$V_R = IR \tag{6.16}$$

The voltage drop across the inductor, L is as follows (due to *Faraday's law* of induction):

$$V_L = L\frac{dI}{dt} \tag{6.17}$$

Resultant relationship has the following form:

$$L\frac{dI}{dt} = V - IR \tag{6.18}$$

This *ordinary* differential equation is separable; this allows analytical solutions. Detailed process is presented below:

$$\begin{aligned}
&\tfrac{dI}{dt} = \tfrac{V-IR}{L}; \quad \tfrac{dI}{V-IR} = \tfrac{1}{L}dt \\
&-\tfrac{1}{R}\int \tfrac{du}{u} = \tfrac{1}{L}\int dt \quad (\text{by setting } u = V - IR)\,; \quad \Rightarrow \\
&-\tfrac{1}{R}\ln|V - IR| = \tfrac{t}{L} + \text{Constant}_1 \\
&\ln|V - IR| = -\tfrac{R}{L}t + \text{Constant}_2 \\
&(note : -R \text{ gets absorbed in the Constant}_2)
\end{aligned}$$

$$\begin{aligned}
&V - IR = (\text{Constant}_3) \times \mathrm{e}^{-\frac{R}{L}t} \\
&\left(note : \ \text{the } \mathrm{e}^{\text{Constant}_2} \text{ and the } \pm \text{ all gets absorbed in the Constant}_3\right)
\end{aligned}$$

$$\begin{aligned} I\,R &= -\,(\text{Constant}_3) \times \mathrm{e}^{-\frac{R}{L}t} + V; \quad \Rightarrow \\ I &= -\,(\text{Constant}_4) \times \mathrm{e}^{-\frac{R}{L}t} + \frac{V}{R} \\ &\left(note: \ \tfrac{1}{R} \text{ gets absorbed in Constant}_4\right) \end{aligned} \tag{6.19}$$

Initial zero balance enables us to find the value of the **Constant$_4$**:

$$I|_{t=0} = 0; \quad \Rightarrow \quad 0 = -\,(\text{Constant}_4) \times \mathrm{e}^{\frac{0}{L}} + \frac{V}{R}; \quad \Rightarrow \quad \text{Constant}_4 = \frac{V}{R} \tag{6.20}$$

Analytical exact solution of the task can be now written in its final form:

$$I(t) = \frac{V}{R}\left(1 - \mathrm{e}^{-\frac{R}{L}t}\right) \tag{6.21}$$

To complement this solution process, it is often advisable to show results in the graphical format, facilitating dissemination of the results.

For this purpose, we use MATLAB® graphics commands shown in Fig. 6.11, and corresponding plots shown in Fig. 6.12.

Graphical format enables us to come up with the following useful observations:

- The time required for the current flowing in the *LR* series circuit to reach its maximum steady state value is equivalent to about 5 time constants or 5τ.

```
%%
% Designed by Prof P.M.Trivailo (C)2019
% Series Resistance-inductance Electric Circuit
clc; clear;
close('all');
t=[0:0.01:1]*0.1;
k=-100*log(11/30);
I=30*(1-exp(-k*t));
figure; plot(t,I,'b','LineWidth',2);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$I(t)$ [$\cal{A}$]','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
set(gca,'FontSize',16);
```

Fig. 6.11 MATLAB® script for solving electric circuit problem, with main conceptual steps shown

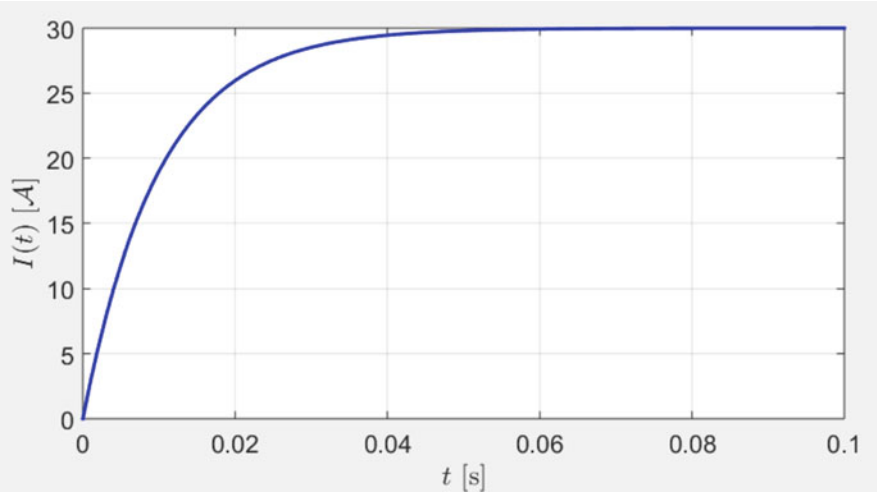


Fig. 6.12 Simulation results for the series resistance-inductance DC electrical circuit: numerical solution is shown with continuous blue line, exact analytical solution is also superimposed and shown with the dashed red line

- This time constant τ is measured by $\tau = L/R$, in seconds, where R is the value of the resistor in Ohms and L is the value of the inductor in Henries. This then forms the basis of an RL charging circuit, where 5τ can also be thought of as "$5 \times L/R$" or the transient time of the circuit.
- In two cases, considered above, $\tau = L/R = 1/k = 0.0100$ s, so transient time is approximately equal to 0.05 s.

Let us proceed now with the numerical solution process, which has been already outlined in the previous subsection. For the current example, this process is shown in Fig. 6.13.

Numerical results (together with exact solution) are shown in terms of the $I = I(t)$ plot in Fig. 6.14.

Newton's Law of Cooling (and Heating): General Comments on Modelling

The rate at which the temperature of an object changes is proportional to the difference between the temperature of the object and the temperature of the surroundings:

$$\frac{dT}{dt} = k\,(T - T_{\mathrm{s}}) \tag{6.22}$$

where: t time; k—the constant of proportionality; $T(t)$—temperature of the object at time t; $T_{\mathrm{s}}(t)$—temperature of the surrounding area at time t.

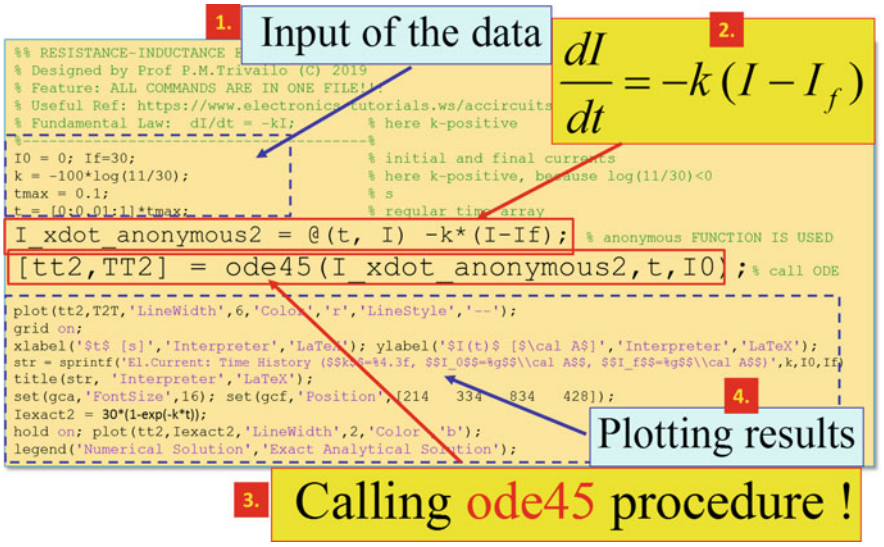


Fig. 6.13 Annotated MATLAB® script for solving electric circuit problem

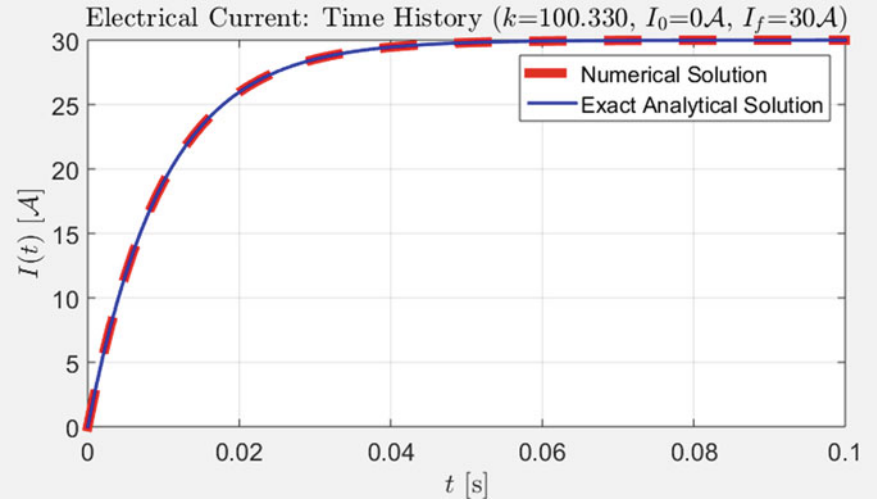


Fig. 6.14 Simulation results for the series resistance-inductance DC electrical circuit

Equation (6.22) is valid for both, cooling and heating, as any of these particular cases is described with appropriate values of k:

- If $k < 0$, Eq. (6.22) describes heating process.
- If $k = 0$, Eq. (6.22) corresponds to the static case, when temperature T is not changing: $T = \text{const}$.

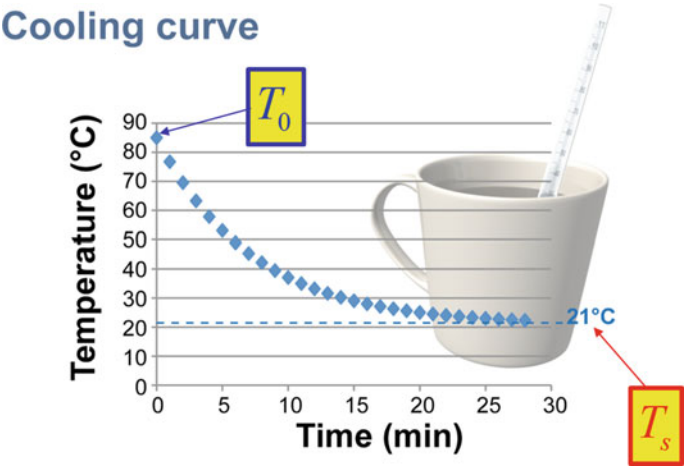


Fig. 6.15 Cooling of the cup of coffee. Image courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

- If $k > 0$, Eq. (6.22) describes cooling process.

The rate of cooling depends on the temperature difference between the two objects. A large temperature difference means rapid cooling.

Let us consider the task of cooling a cup of coffee, illustrated in Fig. 6.15. When the coffee is much hotter than the air, its temperature drops 8 °C in 1 min. A small temperature difference means slower cooling. When the coffee has cooled to 30 °C, the temperature changes by only 1.2 °C per minute.

We first attempt to get an analytical solution of the task, using Eq. (6.22). The main steps involved are as shown below:

$$\begin{aligned}
&\frac{dT}{dt} = k\,(T - T_\mathrm{s})\,; \quad \Rightarrow \quad \frac{dT}{T - T_\mathrm{s}} = k\,dt \\
&\ln|T - T_\mathrm{s}| = kt + \mathrm{Constant}_1; \quad \Rightarrow \quad T - T_\mathrm{s} = (\mathrm{Constant}_2) \times \mathrm{e}^{kt} \\
&\left(\text{the } \mathrm{e}^{\mathrm{Constant}_1} \text{ and the } \pm \text{ all gets absorbed in the } \mathrm{Constant}_2\right) \\
&T = T_\mathrm{s} + (\mathrm{Constant}_2) \times \mathrm{e}^{kt} \\
&\text{For } t = 0: \quad T_0 = T_\mathrm{s} + (\mathrm{Constant}_2) \times \mathrm{e}^0 \quad \Rightarrow \quad (\mathrm{Constant}_2) = T_0 - T_\mathrm{s}
\end{aligned} \tag{6.23}$$

We now can distinguish two different cases: (a) $T_0 > T_\mathrm{s}$, corresponding to the cooling process and (b) $T_0 > T_\mathrm{s}$, corresponding to the heating process.

Figure 6.16 shows the qualitative shapes of $T(t)$ for these two scenarios, i.e. cooling and heating.

Newton's Law of cooling has its limitations [7] and is applicable under the following limitations:

1. This Law applies to cooling by convection and radiation and not by radiation alone. To achieve this condition, the hot body is cooled in a uniform flow of air so that the radiation losses become small as compared to that due to forced convection.

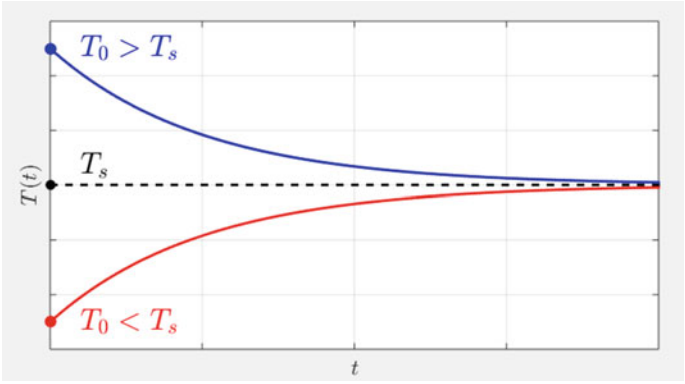


Fig. 6.16 Qualitative time histories for the Cooling and heating scenarios

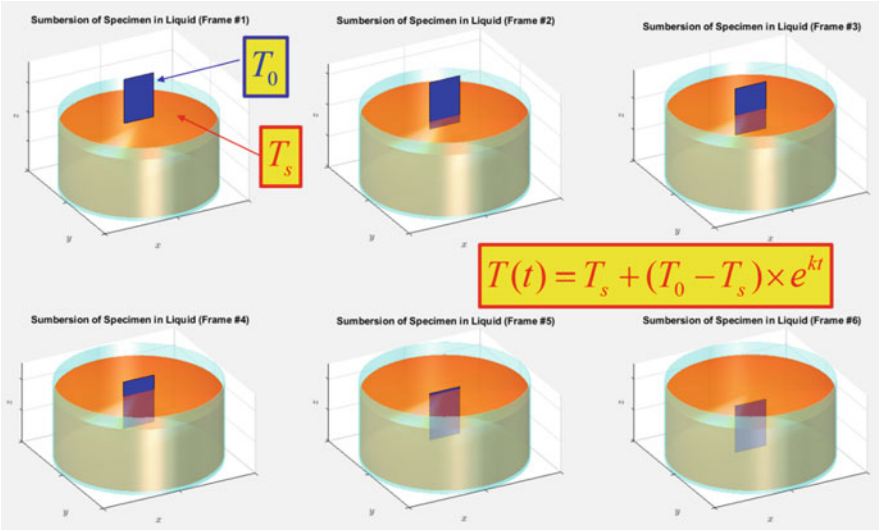


Fig. 6.17 Submersion of the metal specimen in the liquid with different temperature

2. This Law is applicable in still air only for a temperature difference of about 20 K or 30 K, but it is true for all temperature difference in conditions of forced convection of the air.

Newton's Law of Cooling (and Heating): Example-1 (Heating)

For illustration purposes, let us consider the following example. A small metal bar whose temperature is 30 °C is dropped into a container of 75 °C water. Submersion of the specimen is illustrated with Fig. 6.17, using MATLAB® graphics.

After 1 s, the temperature of the bar has increased by 1 °C (i.e. up to 31 °C).

(a) How long will it take for the temperature of the bar to reach 70 °C?
(b) How long will it take for the temperature of the bar to reach 74 °C?

To solve the task, we have to determine first the value of k in the Eq. (6.22). For this purpose, we use the data of the task and substitute known values in the Eq. (6.22) with the following rearrangements:

$$\begin{aligned} T(t) &= T_s + (T_0 - T_s)\,e^{kt_1} \\ 31 &= 75 + (30 - 75)\,e^{k\times 1} \\ (75 - 31) &= (75 - 30)\,e^{k} \\ e^{k} &= \left(\tfrac{75-31}{75-30}\right) = \left(\tfrac{44}{45}\right) \\ k &= \ln\left(\tfrac{44}{45}\right) = -0.0225 \end{aligned} \tag{6.24}$$

Application of the data for $T(t) = 70$ °C enables us to write:

$$\begin{aligned} T(t) &= T_s + (T_0 - T_s)\,e^{kt_{70}} \\ 70 &= 75 + (30 - 75)\,e^{kt_{70}} \\ (75 - 70) &= (75 - 30)\,e^{kt_{70}} \\ e^{kt_{70}} &= \left(\tfrac{75-70}{75-30}\right) = \tfrac{5}{45} = \tfrac{1}{9} \quad \Rightarrow \quad kt_{70} = \ln\left(\tfrac{1}{9}\right) \\ t_{70} &= \tfrac{1}{k}\ln\left(\tfrac{1}{9}\right) = 97.7724 \ \text{(s)} \end{aligned} \tag{6.25}$$

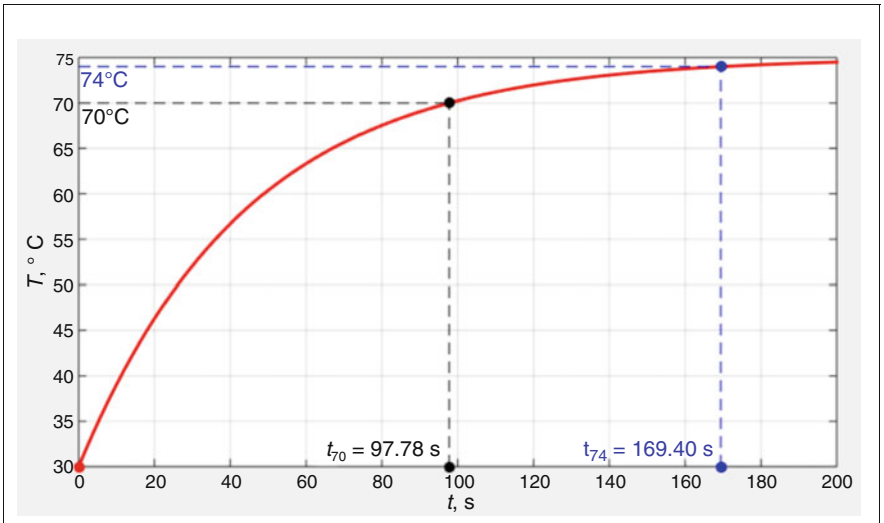
Similarly, application of the data for $T(t) = 74$ °C enables us to write:

$$\begin{aligned} T(t) &= T_s + (T_0 - T_s)\,e^{kt_{74}} \\ 74 &= 75 + (30 - 75)\,e^{kt_{74}} \\ (75 - 74) &= (75 - 30)\,e^{kt_{74}} \\ e^{kt_{74}} &= \left(\tfrac{75-74}{75-30}\right) = \tfrac{1}{45} \quad \Rightarrow \quad kt_{74} = \ln\left(\tfrac{1}{45}\right) \\ t_{74} &= \tfrac{1}{k}\ln\left(\tfrac{1}{45}\right) = 169.3894 \ \text{(s)} \end{aligned} \tag{6.26}$$

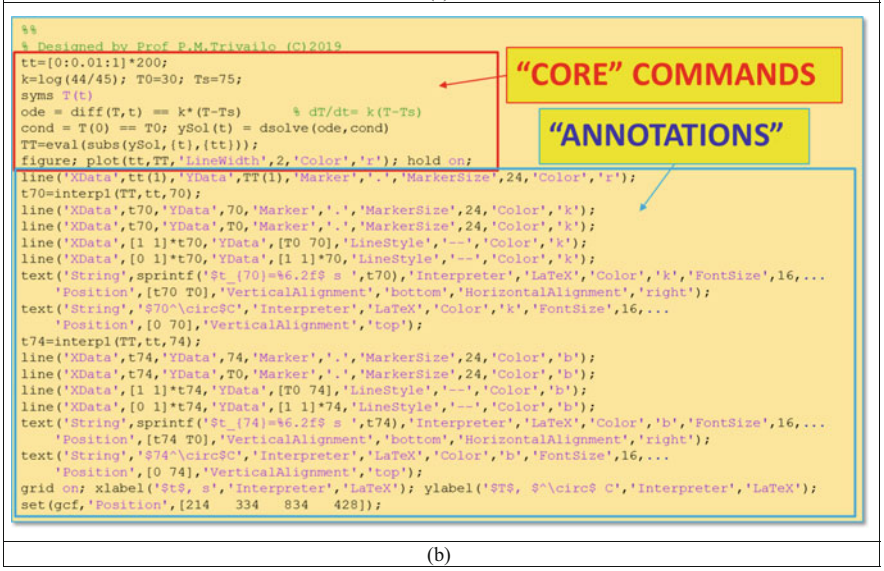
Heating of the liquid process can be represented graphically and is shown in Fig. 6.18. The annotated plot has also two cases (reach of 70 and 74 °C) marked with black and blue dots and lines.

For completeness, we also provide in Fig. 6.18 MATLAB® script, which has been used to generate plot in Fig. 6.18. It shows, that the only few commands required to plot the basic graph; however, many more commands may be required to annotate the graph.

We explore and illustrate significant benefits of the numerical solutions, using this example. Heating equations for the considered case can be easily solved, using MATLAB® "ode" integration procedure. The whole process is illustrated



(a)




```
%%
% Designed by Prof P.M.Trivailo (C)2019
tt=[0:0.01:1]*200;
k=log(44/45); T0=30; Ts=75;
syms T(t)
ode = diff(T,t) == k*(T-Ts)       % dT/dt= k(T-Ts)
cond = T(0) == T0; ySol(t) = dsolve(ode,cond)
TT=eval(subs(ySol,{t},{tt}));
figure; plot(tt,TT,'LineWidth',2,'Color','r'); hold on;
line('XData',tt(1),'YData',TT(1),'Marker','.','MarkerSize',24,'Color','r');
t70=interp1(TT,tt,70);
line('XData',t70,'YData',70,'Marker','.','MarkerSize',24,'Color','k');
line('XData',t70,'YData',T0,'Marker','.','MarkerSize',24,'Color','k');
line('XData',[1 1]*t70,'YData',[T0 70],'LineStyle','--','Color','k');
line('XData',[0 1]*t70,'YData',[1 1]*70,'LineStyle','--','Color','k');
text('String',sprintf('$t_{70}=%6.2f$ s ',t70),'Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[t70 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$70^\circ$C','Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[0 70],'VerticalAlignment','top');
t74=interp1(TT,tt,74);
line('XData',t74,'YData',74,'Marker','.','MarkerSize',24,'Color','b');
line('XData',t74,'YData',T0,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[1 1]*t74,'YData',[T0 74],'LineStyle','--','Color','b');
line('XData',[0 1]*t74,'YData',[1 1]*74,'LineStyle','--','Color','b');
text('String',sprintf('$t_{74}=%6.2f$ s ',t74),'Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[t74 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$74^\circ$C','Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[0 74],'VerticalAlignment','top');
grid on; xlabel('$t$, s','Interpreter','LaTeX'); ylabel('$T$, $^\circ$ C','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
```

(b)

Fig. 6.18 Modelling of a specimen, heated by a hotter surrounding liquid: (**a**) time history of the specimen; (**b**) MATLAB® script

in Fig. 6.19 with the annotated MATLAB® script, where equations (6.22) are programmed as *anonymous functions* and the problem is solved, using `ode45` MATLAB® procedure.

Numerical results are shown in Fig. 6.20 graphically, where the approximated numerical solution is shown with the continuous line and the superimposed exact analytical solution is shown with the dashed line.

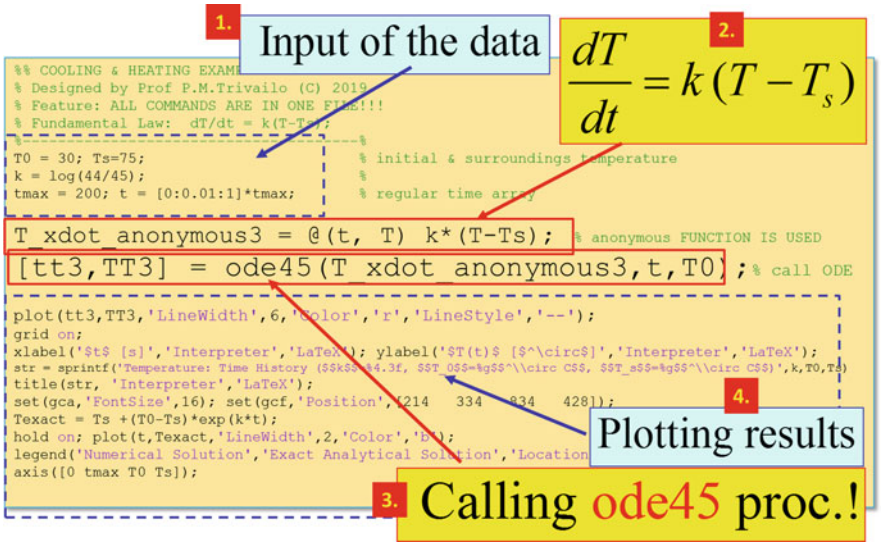


Fig. 6.19 Annotated MATLAB® script to solve task of heating of the specimen, using numerical method

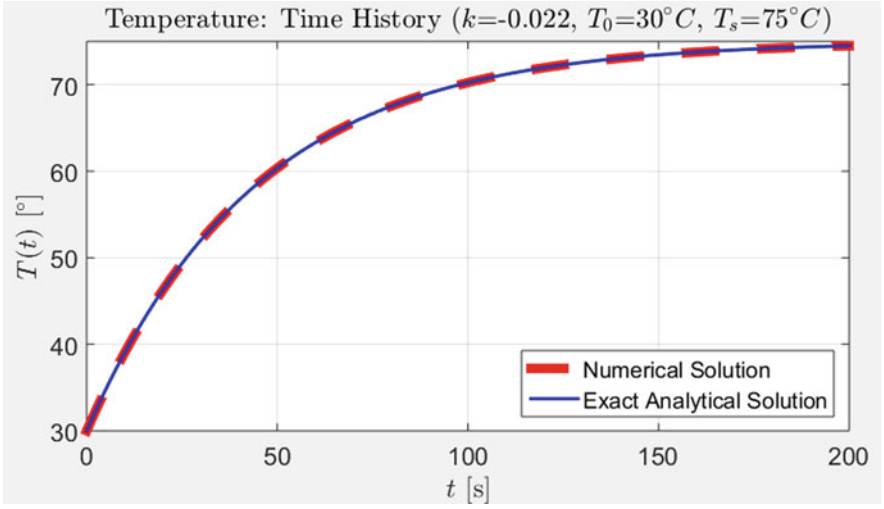


Fig. 6.20 Comparison of the numerical solution for the heated specimen against the exact analytical solution

It should be noted that (as shown in the script in Fig. 6.19), the anonymous function "T_xdot_anonymous3" (it can be given any other convenient name) does not explicitly use time *t*; however, it is a compulsory convention for this function to include *t* as a dummy variable for the "`ode45`" procedure to be able to

perform numerical integration of the relevant differential equation, expressed with the anonymous function.

Also, it is imperative to use in the `ode45` call exactly the same name as used for the anonymous function. At last, the name of the anonymous function should be in compliance of MATLAB® function names, i.e. should not have spaces, algebraic symbols, etc. For details, if necessary, please, refer to the complete MATLAB® documentation.

Newton's Law of Cooling (and Heating): Example-2 (Cooling)

You are given a very hot sample of metal, and wish to know its temperature. You have a thermometer, but it only measures up to 200 °C, and the metal is much hotter than that! You leave the metal in a room kept at 20 °C. After 6 min, it has cooled sufficiently that you can measure its temperature; it is 80 °C. After another 2 min it is 50 °C. What was the initial temperature of the metal? This problem is illustrated with Fig. 6.21, where the red curve shows expected tendency for the time history of the temperature of the specimen.

General solution of the Eq. (6.22) is

$$T(t) = T_s + (T_0 - T_s)\, e^{kt} \tag{6.27}$$

However, it cannot be immediately used, as k and T_0 are not known. To determine first the value of k, we can reset time at $t = 6$ s, introducing new supplementary time variable $\tau = t - 6$ (as shown in Fig. 6.21). Then we can apply Eq. (6.27), resulting in the follow process of relating the known data:

$$\begin{aligned} &T(\tau) = T_s + (T_0 - T_s)\, e^{k\tau} \quad \text{where } \tau = t - 6 \\ &50 = 20 + (80 - 20)\, e^{k\tau_2} \quad \Leftarrow \quad \text{for } \tau = 2 \\ &(50 - 20) = (80 - 20)\, e^{k\tau_2} \\ &e^{k\tau_2} = \left(\tfrac{50-20}{80-20}\right) = \tfrac{30}{60} = \tfrac{1}{2} \quad \Rightarrow \quad k\tau_2 = \ln\left(\tfrac{1}{2}\right) \\ &k = \tfrac{1}{\tau_2} \ln\left(\tfrac{1}{2}\right) = \tfrac{1}{2} \ln\left(\tfrac{1}{2}\right) = -0.3466 \ \ (1/\text{min}) \end{aligned} \tag{6.28}$$

By the way, it would be interesting to note the different temperature rate of change at different instants. For example, for $t = 6$ min

$$\left.\frac{dT}{dt}\right|_{t=6 \text{ min}} = k\,(T - T_s) = -0.3466\,(80 - 20) = -20.7944 \ \ (\text{deg}/\text{min}) \tag{6.29}$$

Two minutes later, when $t = 8$ min, the rate of temperature is reduced by the factor of 2:

$$\left.\frac{dT}{dt}\right|_{t=8 \text{ min}} = k\,(T - T_s) = -0.3466\,(50 - 20) = -10.3972 \ \ (\text{deg}/\text{min}) \tag{6.30}$$

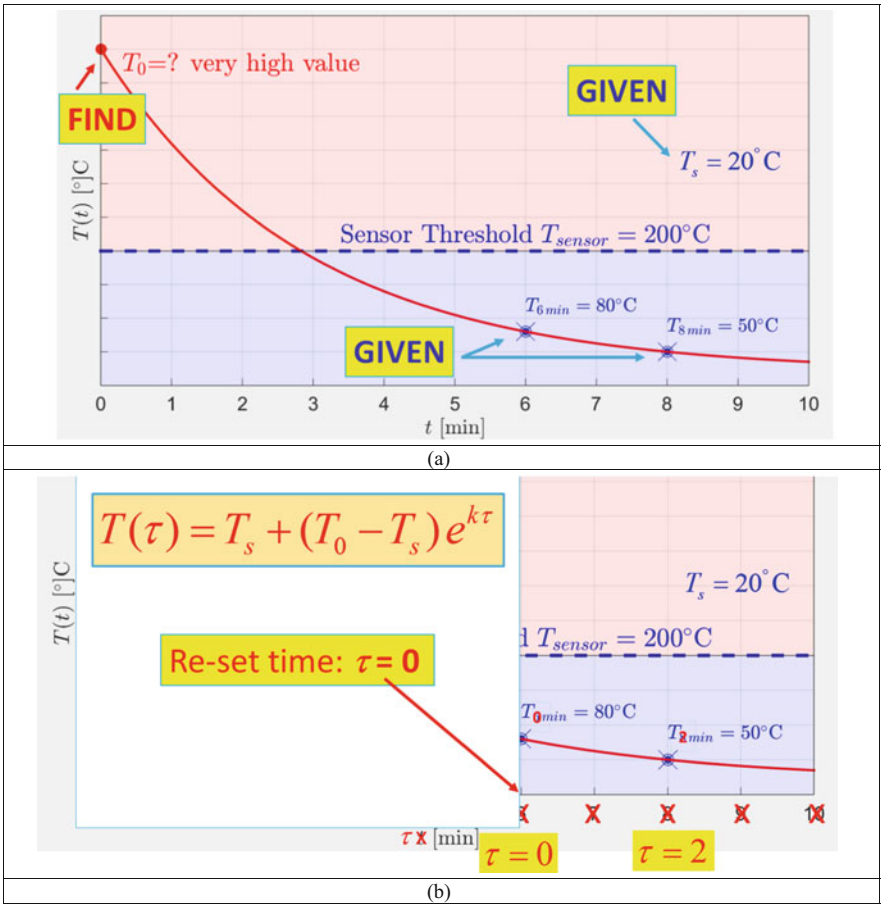


Fig. 6.21 Formulation of the cooling problem with known data is shown in blue, unknown data is shown in red: (**a**) initial formulation; (**b**) time is reset at $t = 6$ s and new time $\tau = t - 6$ is introduced

After the value of k was established, the value of T_0 is still not known. In view of the absence of critical data, in order to proceed with the quest to find a solution to the task, let us guess the unknown value, assuming that $T_0 = 200$ °C. With this in mind, solution to the task can be obtained, using numerical integration of the differential equation for the problem (6.27), using "`ode45`" procedure in MATLAB®. The annotated script is presented in Fig. 6.22.

Results of the simulation are presented in Fig. 6.23a. Graphical representation of results is very useful, as it immediately rejects assumption as invalid, because the resultant cooling curve does not pass any of the desired "waypoints", i.e. points $T = 80$ °C for $t = 6$ s and $T = 50$ °C for $t = 8$ s.

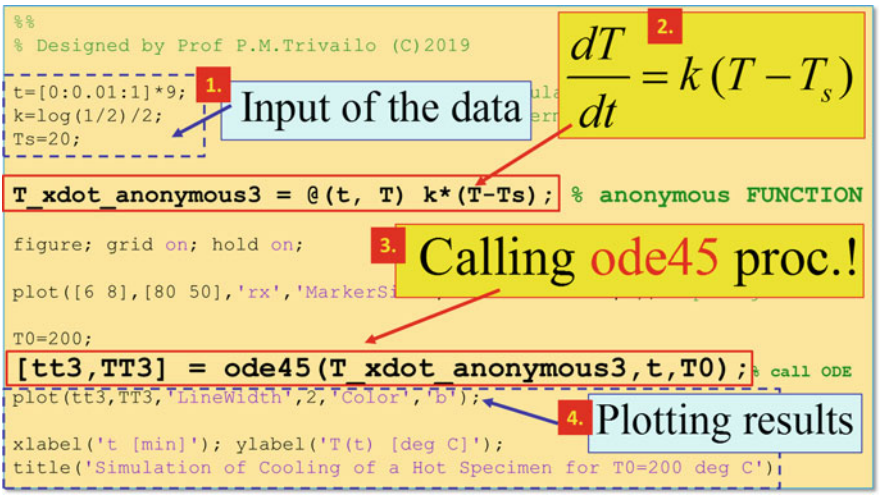


Fig. 6.22 Annotated MATLAB® script for solving cooling problem, where one of the unknowns is guessed as $T_0 = 200\,^{\circ}\text{C}$

This unsuccessful attempt prompts a strategy, which is widely used in simulations of engineering systems: utilizing the advantage of the numerical methods, often enabling fast solution of the task, we can attempt to guess the unknown value of T_0 as an array of several values. Analysing afterwards results of the simulations for these multiple "guess" cases, similar to the previous attempt, we will reject the unsuitable values of T_0, which would not satisfy conditions of the task and will retain the value which is within the accuracy tolerance for the task.

Figure 6.23b shows in blue colour the "guessed" solutions, which should be abandoned, as not satisfying "waypoints" conditions of the task. And the red colour is used to distinguish the cooling curve, which satisfies these conditions and which can be adopted as being solution of the task.

Pond Pollution Example

Consider three ponds connected by streams (see Fig. 6.24a). The first pond has a pollution source, which spreads via the connecting streams to the other ponds. The goal of the exercise is to determine the amount of pollutant in each pond.

Let us introduce the following notations and consider an illustrated example with particular data. We denote volumes of three ponds 1, 2, 3, connected by streams, as V_1, V_2, V_3. The pollution source $f(t)$ is in pond 1. For a specific numerical example, take $f_i/V_i = 0.03$, $1 \leq i \leq 3$, and let $f(t) = 3$ kg/h for the first 48 h, thereafter $f(t) = 0$. We expect due to uniform mixing that after a long time there will be $3 * 48 = 144$ kg of pollutant uniformly deposited, which is 48 kg per pond. Initially, $x_1(0) = x_2\,(0) = x_3(0) = 0$, if the ponds were pristine. The other specifications for the case are:

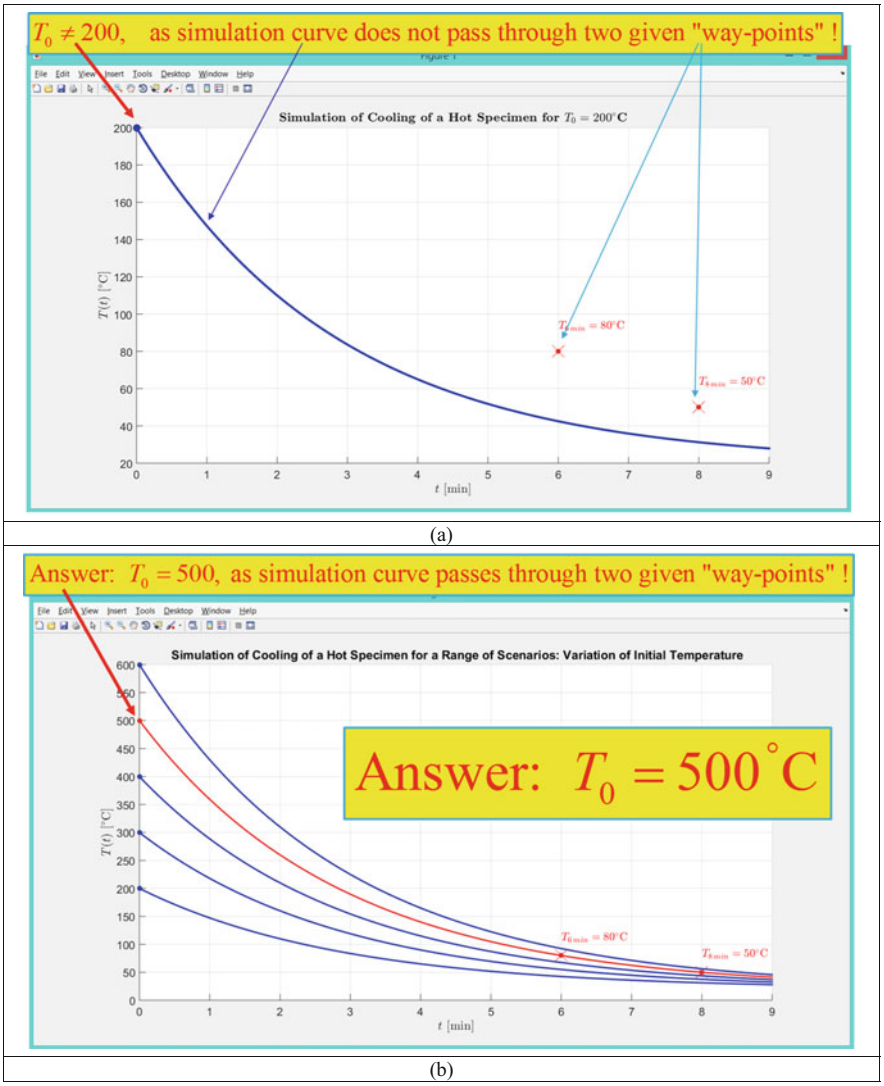
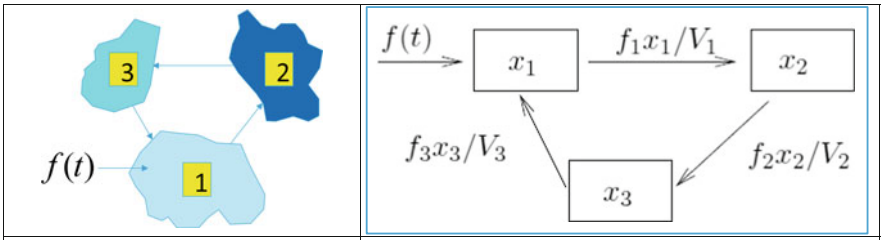


Fig. 6.23 Results of the simulation of the problem, where one of the unknowns T_0: (**a**) is guessed as a single value $T_0 = 200$ °C. (**b**) Are guessed as an array of values $T_0 = [200{:}100{:}600]$ °C

- Symbol $f(t)$ is the pollutant flow rate into pond 1 (kg/h).
- Symbols f_1, f_2, f_3 denote the pollutant flow rates out of ponds 1, 2, 3, respectively (kg/h).
- It is assumed that the pollutant is well-mixed in each pond.
- The three ponds have volumes V_1, V_2, V_3 (m^3), which remain constant.



(a) (b)

Fig. 6.24 System of three ponds: (**a**) sketch with shown interconnection and pollutant; (**b**) notations for the system of differential equations

- Symbols $x_1(t)$, $x_2(t)$, $x_3(t)$ denote the amount (kg) of pollutant in ponds 1, 2, 3, respectively.

The pollutant flux is the flow rate times the pollutant concentration, e.g. pond 1 is emptied with flux f_1 times $x_1(t)/V_1$. A compartment analysis is summarized in the diagram in Fig. 6.24b. The diagram plus compartment analysis gives the following differential equations:

$$\left\{\begin{array}{l} \frac{dx_1(t)}{dt} = \frac{f_3}{V_3}x_3(t) - \frac{f_1}{V_1}x_1(t) + f(t) \\ \frac{dx_2(t)}{dt} = \frac{f_1}{V_1}x_1(t) - \frac{f_2}{V_2}x_2(t) \\ \frac{dx_3(t)}{dt} = \frac{f_2}{V_2}x_2(t) - \frac{f_3}{V_3}x_3(t) \end{array}\right. \Rightarrow \left\{\begin{array}{l} \frac{dx_1(t)}{dt} = 0.03x_3(t) - 0.03x_1(t) + 3 \\ \frac{dx_2(t)}{dt} = 0.03x_1(t) - 0.03x_2(t) \\ \frac{dx_3(t)}{dt} = 0.03x_2(t) - 0.03x_3(t) \end{array}\right. \tag{6.31}$$

The new element in this example is derivation of the exact analytical solution of the system of Eq. (6.31), using *symbolic capabilities* available from MATLAB®. The corresponding MATLAB® script is presented in Fig. 6.25.

Analytical expressions for the solved problem are as follows:

$$\begin{aligned} x_1(t) &= t - \frac{100\,\cos\left(\frac{3\sqrt{3}t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} + \frac{100}{3} \\ x_2(t) &= t - \frac{200\sqrt{3}\sin\left(\frac{3\sqrt{3}t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} \\ x_3(t) &= t + \frac{100\,\cos\left(\frac{3\sqrt{3}t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} - \frac{100}{3} \end{aligned} \tag{6.32}$$

Utilizing further MATLAB® symbolic plotting commands (`fplot` or `ezplot` instead, if MATLAB® version before R2016a is used), we can plot solutions, given by Eqs. (6.32). They are presented in Fig. 6.26.

With the exact benchmark results, we will solve the same task, using "`ode`" numerical solvers for the matrix ordinary differential equations, available in MAT-

```
%% POLLUTED PONDS EXAMPLE: SYMBOLIC SOLUTION
% Designed by Prof P.M.Trivailo (C) 2019
%----------------------------------------
% Define matrices and the matrix equation.
syms x1(t) x2(t) x3(t)
r=[-1  0  1;  1 -1  0;  0  1 -1]*0.03;
D=[3; 0; 0]; X = [x1; x2; x3];
odes = diff(X) == r*X + D

% Solve the matrix equation using "dsolve".
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes);

% Solve the system with zero initial conditions.
C = X(0) == [0; 0; 0];
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes,C)
```

Fig. 6.25 MATLAB® script for solving Eqs. (6.31), using MATLAB® symbolic capabilities

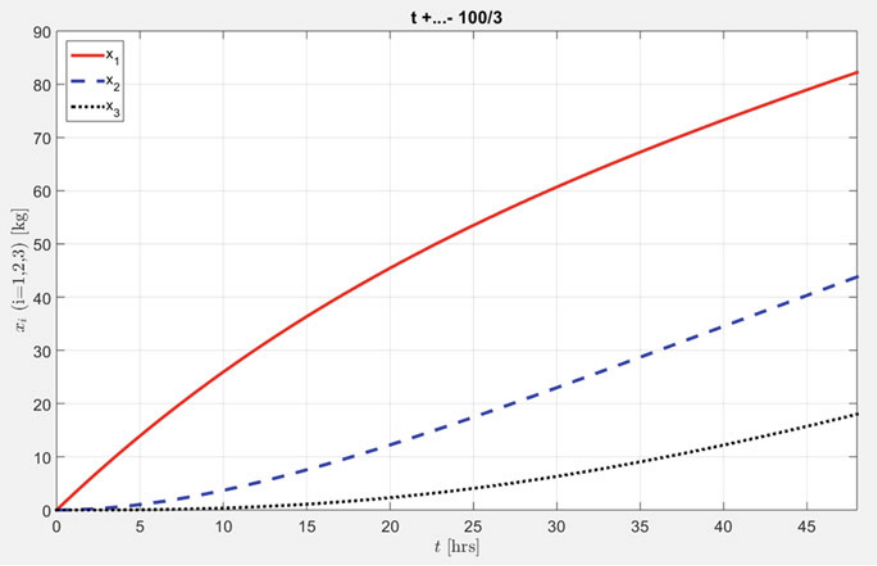


Fig. 6.26 MATLAB® exact solutions for pollutants in lakes 1, 2, 3, obtained with MATLAB® symbolic solving capabilities and plotted, using MATLAB® symbolic plotting capabilities

LAB®. The annotated script, performing this task, is given in Fig. 6.27 with plotted numerical results shown in Fig. 6.28. Comparison of the results in Figs. 6.26 and 6.28 demonstrates close correlation between analytical exact and numerical approximate solutions.

```
%% POLLUTED PONDS EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
%----------------------------------------
clear; close all; clc;
r=[-1  0  1;...
    1 -1  0;...
    0  1 -1]*0.03;
D=[3; 0; 0];
tmax=24*2; t=[0:0.01:1]*tmax; % hrs
mass_xdot_anonymous = @(t, x) (r*x+D);
[tt,zz]=ode45(mass_xdot_anonymous,t,[0;0;0]);
plot(tt,zz(:,1),'r-',tt,zz(:,2),'b--',tt,zz(:,3),'k:','LineWidth',3);
legend('x_1','x_2','x_3','Location','NorthWest');
xlabel('$t$ [hrs]','Interpreter','LaTeX');
ylabel('$x_i$ (i=1,2,3) [kg]','Interpreter','LaTeX');
grid on
strl='Final pollutant amounts in ponds: ';
str=sprintf('%s$$x_1$$=%5.2f kg; $$x_2$$=%5.2f kg; $$x_3$$=%5.2f kg; TOTAL=%6.2f kg',strl,zz(end,:),sum(zz(end,:)));
title(str,'Interpreter','LaTeX')
set(gca,'FontSize',16);  set(gcf,'Position',[488 -90 1361 852]);
```

CORE or "HEART" OF THE SCRIPT !!!

"DECORATIONS" commands

Fig. 6.27 MATLAB® script for solving "three ponds" task numerically

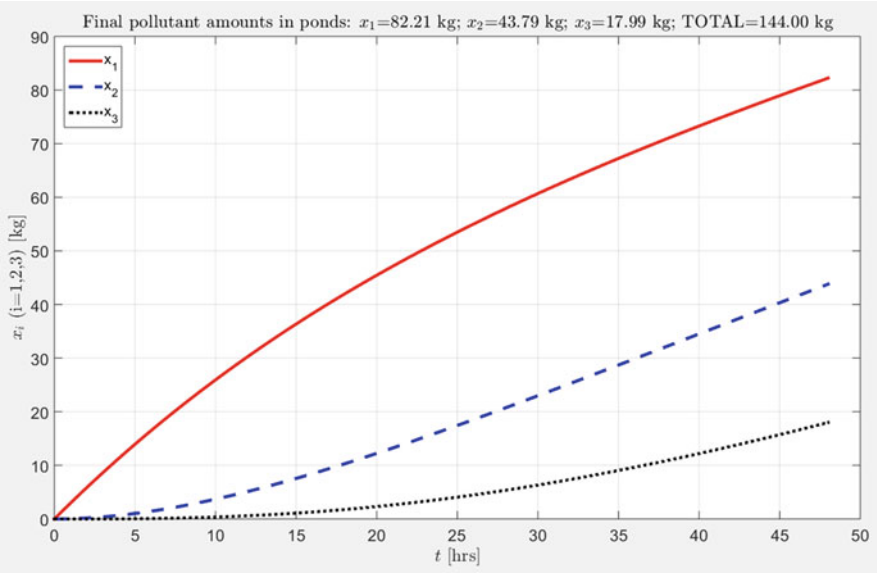


Fig. 6.28 MATLAB® approximate numerical solutions for pollutants in lakes 1, 2, 3, obtained with MATLAB® solving differential equations capabilities

6.2.5 *Numerical Simulation of the Problems, Described with the Second Order Ordinary Differential Equations*

Falling Mass Example-1 (No Air Resistance)

As the next representative case, let us consider a falling mass m, assuming no air resistance. This task is illustrated with Fig. 6.29. In this case, for solution process and further interpretation of results, we need to assign positive sign conventions. Let us place the coordinate system zOh at the datum level and direct axis z up. In this case, application of Newton's Second Law to the free-body diagram of the mass, after obvious simplifications, leads to the following equation of motion of the falling mass, which is a *second order ordinary differential equation*:

$$\ddot{z} = g \tag{6.33}$$

Of course, with this simplified formulation of the task, it can be solved analytically. Integration of the differential equation (6.33) allows analytical solution:

$$\frac{d^2z}{dt^2} = -g; \quad \Rightarrow \quad \frac{dz}{dt} = -gt + C_1; \quad \Rightarrow \quad z = -\frac{gt^2}{2} + C_1 t + C_2 \tag{6.34}$$

Application of Initial Conditions enables us to find the values of constants C_1 and C_2:

$$\begin{array}{l} z|_{t=0} = z_0; \quad \Rightarrow \quad z_0 = C_2 \\ \frac{dz}{dt}\big|_{t=0} = v_0; \quad \Rightarrow \quad v_0 = C_1 \end{array} \tag{6.35}$$

Therefore, the exact analytical solution of the task (which is valid within the assumptions in the task) can be presented as follows:

$$z(t) = z_0 + v_0 t - \frac{gt^2}{2} \tag{6.36}$$

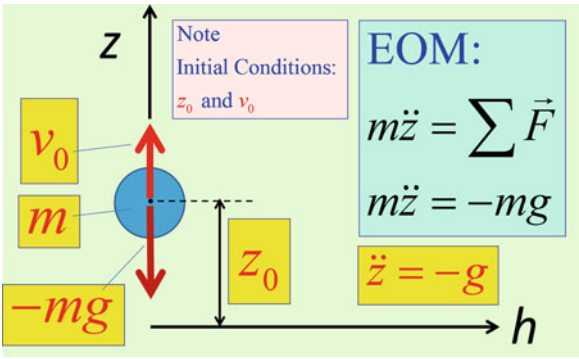
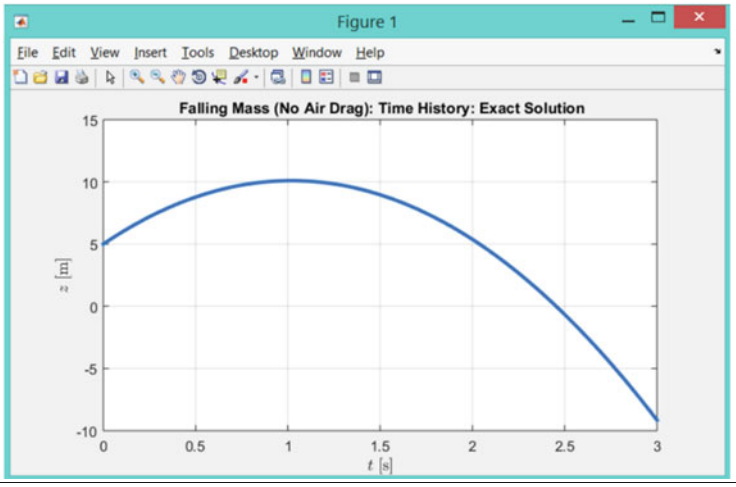


Fig. 6.29 Model of a falling mass with accepted notations shown

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
%---------------------------------------
        %                 ^ z
% Initial Conditions:     |
v0=10;   % m/s            |
z0=5;    % m              |
tmax=3;  % s              +--------------> h
t=[0:0.01:1]*tmax;
zz=z0+v0*t-g*t.^2/2;  % Note: pseudo-square operation !!!
plot(tt,zz,'LineWidth',3,'Color',[0 0.5 1]);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History: Exact Solution');
set(gca, 'FontSize',12);
```

(a)



(b)

Fig. 6.30 Simulation of the falling mass with air resistance ignored: (**a**) MATLAB® script implementing exact solution; (**b**) exact solution plotted as $z = z(t)$

This solution can be programmed as MATLAB® script and can be used to generate solution plot of the height of the mass as a function of time $z = z(t)$. These are presented in Fig. 6.30.

With this solution, which we can use as a benchmark, let us solve the same task, using numerical technique and employing one of the "ode" MATLAB® differential equations solvers. However, these solvers are applicable to the first-order differential equations and their systems, therefore, in order to access "ode" solvers, we will need first to reformulate the task.

The aim of this supplementary reformulation step is to reduce second order equation (6.33) to the system of the *first order* differential equations. This process

is as follows. Instead of z, used in the initial formulation, let us introduce new, so-called state variables x_1 and x_2:

$$\begin{cases} x_1 = z \\ x_1 = \dot{z} \end{cases} \tag{6.37}$$

Then, from Eq. (6.37) $\dot{x}_1 = x_2$, however, this relationship just reflects the choice of the state variables, and does not reflect the physical equation of motion (6.34), which can be rewritten in terms of the new variables as $\dot{x}_2 = -g$. Therefore, instead of the initial single ordinary differential equation (6.33) of the second order, written in terms of z, we can produce a system of two ordinary differential equations of the first order

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases} \tag{6.38}$$

System of Eq. (6.38) is an equivalent to the Eq. (6.33); however, for (6.38) the "`ode`" MATLAB® procedures can be applied.

Corresponding annotated MATLAB® script and resulting plot for the first state x_1, being a nick-name to z, are shown in Fig. 6.31a.

Comparison of the results in Figs. 6.30b and 6.31b shows that they are identical within the solution tolerance. However, it should be remembered, that analytical exact solutions are not always available and numerical technique may be the only reasonable alternative.

For completeness, let us mention that in the example provided, where equations of motion (6.38) can be presented in the compact form, we used an *anonymous function*. This single function can be applied for the system of falling masses, and an illustration example for two simultaneously launched masses is shown in Fig. 6.32. It features supplementing animation of the numerically simulated results.

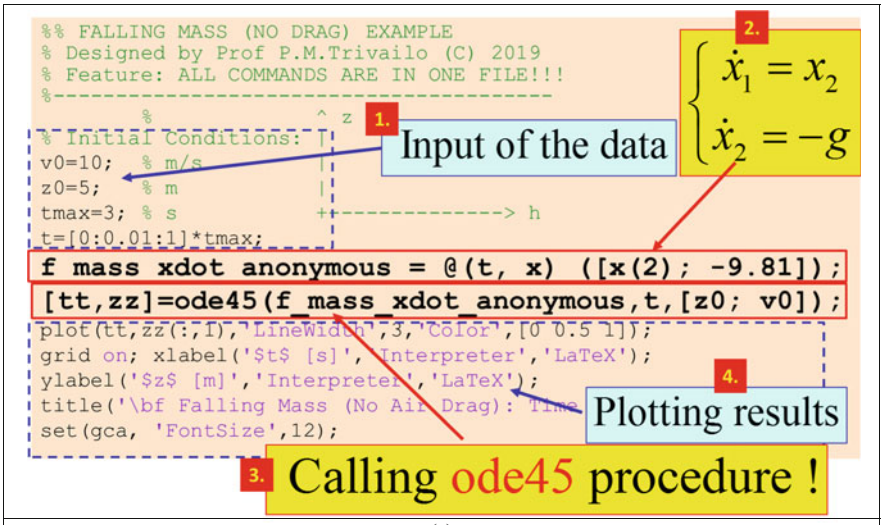
Vibration of a Single Mass-Spring-Damper Linear System Example

Let us consider a Virtual Reality model of a mass-spring linear system with constant k and m, shown in Fig. 6.33a. In this example, translational displacement of the mass is denoted as $q(t)$.

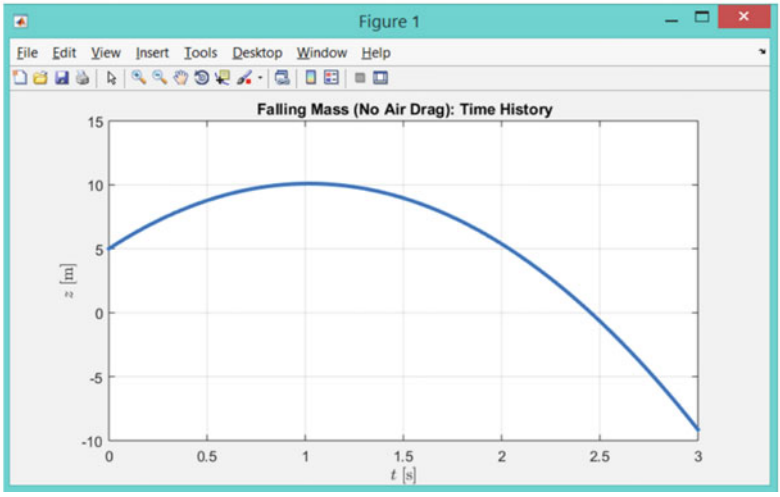
In order to derive equation of motion, we apply "positive" displacement to the mass **m** (Fig. 6.33b); then remove constraints, applying equivalent forces to keep the system in equilibrium (Fig. 6.33c).

The resultant free body diagram is shown in Fig. 6.33d and applying Newton's Second Law for the "m = const" case, we can write the differential equation of motion of the system as follows:

$$\sum F = ma \Rightarrow m\ddot{q} = -kq \tag{6.39}$$



(a)

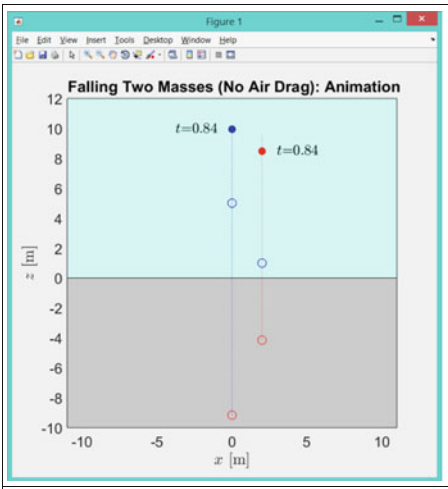


(b)

Fig. 6.31 Simulation of the falling mass with air resistance ignored: (**a**) MATLAB® script implementing exact solution; (**b**) exact solution plotted as $z = z(t)$

If needed, the excitation force F and viscous damping c can be also added (see corresponding system in Fig. 6.34), which would expand this equation to the flowing format:

$$m\ddot{q} + c\dot{q} + kq = F \tag{6.40}$$



(a)

```
%% Basic solution/animation script:
% Designed by Prof P.M.Trivailo (C) 2019
%--------------------------
t=[0:0.005:1]*3;
v0=10; z0=5; v02=13; z02=1;

fm_xdot = @(t, x) ([x(2); -9.81]);

[tt,zz]=ode45(fm_xdot,t,[z0; v0]);
[tt2,zz2]=ode45(fm_xdot,t,[z02; v02]);

g=line('XData',0,'YData',z0,...
    'Marker','.','MarkerSize',36);
g2=line('XData',2,'YData',z0,...
'Marker','.','MarkerSize',36,'Color','r');

axis([-11 11 -10 12]); grid on;

for i=1:length(tt)
    set(g,'YData',zz(i,1));
    set(g2,'YData',zz2(i,1));
    drawnow;
    pause(0.01);
end
```

(b)

Fig. 6.32 Simulation of two falling masses with air resistance ignored: (**a**) animation screen; (**b**) MATLAB® basic script for simulation and animation of two falling masses

The feature of this example is in presenting Eq. (6.40) in the matrix state-space form:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{6.41}$$

or the same can be written in the vector format:

$$\begin{aligned} \dot{\mathrm{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{x} &= \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix} \\ A &= \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \end{aligned} \tag{6.42}$$

The last equation can be programmed in MATLAB® and the system can be simulated. For certainty, let us assume the following hypothetical parameters: $m = 10$ kg; $k = 100$ N/m; $c = 10$ N*s/m; the initial conditions are assumed as follows: $q0 = 3$ m; $q\_dot0 = 15$ m/s.

Proposed simulation script and results of the simulation are shown in Figs. 6.35 and 6.36.

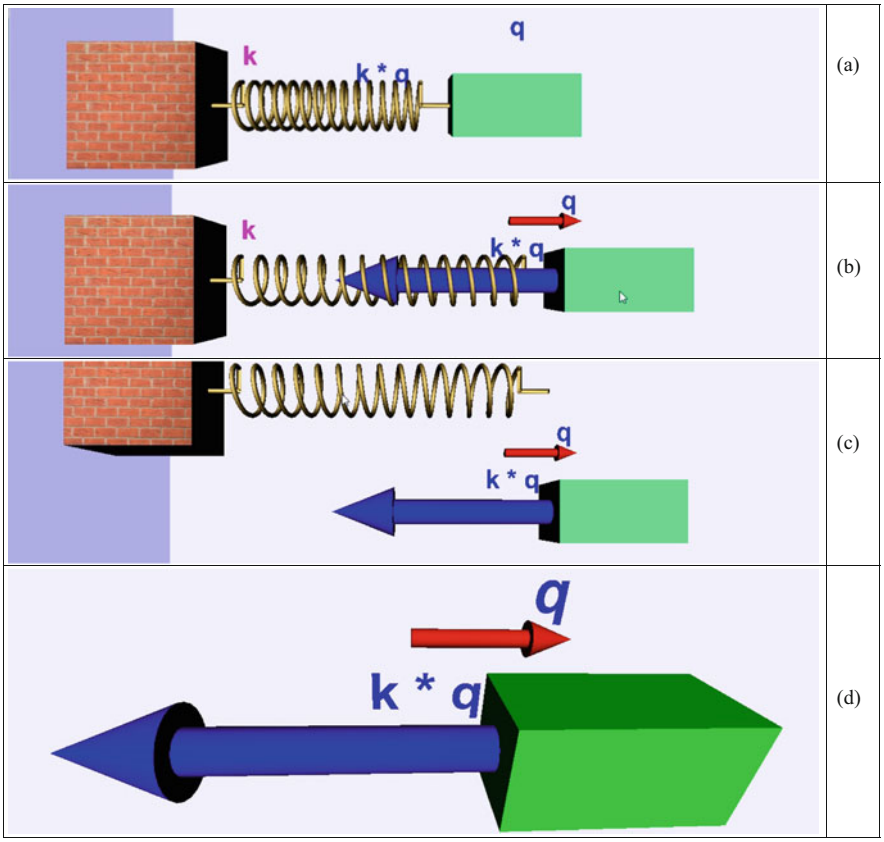
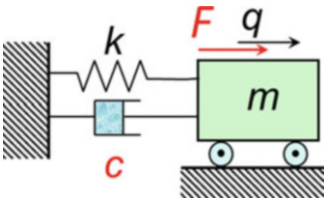


Fig. 6.33 Virtual Reality mass-spring system: (**a**) notations, (**b**) positive displacement is applied to the mass m; (**c**) constraints removed and equivalent forces applied to maintain equilibrium; (**d**) free body diagram

Fig. 6.34
Mass-spring-viscous damper system



Vibration of the Two Mass-Spring Linear System Example

Let us consider a two-DOFs mass-spring system. Its sketch is given in Fig. 6.37a.

Application of Newton's Second Law to the free-body diagrams in Fig. 6.37e, f enables us to write two differential equations (of the second order each) of motion:

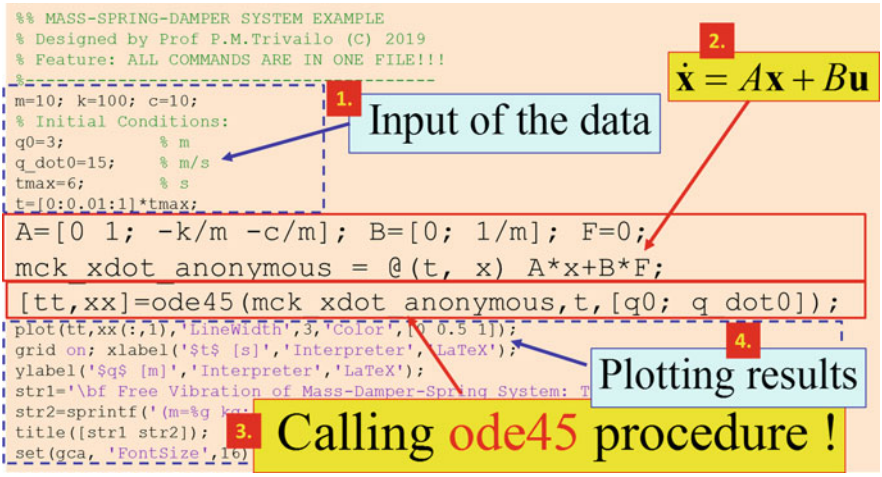


Fig. 6.35 Annotated MATLAB® script for simulating mass-spring-viscous damper system

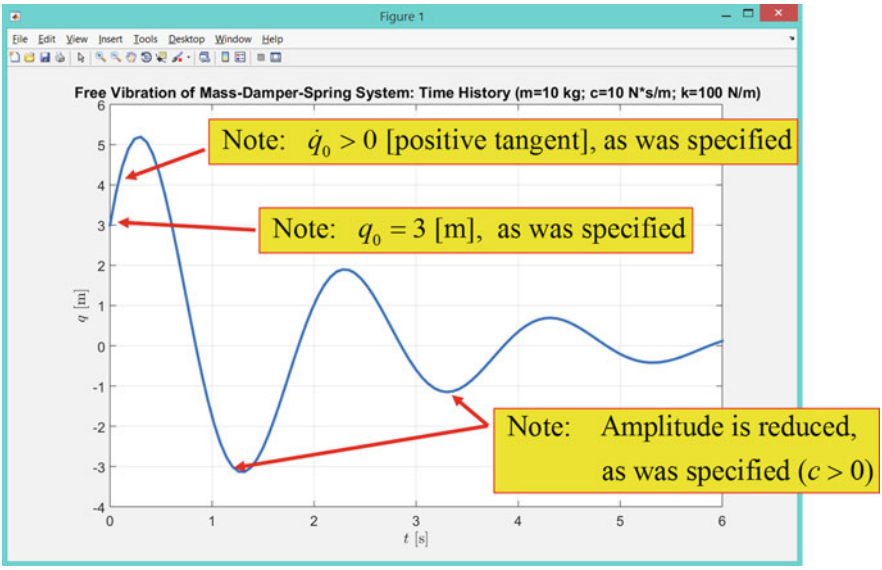


Fig. 6.36 Annotated results of the simulation of the mass-spring-viscous damper system

$$\begin{cases} m_1\ddot{q}_1 + (k_1 + k_2)\, q_1 - k_2 q_2 = 0 \\ m_2\ddot{q}_2 - k_2 q_1 + (k_2 + k_3)\, q_2 = 0 \end{cases} \tag{6.43}$$

For convenience in application of the simulation methods, these equations can be rewritten in the matrix format as follows:

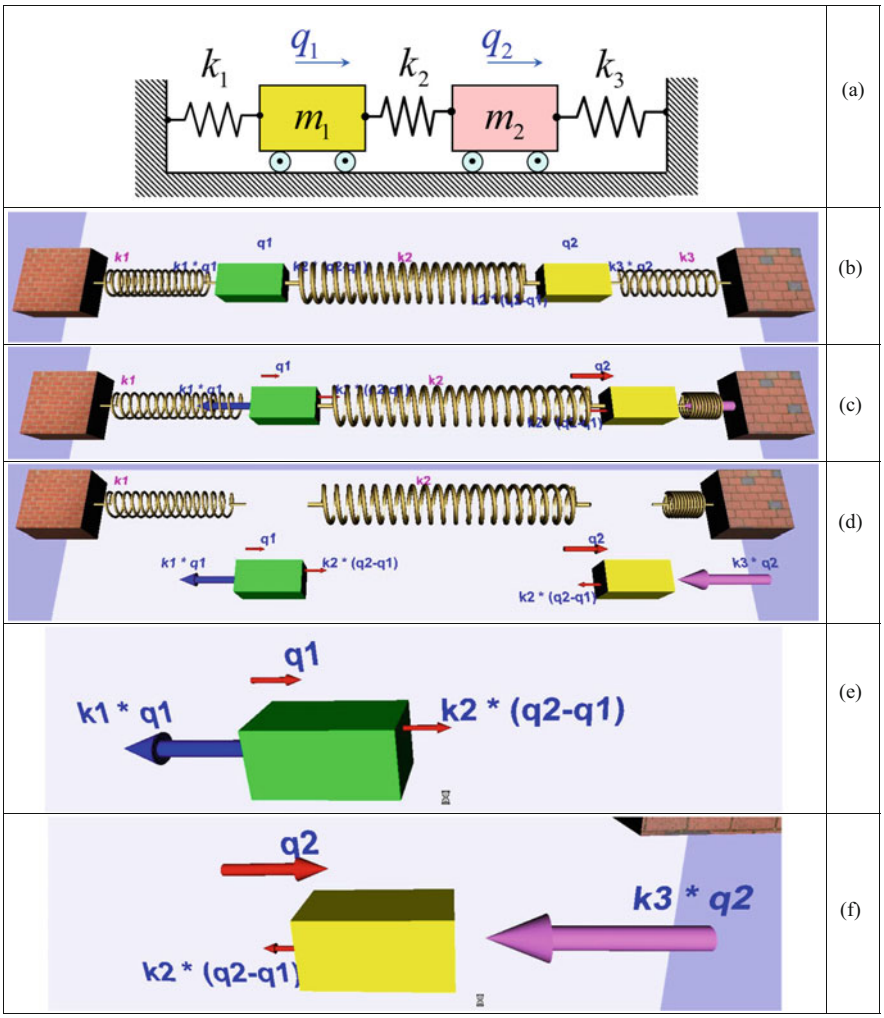


Fig. 6.37 Two DOFs mass-spring system: (**a**) sketch; (**b**) notations in VR; (**c**) positive displacements q_1 and q_2 applied; (**d**) constraints removed and equivalent forces are applied; (**e**) free-body diagram for mass m_1; (**f**) free-body diagram for mass m_2

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad \text{or}$$
$$[m]\{\ddot{q}\} + [k]\{q\} = \{0\} \quad \text{or (with damping and excitation forces added)} \tag{6.44}$$
$$[m]\{\ddot{q}\} + [c]\{\dot{q}\} + [k]\{q\} = \{F\}$$

where $[m]$, $[c]$ and $[k]$ are known as mass, viscous damping and stiffness matrices of the system and for the particular case of the mass-spring system, similarly shown in Fig. 6.37a are equal to:

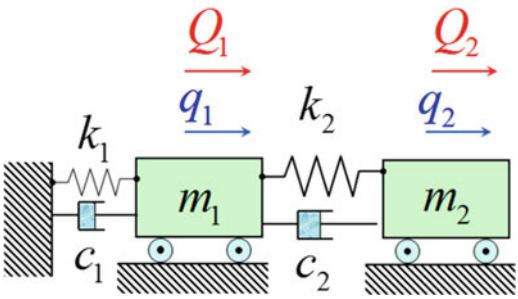


Fig. 6.38 Two DOFs mass-spring-damper system (another example of element's arrangement, different from Fig. 6.37)

$$[m] = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}; \quad [c] = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix}; \quad [k] = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \tag{6.45}$$

It should be emphasized that the structure of matrices $[c]$ and $[k]$ depends upon the way the masses m_1 and m_2 are interconnected. For example, for another case, shown in Fig. 6.38, the system matrices are slightly different from those, given by Eq. (6.45) and are presented by the following relationships:

$$[m] = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}; \quad [c] = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}; \quad [k] = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \tag{6.46}$$

Very often, especially in control applications, matrix Eq. (6.44) is further rewritten in so-called *state-space format* $\{\dot{x}\} = [A]\{x\} + [B]\{u\}$; also it is supplemented with the so-called *output equation* $\{y\} = [C]\{x\} + [D]\{u\}$and the system of equations for simulation purposes can be rewritten as:

$$\begin{cases} \{\dot{x}\} = [A]\{x\} + [B]\{u\} \\ \{y\} = [C]\{x\} + [D]\{u\} \end{cases} \tag{6.47}$$

where

$$[A] = \left[\begin{array}{c|c} [0] & [1] \\ \hline -[m]^{-1} \times [k] & -[m]^{-1} \times [c] \end{array}\right]; \quad [B] = \begin{bmatrix} [0] \\ \hline [m]^{-1} \end{bmatrix}; \quad \{u\} = \begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix} \tag{6.48}$$

Contents of the matrices $[C]$ and $[D]$ are assigned by the designer; however, often, for $[C]$ identity matrix and for $[B]$—zero matrix (of appropriate consistent dimensions) are used:

$$[C] = [1]; \quad [B] = [0] \tag{6.49}$$

enabling to get all system's state in the output vector $\{y\}$.

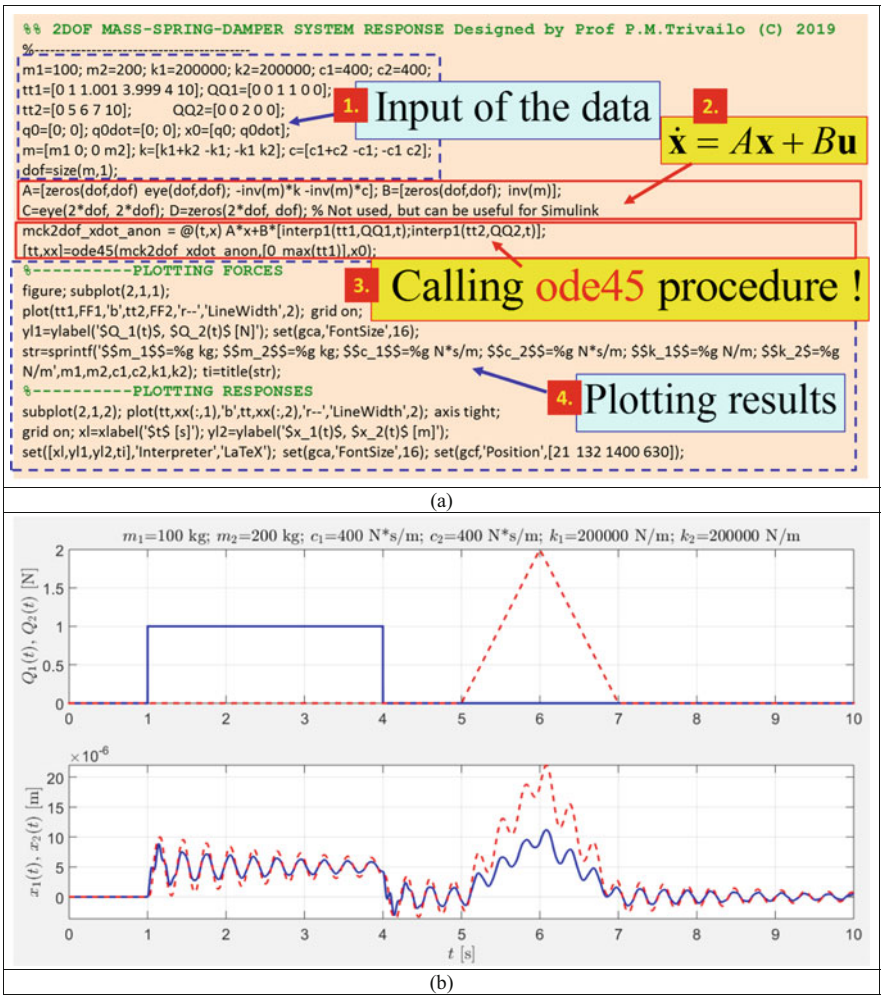


Fig. 6.39 Simulation of the response of the 2DOF mass-spring-viscous damper system, excited with two external forces: (**a**) annotated MATLAB® script; (**b**) response results

To illustrate application of Eqs. (6.47)–(6.49), in Fig. 6.39, we attach the MATLAB® script and results of the simulation of the forced response of the 2DOFs system, presented in Fig. 6.38.

6.3 Modelling and Simulation of Non-linear Systems: Examples of the Selected Simulations

6.3.1 Oscillations of the Pendulum: Comparison of the Results for the Linear and Non-linear Models

The word "pendulum" is from Latin *pendulus*, meaning "hanging". A pendulum is a weight suspended from a pivot so that it can swing freely (see Fig. 6.40). When a pendulum is displaced sideways from its resting, equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. When released, the restoring force acting on the pendulum's mass causes it to oscillate about the equilibrium position, swinging back and forth.

The time for one complete cycle, a left swing and a right swing, is called the period. The period T depends on the length of the pendulum L and also to a slight degree on the amplitude, the width of the pendulum's swing.

For small deflection angles, the period of swinging oscillations can be calculated, using the following relationship:

$$T \approx 2\pi\sqrt{\frac{L}{g}} \tag{6.50}$$

The differential equation of motion for the swinging pendulum can be derived in many different ways. Let us consider Newton's Second Law in the vector format, as applied to *constant* mass m:

$$\sum \vec{\mathbf{F}} = m\ddot{\vec{\mathbf{r}}} \tag{6.51}$$

This relationship for the translational motion is equivalent to the following equation, applicable to the rotational motion of the system with constant moment of inertia I_O, rotating about point O:

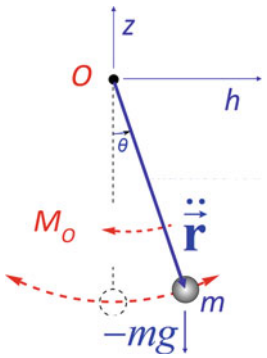


Fig. 6.40 Pendulum: notations and sign conventions

$$\sum M_O = I_O \ddot{\theta} \tag{6.52}$$

Developing the left-hand side of the Eq. (6.50) $\sum M_O = -mg \sin\theta \times L$ and developing the right-hand side of the Eq. (6.51) $I_O \frac{d^2\theta}{dt^2} = mL^2 \frac{d^2\theta}{dt^2}$, we can derive differential equation of motion for the pendulum, which is a *non-linear* equation:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \sin\theta = 0 \tag{6.53}$$

This single ordinary non-linear differential equation of the second order can be now rewritten as a non-linear system of two differential equations of the first order:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{g}{L} \sin\theta \end{cases} \tag{6.54}$$

These equations can be solved, using exactly the same principles and techniques, as were used for the linear systems. As in the examples above, we can describe the differential equations (6.54), using anonymous function (or use a separate function, if equations are not given with compact expressions [8]). The proposed annotated MATLAB® script is shown in Fig. 6.41.

The results of the simulation are presented in Fig. 6.42.

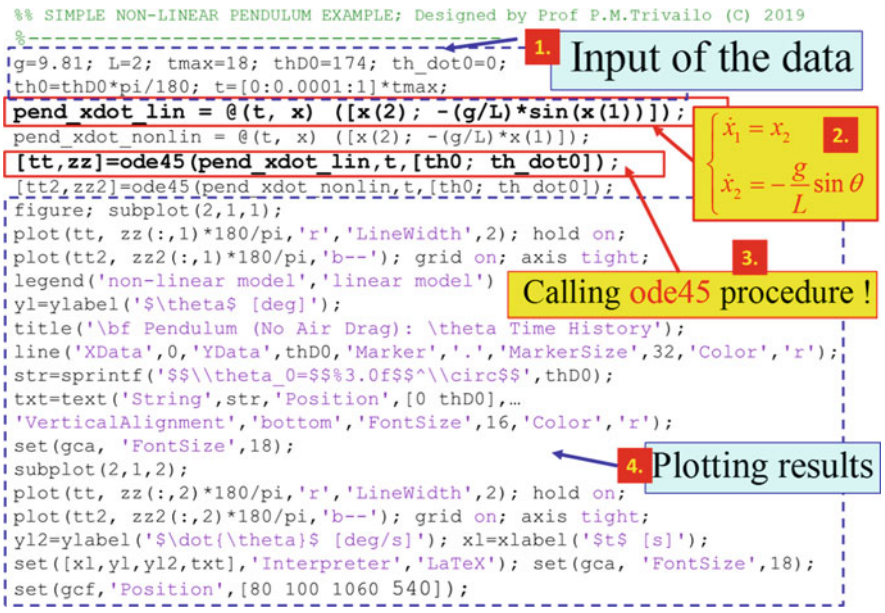


Fig. 6.41 Annotated MATLAB® script to simulate oscillations of the pendulum

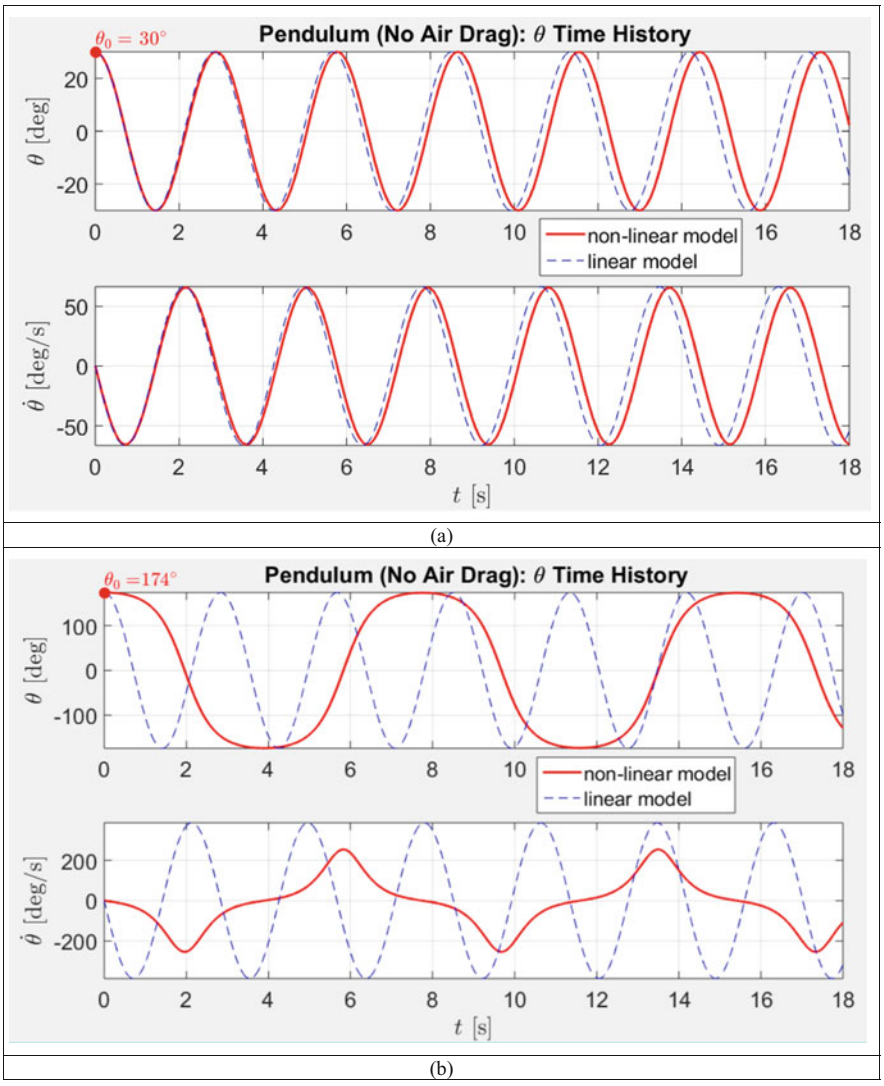


Fig. 6.42 Simulation results for the swinging pendulum, excited via initial conditions: (**a**) $\theta_0 = 30°$ and $\dot{\theta}_0 = 0$ case; (**b**) $\theta_0 = 174°$ and $\dot{\theta}_0 = 0$ case

The first plot in Fig. 6.42a has results for the non-linear model (shown in continuous red lines) and simplified, linear model (shown with dashed blue lines), in both cases excited with the same initial conditions, involving $\theta_0 = 30°$ and $\dot{\theta}_0 = 0$. The second plot in Fig. 6.42b has also results for non-linear and linear models, however, excited with conditions, involving $\theta_0 = 174°$ and $\dot{\theta}_0 = 0$, i.e. having much larger initial deflection angle θ_0.

It is obvious that for relatively small angles, including the $\theta_0 = 30°$, non-linear and linear models produce comparable results (Fig. 6.42a). However, for large angles θ, results for the non-linear model are very different from the results related to the linear system. These differences are very significant for both states, angle $\theta(t)$ and angular velocity $\dot{\theta}(t)$, however, the differences are most prominent for the angular velocity $\dot{\theta}$.

6.3.2 *Simulation of the Projectile Motion (No Air Drag)*

Let us consider an example of the projectile motion. For its modelling, we will use the Cartesian coordinate system with z axis aligned with the direction of the gravitational force, shown in Fig. 6.43. In contrast to the previously considered single falling mass case (which had only one DOF, therefore, two states in the model), the 2D projectile motion case, described in the Cartesian coordinates, would require two coordinates, which we denote as h and z for the horizontal and vertical axes correspondingly.

We first consider the case when the *air resistance is small* and, therefore, ignored. This would enable us to build benchmark result data to be further compared with the results for the air resistance non-linear formulation, to be presented at the end of this section.

To model the system, we employ Newton's Second Law and represent all vectors involved (velocity and applied forces) with the equivalent combination of their h and z components. This results in two ordinary differential equations:

$$\begin{cases} m\ddot{h} = \sum F_h \\ m\ddot{z} = \sum F_z \end{cases} \Rightarrow \begin{cases} m\ddot{h} = 0 \\ m\ddot{z} = -mg \end{cases} \Rightarrow \begin{cases} \ddot{h} = 0 \\ \ddot{z} = -g \end{cases} \tag{6.55}$$

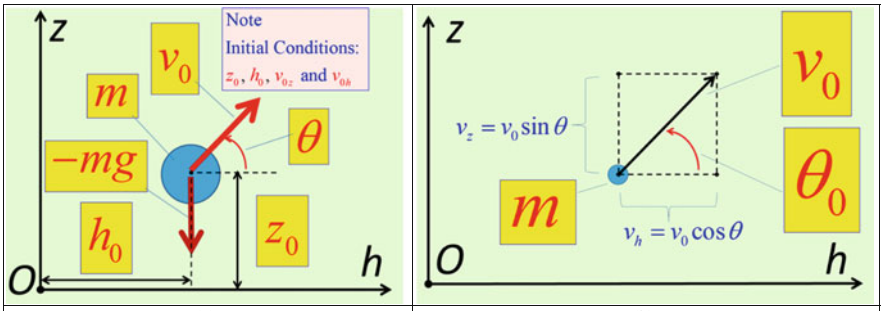


Fig. 6.43 Projectile motion: (**a**) Cartesian coordinates, selected for modelling and notations; (**b**) representation of the velocity vector via its h and z components

As it can be seen from Eq. (6.55), the simplified final equations do not include m, which indicates that the results would be mass insensitive. Also, it can be seen, that the resultant equation of motion (EOM) are *linear*. It should be also noted that, if necessary, the user can change the order of the equations in the system (6.55), placing the second equation in the first position.

The dynamic equation of motion of the projectile (for the no air-resistance case), Eqs. (6.55), are the *second order ordinary differential equations*. In order to access the park of the powerful "ode" numerical solvers in MATLAB®, we will need to rewrite these equations as a set of the first order ordinary differential equations. For this purpose, we introduce the system's states h, $\dot{h}$, z and $\dot{z}$.

It would be important to mention that the final format of the new set of equations would depend upon the order in which these states are "packed" in the vector of states $\mathbf{x}$. We present a few possibilities for numbering of the system's states.

For example,

$$\text{if the order of states is selected as } \begin{cases} x_1 = z \\ x_2 = \dot{z} \\ x_3 = h \\ x_4 = \dot{h} \end{cases} \text{ then the EOM are : } \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = 0 \end{cases} \tag{6.56}$$

There are two major implications of this choice of the states. Firstly, when the initial conditions are formulated, the following vector must be passed on to the solver:

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} z_0 \\ v_{0z} \\ h_0 \\ v_{0h} \end{bmatrix} \tag{6.57}$$

Secondly, when results are calculated by MATLAB®, the z data would be in the first column of the matrix with all solutions for all states, and h data would be in the third column, as per the first statement in Eq. (6.57). Therefore, for example, for plotting trajectory of the mass, the x_3–x_1 data should be used.

We present similar results in the other cases of numbering of the states.

$$\text{If the order of states is selected as } \begin{cases} x_1 = z \\ x_2 = h \\ x_3 = \dot{z} \\ x_4 = \dot{h} \end{cases} \text{ then the EOM are : } \begin{cases} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = -g \\ \dot{x}_4 = 0 \end{cases} \tag{6.58}$$

and the vector of initial conditions should be specified as $\mathbf{x}_0 = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^{\mathrm{T}}_{t=0} = [z_0 \;\; h_0 \;\; v_{0z} \;\; v_{0h}]^{\mathrm{T}}$ and for plotting 2D trajectory of the mass, x_2–x_1 data should be used. (Here and further, T is used to show that the vector needs to be transposed).

$$\text{If the order of states is selected as } \begin{cases} x_1 = h \\ x_2 = z \\ x_3 = \dot{h} \\ x_4 = \dot{z} \end{cases} \quad \text{then the EOM are :} \quad \begin{cases} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = 0 \\ \dot{x}_4 = -g \end{cases} \tag{6.59}$$

and the vector of initial conditions should be specified as $\mathbf{x}_0 = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^{\mathrm{T}}_{t=0} = [z_0 \;\; h_0 \;\; v_{0z} \;\; v_{0h}]^{\mathrm{T}}$ and for plotting 2D trajectory of the mass, x_1–x_2 data should be used.

$$\text{If the order of states is selected as } \begin{cases} x_1 = h \\ x_2 = \dot{h} \\ x_3 = z \\ x_4 = \dot{z} \end{cases} \quad \text{then the EOM are :} \quad \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 0 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -g \end{cases} \tag{6.60}$$

and the vector of initial conditions should be specified as $\mathbf{x}_0 = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^{\mathrm{T}}_{t=0} = [h_0 \;\; v_{0h} \;\; z_0 \;\; v_{0z}]^{\mathrm{T}}$ and for plotting 2D trajectory of the mass, x_1–x_3 data should be used.

Let us consider numerical implementation of the scheme, given by Eqs. (6.56), i.e. the states are numbered as $\mathbf{x} = \left[z, \;\; \dot{z}, \;\; h, \;\; \dot{h}\right]^{\mathrm{T}}$, then the differential equations of motion (EOM) to be solved, are:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = 0 \end{cases} \tag{6.61}$$

Annotated MATLAB script, enabling simulation and animation of the projectile motion (for the no air resistance case) is shown in Fig. 6.44.

If comparison of the trajectories for various values of one of the parameters, let us say projectile angle is needed, the core of script in Fig. 6.44 can be used as an engine and the modified script is presented in Fig. 6.45a with simulation results in Fig. 6.45b. Note that in the script we are only plotting trajectories for $h > 0$.

It can be seen from Fig. 6.45b, that angles $\theta_0 = 10°$ and $\theta_0 = 80°$ ensue the same horizontal travel distance h for the projectile (restricted by $z \geq 0$ requirement). Another pair of the projectile angle is $\theta_0 = 30°$ and $\theta_0 = 60°$. Another observation is that the maximum horizontal travel distance h is achieved, when $\theta_0 = 45°$.

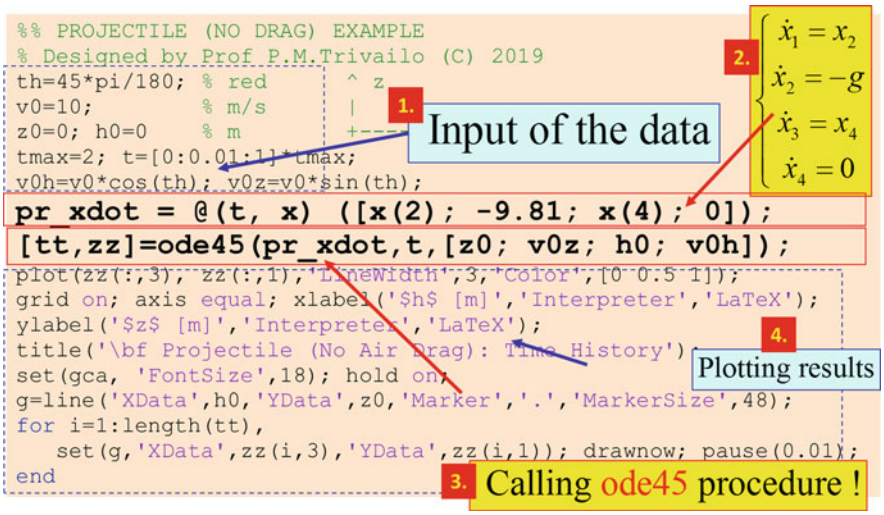


Fig. 6.44 Annotated MATLAB® script, enabling simulation and animation of the projectile motion (for no air resistance case)

The script in Fig. 6.45a can be used to produce a "projectile umbrella"—envelope of trajectories of a projectile with a constant initial speed v_0 and different angles θ_0, discussed in [9]. It would be interesting to note, that the equation for this envelope can be derived analytically. It is given by the following equation:

$$z = \frac{1}{2}\left(\frac{{v_0}^2}{g} - \frac{g}{v_0^2}x^2\right) \tag{6.62}$$

This curve is plotted in Fig. 6.46, where for its numerical validation, we also superimpose the projectile trajectories for a wide range of the angle θ_0: $\theta_0 = [5^\circ : 5^\circ : 175^\circ]$.

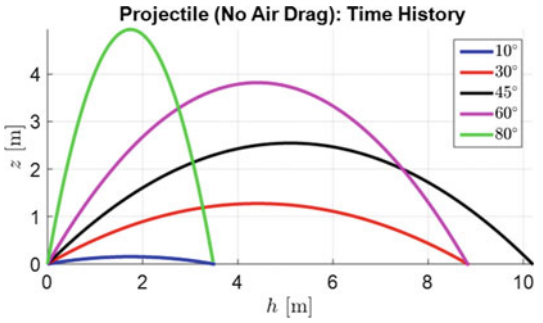
Of course, one of the most critical steps in the simulation process would be validation of the numerical results. In some cases, where the analytical solutions are available, it is advisable to validate numerical solutions against exact analytical solutions. In the case of the projectile motion tasks with no air drag included in the model, numerical results can be validated in many different ways. For example, the following analytical solutions can be used:

- The equation of the parabolic *path* of motion:

$$z = -\frac{1}{2}g\frac{x^2}{v_0^2\cos^2\theta_0} + x\tan\theta_0 \tag{6.63}$$

```
% Designed by Prof P.M.Trivailo (C) 2019
for i=1:length(thAll)
    th=thAll(i); v0h=v0*cos(th); v0z=v0*sin(th);
    [tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
    plot(zz(:,3), zz(:,1),'LineWidth',3,'Color',cols(i));
    idx=max(find(zz(:,1)>=0));
    hCurMax=zz(idx,3);
    zCurMax=max(zz(:,1));
    if hCurMax>hMax, hMax=hCurMax; end;
    if zCurMax>zMax, zMax=zCurMax; end;
    grid on; axis equal; xlabel('$h$ [m]','Interpreter','LaTeX');
    ylabel('$z$ [m]','Interpreter','LaTeX');
    title('\bf Projectile (No Air Drag): Time History');
end
lg=legend('10$^\circ$','30$^\circ$','45$^\circ$','60$^\circ$','80$^\circ$');
set(lg,'Interpreter','LaTeX');
axis([0 hMax 0 zMax]); set(gca, 'FontSize',18); hold on;
```

(a)



(b)

Fig. 6.45 (**a**) MATLAB® script, enabling comparative simulations of the projectile motion (for no air resistance case) for various θ_0; (**b**) simulation results

- The *range* R of the projectile on a flat ground ($z = 0$) and the time t_R the projectile reaches the range R are:

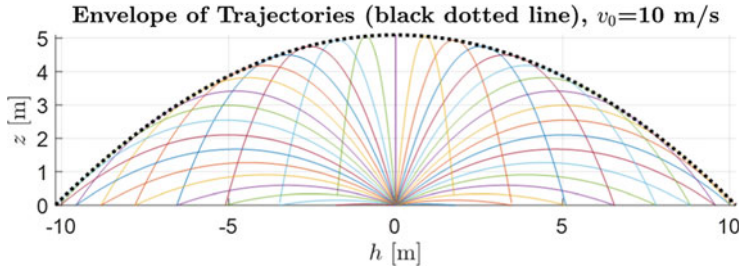
$$R = \frac{v_0^2}{g} \sin(2\theta_0); \quad t_R = 2\frac{v_0}{g} \sin\theta_0 \tag{6.64}$$

- The *highest point* H and the time it is reached are:

$$H = \frac{v_0^2}{2g} \sin^2\theta_0; \quad t_H = \frac{1}{2} t_R = \frac{v_0}{g} \sin\theta_0 \tag{6.65}$$

```
% Designed by Prof P.M.Trivailo (C) 2019
g=9.81; v0=10;  % m/s^2; m/s       ^z
z0=0;   h0=0;   % m                 |
tmax=2; t=[0:0.01:1]*tmax; %        +---------------> h
pr_xdot = @(t, x) ([x(2); -g; x(4); 0]);
thAll=[5:5:195]*pi/180; % rad
figure; hold on; hMax=0; zMax=0;
%--- Plotting simulated projectile trajectories for various angles "theta"
for i=1:length(thAll)
    th=thAll(i); v0h=v0*cos(th); v0z=v0*sin(th);
    [tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
    plot(zz(:,3), zz(:,1));
    idx=max(find(zz(:,1)>=0)); hCurMax=zz(idx,3); zCurMax=max(zz(:,1));
    if hCurMax>hMax, hMax=hCurMax; end; if zCurMax>zMax, zMax=zCurMax; end;
end
grid on; axis equal; xl=xlabel('$h$ [m]'); yl=ylabel('$z$ [m]');
tit=title(sprintf('\\bf Envelope of Trajectories (black dotted line),
$$v_0$$=%g m/s',v0));
set([xl,yl,tit],'Interpreter','LaTeX');
%--- Plotting Envelope (given by exact analytical expression)
hEnvel=[-1:0.01:1]*hMax; zEnvel=(v0^2/g-(g/v0^2)*hEnvel.^2)/2;
plot(hEnvel,zEnvel,'k:','LineWidth',3);
axis([-hMax hMax 0 zMax]); set(gca, 'FontSize',18);
```

(a)



(b)

Fig. 6.46 (**a**) MATLAB® script to plot the analytical expression for the envelope of the projectile trajectories (for "no air resistance" case and $v_0 = 10$ m/s) and embraced numerically simulated projectile trajectories for various $\theta_0 = [5{:}5{:}175]$ degrees; (**b**) simulation results

6.3.3 Simulation of the Projectile Motion (With Air Drag)

In contrast to the task formulation used in the previous section, now we include in the task formulation the air resistance force acting on the projectile. Let us assume that it is proportional to the squared velocity v of the mass m and a coefficient C_d:

$$|F_{\text{air}}| = C_d v^2 = C_d \left(v_h^2 + v_z^2 \right) \tag{6.66}$$

This force is not generally aligned with any of the selected axes, h and z, therefore, in order to formulate the h and z component equations of motion, we

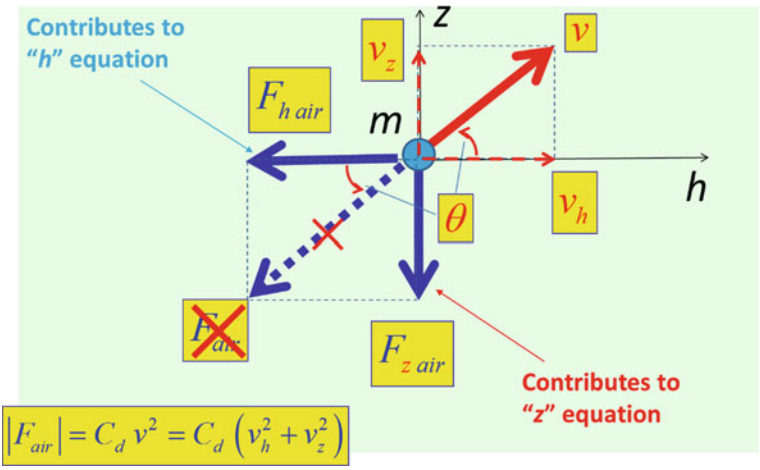


Fig. 6.47 Cartesian coordinate system h–z, used for simulation of the projectile motion, with mass m velocity v and air resistance force F_{air}, resolved along these directions

need to resolve the F_{air} along the h and z directions. For this purpose, we first need to declare the assumed sign conventions, shown in Fig. 6.47. This Figure also shows corresponding representation of the velocity and air resistance force components, resolved along h and z directions. On Fig. 6.47, the air resistance force F_{air} (shown with a dashed arrow) is represented as an equivalent sum of the $F_{h\,\text{air}}$ and $F_{z\,\text{air}}$ force components:

$$\begin{aligned} |F_{\text{air}}| &= C_d v^2 \\ F_{z\,\text{air}} &= -|F_{\text{air}}| \sin\theta = -|F_{\text{air}}| \frac{v_z}{v} = -C_d v^2 \frac{v_z}{v} = -C_d v_z v = -C_d v_z \sqrt{v_h^2 + v_z^2} \\ F_{h\,\text{air}} &= -|F_{\text{air}}| \cos\theta = -|F_{\text{air}}| \frac{v_h}{v} = -C_d v^2 \frac{v_h}{v} = -C_d v_h v = -C_d v_h \sqrt{v_h^2 + v_z^2} \end{aligned} \tag{6.67}$$

After this representation is completed, the F_{air} force is no longer needed, and therefore marked in the Fig. 6.47 with a red cross.

Differential equations of motion of the projectile, can be derived, using various methods. Using Newton's Second Law, we can write them, in view of the explicit air resistance force components [see Eq. (6.67)] as follows:

$$\begin{cases} m\ddot{h} = F_{h\,\text{air}} = -C_d v^2 \cos\theta \\ m\ddot{z} = F_{z\,\text{air}} - mg = -mg - C_d v^2 \sin\theta \end{cases} \tag{6.68}$$

These equations of motion are *non-linear*, second order, ordinary differential equations. However, the treatment of this non-linear model for obtaining numerical solution is very much the same, as was used for the linear case, presented in the previous subsection. We first rewrite these equations in terms of system's states,

specifying the selected order of the states, for example, if the order of the states selected as $\mathbf{x} = \left[z,\ \dot{z},\ h,\ \dot{h}\right]^{\mathrm{T}}$, then the equations of motion of the system to be used in numerical simulation process are:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g - \frac{1}{m} C_d v^2 \sin\theta \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -\frac{1}{m} C_d v^2 \cos\theta \end{cases} \Rightarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g - \frac{1}{m} C_d x_2 \sqrt{x_2^2 + x_4^2} \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -\frac{1}{m} C_d x_4 \sqrt{x_2^2 + x_4^2} \end{cases} \tag{6.69}$$

Then, in view of the compact expressions for the states, we program them in the MATLAB® script, using anonymous function and employ one of the "ode" integration procedures to get numerical solutions. The recommended structure of the script, as in many preceding examples, remains the same and includes the following critical stages: (6.1) input of the data; (6.2) formulation of the states; (6.3) calling "ode45" integration procedure to solve differential Eq. (6.57) and (6.4) plotting results of the numerical simulation. The annotated MATLAB® script, implementing procedure of solving *non-linear formulation* for the projectile motion, is given in Fig. 6.48.

The script in Fig. 6.48 can be further developed to run concurrent comparative animations for two projectiles, described with linear model (which does not include air resistance) and non-linear model (which includes air resistance).

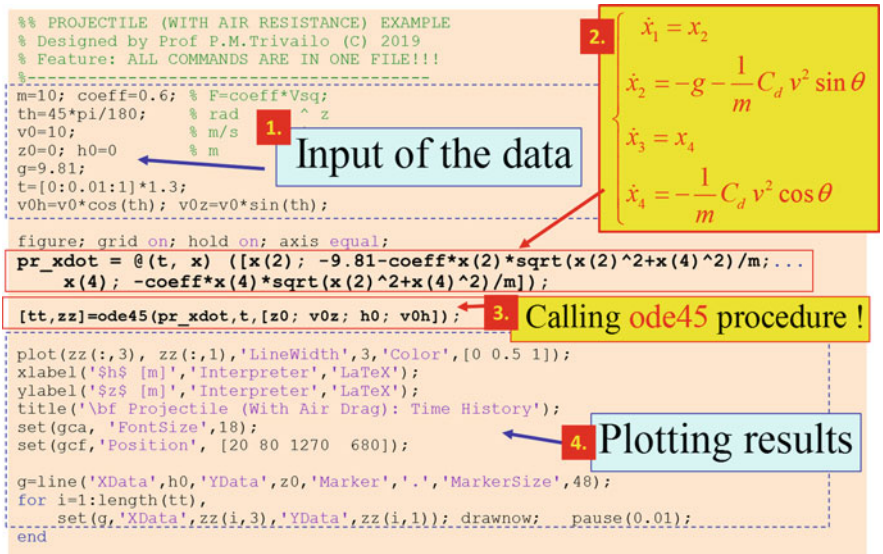


Fig. 6.48 Annotated MATLAB® script, enabling simulation and animation of the projectile motion (for the case, including air resistance)

```
%% COMPARE PROJECTILES WITHOUT and WITH AIR RESISTANCE
% Designed by Prof P.M.Trivailo (C) 2019
%----------------------------------------
m=10; Cd=0.1;           % F=Cd*v^2;
thd=45; th=thd*pi/180;  % rad       ^ z
v0=30;                  % m/s       |
z0=0; h0=0;             % m         +--------------> h
g=9.81;
v0h=v0*cos(th); v0z=v0*sin(th);
tmax=2*v0*sin(th)/g; hmax=v0^2*sin(2*th)/g;
t=0:0.02:tmax;

pr_xdot_L  = @(t, x) ([x(2); -9.81; x(4); 0]);
pr_xdot_NL = @(t, x) ([x(2); -9.81-Cd*x(2)*sqrt(x(2)^2+x(4)^2)/m;...
                       x(4); -Cd*x(4)*sqrt(x(2)^2+x(4)^2)/m]);

[tL, zL]  = ode45(pr_xdot_L, t,[z0; v0z; h0; v0h]);
[tNL,zNL] = ode45(pr_xdot_NL,t,[z0; v0z; h0; v0h]);

figure; grid on; hold on; axis equal;
plot(zL(:,3), zL(:,1),'LineWidth',3,'Color','b','LineStyle',':');
plot(zNL(:,3), zNL(:,1),'LineWidth',3,'Color','r');
axis tight; ax=axis; yMaxNew=ax(4)*1.1; axis([ax(1:3) yMaxNew]);

pSKY=patch('XData',[ax(1:2) ax(2:-1:1)],'YData',[0 0 yMaxNew yMaxNew],'FaceColor',[0.6 1 1]); % sky
pGROUND=patch('XData',[ax(1:2) ax(2:-1:1)],'YData',[0 0 ax(3) ax(3)],'FaceColor',[.4 .4 .4]); % ground
alpha([pSKY,pGROUND],0.4); grid on

leg=legend(' Linear model: air resistance ignored',' Non-linear model: air resistance included');
set(leg,'Location','SouthWest','Color',[1 1 .6]); xl=xlabel('$h$ [m]'); yl=ylabel('$z$ [m]');
str='Compare projectiles with and without air resistance, ';
tit=title(sprintf('%s$$v_0$$=%g m/s; $$\\theta_0$$=%3.0f$$^\\circ$$; $$C_d$$=%3.1f kg/m',str,v0,thd,Cd));
txtL=text('Color','b'); txtNL=text('Color','r'); set([xl,yl,tit,txtL,txtNL],'Interpreter','LaTeX');
set([txtL,txtNL],'FontSize',16,'HorizontalAlignment','left','VerticalAlignment','bottom');

set(gca, 'FontSize',18); set(gcf,'Position', [20 80 1270  680]);

g=line('MarkerEdgeColor','r'); g2=line('MarkerEdgeColor','b');
set([g,g2],'XData',h0,'YData',z0,'Marker','o','MarkerSize',12,'MarkerFaceColor','w','LineWidth',2);
for ii=1:length(tNL),
    tc=tNL(ii); str=sprintf('$$\\,\\,$$3.1f s',tc);
    set([txtL, txtNL],'String',str);
    set(txtL,'Position',zL(ii,[3,1])); set(txtNL,'Position',zNL(ii,[3,1]));
    if rem(tc*100,1*100)==0,
        txtLstamp=text('Position',zL(ii,[3,1])); txtNLstamp=text('Position',zNL(ii,[3,1]));
        set([txtLstamp,txtNLstamp],'String',str,'Interpreter','LaTeX','FontSize',16,...
                                   'HorizontalAlignment','left','VerticalAlignment','bottom');
        liLstamp =line('XData',zL(ii,3), 'YData',zL(ii,1), 'MarkerEdgeColor','b');
        liNLstamp=line('XData',zNL(ii,3),'YData',zNL(ii,1),'MarkerEdgeColor','r');
        set([liLstamp,liNLstamp],'Marker','o','MarkerSize',8,'LineWidth',2,'MarkerFaceColor','w');
    end;
    set(g,'XData',zNL(ii,3),'YData',zNL(ii,1)); set(g2,'XData',zL(ii,3),'YData',zL(ii,1));
    drawnow;   % pause(0.01);
end
```

Fig. 6.49 Versatile MATLAB® script, enabling simultaneous comparative simulations and animations of two projectiles, described with linear and non-linear models, which do not include and include air resistance correspondingly

This versatile script is presented in Fig. 6.49 and the final snap-shot screen of the animated window is given in Fig. 6.50.

Comparison of the projectile trajectories for the *linear* and *non-linear* models in Fig. 6.50 enables us to reveal very significant differences in their shapes and key characteristics, including the range R of the projectile on a flat ground ($z = 0$) and the time t_R the projectile reaches the range R, highest point H and the time it is reached t_H. This particular case comparison highlights, in general case, limitations of the linear model, use of which would lead to significant errors in prediction of the projectile time history.

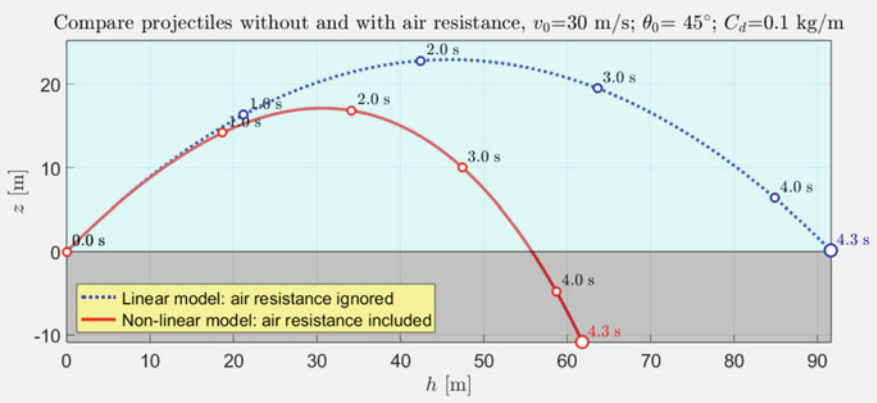


Fig. 6.50 Final snapshot from the animation screen, presenting simultaneous comparative simulations and animations of two projectiles, described with linear and non-linear models

And simplicity of the modelling of the fully non-linear model facilitates use of the non-linear models and contributes to the improvement of the accuracy of the modern simulations.

6.4 Advancing Modelling and Simulation of Engineering Systems, Selecting and Employing Most Beneficial Coordinate Systems

6.4.1 Example of Utilization of the Inclined Cartesian Coordinate System

Let us consider the following task, associated with a flat surface, inclined at angle α with respect to the horizon (shown in Fig. 6.53a). Assume that during the experiment, mass m was released and travelled distance $AB = h$ before it hit the incline. Then, mass m bounces from the inclined surface. Simulate the motion of the mass after the impact. In particular, determine the distance BC along the incline, which the mass will fly until it collides with the incline surface for the second time at point C, assuming elastic impact, which is characterized with no any energy losses during the impact.

The treatment of the task without or with air resistance is the same. This task of simulating mass motion between points A and B can be solved, using falling mass methodology, presented in one of the previous subsections. The corresponding simulation MATLAB script is given in Fig. 6.51 and the results of the simulation (for the $m = 1$ kg, $H = 2$ m, $C_d = 0.1$ and $C_d = 0.8$) are presented in Fig. 6.52.

```
%% RELEASED MASS ABOVE INCLINE: SEGMENT "AB" (with air resistance)
% Designed by Prof P.M.Trivailo (C) 2019
%---------------------------------------
m=10; H=5; Cd=0.1;       % F=Cd*v^2;
%m=1; H=2; Cd=0.8;       % F=Cd*v^2;
g=9.81; t=0:0.02:1.1;

pr_xdot_NL_1 = @(t, x) ([x(2); -9.81-Cd*x(2)*sqrt(x(2)^2+x(4)^2)/m;...
                         x(4); -Cd*x(4)*sqrt(x(2)^2+x(4)^2)/m]);
[tNL,zNL] = ode45(pr_xdot_NL_1,t,[H; 0; 0; 0]);

%------Plotting displacement "z"---
figure; subplot(2,1,1); grid on; hold on;
plot(tNL, zNL(:,1),'LineWidth',3,'Color','r');
xl1=xlabel('$t$ [s]'); yl1=ylabel('$z$ [m]');

tB=interp1(zNL(:,1),tNL,0); vB=interp1(tNL,zNL(:,2),tB);
ptA1=line('XData',0,'YData',H); ptB1=line('XData',tB,'YData',0);
set([ptA1,ptB1],'MarkerEdgeColor','r');
txtA1=text('String',' $A$','Position',[0,H], 'Color','r');
txtB1=text('String',' $B$','Position',[tB,0],'Color','r');
title(sprintf('$$m=%g$$ kg; $$H=%g$$ m; $$C_d=%g$$ kg/m case: $$t_B=%6.4f$$ s; $$v_B=%6.4f$$ m/s',...
                m,H,Cd,tB,vB),'Interpreter','LaTeX');
axis([0 tB 0 H*1.2]); set(gca, 'FontSize',16);
%------Plotting velocity "z_dot"---
subplot(2,1,2); grid on; hold on;
plot(tNL, zNL(:,2),'LineWidth',3,'Color','b');
xl2=xlabel('$t$ [s]'); yl2=ylabel('$\dot{z}$ [m/s]');
set([xl1,xl2,yl1,yl2],'Interpreter','LaTeX');

ptA2=line('XData',0,'YData',0); ptB2=line('XData',tB,'YData',vB);
set([ptA2,ptB2],'MarkerEdgeColor','b');
set([ptA1,ptA2,ptB1,ptB2],'Marker','o','MarkerSize',10,'LineWidth',3,'MarkerFaceColor','w');
txtA2=text('String',' $A$','Position',[0,0],  'Color','b');
txtB2=text('String',' $B$','Position',[tB,vB],'Color','b');
set([txtA1, txtB1, txtA2, txtB2],'FontSize',24,'Interpreter','LaTeX',...
    'VerticalAlignment','bottom','HorizontalAlignment','left');
axis([0 tB vB 0]);

set(gcf,'Position',[80   180   928   560]); set(gca, 'FontSize',16);
```

Fig. 6.51 MATLAB script, enabling simulation of the mass m motion during segment AB

The task of simulating mass motion between points B and C can be solved, using projectile motion methodology presented earlier and utilizing traditional Cartesian coordinate system h–z, aligned with horizontal and vertical directions (see Fig. 6.53a). However, selection of the *inclined Cartesian coordinate system* $\overline{h}\ \overline{O}\ \overline{z}$ (shown in Fig. 6.53b) could offer multiple advantages, for example, simplicity in monitoring conditions of the impact at point C (indeed, this condition would simply satisfy a simple requirement of the current $\overline{z}$ coordinate of the mass to be equal to zero). However, utilization of this coordinate system would require formulation of the equations of motion in the selected coordinate system. For this purpose, let us consider the instant after the impact (presented in Fig. 6.53b). This figure shows for the segment BC the initial velocity v_0, value of which must be equated to the impact velocity at point B, (i.e. final calculated velocity for the segment AB). Assuming perfect elastic impact, we can also conclude that the angle of incidence γ_1 must be equal to the angle of reflection γ_2 (see Fig. 6.53b) and equal to the angle α.

This enables us to establish for segment BC the initial projectile launching angle θ_0 in the inclined Cartesian coordinates $\overline{h}\ \overline{O}\ \overline{z}$:

$$\gamma_1 = \gamma_2 = \alpha \quad \Rightarrow \quad \theta_0 = \frac{\pi}{2} - \alpha \tag{6.70}$$

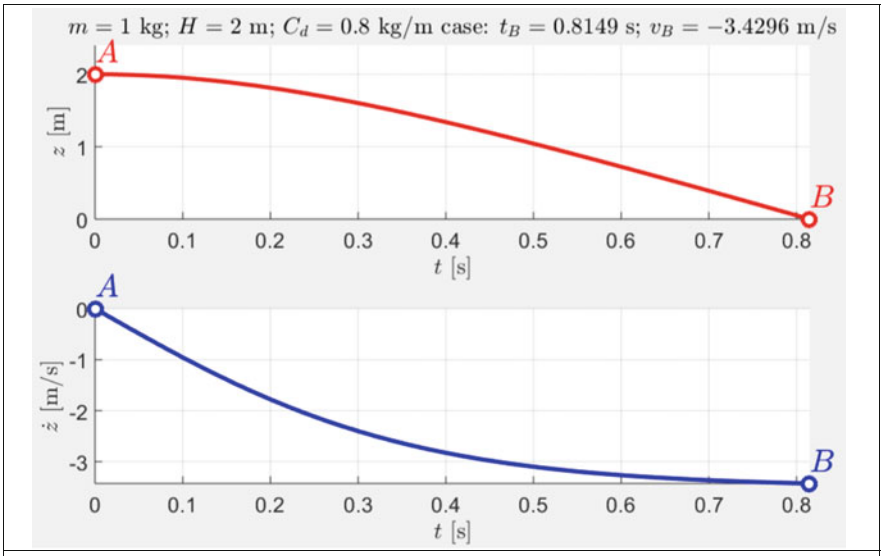


Fig. 6.52 Results of the simulation for the released mass m (falling mass segment AB): (**a**) $m = 1$ kg; $H = 2$ m; $C_d = 0.8$ kg/m; (**b**) $m = 10$ kg; $H = 5$ m; $C_d = 0.1$ kg/m

We now consider the forces, applied to the mass, being the gravitational force mg and air resistance force F_{air}. These forces, in general case, are not aligned with any of the selected coordinate axes, therefore, we need to resolve them along the

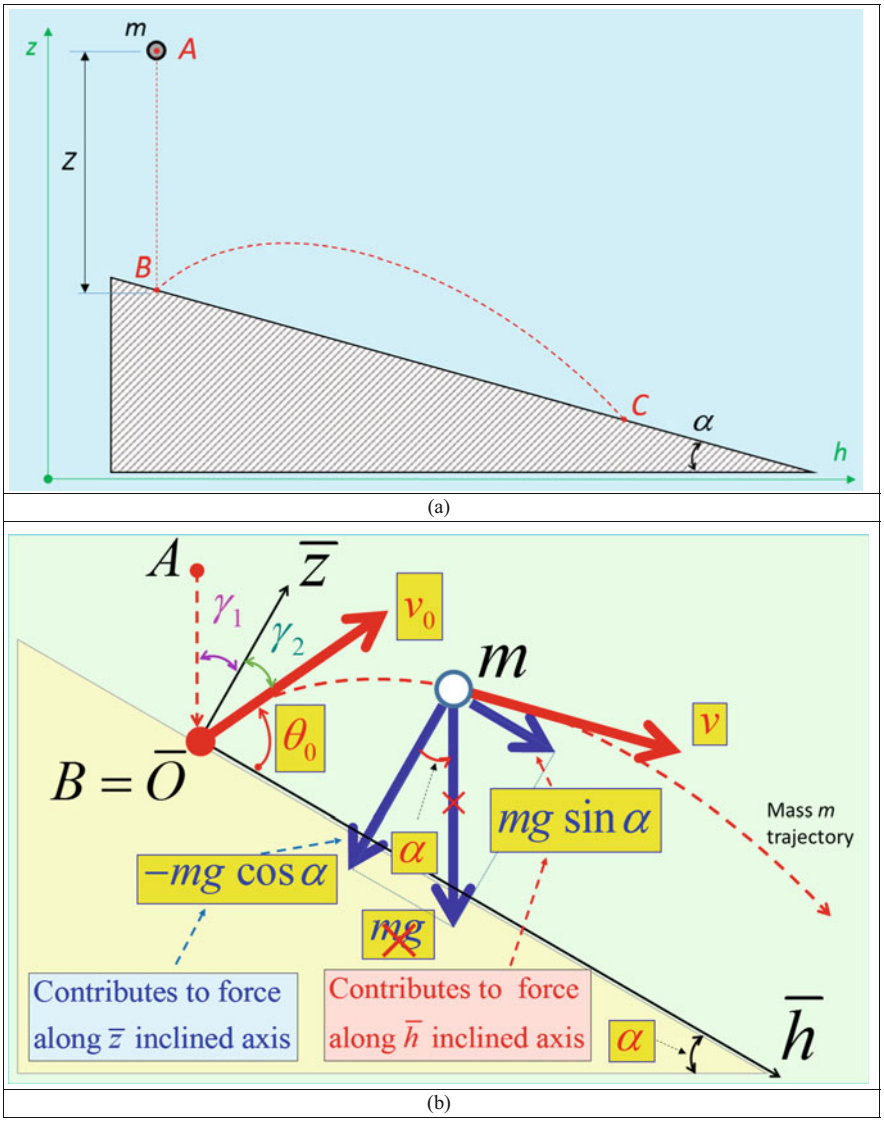


Fig. 6.53 Mass m, dropped from point A and colliding with the inclined surface: (**a**) main notations; (**b**) bounced mass m is shown after the impact, together with employed inclined Cartesian coordinate system $\bar{h}\ \bar{O}\ \bar{z}$

coordinate axes and represent with their axial components. For brevity, we will illustrate in detail the process, using mg force only, and shown in Fig. 6.53b.

The Fig. 6.53b shows this force as a vector and also shows that it can be represented as an equivalent sum of two components $mg \sin \alpha$ and $-mg \cos \alpha$,

obtained when mg is resolved along $\overline{h}$ and $\overline{z}$ directions correspondingly. After this representation is completed, vector mg is no longer needed; therefore, this force is shown as crossed out with red lines in Fig. 6.53b. The air resistance force can be also resolved in a similar way along $\overline{h}$ and $\overline{z}$ directions. With these in mind, the equations of motion of the system can now be written in the inclined Cartesian coordinates as follows:

$$\begin{cases} m\ddot{\overline{h}} = \sum F_{\overline{h}} \\ m\ddot{\overline{z}} = \sum F_{\overline{z}} \end{cases} \Rightarrow \begin{cases} m\ddot{\overline{h}} = mg\ \sin\alpha - C_d v^2 \cos\theta \\ m\ddot{\overline{z}} = -mg\ \cos\alpha - C_d v^2 \sin\theta \end{cases}$$
$$\Rightarrow \begin{cases} \ddot{\overline{h}} = g\ \sin\alpha - \frac{1}{m} C_d v^2 \cos\theta \\ \ddot{\overline{z}} = -g\ \cos\alpha - \frac{1}{m} C_d v^2 \sin\theta \end{cases} \tag{6.71}$$

where v is the magnitude of the instantaneous velocity vector $\mathbf{v}$ of the mass and θ is the instantaneous inclination angle for the velocity vector $\mathbf{v}$ in the $\overline{h}\ \overline{O}\ \overline{z}$ coordinate system, as per the Fig. 6.53b. Let us select the system's states as follows:

$$\mathbf{x} = \left[\overline{z},\ \dot{\overline{z}},\ \overline{h},\ \dot{\overline{h}}\right]^{\mathrm{T}} \tag{6.72}$$

then the equation of motion of the mass m can be rewritten in the following form:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g\cos\alpha - \frac{1}{m} C_d v^2 \sin\theta \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = g\sin\alpha - \frac{1}{m} C_d v^2 \cos\theta \end{cases} \Rightarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g\cos\alpha - \frac{1}{m} C_d x_2 \sqrt{x_2^2 + x_4^2} \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = g\sin\alpha - \frac{1}{m} C_d x_4 \sqrt{x_2^2 + x_4^2} \end{cases} \tag{6.73}$$

If we place the origin of the coordinate system $\overline{h}\ \overline{O}\ \overline{z}$ at the point B, then the vector of initial conditions, required by MATLAB solvers, in view of Eq. (6.56), is

$$\mathbf{x}_0 = \left[0,\ v_0\ \sin\theta_0,\ 0,\ v_0\ \cos\theta_0\right]^{\mathrm{T}} = \left[0,\ v_0\ \cos\alpha,\ 0,\ v_0\ \sin\alpha\right]^{\mathrm{T}} \tag{6.74}$$

Equations (6.73), complemented with initial conditions (6.74), can be solved numerically, using MATLAB `ode` procedures. Corresponding MATLAB script and numerical results are presented in Figs. 6.54 and 6.55 for the following two contrast cases: (a) $m = 1$ kg; $H = 2$ m; $C_d = 0.8$ kg/m; $\alpha = 30°$; (b) $m = 10$ kg; $H = 5$ m; $C_d = 0.1$ kg/m; $\alpha = 30°$.

It should be emphasized that solution of Eqs. (6.73), shown in Fig. 6.55, produces results in the inclined Cartesian coordinate system, facilitating determination of the

```
%% RELEASED MASS ABOVE INCLINE: SEGMENT "BC", bouncing (with air resistance)
% Designed by Prof P.M.Trivailo (C) 2019
%-----------------------------------------
ald=30; al=ald*pi/180;  % rad
ca=cos(al); sa=sin(al);
pr_xdot_NL_2 = @(t, x) ([x(2); -g*ca-Cd*x(2)*sqrt(x(2)^2+x(4)^2)/m;...
                         x(4);  g*sa-Cd*x(4)*sqrt(x(2)^2+x(4)^2)/m]);
[tNL,xNL] = ode45(pr_xdot_NL_2,t,[0; abs(vB)*ca; 0; abs(vB)*sa]);

%--- PLOT TRAJECTORY in THE INCLINED COORDINATE SYSTEM "h_bar O_bar z_bar"
figure;
plot(xNL(:,3),xNL(:,1),'LineWidth',3,'Color','r');
grid on; axis equal
xl1=xlabel('$\bar{h}$ [m]'); yl1=ylabel('$\bar{z}$ [m]');
%
tC=interp1(xNL(2:end,1),tNL(2:end),0); vC=interp1(tNL,sqrt(xNL(:,2).^2+xNL(:,4).^2),tC);
hC=interp1(tNL,xNL(:,3),tC);
ptB1=line('XData',0,'YData',0); ptC1=line('XData',hC,'YData',0);
set([ptB1,ptC1],'MarkerEdgeColor','r');
txtB1=text('String',' $B$','Position',[0,0], 'Color','r');
txtC1=text('String',' $C$','Position',[hC,0],'Color','r');
title(sprintf('$$m=%g$$ kg; $$H=%g$$ m; $$C_d=%g$$ kg/m case: $$t_C=%6.4f$$ s; $$v_C=%6.4f$$ m/s; $$\\bar{h}_C=%6.4f$$ m',...
      m,H,Cd,tC,vC,hC),'Interpreter','LaTeX');
axis([0 hC 0 max(xNL(:,1))]);
set(gca, 'FontSize',16); set([xl1,yl1],'Interpreter','LaTeX');

set([ptB1,ptC1],'Marker','o','MarkerSize',10,'LineWidth',3,'MarkerFaceColor','w');
set([txtB1, txtC1],'FontSize',24,'Interpreter','LaTeX',...
    'VerticalAlignment','bottom','HorizontalAlignment','left');
set(txtB1,'HorizontalAlignment','right');
set(gcf,'Position',[80 80 1100  660]); set(gca, 'FontSize',16);

%--- PLOT TRAJECTORY in NOT INCLINED COORDINATE SYSTEM "hOz"
figure;
ca=cos(-al); sa=sin(-al); TT=[ca -sa; sa ca]; % Rotation matrix
hzNLrota=transpose(TT*[xNL(:,3) xNL(:,1)]');  % Rotated coordinates
plot(hzNLrota(:,1),hzNLrota(:,2),'LineWidth',3,'Color','r');
grid on; axis equal; xl1=xlabel('$h$ [m]'); yl1=ylabel('$z$ [m]');
hCrota=TT*[hC; 0]; % Rotated (line) segment BC
ptB1=line('XData',0,'YData',0); ptC1=line('XData',hCrota(1),'YData',hCrota(2));
set([ptB1,ptC1],'MarkerEdgeColor','r');
txtB1=text('String',' $B$','Position',[0,0], 'Color','r');
txtC1=text('String',' $C$','Position',hCrota,'Color','r');
title(sprintf('$$m=%g$$ kg; $$H=%g$$ m; $$C_d=%g$$ kg/m case: $$t_C=%6.4f$$ s; $$v_C=%6.4f$$ m/s',...
      m,H,Cd,tC,vC),'Interpreter','LaTeX');
axis([0 hCrota(1) hCrota(2) max(hzNLrota(:,2))]);
set(gca, 'FontSize',16); set([xl1,yl1],'Interpreter','LaTeX');
patch('XData',[0 hCrota(1) 0],'YData',[-[1 1]*hCrota(1)*tan(al) 0],'FaceColor','y');

set([ptB1,ptC1],'Marker','o','MarkerSize',10,'LineWidth',3,'MarkerFaceColor','w');
set([txtB1, txtC1],'FontSize',24,'Interpreter','LaTeX',...
    'VerticalAlignment','bottom','HorizontalAlignment','left');
set(txtB1,'HorizontalAlignment','right');
set(gcf,'Position',[80 80 1100  660]); set(gca, 'FontSize',16);
```

Fig. 6.54 MATLAB script, enabling simulation of the mass m motion during segment BC

instant, when mass hits the incline for the second time at point C. The results are showing that these distances in two contrast cases are equal to 1.19 and 17.04 m correspondingly.

For interpretation of the BC segment trajectories in the conventional, not inclined hOz coordinate system, trajectories in Fig. 6.55 must be rotated by the angle of incline α (in the clockwise direction for the considered cases).

Rotation of the geometric objects can be done using matrix transformation of their coordinates:

$$[h-z\text{ coordinates}]_{\text{after rotation}} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} [\overline{h}-\overline{z}\text{ coordinates}]_{\text{before rotation}} \tag{6.75}$$

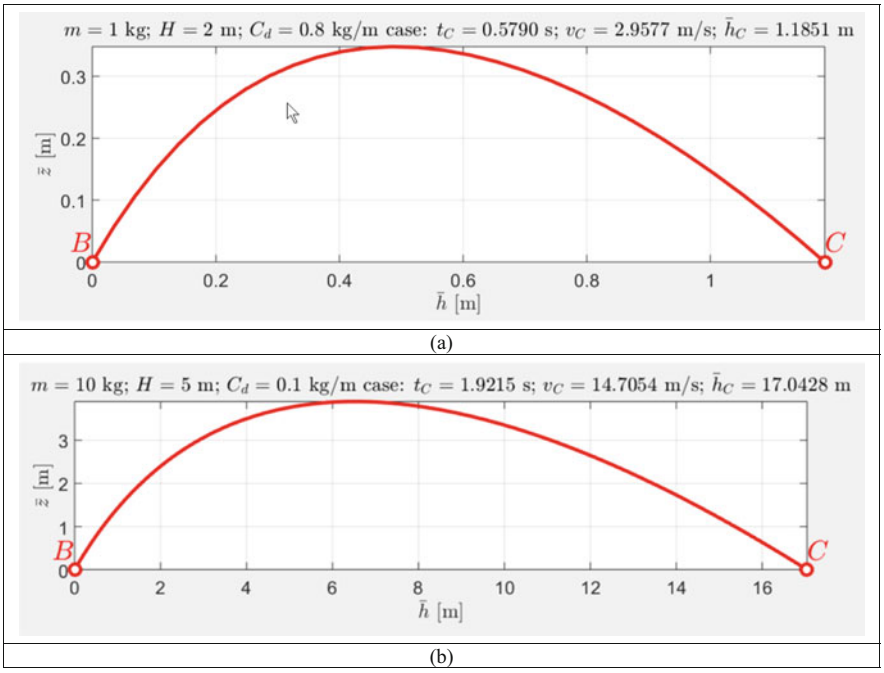
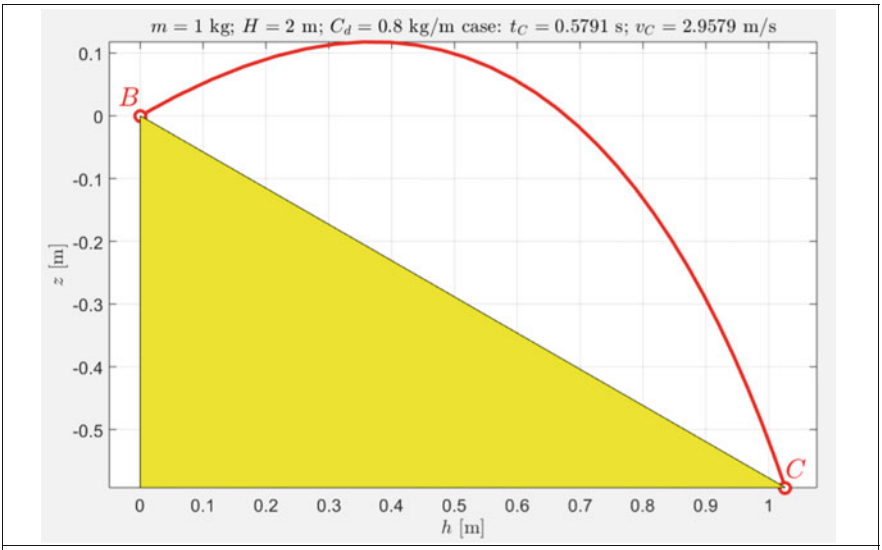


Fig. 6.55 Simulated trajectories for the bouncing mass m (segment BC) in the inclined Cartesian coordinate system $\overline{h}\ \overline{O}\ \overline{z}$ with the air resistance included in the model: (**a**) $m = 1$ kg; $H = 2$ m; $C_d = 0.8$ kg/m; (**b**) $m = 10$ kg; $H = 5$ m; $C_d = 0.1$ kg/m

The "rotated" trajectories of the mass in the two contrast study cases are produced in Fig. 6.56.

6.4.2 *Example of Utilization of the Normal and Tangential Coordinate System*

Use of moving coordinate systems, alternative to Cartesian, can be very useful in engineering and in mechanics, in particular. We will illustrate, that utilization of the normal and tangential components of the projectile velocity could give an interesting insight on the trajectory of the motion. In dynamics, use of normal and tangential coordinates simplifies description of the velocity vector and air resistance forces. We will show how these coordinates can be used to monitor position of the *instantaneous centre of rotation* and the value of the *instantaneous radius of rotation*.



(a)

(b)

Fig. 6.56 Simulated trajectories for the bouncing mass m (segment BC) in the not inclined Cartesian coordinate system hOz with the air resistance included in the model: (**a**) $m = 1$ kg; $H = 2$ m; $C_d = 0.8$ kg/m; (**b**) $m = 10$ kg; $H = 5$ m; $C_d = 0.1$ kg/m

It can be derived that the acceleration of the moving mass m, including projectile, can be expressed in the normal and tangential coordinates as follows [10]:

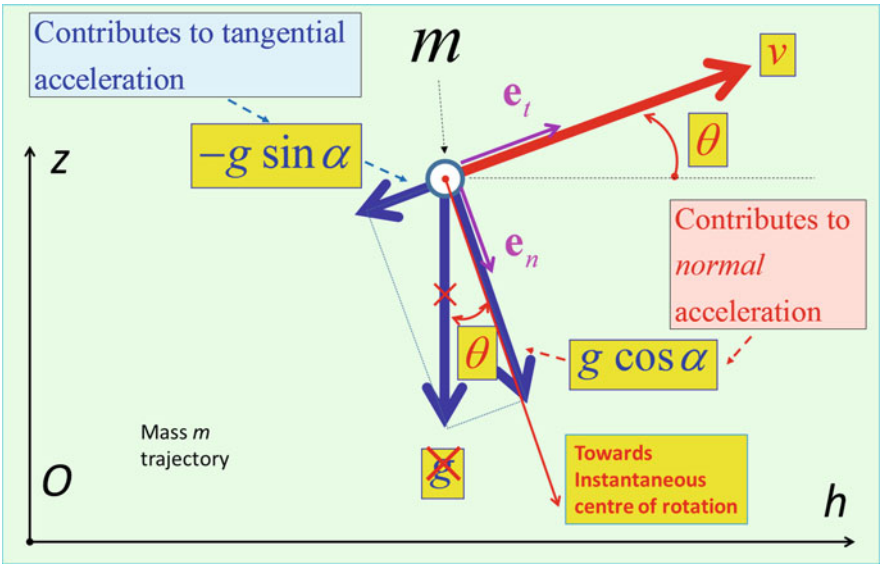


Fig. 6.57 Representation of the vector of gravitational acceleration with normal and tangential components

$$\mathbf{a} = \frac{dv}{dt}\mathbf{e}_t + \frac{v^2}{\rho}\mathbf{e}_n \tag{6.76}$$

In Eq. (6.76) $\mathbf{e}_t$ and $\mathbf{e}_n$ are denoting unit vectors, tangent and normal to the path, correspondingly and ρ is denoting the instantaneous radius of rotation of the mass–radius of the imagined circle, closely conforming with the current infinitesimal segment of the path around current instantaneous position of the moving point.

For our study, we will consider projectile mass m model, ignoring air resistance. In this case, the gravitational acceleration g is the only acceleration, applied to the mass. Figure 6.57 shows that the vector of the gravitational acceleration $\mathbf{g}$ can be resolved along the *normal* and *tangential* directions (i.e. $\mathbf{e}_n$ and $\mathbf{e}_t$ directions):

$$\mathbf{g} = g_n\mathbf{e}_n + g_t\mathbf{e}_t = (g\ \cos\theta)\,\mathbf{e}_n + (-g\ \sin\theta)\,\mathbf{e}_t \tag{6.77}$$

From Eqs. (6.76) and (6.77), it can be identified that the radius of the instantaneous rotation of the projectile mass m can be calculated as:

$$\rho = \frac{v^2}{g\cos\theta} = \frac{v_h^2 + v_z^2}{g \times \arctan\left(\frac{v_z}{v_h}\right)} \tag{6.78}$$

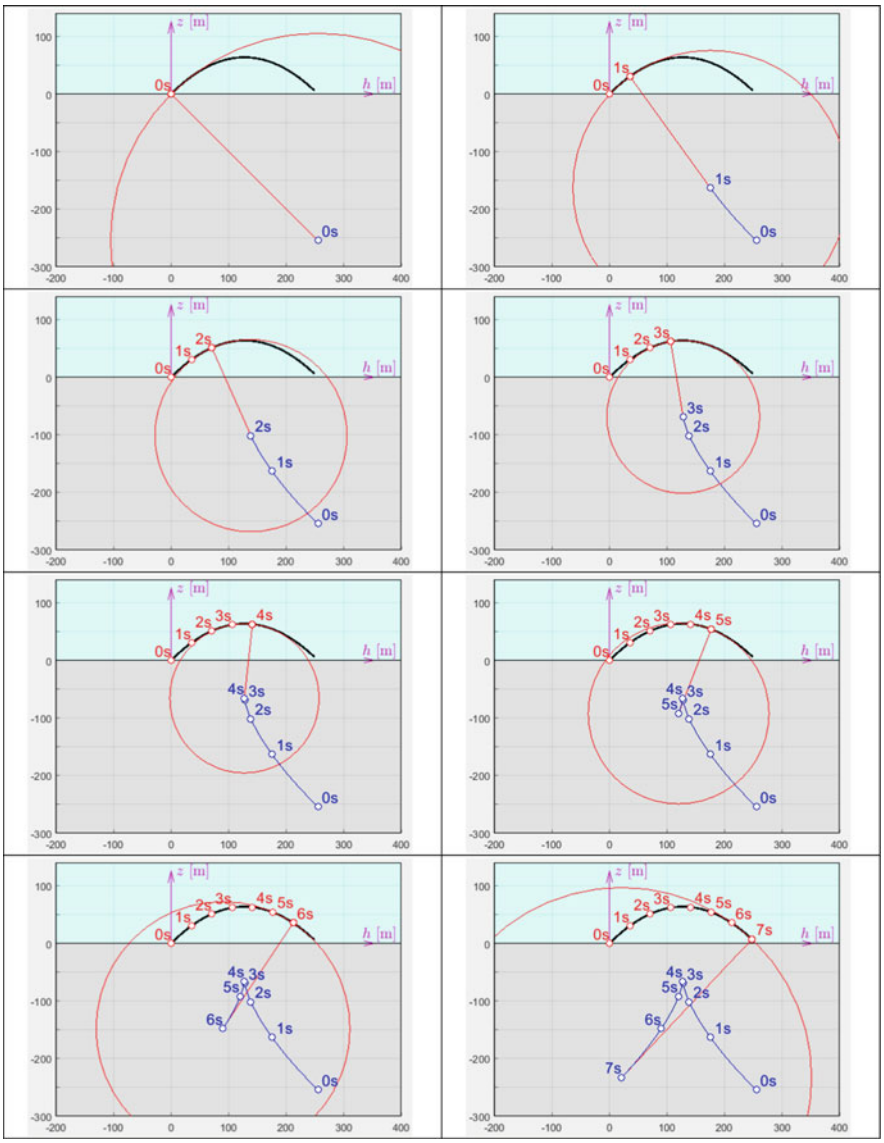


Fig. 6.58 Evolution of instantaneous circles of rotation (red circles, red radii) and migration of their (blue) centres: snapshots from animation, taken at $t = 0{:}7$ s [air resistance ignored]

Evolution of the instantaneous circles of rotation and location of their centres were animated and the snapshots are shown for the example ($v_0 = 50$ m/s; $\theta_0 = 45°$) in Fig. 6.58.

In Fig. 6.58, the projectile's trajectory is shown with black line, instantaneous circles of rotation—with red circular lines, instantaneous radii—with red straight

lines, instantaneous centres of rotation—with blue small circles, and matching radii points, lying on the path trajectory—with red small circles.

It is quite amazing to realize that exactly the same script can be used for animating evolution of the instantaneous circles of rotation and position of the instantaneous radii for the case with air resistance taken into consideration! However, this no-difference in treatment of the linear and non-linear cases (for the instantaneous centres of rotation analysis) is becoming evident, when we recall that the air resistance acceleration is always aligned with the tangential direction and, therefore, does not contribute to the normal acceleration used for the associated calculations of R. For completeness in Fig. 6.59, similar to Fig. 6.58, we present snapshots of the animated instantaneous radii and circles of rotation for the air resistance case with $C_d = 0.06$ kg/m.

Side by side comparison of Figs. 6.58 and 6.59 for linear and non-linear models shows that the radii of instantaneous rotations for the non-linear cases are smaller, especially when the projectile is at its perigee segment of the trajectory. This can be confirmed with the plot of the explicit comparison of the radii of the instantaneous rotations of the projectiles, presented in Fig. 6.60.

To conclude consideration of the projectile models, we need to mention that when state solution is available, it can be further used for analysis and/or interpretation of any characteristics of interest of the system. For example, we can easily plot time histories for the velocities of the projectiles, modelled with linear and non-linear models, shown in Fig. 6.61. For this purpose, a relationship of the velocity v should be produced and programmed for plotting:

$$v = \sqrt{v_h^2 + v_z^2} = \sqrt{x_2^2 + x_4^2} \tag{6.79}$$

It should be reminded, that the indices in relationship (6.79) are sensitive to the order of the states, in the illustration case stipulated with Eqs. (6.56) and (6.72), i.e.

$$\begin{aligned} \mathbf{x}_{\text{linear}} &= \left[z,\ \dot{z},\ h,\ \dot{h} \right]^{\mathrm{T}} \\ \mathbf{x}_{\text{non-linear}} &= \left[\overline{z},\ \dot{\overline{z}},\ \overline{h},\ \dot{\overline{h}} \right]^{\mathrm{T}} \end{aligned} \tag{6.80}$$

Also, when relationship (6.79) is programmed in MATLAB, pseudo-operations must be used, i.e. `sqrt(x2.^2 + x4.^2)`.

6.4.3 *Example of Utilization of the Polar Coordinate System in Spacecraft Dynamics*

Polar coordinate systems are often used to describe the curvilinear motion of a point mass. Being very popular in mechanics, they are widely used for modelling

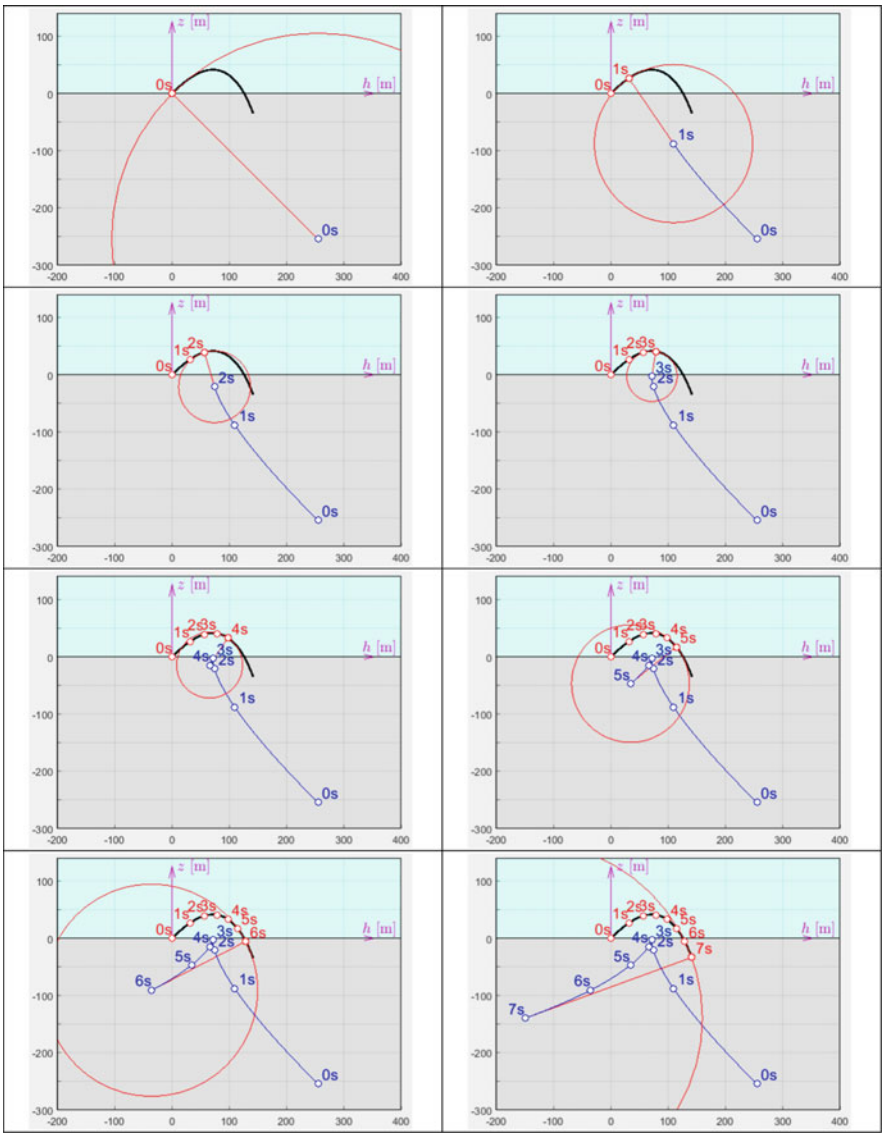


Fig. 6.59 Evolution of instantaneous circles of rotation (red circles, red radii) and migration of their (blue) centres: snapshots from animation, taken at $t = 0{:}7$ s [air resistance included]

of the radar sensed aircraft and for the development of the classical orbital mechanics models. The general analytical expressions for the *radial* and *transverse* components of the acceleration in polar coordinates are given with the following expressions:

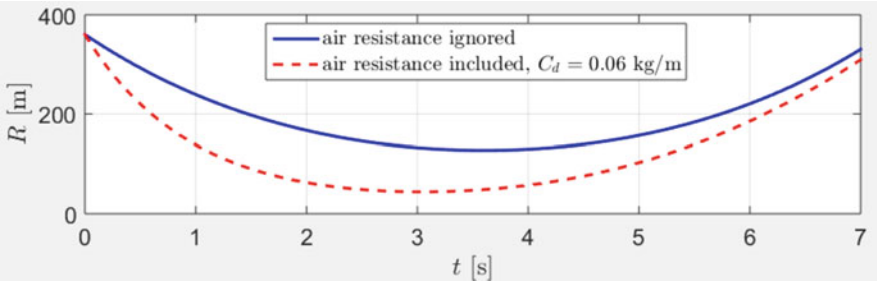


Fig. 6.60 Comparison of the radii of instantaneous rotations for the projectiles, described with linear (with air resistance ignored) and non-linear (air resistance included) models

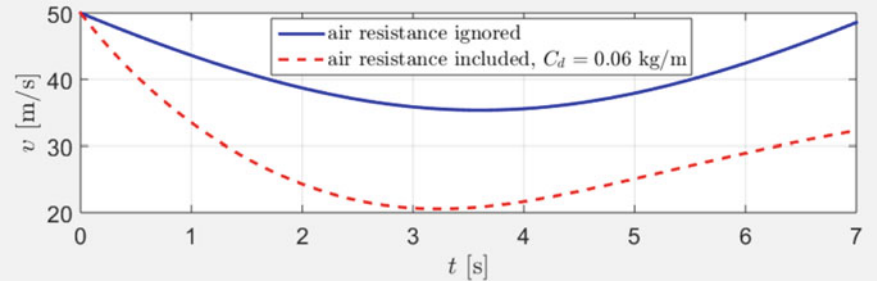


Fig. 6.61 Comparison of the velocity of the projectiles, described with linear (with air resistance ignored) and non-linear (air resistance included) models

$$\begin{cases} a_r = \frac{d^2r}{dt^2} - r\left(\frac{d\theta}{dt}\right)^2 = \frac{d^2r}{dt^2} - r\omega^2 \\ a_\theta = r\frac{d^2\theta}{dt^2} + 2\frac{dr}{dt}\frac{d\theta}{dt} = r\alpha + 2\frac{dr}{dt}\omega \end{cases} \tag{6.81}$$

where the term $-r\omega^2$ in the radial component of the acceleration is called centripetal acceleration, and the term $2(dr/dt)\omega$ in the transverse component is called Coriolis acceleration. With this in mind, total acceleration, acting on the moving point can be expressed with a single equation, utilizing polar coordinates unit vectors $\mathbf{e}_r$ and $\mathbf{e}_\theta$:

$$\mathbf{a} = \left(\ddot{r} - r\dot{\theta}^2\right)\mathbf{e}_r + \left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right)\mathbf{e}_\theta \tag{6.82}$$

On the other hand, if we consider a spacecraft of mass m in its orbit around the Earth and assume that the gravitational force is the only force, acting on the spacecraft, total acceleration can be calculated explicitly, using Newton's law of universal gravitation:

$$\mathbf{a} = -\frac{Gm_1M_\oplus}{r^2}\mathbf{e}_r \tag{6.83}$$

where G is the gravitational constant [6.674×10^{-11} N $\cdot$ (m/kg)^2] and $M_\oplus$ is the mass of the Earth [$M_\oplus = 5.97219 \times 10^{24}$ kg]. It should be noted, that Eq. (6.83) does not have *transverse* component, as the gravitational force is aligned with the radial direction. Comparison of Eqs. (6.82) and (6.83) allows us to write two equations of motion of the point in the following from two coupled *ordinary differential equations of the second order*:

$$\begin{cases} m\left(\ddot{r} - r\dot{\theta}^2\right) = -\frac{GmM_\oplus}{r^2} \\ m\left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right) = 0 \end{cases} \tag{6.84}$$

The methodology of treatment of these types of equations for obtaining numerical solution was illustrated with the example of the vibration of mass-spring systems. Firstly, we will rewrite these second order equations in terms of the states of the system, which would reduce the order of the new set of equations from the second to the first.

Let us assume the following states of the system and their order in the states vector $\mathbf{x} = [x_1, x_2, x_3, x_4]^{\mathrm{T}}$:

$$\mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} r \\ \theta \\ \dot{r} \\ \dot{\theta} \end{Bmatrix} \tag{6.85}$$

In view of this statement, we can write first two state equations: $\dot{x}_1 = x_3$ and $\dot{x}_2 = x_4$. Two other equations, reflecting the physical side of the modelling of the simulated system can be obtained from Eq. (6.72). Total system of four ordinary differential equations of the first order, being more convenient equivalent of the Eqs. (6.84), can now be written as:

$$\begin{cases} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = x_1 {x_4}^2 - \frac{GM_\oplus}{r^2} \\ \dot{x}_4 = -\frac{2x_3x_4}{x_1} \end{cases} \tag{6.86}$$

Remarkable, that the mass m is no longer participating in the states equations of motion (6.86), also observed in the case of the linear projectile model, where the air resistance was ignored. We need to stress out, that the basic spacecraft dynamics model state equations (6.86) are *non-linear*, however, as it will be seen, conceptually, the solution process for these equations would be very much the same to the treatment of the process for linear systems. This just confirms one of the main advantages of the numerical methods—uniform treatment of the simulated systems, linear and non-linear.

```
%% BASIC SPACECRAFT ORBITAL DYNAMICS
% Designed by Prof P.M.Trivailo (C) 2019
%-----------------------
G=6.673*10^(-11); % m^3/(kg*s^2)
M=5.97219*10^24;  % kg
rE=6371008.7714;  % m
th0d=60;
%--- Cases-1,2,3,4 (comment/uncomment appropriate)
%r0=1.2*rE; theta0=th0d*pi/180; rdot0=0; thetadot0=-7000/r0; tmax=6000;  tstep=500;  % Case-1
r0=1.2*rE; theta0=th0d*pi/180; rdot0=0; thetadot0=-9000/r0; tmax=22000; tstep=2000; % Case-2

tt=[0:0.005:1]*tmax; x0=[r0; theta0; rdot0; thetadot0];

% x(1) - r; x(2) - theta; x(3) - r_dot; x(4) - theta_dot;
space_xdot  = @(t, x) ([x(3); x(4); x(1).*x(4).^2 - G*M./(x(1).^2); -2*x(3).*x(4)./x(1)]);
[t, x]  = ode45(space_xdot, tt,x0);

%=====
figure; pol=polar(x(:,2),x(:,1)); set(pol,'LineWidth',2,'Color','r'); hold on;
cdata = imread('composite.jpg');
[X, Y, Z] = ellipsoid(0, 0, 0, rE, rE, rE, 180);
globel = surf(X, Y, -Z,  'FaceColor','texturemap', 'CData', cdata, 'EdgeColor', 'none');
grid on; axis equal; xl=xlabel('$x$ [m]'); yl=ylabel('$y$ [m]');  zl=zlabel('$z$ [m]');
set([xl,yl,zl],'Interpreter','LaTeX'); set(gca, 'FontSize',20);

idx=max(find(x(:,2)<2*pi)); tRegular=[0:tstep:t(idx)-tstep];
rRegular=interp1(t,x(:,1),tRegular); thRegular=interp1(t,x(:,2),tRegular);

line('XData',rRegular.*cos(thRegular),'YData',rRegular.*sin(thRegular),'ZData',zeros(size(rRegular)),...
    'LineStyle','none','LineWidth',1,'Marker','o','MarkerSize',6,'MarkerEdgeColor','k','MarkerFaceColor','w');

for ii=1:length(tRegular);
    tc=tRegular(ii); xc=rRegular(ii).*cos(thRegular(ii)); yc=rRegular(ii).*sin(thRegular(ii));
        text('String',sprintf('%2.0fs',tc),'Position',[xc, yc],'Color','k',...
            'HorizontalAlignment','center','VerticalAlignment','bottom','FontSize',14,'FontWeight','bold');
end
```

Fig. 6.62 MATLAB script, enabling simulation of the basic orbital dynamics equations

MATLAB script enabling simulation and visualization of the spacecraft orbits for the following two contrast cases of the initial conditions is presented in Fig. 6.62, and results of the simulation are in Figs. 6.63 and 6.64. The initial conditions, used for these cases:

(a) $r_0 = 1.2R_\oplus$; $\theta_0 = 60^\circ$; $\dot{r} = 0$; $\dot{\theta} = -7000/r_0$
(b) $r_0 = 1.2R_\oplus$; $\theta_0 = 60^\circ$; $\dot{r} = 0$; $\dot{\theta} = -9000/r_0$

To bring visual realism to the plots in Figs. 6.63 and 6.64, we are plotting the Earth as a sphere, using Earth texture, as per advice in [11]. We also place time markers along spacecraft trajectories, plotted as polar lines in Fig. 6.63 and 3D Cartesian line in Fig. 6.64. In Fig. 64, we also distinguish the spacecraft plane as a yellow semi-transparent plane, bounded with the trajectory line (shown with red colour).

The first observation from Fig. 6.63 and other similar cases, which could be generated with the script in Fig. 6.62 enables us to rediscover Kepler's first Law, reformulating for the spacecraft-Earth system and stating that "The orbit of a spacecraft is an ellipse with the earth at one of the two foci".

The second observation would reflect importance of the initial conditions: case (b) has only higher initial transverse initial velocity, as compared to case (a), with all other conditions being the same. It is interesting to observe, that only this change in a single parameter significantly influences the shape of the orbit, making it elliptical with much higher eccentricity.

As an add-on to this example, we will further use case (b) to rediscover Kepler's Second Law, applied to the spacecraft-Earth system and stipulating that "A line

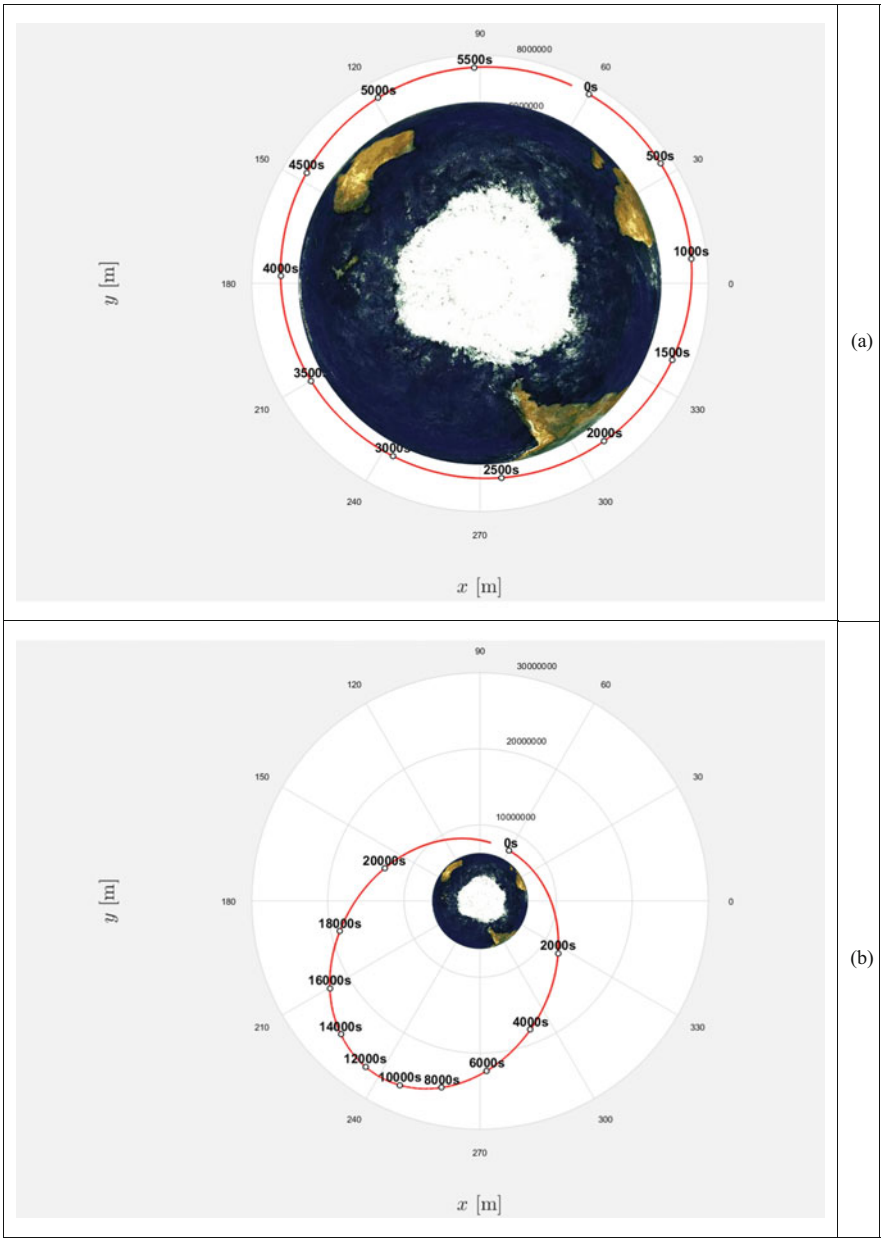


Fig. 6.63 Simulated spacecraft orbits for two contrast cases

segment joining a spacecraft and the Earth sweeps out equal areas during equal intervals of time".

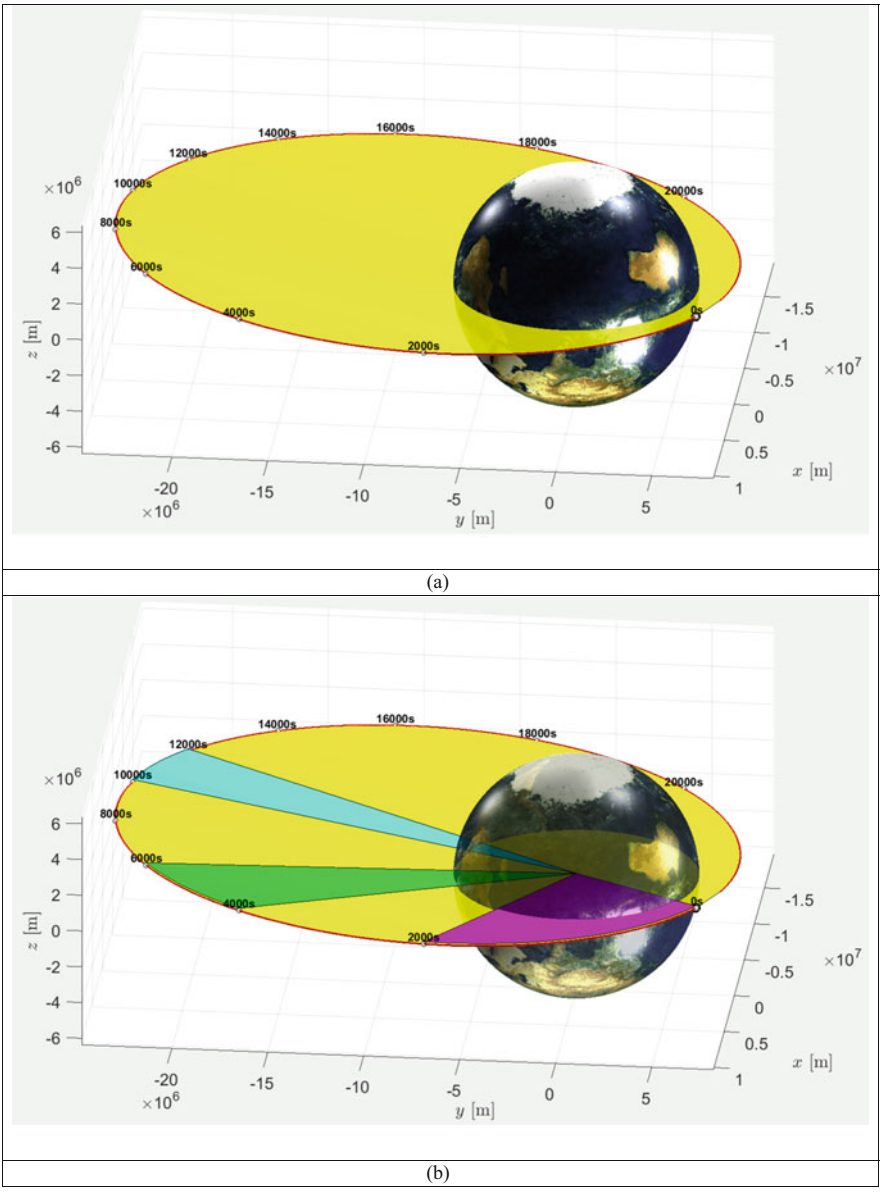


Fig. 6.64 Simulated spacecraft orbit for the illustration initial conditions case $r_0 = 1.2R_\oplus$; $\theta_0 = 60^\circ$; $\dot{r} = 0$; $\dot{\theta} = -9000/r_0$: (**a**) 3D view, Earth shown as non-transparent body; (**b**) areas, swept out by the radius vector pinned at the Earth centre for 0–2000s (magenta); 4000–6000 s (green) and 10,000–12,000 s (cyan) equal intervals with Earth shown as a semi-transparent body

To prove this, let us consider first the area, swept out by the radius vector pinned at the Earth centre, during 0–2000 s interval (shown with magenta colour in Fig. 6.64b). The area could be easily calculated, using MATLAB `polyarea(X,Y)`

command and is equal to $6.87 \cdot 10^7$ km^2. Within the accuracy of 0.1% and 0.01% correspondingly, the areas of two other areas, corresponding to the intervals of 4000–6000 (shown in green) and 10,000–12,000 s (shown in cyan) are equal to the same value, confirming Kepler's Second Law.

Finally, in the subsection, we will verify Kepler's Third Law. Being applied to the spacecraft-Earth system, it states that "The square of the orbital period of a spacecraft is directly proportional to the cube of the semi-major axis of its orbit". This would enable us to write the following relationship for two cases of the spacecraft orbits:

$$\left(\frac{T_B}{T_A}\right)^2 = \left(\frac{r_B}{r_A}\right)^3 \tag{6.87}$$

Using simulation results for the cases (a) and (b) in Fig. 6.63, we can calculate the left-hand side of Eq. (6.87), i.e. ratio of the corresponding periods (in hrs), which would be equal to $(6.1906/1.6933)^2 = 13.4$. It can be verified, that the right-hand side of Eq. (6.87), with the accuracy of 0.2%, would be equal to the same value: $(17{,}125.9/7212.46)^3 = 13.4$. This concludes elementary numerical verification of Kepler's Third Law, using cases (a) and (b). For more generalized verifications, more comparative examples should be involved.

Finally, once again it should be mentioned that original Kepler's Laws were initially formulated for the planetary motion, and in this subsection they were reformulated for the spacecraft-Earth system. Classical *Kepler's Laws of planetary motion* [11], widely used in astronomy, are three scientific laws describing the motion of planets around the Sun. For completeness, we present Kepler's Laws in the original iconic form:

1. The orbit of a planet is an ellipse with the Sun at one of the two foci.
2. A line segment joining a planet and the Sun sweeps out equal areas during equal intervals of time.
3. The square of the orbital period of a planet is directly proportional to the cube of the semi-major axis of its orbit.

These Laws have wide application in astronomy and in order to illustrate one of them, we estimate the period of rotation about Sun for Jupiter. If the average orbital radii for Earth and Jupiter are available (equal to $r_{\mathrm{E}} = 1.5 \times 10^8$ m and $r_{\mathrm{J}} = 7.8 \times 10^8$ m), then the period T_{J} of rotation of Jupiter around Sun in terms of the Earth's period $T_{\mathrm{E}} = 1$ year, is calculated, using Eq. (6.87):

$$T_{\mathrm{J}} = T_{\mathrm{E}}\left(\frac{r_{\mathrm{J}}}{r_{\mathrm{E}}}\right)^{3/2} = 11.9 \text{ years} \tag{6.88}$$

6.5 Finite Element Method (FEM) as a Powerful "Vehicle" for Reducing Modelling of Continuous Systems to the Modelling of Discrete Systems: Applications in Dynamics

6.5.1 Some General Comments on the FEM in Modelling and Simulation

The finite element method (FEM) is a numerical method for solving problems of engineering and mathematical physics. Studying or analysing a phenomenon with FEM is often referred to as finite element analysis (FEA).

Typical problem areas of interest include structural analysis, heat transfer, fluid flow, mass transport and electromagnetic potential. The analytical solution of these problems generally require the solution to boundary value problems for partial differential equations. The finite element method formulation of the problem results in a system of algebraic equations. The method approximates the unknown function over the domain. To solve the problem, it subdivides a large system into smaller, simpler parts that are called finite elements. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. FEM then uses variational methods from the calculus of variations to approximate a solution by minimizing an associated error function [12].

The FE method was developed more by engineers using physical insight than by mathematicians using abstract methods. It was first applied to problems of stress analysis and has since been applied to other problems of continua [13]. The applications of the method and bibliography on the subject are immense and the interested reader is referred to the classical text [14] for more details.

The focus of this section, however, will be on the examples from dynamics, involving vibrations of multi-DOFs mass-spring systems and elastic rods and beams in axial and lateral vibrations. We will show that the FEM can be immediately applied to the ***discrete*** deterministic systems (i.e. with finite DOFs), like mass-spring systems, used for the concise and visual introduction into the FEM, presented earlier in the section. In the following examples, we will show that the FEM can be applied to the ***continuous*** deterministic systems, like vibrating beams, trusses, plates, shells, solids, etc.

We will illustrate that the FEM in these cases would be playing a role of the powerful "vehicle", enabling reduction of the systems with infinite number of degrees-of-freedom (DOFs) to the systems with finite number of DOFs, further conversion of these discrete models to the states formulation and solution, using mathematical solvers, for example, "ode" MATLAB solvers for ordinary differential equations.

Visual Introduction into FEM, Using Multi Mass-Spring System Examples

Let us consider a simple unconstrained mass-spring system, similar to the system in Fig. 6.65. It has two springs, which we will treat as "finite elements", characterized with their nodal displacements. If we number masses in an arbitrary way, in general case, each of the particular spring elements would have own two numbers, i and j ($i \neq j$), to describe their boundaries and to denote its stiffness k_{ij}.

Let us consider two experiments with the spring element k_{ij} (illustrated in Fig. 6.66), firstly applying a unit displacement to its left boundary and then applying a unit displacement to its right boundary. This would enable us to establish resultant forces F, which would be required to implement these experiments. These findings can be expressed as two equations, which could be written in a single matrix equation:

$$\begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix} \begin{Bmatrix} q_i \\ q_j \end{Bmatrix} = \begin{Bmatrix} F_i \\ F_j \end{Bmatrix} \tag{6.89}$$

The square matrix involved in Eq. (6.89) is called element stiffness matrix $\left[k_{ij}^{e}\right]$. If we create a global stiffness matrix $[K]$, which would have dimension sufficient to embrace all degrees-of-freedom, then the elasticity of the spring k_{ij} could be

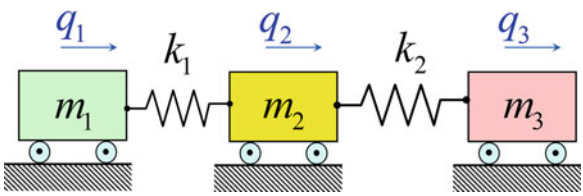


Fig. 6.65 Unconstrained 3DOFs mass-spring system

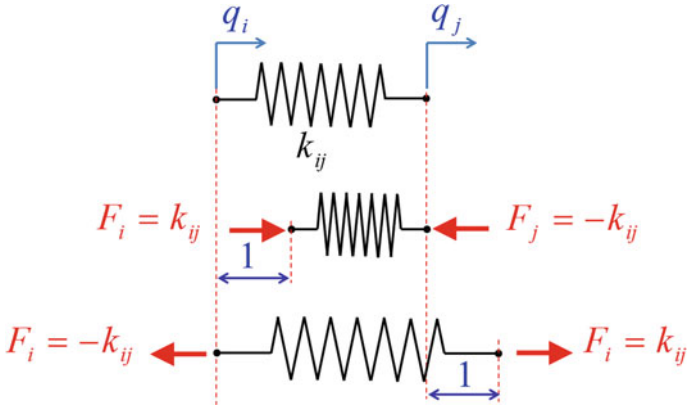


Fig. 6.66 Consecutive application of the unit displacements to the boundaries of the spring k_{ij}

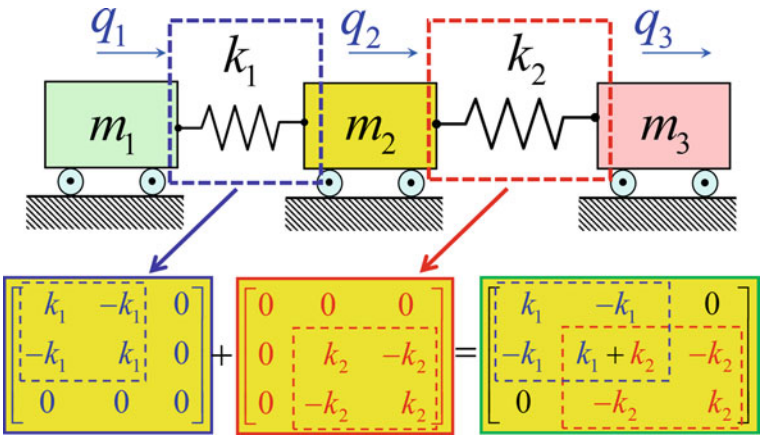


Fig. 6.67 Process of assembling global stiffness matrix of the 3DOF mass-spring system

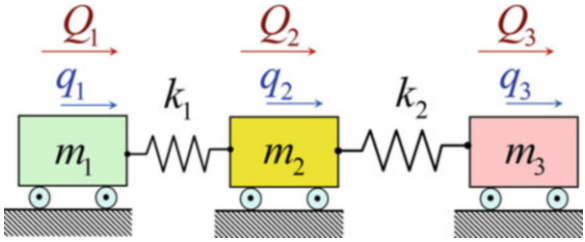


Fig. 6.68 Dynamic excitation of the unconstrained 3DOFs mass-spring system

reflected in the matrix $[K]$ by adding components of $\left[k_{ij}^{e}\right]$ to the relevant cells in $[K]$. This process is illustrated in Fig. 6.67.

This result enables us to write general dynamics equation for the system for an even more general case, when the arbitrary external forces Q_1, Q_2, Q_3 are applied to the system (Fig. 6.68):

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{Bmatrix} \tag{6.90}$$

This equation is the same as it would be an equation derived using traditional method, based on the free-body diagram and Newton's Second Law.

As further development of this topic, we now present two main techniques to model systems which have some of the boundaries of their springs being constrained, i.e. possess with constrained boundary conditions. The simplest case is presented in Fig. 6.69a. The original system with one spring, left boundary of which is constrained, is shown in Fig. 6.69a. To derive global stiffness matrix, we employ a *supplementary* unconstrained system, which is created by adding to the

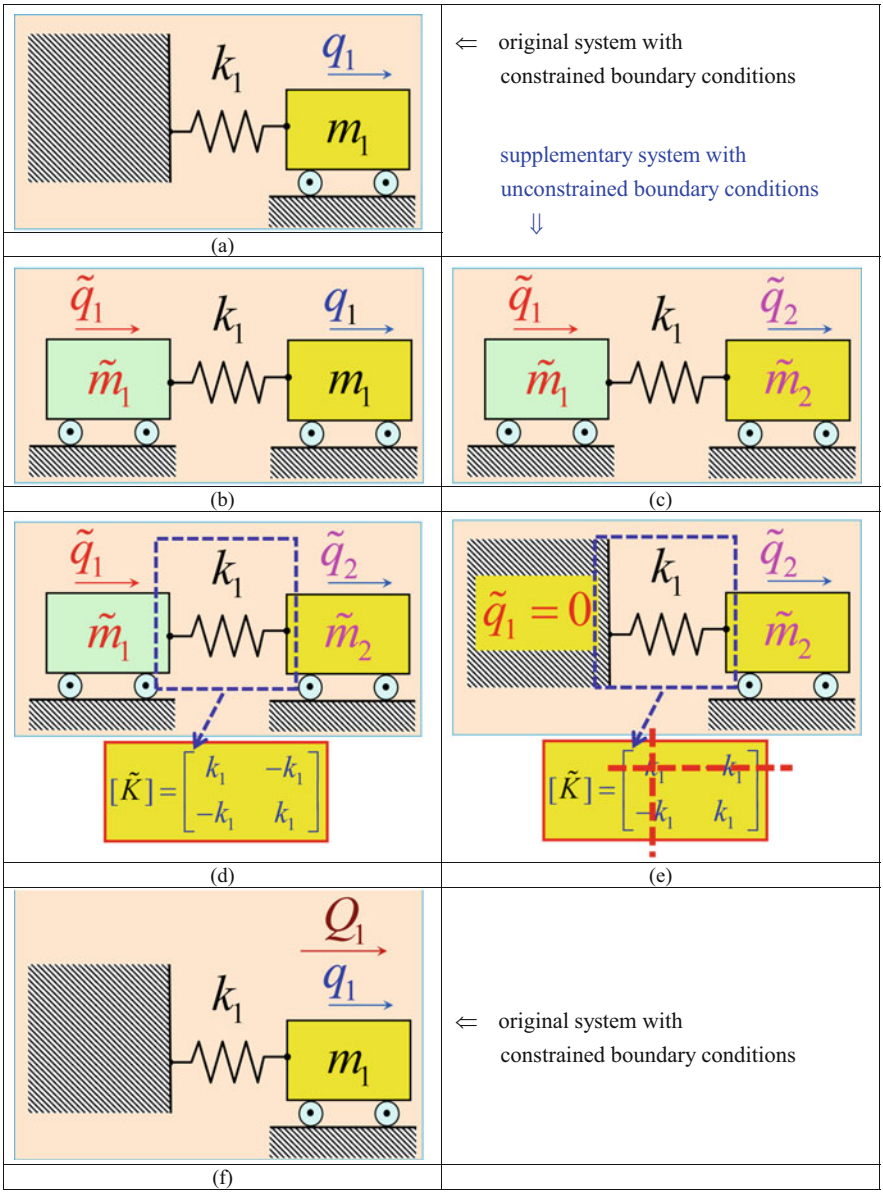


Fig. 6.69 Process of taking into account constrained DOFs in the global matrices derivation

original system a dummy mass, replicating left wall (Fig. 6.69b). Essentially, we add additional DOF ($\tilde{q}_1$) to the system, which has an additional mass ($\tilde{m}_1$). In the supplementary system, for consistency and programming convenience, we rename notations (Fig. 6.69c). Now supplementary system has consistent numbering and

notations and we can produce its global stiffness matrix, using assembling process, explained earlier (Fig. 6.69d). By the way, global stiffness matrix $\left[\tilde{K}\right]$ for the supplementary system (in this purposely simplified case) is equal to the element stiffness matrix.

The static equation for the supplementary unconstrained system can be written as follows:

$$\left[\tilde{K}\right] = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \tag{6.91}$$

A requirement to constrain left end of the spring would mean that $\tilde{q}_1 = 0$ and the first equation, corresponding to $\tilde{q}_1$ and the information in the first column of $[\tilde{K}]$ are becoming irrelevant and can be crossed out, as shown in Fig. 6.69e.

Now we can return to the initial original system, replacing the dummy mass in the supplementary system with a wall. Returning to the initial original system and also notations (Fig. 6.69f), we can now use the inherited "truncated" or "condensed" matrix $\left[\tilde{K}\right]$as a global stiffness matrix of the original system (i.e. $[K] = [k_1]$) and can write an equation of motion of the original system, derived using the FEM, as follows:

$$m_1\ddot{q}_1 + k_1 q_1 = Q_1 \tag{6.92}$$

Another technique to take into account constrained DOFs in the system is also involving creation of the unconstrained supplementary system, but suggests assignment of very large value (let say, 10^{12} kg for the main mass being a few kilograms) to the added mass [15]. This is a very convenient method for numerical implementation, as it does not require truncation of the global matrices for the supplementary system and additional renumbering manipulations. More advanced examples may involve, in a general case, n masses, as in Fig. 6.70. The process of treating constrained system is shown in Fig. 6.71a–6.71d for $n = 4$. Figures 6.72a, b also present MATLAB scripts to solve eigenvalue problems.

In the current illustrated example, the following parameters are assumed: $n = 4$; $m = 7$ kg; $k = 500$ N/m. Figure 6.72a shows that the first natural frequency of the

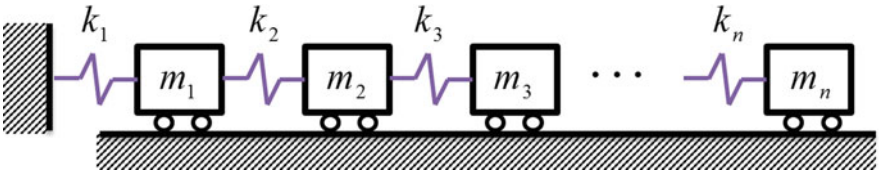


Fig. 6.70 n-DOFs mass-spring system

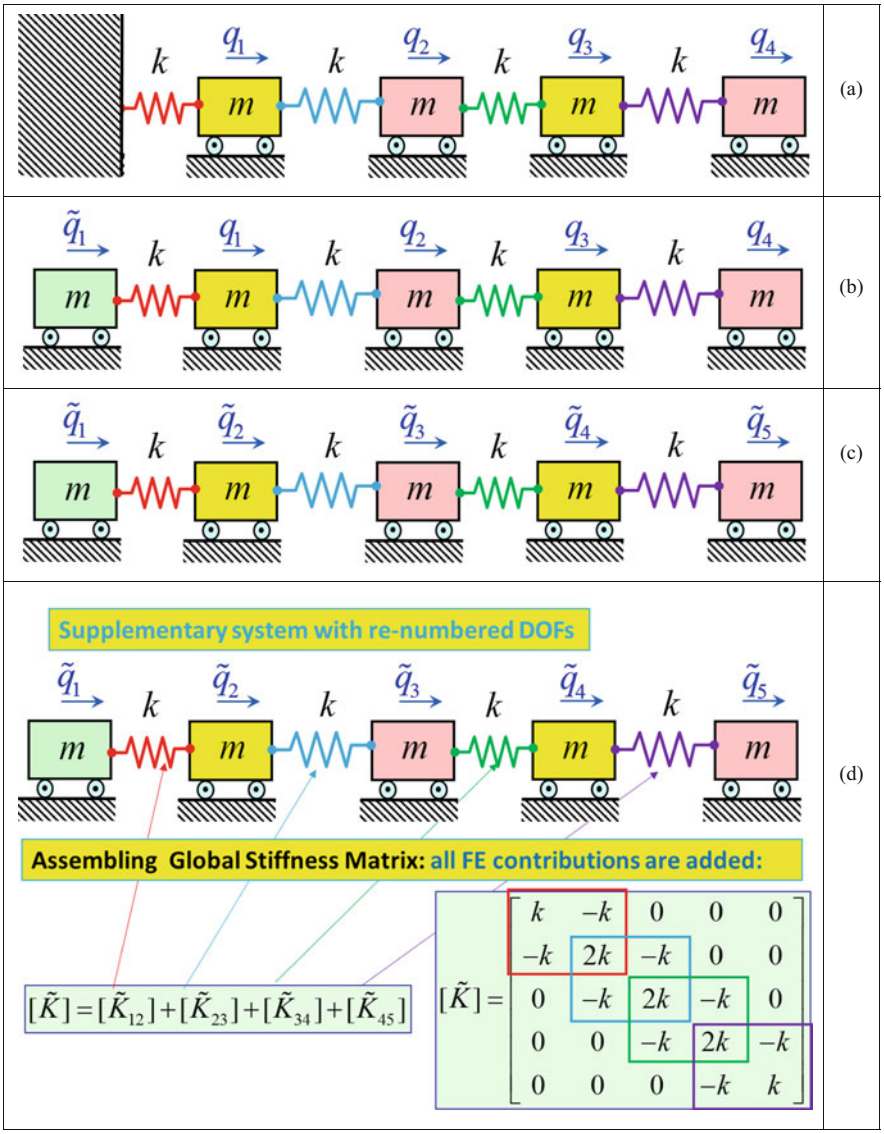


Fig. 6.71 Mass-spring system example: (**a**) original system; (**b**) supplementary system; (**c**) supplementary system renumbered; (**d**) assembly of the stiffness matrix for the supplementary system

supplementary system is equal to zero, which would be an expected result, as the system is not constrained and admits rigid body motion.

Return to the original system would imply that $\tilde{q}_1 = 0$ and, mathematically, this would mean that we can get global stiffness matrix from the supplementary

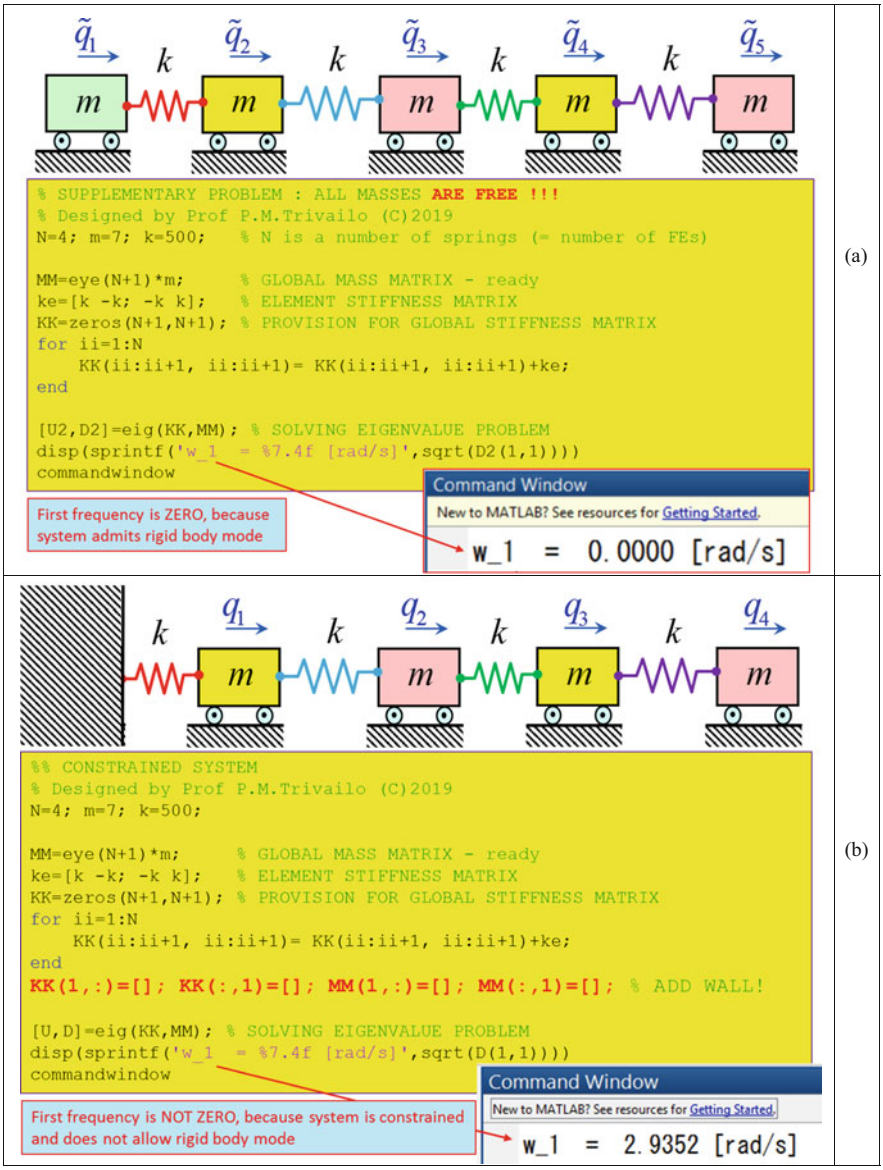


Fig. 6.72 Mass-spring system example: (**a**) calculation of the first natural frequency of the supplementary system; (**b**) calculation of the first natural frequency for original system

system matrix by crossing out its first row and column. MATLAB script for these calculations is shown in Fig. 6.72b, together with the calculated first natural frequency for the original system, which is equal to $\omega_1 = 2.9352$ rad/s.

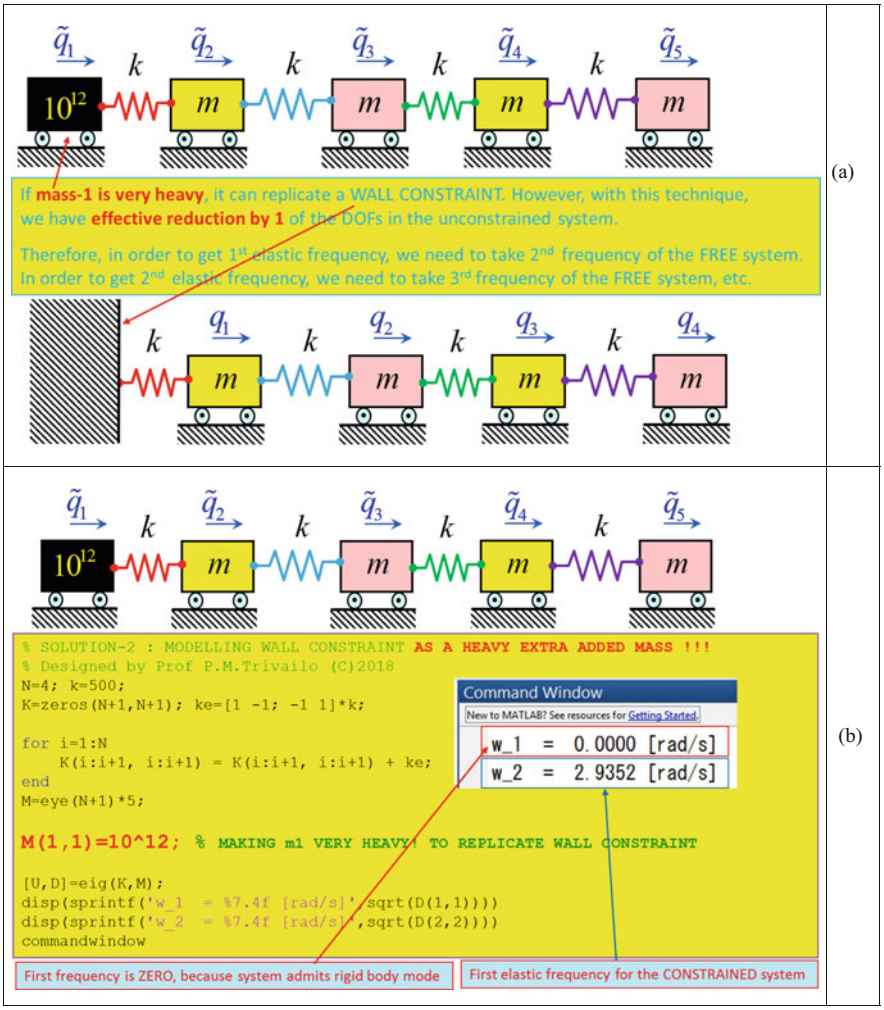


Fig. 6.73 Unconstrained supplementary system in taking into account constrained masses: (**a**) method of replicating a wall by adding a heavy mass; (**b**) analysis of the first natural frequencies

Lastly, in Fig. 6.73 we illustrate second method of modelling constrained system in Fig. 6.71a. This is very simple for programming and quite an efficient method, suggesting that the constraining wall in Fig. 6.71a could be replaced with a very heavy mass, shown in black colour on the diagram, Analysis of the natural frequencies shows that the first natural frequency for the supplementary system with heavy mass is equal to zero, and its second natural frequency is equal to the first elastic frequency of the original constrained mass (see Fig. 6.73).

6.5.2 *Utilization of the FEM for Modelling of the Axial Vibrations of Elastic Rods*

Axial vibration of elastic rods is a classical task. Indeed, many structures and structural elements are subjected to longitudinal loadings and can vibrate in the longitudinal (axial) direction. The typical examples of them are: rockets, turbine blades, helicopter blades, elements of magnetostriction devices, etc.

These complex vibrations can be nicely illustrated/visualized, using conceptual analogies from physics (see Fig. 6.74). Figure 6.74a displays a sound wave propagation, and Fig. 6.74b shows coiling spring, which can be used to easily send longitudinal waves along the spring: for this, free end should be pulled in and out.

However, in the context of this book chapter, axial vibration of rods task attracts our attention with the availability of analytical solutions, which we will establish first to use them as a benchmark for the numerical approximate solution; we will also obtain after the analytical (exact) counterparts.

Axial vibrations of the elastic rods can be described with the partial differential equation [17]:

$$\frac{\partial}{\partial x}\left[E\mathcal{A}(x)\frac{\partial u\,(x,t)}{\partial x}\right]+f\,(x,t)=m(x)\frac{\partial^2 u\,(x,t)}{\partial t^2} \tag{6.93}$$

Equation of motion and the boundary conditions of the problem constitute what is referred to as a *boundary-value problem*.

In case of homogeneous rods ($E\mathcal{A}=$ const, $m=$ const), these equations for the free vibration case can be simplified to the format, known as the *wave equation* [18]:

$$\frac{\partial^2 u}{\partial x^2}=\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}\quad \text{where}\ \ c=\sqrt{E\mathcal{A}/m}=\sqrt{E/\rho} \tag{6.94}$$

In Eq. (6.94) x is a coordinate of the cross section of the rod, $u(x,t)$ is a function of axial displacements of its cross sections, c is the wave propagation velocity along the rod, E is its modulus of elasticity, m is the mass per unit length and ρ is the density of the material and $\mathcal{A}$ is the area of the cross section (Fig. 6.75).

The general solution of Eq. (6.94) can be obtained, using the method of separation of variables, assuming solution in the form $u(x,t)=U(x)\cdot F(t)$, and has the following form:

$$u\,(x,t)=\left(A\sin\frac{\omega}{c}x+B\cos\frac{\omega}{c}x\right)\times(C\sin\omega t+D\cos\omega t) \tag{6.95}$$

where arbitrary constants A, B, C, D depend on the *boundary conditions* and the *initial conditions*.

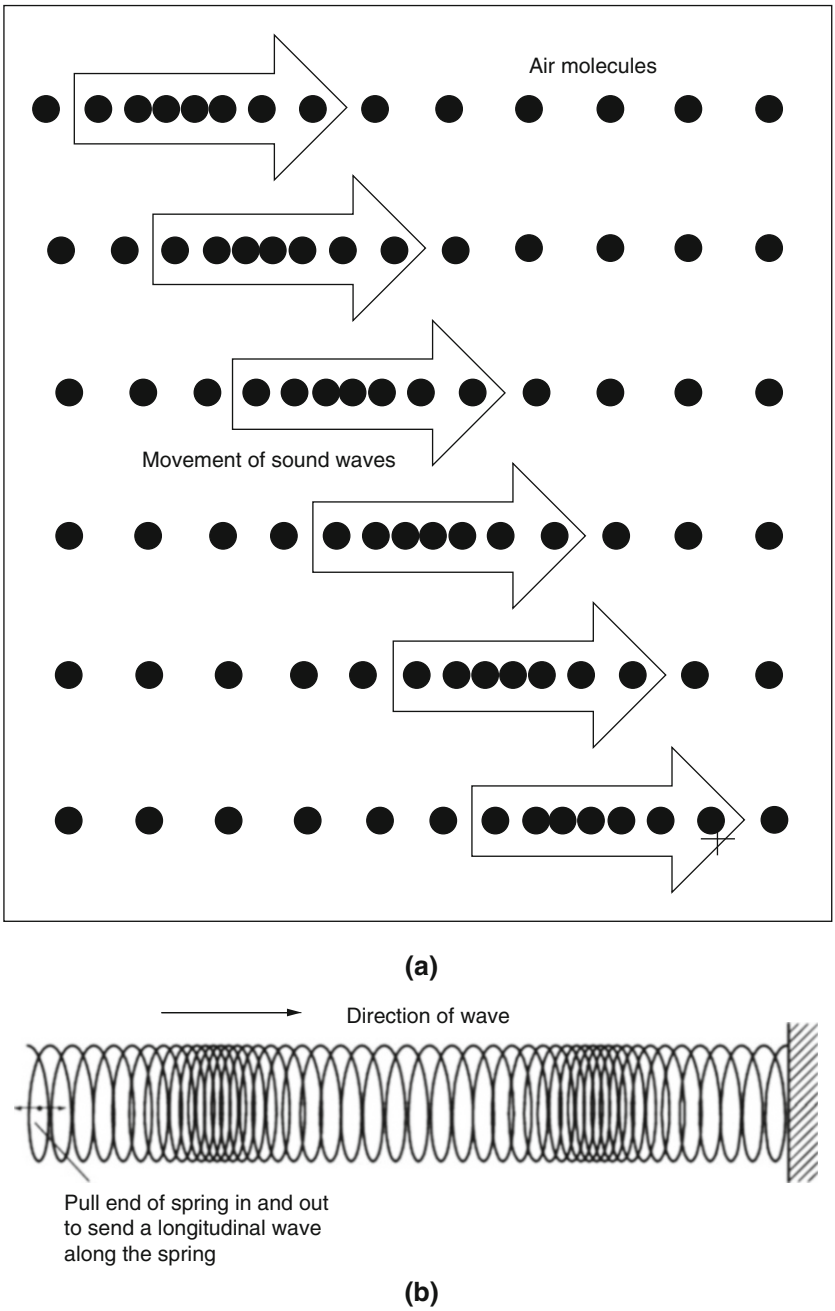


Fig. 6.74 Analogies with axially vibrating rods: (**a**) propagation of a sound wave; (**b**) propagation of a transverse wave in a coiling spring [16]

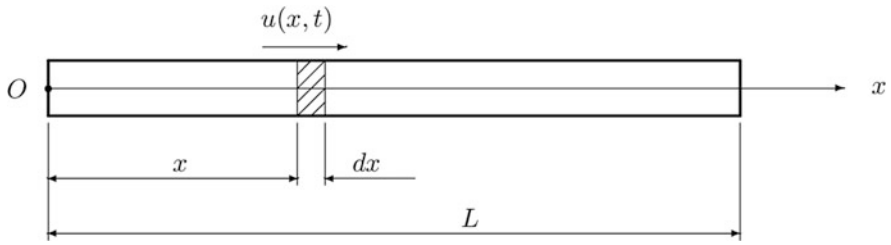


Fig. 6.75 Homogeneous elastic rods in axial vibrations: notations

Since the bar has free ends, the axial force, which is proportional to dU/dx, must be zero at each extremity. Thus, the boundary conditions for this problem may be written as

$$EA\frac{\partial u\,(x,t)}{\partial x}\bigg|_{x=0} = 0, \quad EA\frac{\partial u\,(x,t)}{\partial x}\bigg|_{x=L} = 0 \tag{6.96}$$

The first boundary condition will require that $A = 0$, so

$$u\,(x,t) = \left(B\cos\frac{\omega}{c}x\right) \times (C\sin\omega t + D\cos\omega t) \tag{6.97}$$

The second boundary condition then leads to the *characteristic equation*:

$$\sin\frac{\omega L}{c} = 0, \quad \text{or} \quad \frac{\omega_r L}{c} = r\pi, \quad r = 1, 2, 3, \ldots \tag{6.98}$$

to which corresponds the infinite set of mode shapes, described with *eigenfunctions*:

$$U_r(x) = B_r\cos\frac{r\pi x}{L} \tag{6.99}$$

The first natural elastic modes for the free-free rod are plotted in Fig. 6.76 [17], where the modes have been normalized by letting $B_r = 1$. We note that the first mode has one node, the second has two nodes and the third has three odes. In general, the rth elastic mode has r nodes ($r = 1,2,\ldots$).

The system natural frequencies are:

$$\omega_r = \frac{r\pi c}{L} = r\pi\sqrt{\frac{E}{\rho L^2}}, \quad r = 1, 2, 3, \ldots \tag{6.100}$$

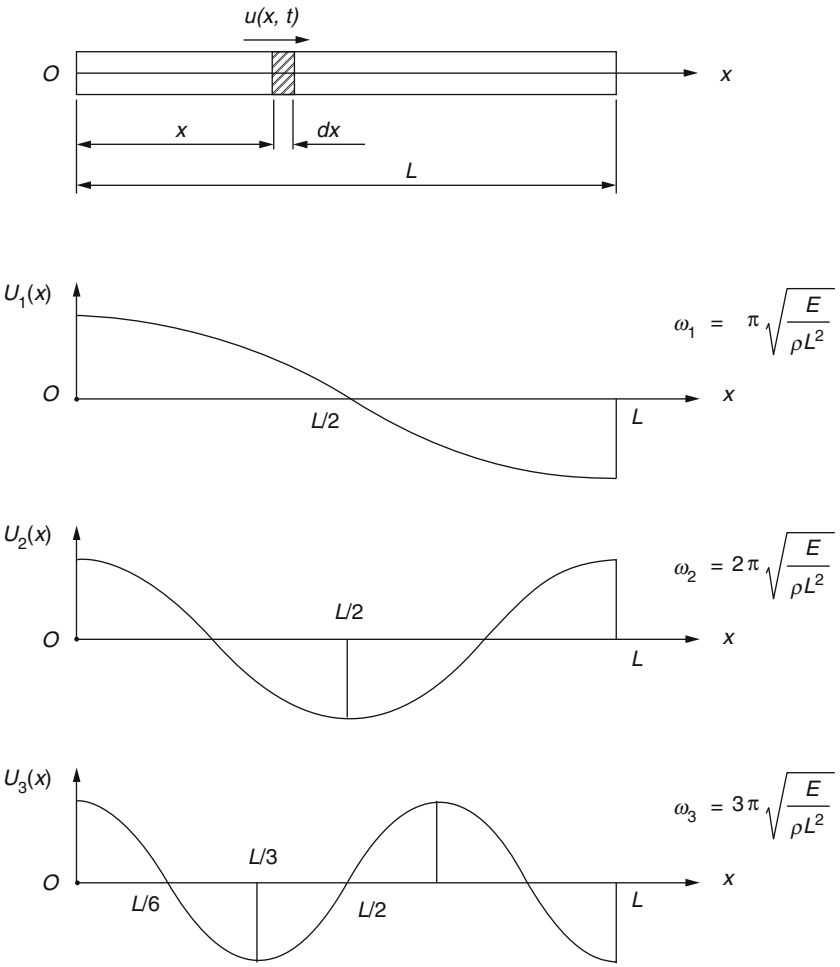


Fig. 6.76 The first three natural elastic modes of longitudinal vibration for a free-free bar

Note that for the free-free boundary conditions, system allows rigid-body mode, characterized with zero frequency, and this specific case can be obtained from Eq. (6.100) using $r = 0$. Non-zero values of r are related to the so-called "elastic" modes. It should be also noted, that the system has ***infinite*** number of natural frequencies, and can be called as a system with infinite number of degrees-of-freedom.

In the more general case of free vibration initiated in any manner, the solution will contain many of the normal modes:

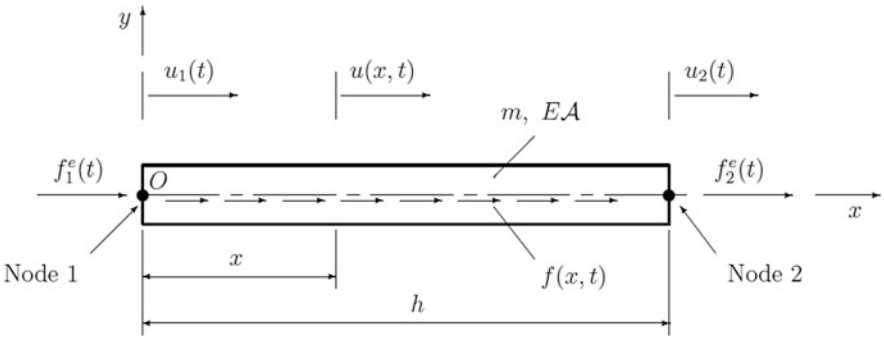


Fig. 6.77 Truss finite element: main notations

$$u\,(x,t) = \sum_{r=1}^{\infty} \cos\frac{r\pi x}{L}\left(C_r \sin\omega_r t + D_r \cos\omega_r t\right), \quad \omega_r = \frac{r\pi c}{L}, \quad r = 1, 2, 3, \ldots \tag{6.101}$$

The arbitrary constants C_r, D_r depend on the *initial conditions*. As with increased need for denoting too many parameters in this chapter, we are quickly running out of notations, in the current context, coefficient A, B, C and D should not be mixed up with the matrices in the state-space formulation, considered earlier in previous sections.

With the exact benchmark solutions Eq. (6.100) for the natural frequencies established, we will obtain the *natural frequencies* of the same system, using the FEM, i.e. alternative method. The justification for this would be in the ability to solve a much wider range of problems, including those, where derivation of the analytical solutions would not be possible, for example in non-linear formulations.

Let us now consider a single finite element (see Fig. 6.77) with two nodal axial displacements, u_1 and u_2, being two degrees-of freedom.

Using so-called Hermite functions, H_1 and H_2, also known as shape functions [14] (shown in Fig. 6.78), it would be possible to derive an expression for calculation of the axis displacement at any internal point with arbitrary coordinate x, using nodal values [17]:

$$\begin{aligned} &u\,(x,t) = H_1(x)u_1(t) + H_2(x)u_2(t) = \left(1 - \tfrac{x}{h}\right)u_1(t) + \tfrac{x}{h}u_2(t), \quad \text{where} \\ &H_1(x) = \left(1 - \tfrac{x}{h}\right) \quad \text{and} \quad H_2(x) = \tfrac{x}{h} \end{aligned} \tag{6.102}$$

Using Eq. (6.102), it is possible to derive single finite element mass and stiffness matrices:

$$\left[k^{\mathrm{e}}\right] = \frac{EA}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}; \quad \left[m^{\mathrm{e}}\right] = \frac{mh}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{6.103}$$

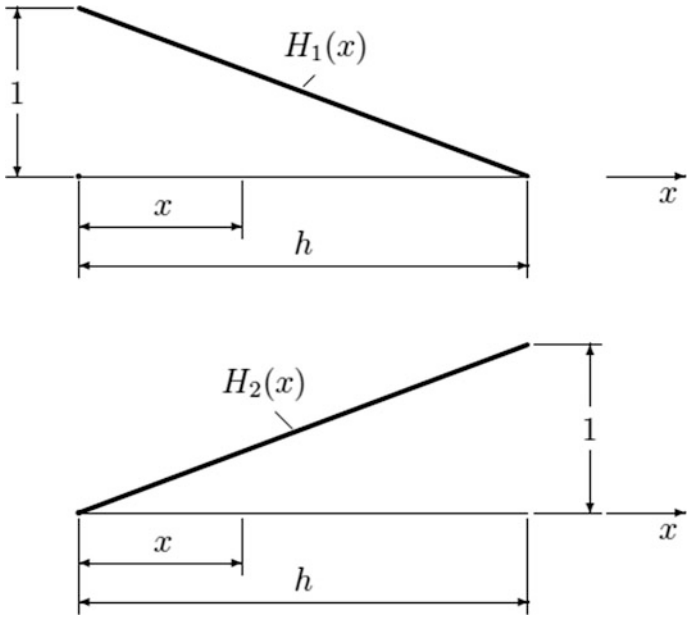


Fig. 6.78 Shape functions for a truss finite element

and equivalent nodal forces:

$$\begin{aligned} f_1^e(t) &= \int_0^h f(x,t)\left(1-\tfrac{x}{h}\right)\,dx = \int_0^h f(x,t)\,H_1(x)dx \\ f_2^e(t) &= \int_0^h f(x,t)\left(\tfrac{x}{h}\right)dx = \int_0^h f(x,t)\,H_2(x)\,dx \end{aligned} \tag{6.104}$$

Let us assess, how expressions (6.103) work. Consider a rod, modelled with a single finite element only. Then, for the static case, system can be modelled with the following relationship:

$$\begin{aligned} &[k^e]\{u\} = \{F\} \quad \text{(matrix equation)}, \text{ or,} \\ &\frac{EA}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \text{ (the same, but in the expanded format)} \end{aligned} \tag{6.105}$$

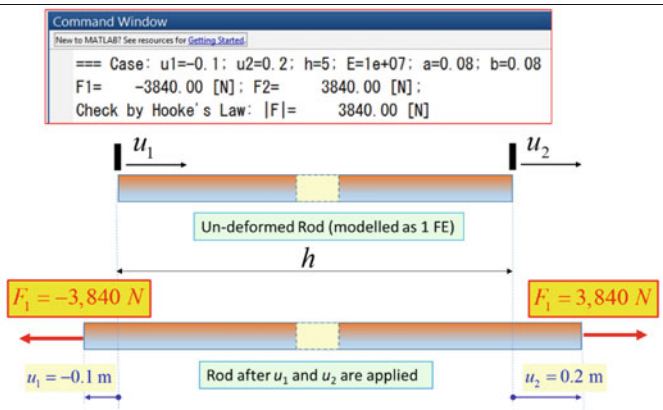
For the illustration purposes, let us assume the following numeric values for the example: $a = 0.08$ m; $b = 0.08$ m; $h = 5$ m; $u_1 = -0.1$ m; $u_2 = 0.2$ m; $E = 0.01 * 10^9$ Pa. Calculation of the forces, required to ensure end rod's displacements, u_1 and u_2, using Eq. (6.105) produces verifiable results, illustrated in Fig. 6.79b.

As next sophistication of the model, we model the bar ($L = 1$ m; $E = 0.01 * 10^9$ Pa; $\rho = 1.2 * 10^3$ kg/m^3; $a = 0.08$ m; $b = 0.08$ m) introducing 50

```
%% BASIC STATIC MODELLING OF THE ROD WITH 1 TRUSS FE
% Designed by Prof P.M.Trivailo (C) 2019
%-------------------------

h=21; a=0.08; b=0.08; h=5; u1=-0.1; u2=0.2; E=0.01*10^9; % m and Pa
A=a*b; EA=E*A;
% --- Calculate Forces using FEM ---
ke=(EA/h)*[1 -1; -1 1]; F=ke*[u1;u2];
clc;
disp(sprintf('=== Case: u1=%g; u2=%g; h=%g; E=%g; a=%g; b=%g',u1,u2,h,E,a,b))
disp(sprintf('F1=%12.2f [N]; F2=%12.2f [N]; ',F(1),F(2)))
% --- Checking Axial Forces using Hooke's Law ---
Fchk=(u2-u1)*EA/h;
disp(sprintf('Check by Hooke''s Law: |F|=%12.2f [N]',Fchk))
commandwindow
```

(a)



(b)

Fig. 6.79 Basic example of static modelling of the rod with only one truss finite element: (**a**) MATLAB script; (**b**) FEM solution for $F_{1,2}$ and its verification against Hooke's Law

finite elements and calculate natural frequencies of the rod. These numerical model results will be compared with the exact solutions, given by Eq. (6.100).

Complete MATLAB script, assembling the system's global matrices, performing required calculations, solving the eigenvalue problem and plotting approximated numerical and exact analytical results is given in Fig. 6.80. This is a universal script, enabling simulations of rods modelled with any number of the FEs. To change this number, the value of NumFE variable should be changed. Considering two contrast cases with NumFE $= 1$ and NumFE $= 50$, we generate plots to compare analytical and FEM results (see Fig. 6.81). Figure 6.81 shows that in addition to the elastic frequencies, the FEM models enabled calculation of the rigid-body mode frequency with zero value $\omega_0 = 0$. And this observation is valid for any number of the finite elements in the model, even for one element model. Also, Fig. 6.81 shows that the accuracy of the FEM predictions can be improved, using higher number of the elements in the model. In our specific example, with 50 finite elements model, at least for 20 first frequencies high accuracies could be achieved.

Next step in sophistication of the FEM modelling is in considering response tasks of the system with *constrained* boundary or boundaries.

```
%% ----- EXAMPLE: FEM modelling of free-free axial rod
% Designed by Prof P.M.Trivailo (C) 2018
%=== ENTERING DATA =========
L=1; E=0.01*10^9; rho=1.2*10^3; a=0.08;  b=0.08;  A=a*b;  EA=E*A;
c=sqrt(E/rho); NumFE=50; h=L/NumFE; m=A*1*rho;

%=== BUILDING [M] & [K] global matrices =========
ke=(EA/h)*[1 -1; -1 1]; me=(m*h/6)*[2 1; 1 2];
M=zeros([1,1]*(NumFE+1)); K=zeros([1,1]*(NumFE+1));
for ii=1:NumFE
    idx=[1 2]+(ii-1);
    M(idx, idx)=M(idx, idx)+me;   K(idx, idx)=K(idx, idx)+ke;
end
%=== Solving Eigenvalue Problem =========
[U,D]=eig(K,M);

%=== Plotting FEM and exact frequencies =========
w1_exact=1*pi*sqrt(E/(rho*L^2));

figure; grid on; hold on; xlim([1 NumFE+1]);
for i=1:NumFE+1
    w_exact=(i-1)*w1_exact; w_FEM=sqrt(abs(diag(D(i,i))));
    disp(sprintf('w_%2i = %7.2f rad/s   w_exact_%2i = %7.2f rad/s',i,w_FEM,i,w_exact))
    plot(i,w_exact,'ob','MarkerSize',8); % plot i-th exact frequency
    plot(i,w_FEM,'xr','MarkerSize',8);   % plot i-th FEM frequency
end

%=== "Documentation": ading legend, labels and title =========
str=sprintf('approximated frequencies (FEM solution with NumFE=%i)',NumFE);
legend('exact frequencies (analytical solution)',str,'Location','NorthWest');
xl=xlabel('Frequency Number $r$'); yl=ylabel('$\omega_r$ [rad/s]');
set([xl,yl],'Interpreter','LaTeX'); title('\bf Free-Free Rod');
%
set(gca,'FontSize',16); set(gcf,'Position', [80 200 1100 360]);
if NumFE<10, set(gca,'XTick',[1:NumFE+1]); end
commandwindow
```

Fig. 6.80 MATLAB script, enabling calculation of the natural frequencies of the rod in axial vibration and comparison with exact analytical solutions

For this illustration, let us consider a uniform rod of length L and cross section $a \times b = \mathcal{A}$, with one end fixed and the other free. This rod is stretched under a static load P_0, as shown in Fig. 6.82, and suddenly released from rest at time $t = 0$. From these initial conditions, determine the longitudinal displacements $u(x,t)$.

For certainty, we assume the following numeric parameters:

- $a = 0.08$ m
- $b = 0.08$ m
- $L = 5$ m
- $\varepsilon = 0.1$
- $E = 0.01 * 10^9$ Pa

The exact analytical solution for the task can be given by the following expression [17]:

$$u(x,t) = \frac{8\varepsilon L}{\pi^2} \sum_{r=1}^{\infty} \frac{(-1)^{r-1}}{(2r-1)^2} \sin \frac{(2r-1)\pi x}{2L} \times \cos \frac{(2r-1)\pi ct}{2L} \qquad (6.106)$$

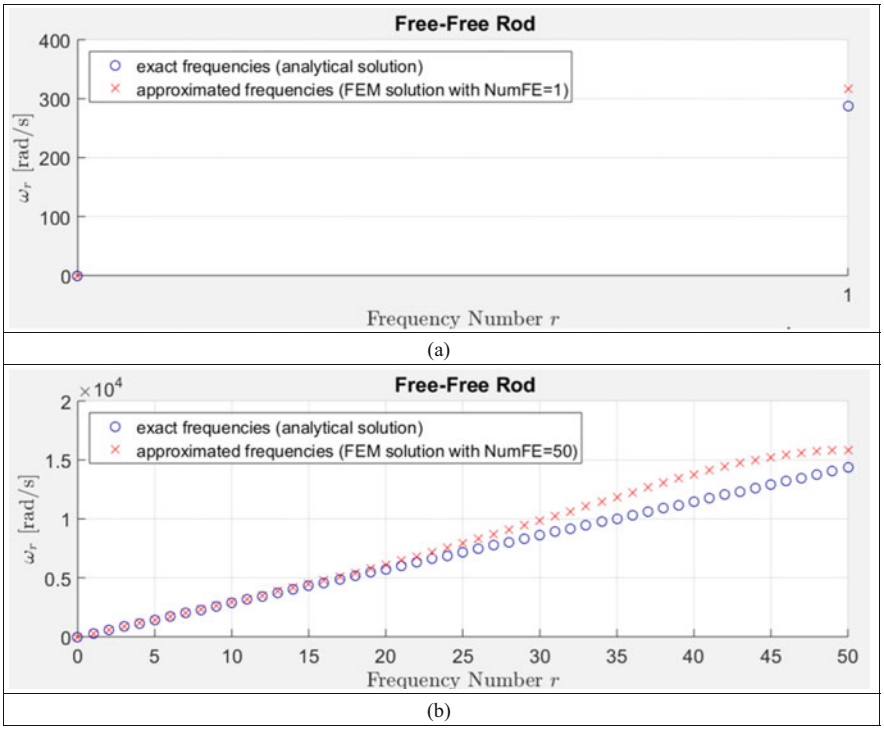


Fig. 6.81 Natural frequencies of the rod in axial vibrations: exact and FEM values: (**a**) finite element model with 1 FE; (**b**) finite element model with 50 FEs

It involves summation with infinite number of contributions from all possible natural modes.

Figure 6.83b–d show the contribution of the first three modes to the total response of the bar. It may be appreciated that the amplitudes of various modes of vibration are rapidly decreasing as r increases.

As one of the features of this example, let us represent Eq. (6.106) as a 3D surface plot. Why this representation is a recommended format?—Because informatively it is equivalent to the animation. However, animation cannot be included in the report, but surface plot can be.

The appropriate script is reproduced in Fig. 6.84 and result of its execution is shown in Fig. 6.85.

As in previous examples, after obtaining analytical solution, we proceed with its alternative—numerical solution. At the initial stage of the process, we consider the simplest FEM model, comprising only one single finite element. To model clamped-free system, we first build the system's matrices for the free-free system, which would be a supplementary system. For one-FE model of the free-free rod with $h = L$, the dynamic equations of motion can be formulated, based on the Newton's Second Law, and can be written in the following matrix form:

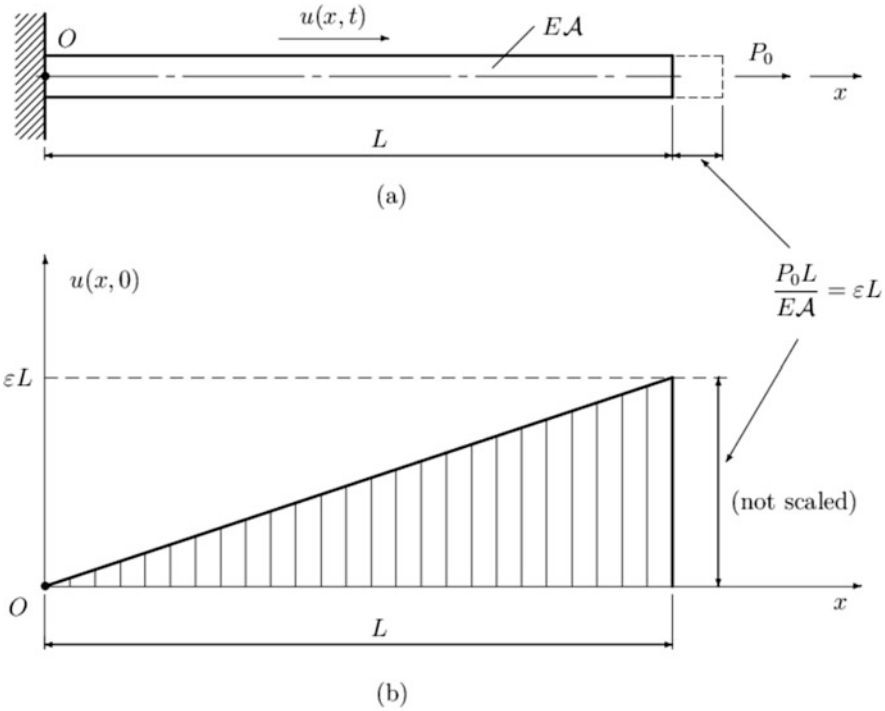


Fig. 6.82 A clamped-free uniform rod: (**a**) study case system; (**b**) initial conditions to excite its longitudinal vibrations

$$\frac{mh}{6}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}\begin{Bmatrix}\ddot{u}_1\\ \ddot{u}_2\end{Bmatrix}+\frac{EA}{h}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{Bmatrix}u_1\\ u_2\end{Bmatrix}$$
$$=\begin{Bmatrix}0\\ 0\end{Bmatrix}\ \text{(for supplementary free-free rod)} \tag{6.107}$$

Applying boundary conditions, we need to constrain left boundary ($u_1 = 0$), which would be equivalent to crossing out the first equation and removing first rows on the remaining $[m^e]$ and $[k^e]$ matrices. Thus, for the study case with clamped-free boundaries, we can reduce the previous Eq. (6.107) (formulated for the supplementary system) to the following:

$$\begin{aligned}&\tfrac{mh}{6}[2]\{\ddot{u}_2\}+\tfrac{EA}{h}[1]\{u_2\}=\{0\}\ \text{(for main system, fixed-free rod) or}\\ &[M]\ddot{u}_2+[K]u_2=[0]\ \left(\text{here } [M]=\tfrac{mh}{6}[2] \text{ and } [K]=\tfrac{EA}{h}[1]\right)\end{aligned} \tag{6.108}$$

Because Eq. (6.108) is the second order equation, in order to access the differential equation solvers, we now reformulate our task in terms of states, which results in reduction of the order of the equations to the first order at the expense of increase of the number of equations.

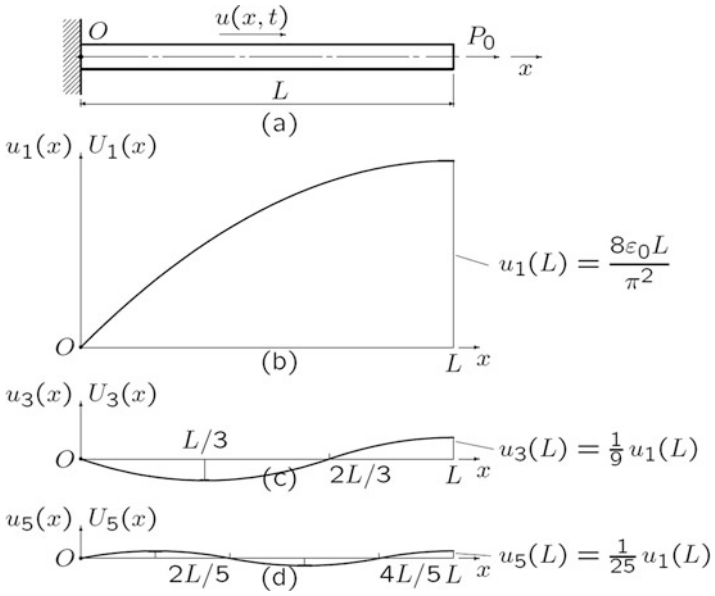


Fig. 6.83 Rod in axial vibrations: (**a**) key notations; (**b**)–(**d**) scaled contribution of the first three modes to the total longitudinal response of the rod, excited with initial conditions

```
%% EXCITED ROD: 3D SURFACE PLOT FOR EXACT SOLUTION
% Designed by Prof P.M.Trivailo (C) 2019
clear; clc; close all
L=5; E=0.01*10^9; rho=1.2*10^3; tmax=0.5; eL=0.1; e=eL/L; c=sqrt(E/rho);

[TT,XX]=meshgrid([0:.001:tmax],[0:.01:1]*L);

coef=2*e/L;
zzAnalyt=0*TT; NN=100;
for i=1:size(TT,1)
    for r=1:NN
        r2=(2*r-1)*pi/(2*L);
        zzAnalyt(i,:)=zzAnalyt(i,:)+...
            coef*(-1)^(r-1).*sin(r2*XX(i,:)).*cos(r2*c*TT(i,:))/(r2^2);
    end
end
figure; hold on; grid on; rotate3d on;
sur=surf(TT,XX,zzAnalyt); colormap jet; lighting phong; shading interp;
contour3(TT,XX,zzAnalyt,[-1:0.2:1]*eL,'k');

%--- hDecoratedh plotting ---
colorbar; camlight; camlight right; view([29, 12]);
xl=xlabel('$t$ [s]'); yl=ylabel('$x$ [m]'); zl=zlabel('$u(x,t)$ [m]');
tit=title(sprintf('\\bf Analytical exact solution for summation index $N=%i$',NN));
set([xl,yl,zl,tit],'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[70 60 1160 680]);
```

Fig. 6.84 MATLAB script, plotting exact solution for the rod, excited with initial conditions, as a 3D surface

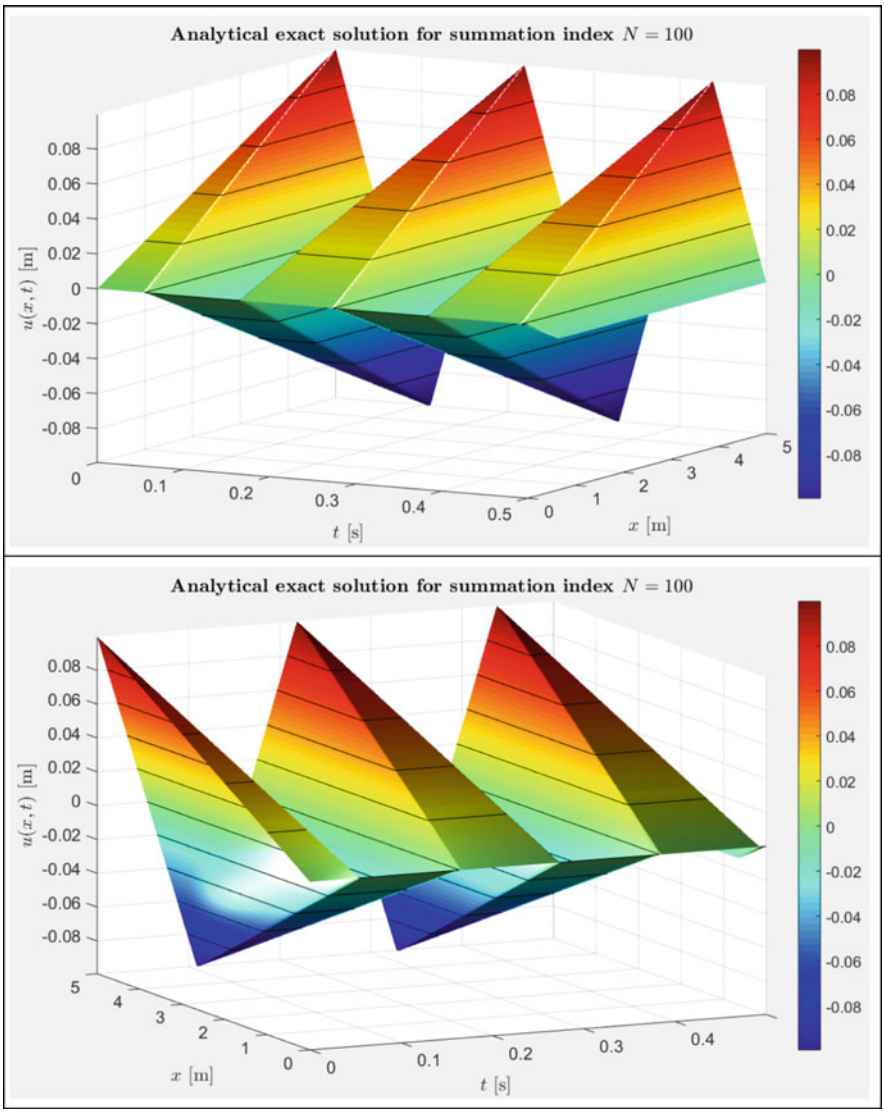


Fig. 6.85 A clamped-free uniform rod: exact analytical (solution with $N = 100$ members kept in the summation), plotted as a 3D surface (this is the same plot, but viewed from two different viewpoints)

If introduced states are $\mathbf{x} = [x_1, x_2]^{\mathrm{T}} = [u_2, \dot{u}_2]^{\mathrm{T}}$, then the FEM dynamics matrix equation for the clamped-free rod, modelled with one finite element can be written in the state form:

```
%% BASIC RESPONSE OF THE CLAMPED-FREE ROD
% Designed by Prof P.M.Trivailo (C) 2019
%---------------------------
a=0.08; b=0.08; h=5;          % Data in [m]
E=0.01*10^9;                  % Young Modulus [Pa]
rho=1.2*10^3;                 % density [kg/m^3]
tmax=0.5;                     % Simulation time [s]
e=0.1; A=a*b; EA=E*A; m=rho*a*b*1;

% --- SUPPLEMENTARY SYSTEM: Modelling rod with only 1 FE ---
ke=(EA/h)*[1 -1; -1 1]; me=(m*h/6)*[2 1; 1 2];

% --- MAIN SYSTEM: Constraining Left Boundary ---
me(:,1)=[]; me(1,:)=[]; ke(:,1)=[]; ke(1,:)=[];

invMxK=inv(me)*ke; x0=[e; 0];
FEM_xdot_anon = @(t,x) [x(2); -invMxK*x(1) ];
[tt,xx] = ode45(FEM_xdot_anon,[0 tmax],x0);
figure; plot(tt,xx(:,1),'b','LineWidth',2); grid on;

xl=xlabel('$t$  [s]'); yl=ylabel('$u_2(t)$   [m]')
str=sprintf('$u_2$ Time History (Axial displacement of tip of fixed rod, modelled with only 1 FE)');
ti=title(str,'FontWeight','bold'); set([xl,yl,ti],'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[120  30  1200  240]);
```

Fig. 6.86 MATLAB script, enabling FEM modelling of the response of the clamped-free uniform rod, modelled with one FE

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{Bmatrix} x_2 \\ -\text{inv}\,([M]) * [K] * x_1 \end{Bmatrix} \tag{6.109}$$

involving inversion of the mass matrix, which is always positive for the engineering systems.

Corresponding MATAB script, enabling solution of this matrix equation is given in Fig. 6.86.

Approximated response of the system, modelled with only one FE, is presented in Fig. 6.87.

We can further enhance representation of the results of the simulation, using 3D surface plotting, shown in the annotated Fig. 6.87b. The corresponding MATLAB script is presented in Fig. 6.88.

As a matter of clarification, we need to stress out that, in most cases, modelling of the system with only one FE would not give sufficient accuracy, and such simplistic modelling was used only for illustration purposes. Therefore, as it will be further shown, the results of the simplistic models may not be realistic and reliable. Saying this, when it comes to the FEM implementation, starting programming with a simple model may not be a bad idea, as this approach enables to proceed quicker with the development of the prototype of the program, which could be easily converted later on in the high fidelity program.

Indeed, if we increase the number of the FEs in the model to 50, the shape of the 3D surface plot representing FEM solution is closely replicating the shape of the same plot, corresponding to the exact (analytical) solution. Placement side by side of these plots in Fig. 6.89 confirms this observation.

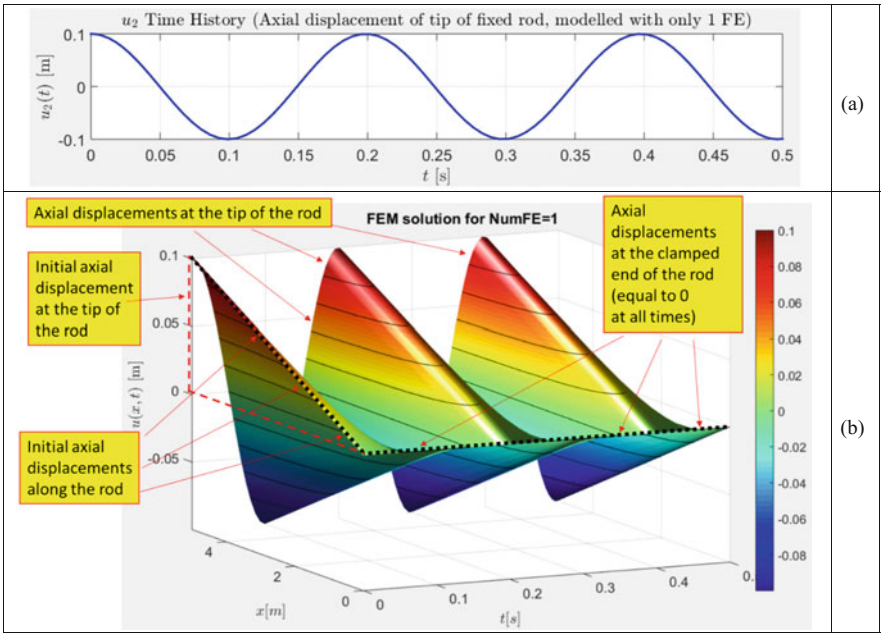


Fig. 6.87 FEM response of the clamped-free uniform rod, modelled with one FE: (**a**) plot for the tip point; (**b**) annotated 3D surface plot for all points along the rod

```
%% 3D SURFACE PLOT FOR the FEM MODEL
% Designed by Prof P.M.Trivailo (C) 2019
%--------------------------
figure; hold on; grid on; rotate3d on;
[TT,XX]=meshgrid(tt,[0 h]); ZZ=[0*xx(:,1)'; xx(:,1)'];
surf(TT,XX,ZZ);
colormap jet; lighting phong; shading interp;
contour3(TT,XX,ZZ,[-1:0.2:1]*e,'k');
%--- "Decorated" plotting ---
colorbar; hh=camlight; camlight left;
set(hh,'Position',[0.09,0.04,0.03]); view([-25, 14]);
xl=xlabel('$t [s]$'); yl=ylabel('$x [m]$');
zl=zlabel('$u(x,t)$ [m]');
set([xl,yl,zl],'Interpreter','LaTeX');
title('\bf FEM solution for NumFE=1');
set(gca,'FontSize',16);
set(gcf,'Position',[70 60 1160 680]);
```

Main Commands

Fig. 6.88 FEM response of the clamped-free uniform rod, modelled with one FE (Continuation of the script in Fig. 6.86)

If we further increase the number of finite elements in the model up to 200, we can see that the shape of the 3D surface plot (see Fig. 6.91b) rapidly converges to the shape of the exact analytical solution (Fig. 6.85).

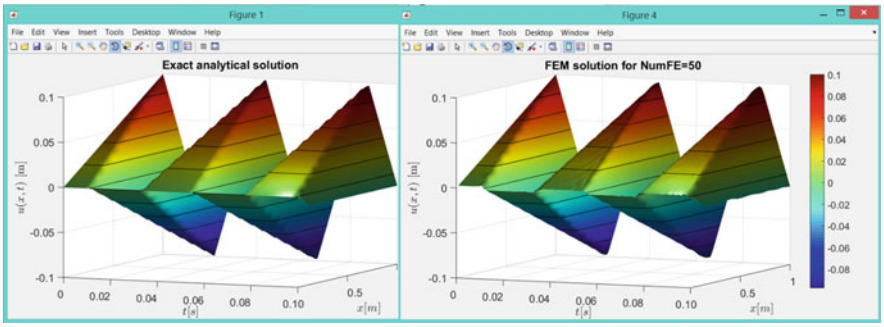


Fig. 6.89 Side by side comparison of the plots, corresponding to the exact solution and the FEM solution, obtained with 50 finite elements approximation (the same viewpoints as used)

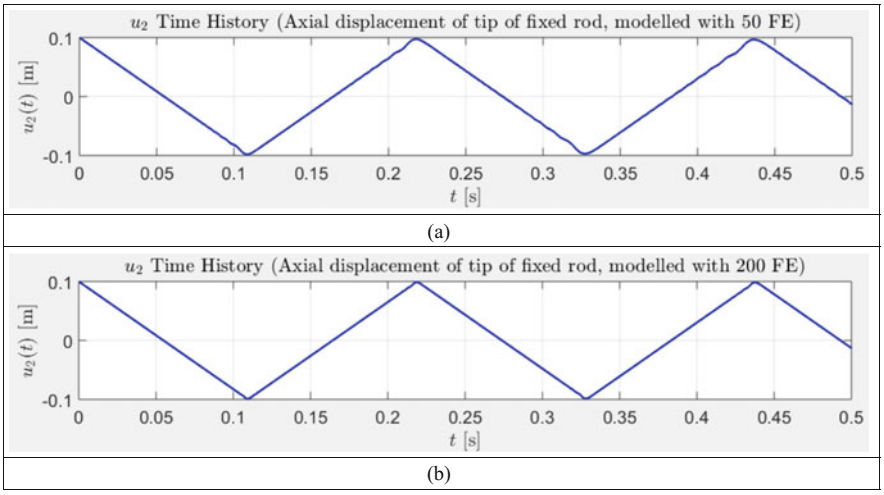


Fig. 6.90 FEM tip point responses of the clamped-free uniform rod: (**a**) model with 50 finite elements; (**b**) model with 200 finite elements

It is possible to observe that the plots for the models with lower number of FE have more "ripples", also reflected with the time histories of the displacement at the free end of the rod, presented in Fig. 6.90a. The segments of the plot for the 200 FE model, shown in Fig. 6.90b, more closely resemble straight lines.

For completeness, in Fig. 6.92 we provide complete script for the FEM modelling of the bar with arbitrary number of finite elements.

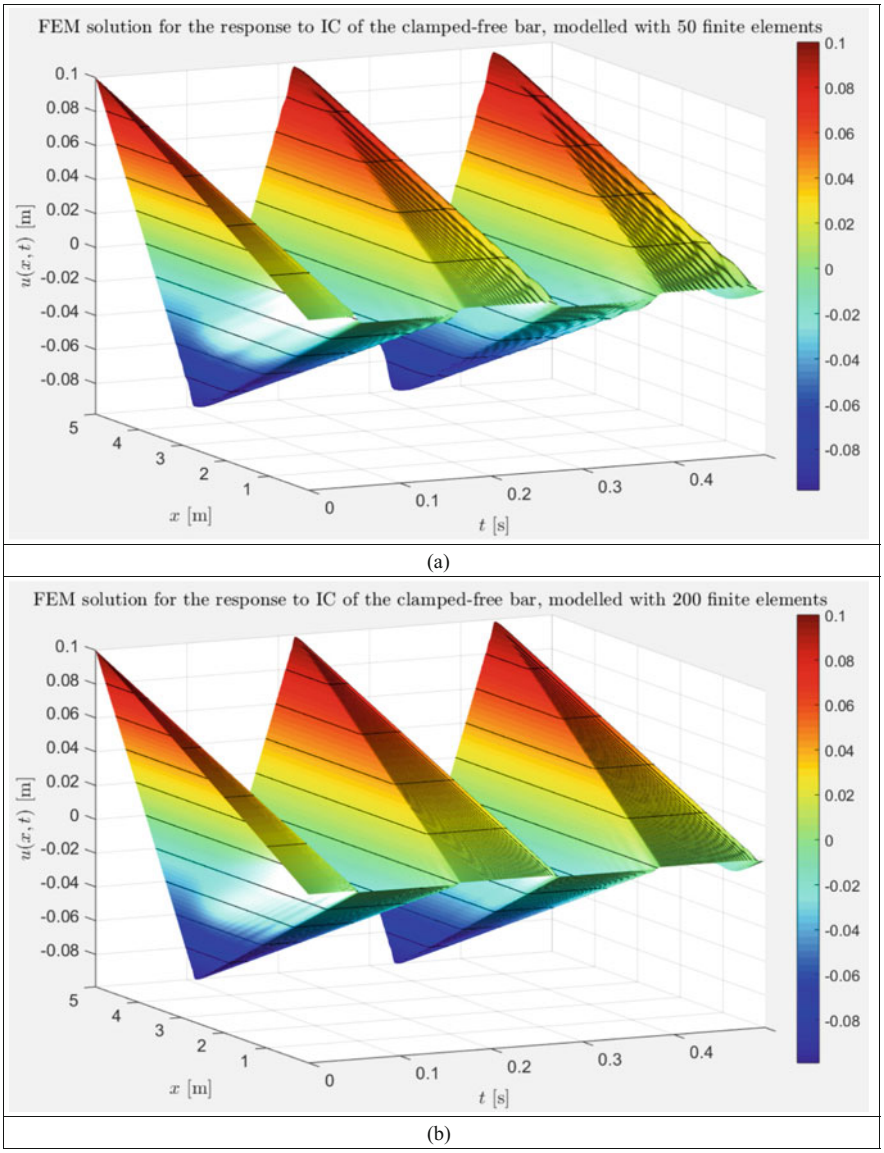


Fig. 6.91 FEM response of the clamped-free uniform rod: (**a**) FE model with 50 finite elements (**b**) FE model with 200 finite elements

```
%% 3D SURFACE PLOT FOR the CLAMPED-FREE FEM MODEL OF THE ROD with any NumFE
% Designed by Prof P.M.Trivailo (C) 2019
%--------------------------
clear; clc; close('all')
L=5; E=0.01*10^9; a=0.08; b=0.08; eL=0.1; e=eL/L; rho=1.2*10^3; m=rho*a*b*1;
A=a*b; EA=E*A; tmax=0.5; NumFE=50; h=L/NumFE;

% Elementary [ke] and [me] matices ---
ke=(EA/h)*[1 -1; -1 1]; me=(m*h/6)*[2 1; 1 2];

% --- SUPPLEMENTARY (FREE-FREE) SYSTEM ---- :
M=zeros(NumFE+1); K=zeros(NumFE+1);
for i=1:NumFE
    idx=[1 2]+(i-1); M(idx,idx)=M(idx,idx)+me; K(idx,idx)=K(idx,idx)+ke;
end

% --- MAIN SYSTEM: Constraining Left Boundary ---
M(:,1)=[]; M(1,:)=[]; K(:,1)=[]; K(1,:)=[];

invMxK=inv(M)*K;
% --- Formulating Boundary Conditions & EOM ---
x0=[eL*[1:NumFE]/NumFE, zeros(1,NumFE)]';
FEM_xdot_anon = @(t,x) [eye(NumFE)*x(NumFE+1:2*NumFE); -invMxK*x(1:NumFE)];
[tt,xx] = ode45(FEM_xdot_anon,[0 tmax],x0);

%
[TT,XX]=meshgrid(tt,L*[1:NumFE]/NumFE);
figure; hold on; grid on; rotate3d on;
sur=surf(TT,XX,xx(:,1:NumFE)'); colormap jet; lighting phong; shading interp;
contour3(TT,XX,xx(:,1:NumFE)',[-0.1:0.02:0.1],'k');

%--- hDecoratedh plotting ---
cb=colorbar; view([-25,14]); camlight left;
hh1=camlight; set(hh1,'Position',[0.09,0.04,0.03]);
xl=xlabel('$t$ [s]'); yl=ylabel('$x$ [m]'); zl=zlabel('$u(x,t)$ [m]');
tit=title(sprintf(['FEM solution for the response to IC of the ',...
    'clamped-free bar, modelled with %i finite elements'],NumFE));
set([xl,yl,zl,tit],'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[70 60 1160 680]);
```

Fig. 6.92 MATLAB complete script for simulation and representation of results for the response of the clamped-free rod in axial vibrations, modelled with arbitrary number of FEs

6.5.3 *Utilization of the FEM for Modelling of the Lateral Vibrations of Elastic Beams*

In the previous subsections, a number of continuous systems were discussed, such as strings in transverse vibrations and rods in axial vibrations. All these systems are governed by the *second-order partial differential equations* in space and time, and are analogous in nature. The corresponding boundary-value problem comprises two boundary conditions, one at each end.

By contrast, a beam in flexure is governed by *fourth-order partial differential equations* in space and the corresponding boundary-value problem requires two boundary conditions at each end.

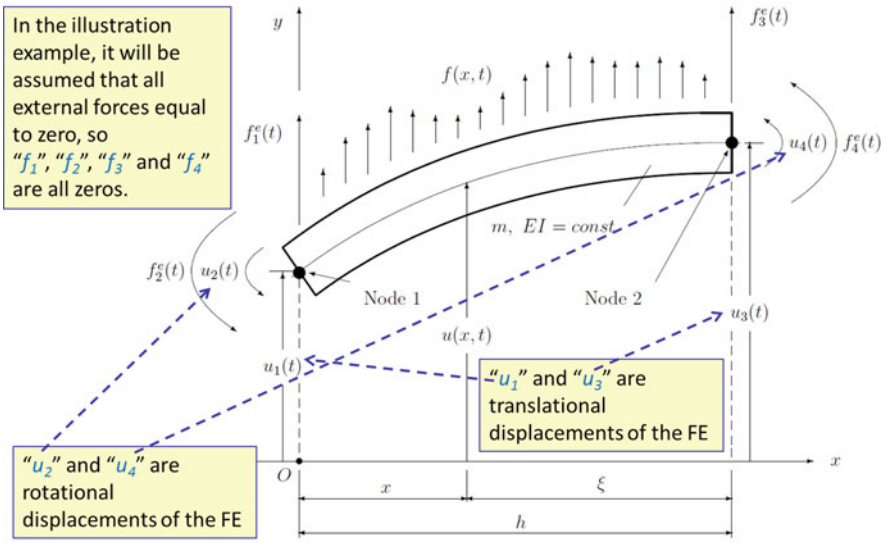


Fig. 6.93 Main notations for the "Euler–Bernoulli Beam" Finite Element

In this subsection, we derive this boundary-value problem for a selected example and obtain the response of this system to initial excitation using finite element method (FEM), as analytical solution for the general case would not be available.

The examples in this subsection are supplemented with detailed MATLAB scripts, which have, for certainty, particular numbers, including the number of the finite elements. However, all these numbers can be easily changed to correspond to the new requirements/conditions.

Let us assume that in the analysis of the beam, it is subdivided into smaller size finite elements. One of these elements is shown in Fig. 6.93, outlining main notation. It is known as "Euler-Bernoulli beam" finite element, and has two nodal translational displacements, u_1 and u_3, and two nodal angular displacements, u_2 and u_4, being four degrees-of freedom of the finite element.

It is possible to rigorously prove [17] that the following function can be used for interpolation of the lateral displacement at any internal point within the finite element:

$$u(x,t) = H_1(x)u_1(t) + H_2(x)u_2(t) + H_3(x)u_3(t) + H_4(x)u_4(t) \qquad (6.110)$$

where functions H_1, H_2, H_3 and H_4 are so-called "*shape functions*" (also known as Hermite interpolation functions or Hermite cubics):

$$\begin{aligned} H_1(x) &= 1 - 3\left(\tfrac{x}{h}\right)^2 + 2\left(\tfrac{x}{h}\right)^3 \\ H_2(x) &= h\left[\left(\tfrac{x}{h}\right) - 2\left(\tfrac{x}{h}\right)^2 + \left(\tfrac{x}{h}\right)^3\right] \\ H_3(x) &= 3\left(\tfrac{x}{h}\right)^2 - 2\left(\tfrac{x}{h}\right)^3 \\ H_4(x) &= h\left[-\left(\tfrac{x}{h}\right)^2 + \left(\tfrac{x}{h}\right)^3\right] \end{aligned} \tag{6.111}$$

Utilization of Eq. (6.111) could further lead to the establishment of the element stiffness and mass matrices:

$$\left[k^{\mathrm{e}}\right] = \frac{EI}{h^3}\begin{bmatrix} 12 & 6h & -12 & 6h \\ 6h & 4h^2 & -6h & 2h^2 \\ -12 & -6h & 12 & -6h \\ 6h & 2h^2 & -6h & 4h^2 \end{bmatrix}; \quad \left[m^{\mathrm{e}}\right] = \frac{mh}{420}\begin{bmatrix} 156 & 22h & 54 & -13h \\ 22h & 4h^2 & 13h & -3h^2 \\ 54 & 13h & 156 & -22h \\ -13h & -3h^2 & -22h & 4h^2 \end{bmatrix} \tag{6.112}$$

For the FEM modelling of the system with external excitations, the expressions for the nodal forces for a 2D beam finite element would be also needed:

$$\begin{aligned} f_1^{\mathrm{e}}(t) &= \int_0^h f(x,t)\, H_1(x)dx = \int_0^h f(x,t)\left[1 - 3\left(\tfrac{x}{h}\right)^2 + 2\left(\tfrac{x}{h}\right)^3\right]dx \\ f_2^{\mathrm{e}}(t) &= \int_0^h f(x,t)\, H_2(x)dx = \int_0^h f(x,t)\left[3\left(\tfrac{x}{h}\right)^2 - 2\left(\tfrac{x}{h}\right)^3\right]dx \\ f_3^{\mathrm{e}}(t) &= \int_0^h f(x,t)\, H_3(x)dx = \int_0^h f(x,t)\left[3\left(\tfrac{x}{h}\right)^2 - 2\left(\tfrac{x}{h}\right)^3\right]dx \\ f_4^{\mathrm{e}}(t) &= \int_0^h f(x,t)\, H_4(x)dx = \int_0^h f(x,t)\, h\left[-\left(\tfrac{x}{h}\right)^2 + \left(\tfrac{x}{h}\right)^3\right]dx \end{aligned} \tag{6.113}$$

however, in the illustration examples here, we will consider excitations of the beams via initial conditions and not external forces; therefore, these nodal forces will be assumed to be zeros.

Notations and relevant units, used in the Eqs. (6.112) and (6.113) are explained below:

- "m" [kg/m]—mass per unit length of the beam;
- "$[m^{\mathrm{e}}]$" elementary mass matrix;
- "E" [Pa]—Young's Modulus;
- "I" [m^4]—cross-sectional area moment of inertia (also known as Second moment of area);
- "EI" [N/m]—bending stiffness of the beam;
- "$[k^{\mathrm{e}}]$" elementary stiffness matrix;
- "h" [m]—length of the finite element.

Individual element matrices can be calculated, using Eq. (6.112) and their elements should be added to the appropriate cells in the global matrices during the assembling process.

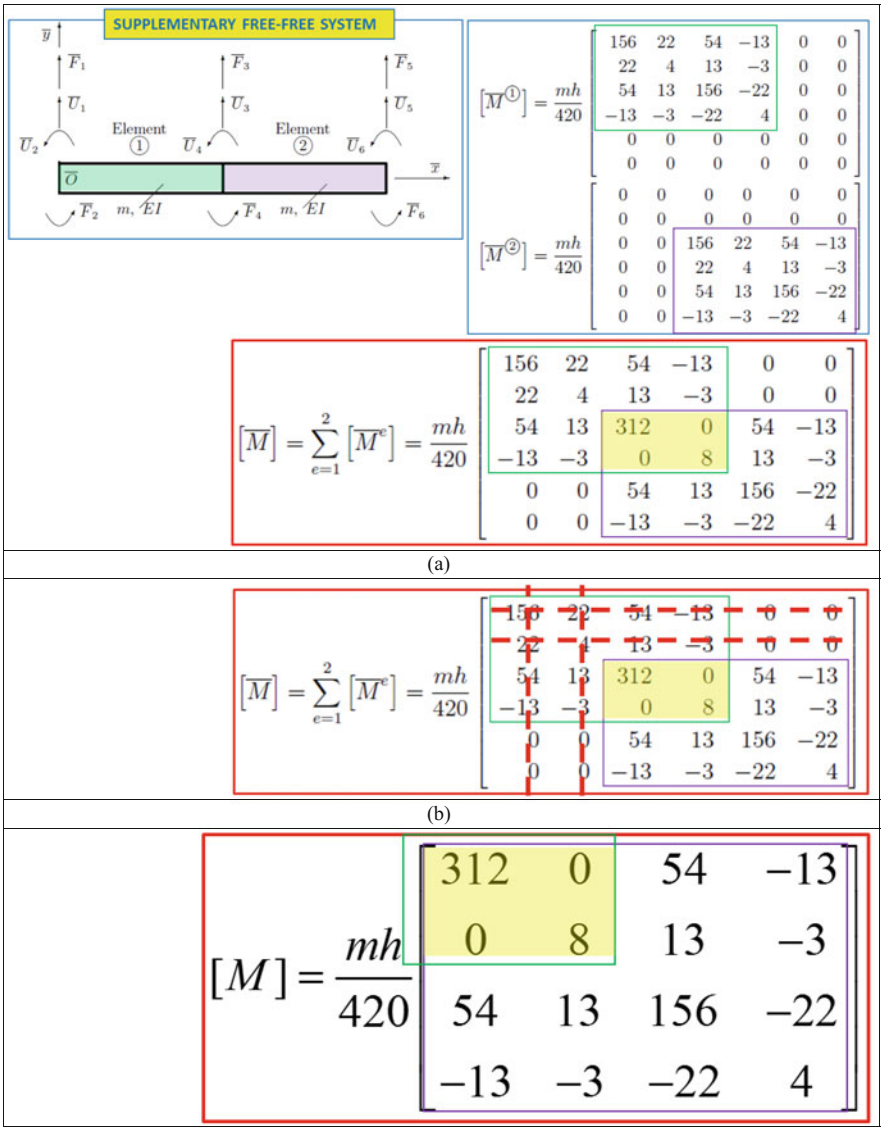


Fig. 6.94 Process of assembling of the global mass matrix: (**a**) assembly for the supplementary system; (**b**) truncation of the matrix; (**c**) global mass matrix for the original system (note: for convenience of the explanation, *h* is assumed to be equal to 1)

This process is explained schematically in detail in Fig. 6.94 for simple example, involving only *two finite elements*.

The feature of the example is in the constrained (clamped) end. For these cases, in the assembling process, we initially ignore original boundary conditions

of the system, and essentially assemble global matrices for the free-free (called supplementary) system. For more visual explanation, let us assume that the length of the finite element $h = 1$ m, in this case element matrices have no "h" parameters in square brackets and are given by numbers only. Of course, in the general case, use of full general expressions for the elementary matrices would be needed.

It is shown in Fig. 6.94, that during the assembly, a node, adjacent to two elements, contributes to the degrees of freedom, associated with this border node. In the illustrated example, middle node contributes to the third and fourth degrees-of-freedom, which results in the "overlapped" area in the global mass matrix, marked with yellow colour in Fig. 6.94, where resultant values are cumulative numbers, containing contributions from the element #1 and element #2.

Requirement to constrain translational and rotational DEFs at the left end of the beam results in making first two DOFs of the supplementary system irrelevant, which allows us to eliminate two rows and two columns in the global matrix of the supplementary system (Fig. 6.94b). These deleted rows and columns correspond to one translational and one rotational constrained DOFs. It may be of interest to mention, that "deletion" of rows and columns (which are also called sometimes as "matrix condensation" or "truncation") can be easily programmed in MATLAB. For example, in order to delete the first column in the matrix $[M]$, the following command can be used:

$$M\,(:, 1) = [] \tag{6.114}$$

and for deletion of the first row in the matrix $[M]$, the following command can be employed:

$$M\,(1, :) = [] \tag{6.115}$$

If the number of finite elements is more than two, the assembly process is very much the same: regardless of the initial conditions, it is advisable to introduce a supplementary free-free beam and assemble its global matrices, which could be further adjusted to the constrained boundary conditions by truncating rows and columns, associated with the constrained DOFs.

For example, assembly process for the clamped-free stepped beam with five finite elements starts with the assembly of the global matrix for the supplementary system and is schematically shown in Fig. 6.95. The example in Fig. 6.95 emphasizes that the individual properties of the finite elements can differ, which makes the FEM a genuinely unique simulation tool, enabling treatment of the systems, for which analytical solutions would no longer be available.

Let us now consider the *main study case system* in this subsection, shown in Fig. 6.96. It involves a simple uniform beam of length L, constant mass per unit length m and constant bending stiffness EI. The beam is initially held at rest and then is dropped from the horizontal position from the height $y = H$. When it reaches the

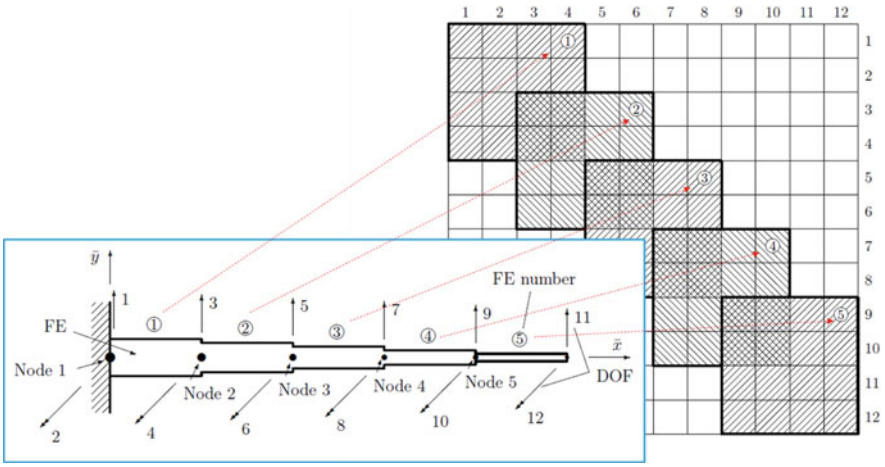


Fig. 6.95 Schematic procedure of assembling of the *global* stiffness matrix for the supplementary (free-free) system, related to the clamped cantilever beam study case example

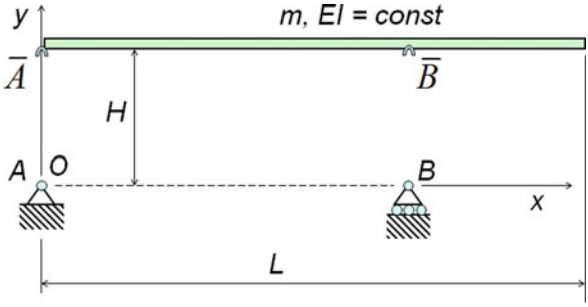


Fig. 6.96 A uniform beam (shown at the approaching stage ($t < 0$)

level $y = 0$, it collides with the obstacles A and B at $x = 0$ and $x = a$, which both automatically lock as pins on the beam's matching points A and B, arresting the motion of the beam.

Assume the instant of the collision as initial time $t = 0$ and consider the idealized case where there are no energy losses due to friction, impact, etc., and where no rebound occurs at the supports.

As an illustration example, we will program in MATLAB the Finite Element Model of the beam (modelled with 10 equal length FEs) and determine the first natural frequency of the beam in rad/s.

We assume the following parameters for the simulation case of the beam:

- Width and height of the beam: $b = 60$ mm and $h = 10$ mm; length of the beam: $L = 2$ m
- Young Modulus of the beam: $E = 200$ GPa
- Density of the material: $\rho = 7500$ kg/m^3 (stainless steel)
- Dropping height: $H = 2$ m
- Location of the right automatic hinge: $a = 0.6 \times L = 0.6 \times 2$ m $= 1.2$ m

The recommended process for simulation of the system and solving specified tasks may include the following critical steps.

- *Step-1:* Modelling of the system with adequate number of finite elements with their boundaries compatible with the requirements of the study case.

  Important comment is related to the subdivision of the beam into the finite elements. It must comply with the requirement of the case to "arrest" points A and B. This could only be achieved if after subdivision of the beam into the finite elements, points A and B coincide with the nodes of the FE model. If the beam is divided into 10 equal length finite elements (with $h = L/10 = 0.2$ m), these requirements would be satisfied: point A would be the left end of the first finite element #1, and point B would be a connecting point for the sixth (#6) and seventh (#7) finite elements.
- *Step-2:* Assembly of the *global mass* and *global stiffness* matrices of the system and formulation of the dynamics equations of motion, which would be subject to the *boundary conditions*. For the original systems with constrained DOF, introduction of the supplementary free-free systems may be required.

  In the considered example, we will need to "start" clocks (i.e. set $t = 0$) at the moment, when beam touches points A and B and rotational hinges are activated, so beam is no longer free-free. In order to implement the *boundary conditions* of the beam, conforming with the requirement that from the moment $t = 0$, the beam is constrained (hinged) at points A and B, we need to assemble global matrices for the supplementary system first. Then, corresponding "reduction" of both global mass and stiffness matrices can be applied. Alternatively, large values could be added to the cells in the global mass matrix, corresponding to the constrained DOFs. Once again, it should be stressed out, that when the DOFs in the system are constrained, we start simulation process, considering first the supplementary system, which is completely free; then, we adjust supplementary solution matrices to the *main* (i.e. constrained) case.
- *Step-3:* Reformulation of the equations of motion, using system's states. This is necessary to be able to access the ordinary differential equations solvers (like "`ode`" in MATLAB), which are usually available for the systems, described with the first order differential equations. For the vibrating systems, described with the second order differential equations, this reformulation typically leads to twice increased number of equations. For linear systems, the resultant equations have the following form: $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ and for non-linear systems have the form: $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$, with various solvers available for both forms.
- *Step-4:* Formulation of the *initial conditions* for the task.

  In the study case for this purpose, we need to determine the velocity of the beam after it flies 2 m and reaches A–B level. For this purpose, the Law of Conservation of Energy can be used. Ignoring the air resistance, and assuming that potential energy of the beam (initially lifted at the 2 m height) is completely converted into the kinetic energy of the beam, when it touches A and B, we can write:

$$mgH = \frac{1}{2}mv^2 \quad \Rightarrow \quad v = \sqrt{2gH} = \sqrt{2 \times 9.81 \times 2} = 6.2642 \ \ (\text{m/s}) \tag{6.116}$$

For this subsection example, the supplementary system (beam) with 10 FEs would have 11 nodes and 22 DOFs, thus, 44 states, representing:

- lateral displacements (states 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21);
- angular displacements (states 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22);
- nodal velocities (states 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43);
- nodal angular velocities (states 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44).

With such defined states of the supplementary system, we can formulate initial conditions, specifying all initial nodal velocities (states 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43) equal to negative v_0 and all other states equal to zero.

Also, in "transition" from supplementary system global matrices, 13th and first rows and columns should be removed to model hinged attachment of the beam at points *A* and *B*. Also, exclusion of the states #13 and #1 would require elimination of the corresponding displacement and velocity states from the vector of the initial conditions, i.e. elements in the $\mathbf{x}_0$ vector number #35, #14, #13 and #1. Note, that it is recommended to remove rows and columns in the vector of initial conditions, global mass and global stiffness matrices, processing deletions in descending (i.e. not ascending) order. So, for programming convenience, in the global mass matrix, 13th rows and columns should be removed first and only after this step is completed, the first rows and columns can be removed. Similarly, in the "condensation" of the vector of the initial conditions, we delete first 35th cell, then 14th cell, then 13th cell, then first cell. Deletion from the tail does not affect numbering of the preceding rows and columns.

- *Step-5:* Programming of the steps above, numerical solution of the task and graphical interpretation of results.

MATLAB script, implementing these steps is presented in Fig. 6.97, and the results of the simulation in the form of a 3D surface plot are presented in Fig. 6.98.

One of the features of this example is in utilization of the MATLAB standard continuous state-space function "`ss`", (available in SIMULINK with its "state-states" block for linear systems). This further simplifies simulation process, as it does not require programming of the "x_dot" anonymous or stand-alone functions. Also, note, that "ss" function was used in combination with "initial" function, bringing even more conveniences to the user. For more functions, facilitating programming process, please, refer to the massive resources on the MATHWORKS web site, Internet or specialized texts, like [8].

```
%% VIBRATION OF THE BEAM (NumFE=10)
% Programmed by Prof P.M.Trivailo (C) 2019

bb=60*10^(-3); hh=10*10^(-3); m=7500*bb*hh; E=200*10^9; I=bb*hh^3/12; EI=E*I;
c=sqrt(EI/m); L=2; dropH=2; v0=sqrt(2*9.81*dropH); alphaC=0; betaC=0;

% Select the number of the FINITE ELEMENTS below:
NN=10; h=L/NN; dof=(NN+1)*2;

% Element matrices:
mm=(m*h/420)*[156   22*h    54   -13*h; ...
               22*h  4*h^2  13*h  -3*h^2; ...
               54   13*h   156   -22*h; ...
             -13*h -3*h^2 -22*h   4*h^2];

kk=  (EI/h^3)*[12    6*h   -12     6*h; ...
                6*h  4*h^2  -6*h   2*h^2; ...
              -12   -6*h    12    -6*h;...
                6*h  2*h^2  -6*h   4*h^2];

K=zeros(dof,dof); M=zeros(dof,dof);
for ii=1:NN
    idx=[1:4]+(ii-1)*2; K(idx,idx)=K(idx,idx)+kk;  M(idx,idx)=M(idx,idx)+mm;
end
x0=zeros(2*dof,1);      % for NN=10; dof=22;
x0(dof+1:2:2*dof)=-v0;  %<-- x0(23 25 27 29 31 33 35 37 39 41 43) = v0;

% FOR THE UNCONSTRAINED FREE-FREE BEAM:
%    1   2   3   4   5   6   7   8   9   10      <--- FEs
%  1===2===3===4===5===6===7===8===9===10===11   <---NODES
%  ^                   ^
% PIN-1                PIN-2
% DOFs:
%  1   3   5   7   9  11  13  15  17  19  21    <- displacements
%  2   4   6   8  10  12  14  16  18  20  22    <--angles
% 23  25  27  29  31  33  35  37  39  41  43    <--velocities
% 24  26  28  30  32  34  36  38  40  42  44    <--angular velocities

% Take into account the boundary conditions.
% Remove the RIGHT HINGE DOF FIRST!!!!!!!!!!!!!
M(13,:)=[]; K(13,:)=[];M(:,13)=[]; K(:,13)=[];

% Remove the LEFT HINGE DOF THEN!!!!!!!!!!!!
M(1,:)=[]; K(1,:)=[];M(:,1)=[]; K(:,1)=[];

% PROCESS x0 FROM THE END TO THE BEGINNING:
x0(dof+13)=[];   % i.e. for dof=22;  x0(35)=[];    <--NO
x0(dof+1)=[];    % i.e. for dof=22;  x0(23)=[];
x0(13)=[];x0(1)=[];

% FOR THE CONSTRAINED PIN(left end)-PIN(0.6L)-FREE(right end) BEAM:
%    1   2   3   4   5   6   7   8   9   10      <--- FEs
%  1===2===3===4===5===6===7===8===9===10===11   <---NODES
%  ^                   ^
% PIN-1                PIN-2
%
% !!! RENUMBERED !!! DOFs:
%      2   4   6   8  10      13  15  17  19    <- displacements
%  1   3   5   7   9  11  12  14  16  18  20    <--angles
%     22  24  26  28  30      33  35  37  39    <--velocities: ONLY THESE=v0
% 21  23  25  27  29  31  32  34  36  38  40    <--angular velocities

C=alphaC*M + betaC*K;
dof=size(M,1);  % dof IS RECALCULATED: now it is for the CONSTRAINED SYSTEM

[U,D]=eig(K,M); w_all=diag(sqrt(abs(D)));

disp(sprintf('\n========== NUMBER OF FE:  %i ======',NN))
disp(sprintf('\n========== NUMBER OF UNCONSTRAINED DOF: %i ======',dof+4))
disp(sprintf('\n========== NUMBER OF CONSTRAINED DOF: %i ======',dof))
disp(sprintf('\n========== FIRST TWO FREQUENCIES FOR THE CONSTRAINED SYSTEM: ======'))
disp(sprintf('w_1=%8.4f [rad/s];  w_2=%8.4f [rad/s]',w_all(1),w_all(2)))

% RESPONSE:
A = [zeros(dof, dof) eye(dof, dof) ; -inv(M)*K -inv(M)*C];
B = [zeros(dof, dof) ; inv(M)]; C = eye(2*dof, 2*dof); D = zeros(2*dof, dof);
SYS=ss(A,B,C,D); [Y,T,X] = initial(SYS,x0); figure; grid on; rotate3d;
translations=[zeros(size(T)), X(:,2), X(:,4), X(:,6), X(:,8), X(:,10),...
              zeros(size(T)), X(:,13), X(:,15), X(:,17), X(:,19)];

% Plot displacements as 3D surface:
[XX,TT]=meshgrid(linspace(0,L,NN+1),T); surf(XX,TT,translations);
% Plot hinged points:
line('XData',zeros(size(TT(:,1))),'YData',TT(:,1),'ZData',0*TT(:,1),'LineWidth',4,'Color','r'); %A
txl=text('String','$A$','Position',[0 0 0],'FontSize',24,'Color','w','FontWeight','bold');
line('XData',1.2*ones(size(TT(:,1))),'YData',TT(:,1),'ZData',0*TT(:,1),'LineWidth',4,'Color','r'); %B
tx2=text('String','$B$','Position',[1.2 0 0],'FontSize',24,'Color','w','FontWeight','bold');
xl=xlabel('$x$ [m]'); yl=ylabel('$t$ [s]'); zl=zlabel('$y$ [m]');
set([xl,yl,zl,txl,tx2],'Interpreter','LaTeX');
camlight('left'); lighting phong; shading interp; colormap jet;
axis tight; view([-47 28]); set(gca,'FontSize',18);
set(gcf,'Position',[100 120 1400 600]);
```

Fig. 6.97 MATLAB script, enabling simulation of the response of the elastic beam, excited via initial conditions

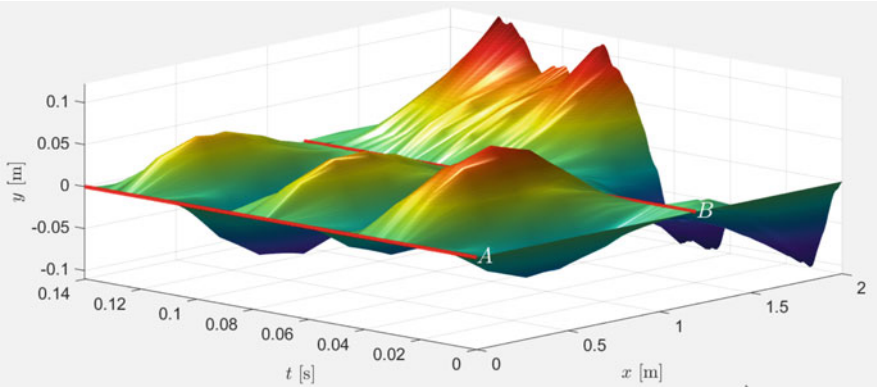


Fig. 6.98 Response of the elastic beam, excited via initial conditions

6.6 Conclusion

This chapter presented guidelines and recommendations for modelling of various engineering systems. These guidelines are unique in the sense that the basic concepts are illustrated with a selection of provocatively simple study cases, all examples are massively illustrated with detailed diagrams and working annotated MATLAB scripts, enabling simulation, static representation of results and even animations, using advanced computer graphics. The high efficiency of several key techniques and methods were presented, including state-space formulation and finite element method. The state-space formulation enables us to access powerful solvers of the ordinary differential equations. And the FEM allows to reduce huge class of problems, described with partial differential equations to the ordinary differential equations. The main intent of this work was to show that with use of modern simulation tools and computer environments, modelling and simulation process of complex systems is amazingly accessible. A uniform and universal recommended process is equally applicable to both linear and non-linear systems and allows comparison of linear and non-linear models and verification of the adopted assumptions. With a wide spectrum of the presented examples, related to the mechanical, aerospace, civil, electrical, environmental and other engineering areas, it is believed that presented modelling and simulation guidelines will be useful for a very wide audience, including engineers, scientists, students and enthusiasts of science and technology.

References

1. McLachlan, N., Nigjeh, B. K., & Trivailo, P. M. (2002). Application of modal analysis to musical bell design. In *Proceedings of the Australian Acoustical Society (AAS) Annual Conference "Innovation in Acoustics and Vibration"*, Adelaide University, 13-15 November, 2002, 10 pp.
2. Guinness: World Records 2009 (Guinness Book of Records), Guinness World Records Ltd., 286 pp.
3. Kruijff, M. (2011). *Tethers in space—A propellantless propulsion in-orbit demonstration*. Oisterwijk: BOXPress. ISBN 978-90-8891-282-5, 433 pp.
4. Williams, P., Hyslop, A. M., Stelzer, M., & Kruijff, M. (2009). YES2 optimal trajectories in presence of eccentricity and aerodynamic drag. *Acta Astronautica, 64*(7–8), 745–769. https://doi.org/10.1016/j.actaastro.2008.11.007.
5. Fujii, H. A., Watanabe, T., Kojima, H., Oyama, K.-I., Kusagaya, T., Yamagiwa, Y., et al. (2009). Sounding rocket experiment of bare electrodynamic tether system. *Acta Astronautica, 64*(2–3), 313–324.
6. Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computer Applied Mathematics, 6*, 19–26.
7. Gupta, S. K., & Kumar A. (2001), Krishna's engineering physics. Volume II (thermal physics). Krishna Prakash Media. ISBN 81 87224 20 7, p. 2.56.
8. Chapman, S. J. (2013). *MATLAB programming with applications for engineers*. Stamford: Cengage Learning. 569 pp. ISBN-13: 978-0-495-66807-7.
9. Jazar, R. N. (2011). *Advanced dynamics. Rigid body, multibody, and aerospace applications*. New York: Wiley. 1324 pp.
10. Bedford, A., & Fowler, W. (1995). *Engineering mechanics. Dynamics*. Boston: Addison-Wesley. 551 pp.
11. Murray, C. D., & Dermott, S. F. (1999). *Solar system dynamics*. Cambridge: Cambridge University Press, XIII+592 pp. ISBN 0-521-57295-9, ISBN 0-521-57597-4.
12. Wikipedia. (2019). Finite element method. Retrieved February 12, 2019, https://en.wikipedia.org/wiki/Finite_element_method.
13. Cook, R. D. (1995). *Finite element modelling for stress analysis*. New York: Wiley. ISBN 0-471-10774-3. 320 pp.
14. Zienkiewicz, O. C., Taylor, R. L., & Zhu, J. Z. (2005). *Finite element method: Its basis and fundamentals* (6th ed.). Oxford: Butterworth-Heinemann. ISBN 0750663200. xiv, 733 pp.
15. Gallagher, R. H. (1974). *Finite element analysis: Fundamentals* (Prentice-Hall civil engineering and engineering mechanics series). Upper Saddle River: Prentice-Hall. 420 pp.
16. Gray, R. 3D earth example. MATHWORKS free file exchange. Retrieved February 10, 2019, from https://au.mathworks.com/matlabcentral/fileexchange/13823-3d-earth-example.
17. Trivailo, P. M. (2015). *Vibrations: Theory & aerospace applications*, Vols. 1&2 (The textbook for senior undergraduate and graduate aerospace students). Melbourne: RMIT Publisher (2008). 247pp +348pp=595 pp., 355 ill, 4 software programs, Bibl.-86; Index - 1580 entries.
18. Rao, S. S. (1995). *Mechanical vibrations* (3rd ed.). Boston: Addison-Wesley. 912 pp.

Chapter 7
On the Description of Large Deformation in Curvilinear Coordinate Systems: Application to Thick-Walled Cylinders

Monir Takla

7.1 Introduction

Increasing requirements for the safety of extremely loaded cylindrical components of structures resulted in considering the inclusion of elastic–plastic deformation in the design calculations. Accordingly, the plastic or elastic–plastic deformation of structures has gained special significance.

Extremely pressurized structures used in traditional and nuclear power generation, oil, gas, and other chemical industries have to endure the design loads reliably, keeping in mind the potentially catastrophic consequences of overloading such structures. As a result, it became necessary to include large plastic or elastic–plastic deformation in the failure analysis of such structures.

This chapter presents an investigation of the elastic–plastic deformation of thick-walled cylinders subject to radial and axial loading. The analysis adopts the von Mises yield criterion and its associated flow rule.

7.1.1 Background

Accurate description of the distribution of large strains and associated stresses leads to improved reliability of the calculated radial and axial loads. It is also vital to calculating failure limit loads needed for the safety design of such structures. Therefore, the correctness of failure analysis is based on the validity and accuracy of

M. Takla (✉)
School of Engineering, RMIT University, Melbourne, VIC, Australia
e-mail: monir.takla@rmit.edu.au

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_7

the description of stresses, strains, and strain rates, which describe the development of the regular stable unique deformations in a current configuration.

Apart from the solution offered earlier by the current author [1, 2], available literature does not provide an accurate prediction for the distribution of stresses and strains in thick-walled pressure vessels under simultaneous radial and axial loading when large elastic–plastic deformation occurs after the elastic limit is exceeded. The reliable solution addresses the nonhomogeneous stress, strain, and velocity fields in thick-walled cylindrical pressure vessels. The cited works, however, do not provide enough details of the mathematical basis describing the large deformation involved in developing the solution. The mathematical description of the large deformation needs further explanation.

7.1.2 Historical Overview

Lame [3] provided a first closed solution for small elastic deformations of a radially loaded closed cylinder under internal and external pressure. Small elastic–plastic deformation of thick-walled cylinders was first investigated by Turner [4], adopting the Tresca yield criterion. Other researchers followed the same approach to provide solutions to cases with various boundary conditions, materials, etc. Their works, however, were limited to the theory of small deformation.

The first solution that used the von Mises yield criterion was provided by Belayev and Sinitskij [5] and Sokolovskij [6]. Some solutions for individual cases were offered by MacGregor et al. [7] and Hodge and White [8] considering small deformations. The principles of the mechanics of continuous media have been long established in the literature [9–14, 25].

Large deformation of a cylinder, constrained in the axial direction, was described by Celep [15], who considered large elastic–plastic deformation and linear material hardening. He obtained a numerical solution by applying the Maximum Distortion Energy yield criterion. Fischer [16] studied plane strain deformation of thick cylinders applying the Maximum Shear yield criterion and considering linear material hardening. Oeynhausen [17] extended the study to include an axial force with the internal pressure, considering nonlinear hardening.

Imaninejad and Subhash [18] studied thick cylinders loaded by proportional radial and axial loading, assuming constant strain ratios throughout the cylinder wall. This assumption violates the basic principle that the axial strain is independent of the radial location, in contrast with the other two strain components. Accordingly, it resulted in an inaccurate solution including singularity.

Elastic–plastic thick-walled cylinders with nonlinear hardening, simultaneously loaded radially and axially, were investigated in Takla [1, 2] adopting the Maximum Distortion Energy yield criterion. However, the definitions of large deformations were only briefly described.

A theoretical solution is generally based on substituting a constitutive law into the equilibrium equations, making sure that the geometric compatibility is maintained.

In the presented analysis, a brief description is provided for the large elastic–plastic deformation. Material-independent fundamentals are first presented. A constitutive law based on applying the von Mises yield criterion in association with the normality rule is discussed. Large deformation is then discussed in detail for a thick-walled cylinder, simultaneously loaded by hydrostatic pressure and axial force, considering nonlinear isotropic hardening. The stress and strain distributions, calculated for prescribed states of large deformation, are used to calculate the applied loads.

7.2 Material-Independent Fundamentals

Large deformation in curvilinear coordinates is described using both spatial and material coordinate systems. Deformation is assumed to remain independent of the axial position. The developed geometric expressions provide the relationships between the geometric variables in the initial configuration and those in the deformed configuration.

7.2.1 Description of Motion

The referential description of the position $\underset{\sim}{x}$ of a material point in three-dimensional space at time t can be expressed by

$$\underset{\sim}{x} = \underset{\sim}{\chi}_{\kappa}\left(\underset{\sim}{X}, t\right), \tag{7.1}$$

where $\underset{\sim}{X}$ is the position of a material point in an arbitrary reference configuration. Although there is a difference between the material and the referential description of motion, both of them are used in this work with the same meaning, i.e., a material point is identified with its position in its reference placement (Lagrange formulation). In order to describe the current state of deformation, the configuration in the initial state is usually used as the reference configuration.

7.2.2 Coordinates and Base Vectors

The motion of a material point can be described, either by the change of its coordinates in a space-fixed coordinate system or by the change of the base vectors in a convective coordinate system in which the coordinates move along with the material points. In the case of axisymmetric, axially homogeneous deformation, the current deformation comprises no rotation. Therefore, the first method will be

adopted. Accordingly, in a curvilinear coordinate system, it is valid for the space formulation:

$$\begin{aligned} \mathrm{d}\underset{\sim}{x} &= \mathrm{d}z_i \underset{\sim}{b}_i \quad \text{(cartesian coordinates)} \\ &= \mathrm{d}x^j \underset{\sim}{g}_j = \mathrm{d}x_k \underset{\sim}{g}^k \quad \text{(general curvilinear coordinates)} \end{aligned}, \tag{7.2}$$

and for the material formulation:

$$\begin{aligned} \mathrm{d}\underset{\sim}{X} &= \mathrm{d}Z_\alpha \underset{\sim}{B}_\alpha \quad \text{(cartesian coordinates)} \\ &= \mathrm{d}X^\alpha \underset{\sim}{G}_\alpha = \mathrm{d}X_\alpha \underset{\sim}{G}^\alpha \quad \text{(general curvilinear coordinates)} \end{aligned}. \tag{7.3}$$

7.2.3 Description of Deformation

The deformation of a body can be represented by the deformation gradient:

$$\mathrm{d}\underset{\sim}{x} = \underset{\sim}{F}\ \mathrm{d}\underset{\sim}{X}, \tag{7.4}$$

where $\underset{\sim}{F}$ is a dual-field tensor, which can be expressed by

$$\begin{aligned} \underset{\sim}{F} &= \frac{\partial \underset{\sim}{\chi}_\kappa\left(\underset{\sim}{X},t\right)}{\partial \underset{\sim}{X}} = \mathrm{Grad}\ \underset{\sim}{\chi}_\kappa\left(\underset{\sim}{X},t\right) \\ \underset{\sim}{F} &= \frac{\partial \underset{\sim}{x}}{\partial \underset{\sim}{X}} = \frac{\partial x^i}{\partial X^\alpha}\underset{\sim}{g}_i \cdot \underset{\sim}{G}^\alpha = F^i_{.\alpha}\underset{\sim}{g}_i \cdot \underset{\sim}{G}^\alpha \end{aligned}. \tag{7.5}$$

For the transformation $\underset{\sim}{\chi}_\kappa$ to be unique and consequently invertible, it is necessary for the Jacobian determinant of the deformation gradient "J" to have a nonzero value, i.e.,

$$J = \frac{\partial \underset{\sim}{\chi}_\kappa}{\partial \underset{\sim}{X}} = \det\ \underset{\sim}{F} \neq 0, \tag{7.6}$$

$$\underset{\sim}{F}^{-1} = \frac{\partial \underset{\sim}{X}}{\partial \underset{\sim}{x}} = \frac{\partial X^\alpha}{\partial x^i}\underset{\sim}{G}^\alpha \cdot \underset{\sim}{g}^i. \tag{7.7}$$

Accordingly, the inverse of the deformation gradient exists, and it can be stated that

$$\mathrm{d}\underset{\sim}{X} = \underset{\sim}{F}^{-1}\ \mathrm{d}\underset{\sim}{x}, \tag{7.8}$$

the Jacobian determinant also represents a measure of the change in volume because

$$J = \frac{\rho_o}{\rho} = \frac{\sqrt{g}}{\sqrt{G}} \left| F^r_{.\alpha} \right|, \tag{7.9}$$

where $\sqrt{G}$ and $\sqrt{g}$ represent the scalar triple products of the base vectors G_i and g_i. The transposed tensors of $\underset{\sim}{F}$ and $\underset{\sim}{F}^{-1}$ are expressed, respectively, as

$$\underset{\sim}{F}^{\mathrm{T}} = \frac{\partial x^i}{\partial X^\alpha} \underset{\sim}{G}^\alpha \cdot \underset{\sim}{g}_i = F^i_{.\alpha} \underset{\sim}{G}^\alpha \cdot \underset{\sim}{g}_i, \tag{7.10}$$

$$\underset{\sim}{F}^{-\mathrm{T}} = \frac{\partial X^\alpha}{\partial x^i} \underset{\sim}{g}^i \cdot \underset{\sim}{G}_\alpha. \tag{7.11}$$

The metric coefficients can be written as

$$\begin{aligned} \underset{\sim}{g}_{ik} &= \underset{\sim}{g}_i \cdot \underset{\sim}{g}_k, \quad \underset{\sim}{g}^{ik} = \underset{\sim}{g}^i \cdot \underset{\sim}{g}^k \\ \underset{\sim}{G}_{\alpha\beta} &= \underset{\sim}{G}_\alpha \cdot \underset{\sim}{G}_\beta, \quad \underset{\sim}{G}^{\alpha\beta} = \underset{\sim}{G}^\alpha \cdot \underset{\sim}{G}^\beta \cdot \end{aligned} \tag{7.12}$$

7.2.4 The Strain Tensor

The symmetric tensor $\underset{\sim}{F}^{-\mathrm{T}} \underset{\sim}{F}$ is positive definite and can be represented by the square of a tensor "$\underset{\sim}{U}$," which is also symmetric and positive definite, defined by

$$\underset{\sim}{U}\underset{\sim}{U} = \underset{\sim}{F}^{\mathrm{T}} \underset{\sim}{F}. \tag{7.13}$$

Defining, through an associative law, the tensor

$$\underset{\sim}{R} = \left(\underset{\sim}{F}^{\mathrm{T}} \right)^{-1} \underset{\sim}{U} \tag{7.14}$$

leads directly to

$$\underset{\sim}{F} = \underset{\sim}{R}\underset{\sim}{U}. \tag{7.15}$$

It is easy to prove that $\underset{\sim}{R}$ is actually an orthogonal tensor. It is also called the "Rotation Tensor." Similar to (7.13), it is valid that

$$\underset{\sim}{V}\underset{\sim}{V} = \underset{\sim}{F}\underset{\sim}{F}^{\mathrm{T}}. \tag{7.16}$$

Considering (7.15) together with the orthogonality of $\underset{\sim}{R}$ leads to the expression

$$\underset{\sim}{V} = \underset{\sim}{R}\underset{\sim}{U}\underset{\sim}{R}^{\mathrm{T}} = \underset{\sim}{F}\underset{\sim}{R}^{\mathrm{T}} = \underset{\sim}{F}\underset{\sim}{R}^{-1}, \tag{7.17}$$

which leads directly to the conclusion

$$\underset{\sim}{F} = \underset{\sim}{V}\underset{\sim}{R}. \tag{7.18}$$

Summarizing (7.15) and (7.18) results in

$$\underset{\sim}{F} = \underset{\sim}{R}\underset{\sim}{U} = \underset{\sim}{V}\underset{\sim}{R}. \tag{7.19}$$

The symmetric tensors $\underset{\sim}{U}$ and $\underset{\sim}{V}$ are designated as right-stretch tensor and left-stretch tensor.

For every symmetric tensor $\underset{\sim}{U}$ exist (at least) three orthogonal vectors; $\underset{\sim}{P}_1$, $\underset{\sim}{P}_2$, and $\underset{\sim}{P}_3$, which maintain their directions during the transformation $\underset{\sim}{U}$

$$\underset{\sim}{U}\underset{\sim}{P}_i = u_{(i)}\underset{\sim}{P}_i. \tag{7.20}$$

They allow the spectral resolution of $\underset{\sim}{U}$ in the form

$$\underset{\sim}{U} = \sum u_{(i)}\underset{\sim}{P}_i \cdot \underset{\sim}{P}_i, \tag{7.21}$$

$u_{(i)}$ and $\underset{\sim}{P}_i$ are the principal values and directions of$\underset{\sim}{U}$. Following from (7.17) and (7.21) that

$$\underset{\sim}{V} = \underset{\sim}{R}\underset{\sim}{U}\underset{\sim}{R}^{\mathrm{T}} = \sum u_{(i)}\underset{\sim}{p}_i \cdot \underset{\sim}{p}_i \tag{7.22}$$

with

$$\underset{\sim}{p}_i = \underset{\sim}{R}\underset{\sim}{P}_i. \tag{7.23}$$

The tensor "$\underset{\sim}{V}$" has generally the same principal values as the tensor "$\underset{\sim}{U}$," but different principal directions $\underset{\sim}{p}_i$ ($\underset{\sim}{p}_i = \underset{\sim}{R}\underset{\sim}{P}_i$), which result from rotating the principal directions of $\underset{\sim}{U}$ by $\underset{\sim}{R}$. Because both of the stretch tensors $\underset{\sim}{U}$ and $\underset{\sim}{V}$ are described entirely through the principal values and directions, the eigenvalues $\varepsilon_{(i)}$ of the strain tensor can be represented as functions of $u_{(i)}$

$$\varepsilon_{(i)} = f\left(u_{(i)}\right). \tag{7.24}$$

The function "f" is required to be monotonically increasing and sufficiently smooth, resulting in two groups of strain tensors, which have the principal directions of the tensors $\underset{\sim}{U}$ and $\underset{\sim}{V}$, respectively.

$$\underset{\sim}{\varepsilon}_o = \underset{\sim}{f}\left(\underset{\sim}{U}\right) = \sum \underset{\sim}{f}\left(u_{(i)}\right) \underset{\sim}{P}_i \cdot \underset{\sim}{P}_i \tag{7.25}$$

and

$$\underset{\sim}{\varepsilon} = \underset{\sim}{f}\left(\underset{\sim}{V}\right) = \sum \underset{\sim}{f}\left(u_{(i)}\right) \underset{\sim}{p}_i \cdot \underset{\sim}{p}_i . \tag{7.26}$$

In both cases, the principal directions are given through the same material orthogonal triad, which is adopted in both reference and current configurations. The corresponding strain tensors are denoted as Lagrange and Euler strain formulations.

The tensor functions $\underset{\sim}{f}\left(\underset{\sim}{U}\right)$ and $\underset{\sim}{f}\left(\underset{\sim}{V}\right)$ are defined in (7.25) and (7.26) by the principal values and directions of their arguments. If the argument $\underset{\sim}{U}$ for an orthogonal tensor $\underset{\sim}{Q}$ is substituted by $\underset{\sim}{Q}\underset{\sim}{U}\underset{\sim}{Q}^{\mathrm{T}}$, only the principal directions change to $\underset{\sim}{Q}\underset{\sim}{P}$; their principal values do not change. By comparison, results for all orthogonal tensors $\underset{\sim}{Q}$ are

$$\underset{\sim}{f}\left(\underset{\sim}{Q}\underset{\sim}{U}\underset{\sim}{Q}^{\mathrm{T}}\right) = \underset{\sim}{Q}\underset{\sim}{f}\left(\underset{\sim}{U}\right)\underset{\sim}{Q}^{\mathrm{T}}. \tag{7.27}$$

According to an equivalent definition, "$\underset{\sim}{f}$" is an isotropic tensor function of tensor variables.

It is beneficial to define a strain measure in such a way that it coincides after linearization with the common strain measure for small deformations. After further restrictions of "$\underset{\sim}{f}$," it comes to

$$\underset{\sim}{f}(I) = 0, \quad \underset{\sim}{f}'(I) = (I). \tag{7.28}$$

A strain tensor with this property is the logarithmic strain tensor.

$$f\left(u_{(i)}\right) = \log\left(u_{(i)}\right), \tag{7.29}$$

i.e.,

$$\underset{\sim}{\varepsilon}_o = \log\left(u_{(i)}\right) \underset{\sim}{P}_i \cdot \underset{\sim}{P}_i = \log\left(\underset{\sim}{U}\right), \tag{7.30}$$

where the term "log (.)" denotes the natural logarithm, alternatively denoted as "ln (.)."

The corresponding Euler strain tensors are obtained using $\underset{\sim}{V}$ instead of $\underset{\sim}{U}$.

$$\underset{\sim}{\varepsilon} = \log\left(v_{(i)}\right) \underset{\sim i}{p} \cdot \underset{\sim i}{p} = \log\left(\underset{\sim}{V}\right). \tag{7.31}$$

A special property of the logarithmic strain tensor is the simple relationship between its trace and the volumetric strain of the deformed element

$$\mathrm{tr}\,(\varepsilon) = \mathrm{tr}(\varepsilon)_o = \sum \log\left(u_{(i)}\right) = \log\left(u_1 u_2 u_3\right) = \log \frac{\mathrm{d}V}{\mathrm{d}V_\mathrm{o}}. \tag{7.32}$$

These strain measures in general also have a disadvantage because a transformation of the principal axes is necessary to calculate the values of its components. These disadvantages, on the other hand, do not apply when the strain measure is needed only for the description of rotation-free deformation.

7.2.5 Velocities

The velocity of a material particle is described by the partial time derivative of the change of its position vector with respect to the observer space

$$\dot{\underset{\sim}{x}} = \dot{\underset{\sim}{x}}\left(\underset{\sim}{x}, t\right) = \dot{\underset{\sim\kappa}{\chi}}\left(\underset{\sim}{X}, t\right) = \frac{\partial}{\partial t}\left(\underset{\sim}{x}, t\right) = \underset{\sim}{v} = v^i \underset{\sim i}{g}\,. \tag{7.33}$$

The over-imposed point is also applied in other cases to designate that the material point is followed during the time differentiation. The material derivative operator

$$(\dot{\cdot}) \equiv \frac{D}{Dt}\,(\;) \equiv \frac{\partial}{\partial t}\,(\;)\Big|_{(X=\text{const.})} \tag{7.34}$$

can be applied to scalar, vector, or tensor functions.

$$\dot{\underset{\sim k}{g}} = \frac{D}{Dt}\underset{\sim k}{g} = \Gamma^i_{jk} v^j \underset{\sim i}{g} \tag{7.35}$$

with the Christoffel symbols

$$\begin{aligned} \Gamma^i_{jk} &= \tfrac{1}{2} g^{ir}\left(g_{kr,j} + g_{rj,k} - g_{jk,r}\right) \\ &= \Gamma^i_{jkr} g^{ir} = \underset{\sim j,k}{g} \cdot \underset{\sim}{g}^i = \underset{\sim k,j}{g} \cdot \underset{\sim}{g}\cdot \end{aligned} \tag{7.36}$$

7.2.6 The Velocity Gradient

The space gradient "grad $\underset{\sim}{v}$" of the velocity field "$\underset{\sim}{v} = \dot{\underset{\sim}{x}}\left(\underset{\sim}{x}, t\right)$" specifies the change of the velocity along $\mathrm{d}\underset{\sim}{x}$. The space velocity gradient "$\underset{\sim}{L}$" is given by

$$\underset{\sim}{L} = \operatorname{grad}\ \underset{\sim}{v} = \frac{\partial \underset{\sim}{v}}{\partial \underset{\sim}{x}} = \frac{\partial \dot{\underset{\sim}{x}}\left(\underset{\sim}{x}, t\right)}{\partial \underset{\sim}{x}} = \operatorname{grad}\ \dot{\underset{\sim}{x}}, \tag{7.37}$$

$$\mathrm{d}\underset{\sim}{v} = \mathrm{d}\dot{\underset{\sim}{x}} = \operatorname{grad}\ \underset{\sim}{v}\ \mathrm{d}\underset{\sim}{x} = \underset{\sim}{L}\ \mathrm{d}\underset{\sim}{x}, \tag{7.38}$$

$$\underset{\sim}{L} = v^i\Big|_j \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = \left\{\frac{\partial v^i}{\partial x^j} + \Gamma^i_{jk} v^k\right\} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j. \tag{7.39}$$

The transpose tensor is expressed as

$$\underset{\sim}{L}^{\mathrm{T}} = v_j\Big|^i \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = \left\{\frac{\partial v_j}{\partial x^i} + \Gamma^i_{jk} v^k\right\} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j \tag{7.40}$$

and

$$\operatorname{Tr}\left(\underset{\sim}{L}\right) = \operatorname{Tr}\left(\operatorname{grad}\ \underset{\sim}{v}\right) = \operatorname{div}\left(\underset{\sim}{v}\right). \tag{7.41}$$

The material time derivative of the deformation gradient is given by

$$\dot{\underset{\sim}{F}} = \underset{\sim}{L}\underset{\sim}{F} = \left(\dot{F}^i_{.\alpha} + F^j_{.\alpha}\Gamma^i_{jr} v^r\right) \underset{\sim i}{g} \cdot \underset{\sim}{G}^\alpha \tag{7.42}$$

so that $\underset{\sim}{L}$can be expressed as

$$\underset{\sim}{L} = \dot{\underset{\sim}{F}}\underset{\sim}{F}^{-1}. \tag{7.43}$$

The space velocity gradient $\underset{\sim}{L}$ can be additively divided into symmetric and skew symmetric tensors, i.e.,

$$\underset{\sim}{L} = \underset{\sim}{D} + \underset{\sim}{W}, \tag{7.44}$$

$$\underset{\sim}{D} = \frac{1}{2}\left(\underset{\sim}{L} + \underset{\sim}{L}^{\mathrm{T}}\right) = \frac{1}{2}\left\{v^i\big|_j + v_j\big|^i\right\} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = d^i_j \underset{\sim i}{g} \cdot \underset{\sim}{g}^j, \tag{7.45}$$

$$\underset{\sim}{W} = \frac{1}{2}\left(\underset{\sim}{L} - \underset{\sim}{L}^{\mathrm{T}}\right) = \frac{1}{2}\left\{v^i\big|_j - v_j\big|^i\right\} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = w^i_j \underset{\sim i}{g} \cdot \underset{\sim}{g}^j, \tag{7.46}$$

where $\underset{\sim}{D}$ is denoted as the strain velocity tensor and $\underset{\sim}{W}$ as the rotation velocity tensor or the spin tensor. The material time derivative of J is obtained from

$$\dot{J} = \frac{DJ}{Dt} = J\ \mathrm{Tr}\left(\underset{\sim}{L}\right) = J v^i\Big|_i. \tag{7.47}$$

7.2.7 Stress Tensors

The symmetric Cauchy stress tensor

$$\underset{\sim}{S} = \sigma^i_{.j} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = \sigma^i_j \underset{\sim i}{g} \cdot \underset{\sim}{g}^j \tag{7.48}$$

is associated with the current configuration. Two additional forms of the stress tensor are further needed, which are the nonsymmetric Lagrange (first Piola–Kirchhoff) stress tensor

$$\underset{\sim \mathrm{L}}{S} = J \underset{\sim}{S}\left(F^{-1}\right)^{\mathrm{T}} = s^{.\alpha}_i \underset{\sim}{g}^i \cdot \underset{\sim \alpha}{G}, \tag{7.49}$$

and the symmetric Kirchhoff stress tensor, which is applied per unit mass of the deformed material element

$$\underset{\sim \mathrm{K}}{S} = J \underset{\sim}{S} = \sigma^i_{K.j} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j. \tag{7.50}$$

The Kirchhoff stress tensor coincides with the Cauchy stress tensor for incompressible material behavior.

Furthermore, the stress deviator needs to be considered.

$$\underset{\sim}{T} = \underset{\sim}{S} - \frac{1}{3}\mathrm{tr}\left(\underset{\sim}{S}\right) \underset{\sim}{I} = t^i_j \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = \left(\sigma^i_j - \frac{1}{3}\sigma^r_r \delta^i_j\right) \underset{\sim i}{g} \cdot \underset{\sim}{g}^j, \tag{7.51}$$

$$\underset{\sim K}{T} = \underset{\sim K}{S} - \frac{1}{3}\mathrm{tr}\left(\underset{\sim K}{S}\right) \underset{\sim}{I} = t^i_{Kj} \underset{\sim i}{g} \cdot \underset{\sim}{g}^j = \left(\sigma^i_{Kj} - \frac{1}{3}\sigma^r_{Kr}\delta^i_j\right) \underset{\sim i}{g} \cdot \underset{\sim}{g}^j, \tag{7.52}$$

where $\delta^i_j = 1$ for $i = \mathrm{j}$ and $\delta^i_j = 0$ for $i \neq j$.

The physical components $\hat{\sigma}^i_j$ of the tensor $\underset{\sim}{S}$ result from

$$\hat{\sigma}^i_j = \frac{\sqrt{g^{ij}}}{\sqrt{g^{ii}}}\sigma^i_j \quad \not\exists\ i, j. \tag{7.53}$$

In case a principal axes system is considered, the physical components are equal to the mixed-variant components of the tensor. The substantial time derivatives are calculated using (7.47) to (7.49). They can be expressed as

$$\underset{\sim}{\dot{S}}_{\mathrm{L}} = J\left[\underset{\sim}{\dot{S}} + \underset{\sim}{S}\,\mathrm{tr}\left(\underset{\sim}{L}\right) + \underset{\sim}{S}\underset{\sim}{L}^{\mathrm{T}}\right]\left(F^{-1}\right)^{\mathrm{T}}, \tag{7.54}$$

$$\underset{\sim}{\dot{S}}_{\mathrm{K}} = J\left[\underset{\sim}{\dot{S}} + \underset{\sim}{S}\,\mathrm{tr}\left(\underset{\sim}{L}\right)\right]. \tag{7.55}$$

An observer-independent objective time derivative is also needed, e.g., the Jaumann derivative given by

$$\overset{o}{\underset{\sim}{S}} = \underset{\sim}{\dot{S}} - \underset{\sim}{W}\underset{\sim}{S} + \underset{\sim}{S}\underset{\sim}{W}, \tag{7.56}$$

$$\overset{o}{\underset{\sim}{S}}_{\mathrm{K}} = \underset{\sim}{\dot{S}}_{\mathrm{K}} - \underset{\sim}{W}\underset{\sim}{S}_{\mathrm{K}} + \underset{\sim}{S}_{\mathrm{K}}\underset{\sim}{W} = J\left[\overset{o}{\underset{\sim}{S}} + \underset{\sim}{S}\,\mathrm{Tr}\left(\underset{\sim}{L}\right)\right]. \tag{7.57}$$

The material and objective time derivatives of the stress deviators (7.50) and (7.51) can be derived in a similar manner.

7.2.8 Equilibrium Conditions

In the case the mass forces and inertia effects are negligible, the equilibrium condition

$$\mathrm{div}\ \underset{\sim}{S} = \sigma^{i}_{j}\Big|_{i}\, \underset{\sim}{g}^{j} = 0 \tag{7.58}$$

must be satisfied with

$$\sigma^{i}_{j}\Big|_{i} = \sigma^{i}_{j,i} + \Gamma^{i}_{ir}\sigma^{r}_{j} - \Gamma^{s}_{ji}\sigma^{i}_{s}. \tag{7.59}$$

The necessity that (7.58) remains satisfied at all times requires that its material time derivative, which represents a condition for continuous equilibrium, must also be satisfied. Accordingly, the condition for continuous equilibrium can be written as

$$\dot{\overline{\mathrm{div}\ \underset{\sim}{S}}} = 0. \tag{7.60}$$

It can be obtained after some calculations [19] in the form

$$\mathrm{div}\left[\underset{\sim}{\dot{S}} + \underset{\sim}{S}\,\mathrm{Tr}\left(\underset{\sim}{L}\right) - \underset{\sim}{S}\underset{\sim}{L}^{\mathrm{T}}\right] = 0. \tag{7.61}$$

It should be noted at this point that the different definitions of the divergence (div) [11] lead to different expressions for (7.61), which revert back to the same equations when calculating their associated components. If the objective time derivative is applied, Eq. (7.61) can be written as

$$\operatorname{div}\left[\frac{1}{J}\overset{o}{\underset{\sim}{S}}_{\mathrm{K}} + \underset{\sim}{W}\,\underset{\sim}{S} - \underset{\sim}{S}\,\underset{\sim}{D}\right] = 0, \tag{7.62}$$

or in mixed variant components as

$$\left[\frac{1}{J}\sigma^{i}_{Kj}|_{o}\underset{\sim}{g}^{j} + w^{\cdot r}_{j}\sigma^{i}_{r} - \sigma^{r}_{j}d^{i}_{r}\right]\Big|_{i} = 0 \tag{7.63}$$

with the components of $\overset{o}{\underset{\sim}{S}}_{\mathrm{K}}$ expressed as $\sigma^{i}_{Kj}\big|_{o}$.

7.3 Constitutive Laws

The constitutive law describes the relationship between stresses or stress increments and strains or strain rates. For an elastic–plastic body, the total deformation can be formally divided into elastic and plastic parts, which leads to a multiplicative decomposition of the deformation gradient,

$$\underset{\sim}{F} = \underset{\sim}{F}_{\mathrm{e}}\,\underset{\sim}{F}_{\mathrm{p}}. \tag{7.64}$$

Such splitting implies the assumption of an incompatible intermediate state. A splitting that deviates from (7.64) is possible ([20, 21, 26, 27]). All these possibilities lead eventually to an additive splitting of the strain rate tensor into elastic and plastic parts.

$$\underset{\sim}{D} = \underset{\sim}{D}_{\mathrm{e}} + \underset{\sim}{D}_{\mathrm{p}} = \left(\overset{\mathrm{e}}{d}{}^{i}_{j} + \overset{\mathrm{p}}{d}{}^{i}_{j}\right)\underset{\sim}{g}_{i}\cdot\underset{\sim}{g}^{j}. \tag{7.65}$$

Both parts of (7.65) do not necessarily need to satisfy all conditions that the total strain rate needs to satisfy.

7.3.1 Elastic Deformation

For small elastic deformation, as, e.g., in metallic materials, the following known relationship is valid.

$$\underset{\sim}{D}_{\mathrm{e}} = \frac{1}{2G}\left[\overset{o}{\underset{\sim}{S}}_{\mathrm{K}} - \frac{\upsilon}{1+\upsilon}\mathrm{Tr}\left(\overset{o}{\underset{\sim}{S}}_{\mathrm{K}}\right)\underset{\sim}{1}\right]. \tag{7.66}$$

This constitutive law can also be written for hypo-elastic materials in the form:

$$\underset{\sim}{D}_{\mathrm{e}} = \frac{1}{2G}\overset{o}{\underset{\sim}{T}}_{K} + \frac{1}{9K}\mathrm{Tr}\left(\dot{\underset{\sim}{S}}_{\mathrm{K}}\right)\underset{\sim}{1} \tag{7.67}$$

with

$$\mathrm{Tr}\left(\overset{o}{\underset{\sim}{S}}_{\mathrm{K}}\right) = \mathrm{Tr}\left(\dot{\underset{\sim}{S}}_{\mathrm{K}}\right). \tag{7.68}$$

The Volumetric Modulus (Bulk's Modulus) is given by

$$K = 2G\frac{1+\upsilon}{3\left(1-2\upsilon\right)} = E\,\frac{1}{3\left(1-2\upsilon\right)}. \tag{7.69}$$

Equation (7.67) can be expressed in component form as

$$\overset{\mathrm{e}}{d}{}^{i}_{j} = \frac{1}{2G}t^{i}_{Kj}\Big|_{o} + \frac{1}{9K}\sigma^{s}_{Ks}\Big|_{o}\delta^{i}_{j}. \tag{7.70}$$

7.3.2 *Plastic Deformation*

Plastic deformation is determined through a yield criterion and a constitutive law.

Yield Criterion

The yield criterion provides an expression for the onset of plastic deformation as a function of the stresses and the material hardening. It can be expressed under the restriction to an isotropic material hardening [20, 21] by

$$F\left(\underset{\sim}{T},\ldots\right) = F\left(J_2, J_3, k^2\right) = f\left(J_2, J_3\right) - k^2\left(W_{\mathrm{p}}\right), \tag{7.71}$$

where

$$J_2 = \mathrm{Tr}\left(\underset{\sim}{T}^2_{K}\right), \tag{7.72}$$

$$J_3 = \mathrm{Tr}\left(\underset{\sim}{T}^3_{K}\right), \tag{7.73}$$

$$W_{\mathrm{p}} = \int_{t_o}^{t} Sp\left(\underset{\sim K}{T}\,\underset{\sim \mathrm{p}}{D}\right) \mathrm{d}\tau \tag{7.74}$$

The quantity k^2 is a scalar function, which describes the material hardening.

Deformation Law

The applied deformation law is based on the theory of plastic potential given by

$$\begin{aligned} \underset{\sim \mathrm{p}}{D} &= \dot{\lambda} \frac{\partial F}{\partial \underset{\sim \mathrm{K}}{S}} \\ \overset{\mathrm{p}}{d}\Big|_{j}^{i} &= \dot{\lambda} \frac{\partial F}{\partial \sigma_{Ki}^{j}} \end{aligned} . \tag{7.75}$$

This approach implies the so-called normality rule, according to which the plastic deformation occurs normal to the yield surface. Accordingly, $\dot{\lambda}$ is an always-positive scalar function of the stresses, through which the yield criterion (7.71) is specified to remain always satisfied during the plastic deformation, i.e.,

$$\dot{F} = 0. \tag{7.76}$$

7.3.3 Formulation of an Associative Constitutive Law

The material law defining of the elastoplastic deformations is obtained according to Eq. (7.65). For the associative constitutive law, the normality rule (7.75) applies. Plastic deformations occur only if the yield criterion (7.71) is satisfied. It should be noted here that

- In the resulting equations, the stress differences appear as significant quantities.
- The current state of stress of the basic solution is calculated from predefined strains and strain rates.
- Large three-dimensional strains in the current state are taken into consideration.
- The velocities in the current state may be in general not unique.

Constitutive Law Adopting Von Mises Yield Criterion

Constitutive laws in literature derived by applying the von Mises yield criterion are expressed as nonlinear equations, which contain components of both the stresses and the strain rates simultaneously. The solution of such equations by numerical integration involves excessive large computational effort. Therefore, an

approximation method is needed to be introduced for the numerical calculation of the current state. The predefined assumptions of the problem should be fully utilized, particularly the incompressibility and the axial homogeneity of the deformation. Besides, a specified load path should be prescribed.

The disadvantage of the von Mises yield criterion regarding the large numerical efforts is compensated on the other hand by the fact that the yield function is continuously differentiable with respect to the stresses, which allows for utilizing the analysis approach in bifurcation and instability investigations [1, 22–24]. For the sake of completeness, the derivation of the material law is briefly sketched here. Limiting considerations to statements of the type

$$\frac{\partial f}{\partial J_3} = 0 \tag{7.77}$$

and

$$\frac{\partial f}{\partial J_2} = 1 \tag{7.78}$$

results from (7.71), the well-known relationship by Huber/Mises/Hencky

$$F = \mathrm{Tr}\left(\underset{\sim K}{T^2}\right) - k^2 = t^i_{Kj} t^j_{Ki} - k^2 = 0 \tag{7.79}$$

as well as a function of the principal stresses

$$F = \left(\sigma^1_{K1} - \sigma^2_{K2}\right)^2 + \left(\sigma^2_{K2} - \sigma^3_{K3}\right)^2 + \left(\sigma^3_{K3} - \sigma^1_{K1}\right)^2 - 3k^2 = 0. \tag{7.80}$$

Applying the normality rule to the above yield criterion with

$$\underset{\sim \mathrm{p}}{D} = \dot{\lambda} 2 \underset{\sim K}{T} \tag{7.81}$$

results in a linear relationship between the components of the strain rate tensor and those of Kirchhoff stress deviator. The quantity $\dot{\lambda}$ is obtained by considering the loading condition (7.76) together with the expression of plastic work (7.74)

$$\dot{\lambda} = \frac{\rho_o}{\left(\frac{\partial k^2}{\partial W_\mathrm{p}}\right)} \frac{\mathrm{Tr}\left(\underset{\sim K}{T} \overset{o}{\underset{\sim K}{T}}\right)}{\mathrm{Tr}\left(\underset{\sim K}{T^2}\right)} \tag{7.82}$$

so that the plastic strain rate tensor leads to

$$\underset{\sim \mathrm{p}}{D} = \frac{2\rho_o}{\left(\frac{\partial k^2}{\partial W_\mathrm{p}}\right)} \frac{\mathrm{Tr}\left(\underset{\sim K}{T} \overset{o}{\underset{\sim K}{T}}\right)}{\mathrm{Tr}\left(\underset{\sim K}{T^2}\right)} \underset{\sim K}{T}. \tag{7.83}$$

Including also the elastic component in the calculation leads to the equation

$$\underset{\sim}{D} = \frac{1}{2G}\overset{o}{\underset{\sim}{T}}_K + \frac{1}{9K}\mathrm{Tr}\left(\underset{\sim}{\dot{S}}_K\right)\underset{\sim}{1} + \frac{2\rho_o}{\left(\frac{\partial k^2}{\partial W_\mathrm{p}}\right)}\frac{\mathrm{Tr}\left(\underset{\sim}{T}_K\,\overset{o}{\underset{\sim}{T}}_K\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2_K\right)}\underset{\sim}{T}_K. \tag{7.84}$$

After a short calculation, a constitutive law is obtained for an elastic–plastic body, valid for small elastic and large plastic deformation with isotropic hardening.

$$\underset{\sim}{D} = \frac{1}{2G}\overset{o}{\underset{\sim}{T}}_K + \frac{1}{9K}\mathrm{Tr}\left(\underset{\sim}{\dot{S}}_\mathrm{K}\right)\underset{\sim}{1} + \delta\left(W_\mathrm{p}\right)\frac{\mathrm{Tr}\left(\underset{\sim}{T}_K\,\underset{\sim}{D}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2_K\right)}\underset{\sim}{T}_K. \tag{7.85}$$

The hardening parameter $\delta(W_\mathrm{p})$ is a dimensionless material property, which has zero value for elastic behavior and has the value of 1.0 for elastic–perfectly plastic behavior

$$\delta\left(W_\mathrm{p}\right) = \frac{1}{1 + \frac{1}{4G\rho_o}\left(\frac{\partial k^2}{\partial W_\mathrm{p}}\right)} \quad 0 \le \delta \le 1. \tag{7.86}$$

For incompressible materials, the constitutive law (7.85) is simplified to

$$\underset{\sim}{D} = \frac{1}{2G}\overset{o}{\underset{\sim}{T}} + \delta\left(W_\mathrm{p}\right)\frac{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)}\underset{\sim}{T}. \tag{7.87}$$

The hardening parameters δ can be obtained as a function of the tangent modulus

$$\delta\left(W_\mathrm{p}\right) = 1 - \frac{E_t}{E}. \tag{7.88}$$

For deformation without rotation, a special nonlinear constitutive law is obtained as

$$\underset{\sim}{D} = \frac{1}{2G}\overset{\cdot}{\underset{\sim}{T}} + \quad \delta\left(W_\mathrm{p}\right)\frac{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)}\underset{\sim}{T}. \tag{7.89}$$

Even if the strain rate tensor $\underset{\sim}{D}$ is known, it is still impossible to find a closed solution for the stress deviator $\underset{\sim}{T}$. Numerical integration is only feasible with considerable effort. Therefore, approximation methods need to be deployed.

A second possible approach is calculating the components of the stress deviator iteratively. In this case, a carefully selected approximation in the first iteration can lead to a very accurate solution in the second iteration.

The approach of such an approximation is based on one of the following cases:

1. The loading path is approximately proportional, provided that the elastic deformation is in the same order as the plastic deformation.
2. The stress deviator "$\underset{\sim}{T}$" can be directly obtained from $\underset{\sim}{D}$, provided that the elastic deformation is small compared to the plastic deformation.

In both cases, an approximation only in the elastic part of the deformation leads to a small deviation from the exact solution. In the second iteration, these deviations become negligible by all measures. The application of the constitutive law for calculating the current state of an elastic–plastic thick-walled cylinder will be explained in Sect. 7.4.

7.3.4 Material Model

The expression for the hardening parameter is usually obtained empirically. An approximation law similar to that of Ramberg–Osgood [19] has been selected. It is expressed as

$$\varepsilon = \frac{\sigma}{E} + \frac{\sigma_o}{B}\left(\frac{\sigma}{\sigma_o} - 1\right)^n \quad \sigma \geq \sigma_o, \tag{7.90}$$

$$\varepsilon = \frac{\sigma}{E} \quad \sigma \leq \sigma_o, \tag{7.91}$$

where σ and ε are the true stress and the Hencky (logarithmic) strain, respectively, which are obtained from a uniaxial tensile or compression test.

The tangent modulus is given by

$$E_\mathrm{t} = \frac{\mathrm{d}\sigma}{\mathrm{d}\varepsilon} = \frac{E}{1 + \frac{E}{B} n \left(\frac{\sigma}{\sigma_o} - 1\right)^{n-1}} \quad \sigma \geq \sigma_o, \tag{7.92}$$

$$E_\mathrm{t} = \frac{\mathrm{d}\sigma}{\mathrm{d}\varepsilon} = E \quad \sigma \leq \sigma_o. \tag{7.93}$$

The curvature is described by

$$\frac{\mathrm{d}E_\mathrm{t}}{\mathrm{d}\varepsilon} = \frac{\mathrm{d}^2\sigma}{\mathrm{d}\varepsilon^2} = \frac{E^3 n\ (n-1)\ \left(\frac{\sigma}{\sigma_o} - 1\right)^{n-2}}{\sigma_o B \left[1 + \frac{E}{B} n \left(\frac{\sigma}{\sigma_o} - 1\right)^{n-1}\right]^3} \quad \sigma \geq \sigma_o, \tag{7.94}$$

$$\frac{\mathrm{d}E_\mathrm{t}}{\mathrm{d}\varepsilon} = \frac{\mathrm{d}^2\sigma}{\mathrm{d}\varepsilon^2} = 0 \quad \sigma \leq \sigma_o. \tag{7.95}$$

Such a material model implies a smooth transition at the elastic–plastic interface as well as a continuous tangent modulus at this interface. The continuity of further derivatives depends on the value of the coefficient n. The tangent modulus changes rapidly once the elastic–plastic limit is reached.

7.4 Application to Thick-Walled Cylinders

Large elastic–plastic deformation is described in detail for a thick-walled cylinder, simultaneously loaded by hydrostatic pressure and axial force, considering nonlinear isotropic hardening. A theoretical solution is obtained by substituting a constitutive law into the equilibrium equations, maintaining geometric compatibility and satisfying boundary conditions. An algorithm is developed for calculating the stress and strain distributions. The applied loads are calculated for prescribed states of large deformation.

7.4.1 Geometric Analysis

Large deformation is described in detail for a thick-walled cylinder. The deformation is axisymmetric and axially homogeneous. The material points move only in the radial and axial directions. However, deformation is a function of the radial coordinate only. Strains are uniquely defined in the deformed state and are also functions of the radial location.

Coordinate Systems

To describe the basic deformation of a thick-walled cylinder, the initial undeformed configuration is selected as the reference configuration. A cylindrical coordinate system is also selected for space-fixed and body-fixed coordinate systems. The space and reference coordinates are defined according to (7.2) and (7.3) as

$$x^j = (r, \phi, z)\,, \tag{7.96}$$

$$X^\alpha = (R, \Phi, Z)\,. \tag{7.97}$$

The corresponding space-fixed Cartesian coordinate z^i are function of the curvilinear coordinates, given by

$$z^1 = r\ \cos\ \phi, \tag{7.98}$$

$$z^2 = r\ \sin\ \phi, \tag{7.99}$$

$$z^3 = z. \tag{7.100}$$

Thus, the following expressions for the base vectors can be obtained.

$$\underset{\sim 1}{g} = \cos\ \phi \underset{\sim 1}{b} + \sin\phi \underset{\sim 2}{b}\,, \tag{7.101}$$

$$\underset{\sim 2}{g} = -r\ \sin\ \phi \underset{\sim 1}{b} + r\ \cos\phi \underset{\sim 2}{b}\,, \tag{7.102}$$

$$\underset{\sim 3}{g} = \underset{\sim 3}{b}\,. \tag{7.103}$$

The metric tensors of the curvilinear coordinate system are

$$g_{ik} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.104}$$

$$g^{ik} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{r^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.105}$$

The Christoffel symbols can be written according to (7.36) as

$$\Gamma^1_{22} = -r, \quad \Gamma^2_{12} = \Gamma^2_{21} = \frac{1}{r}, \tag{7.106}$$

all further Christoffel symbols vanish.

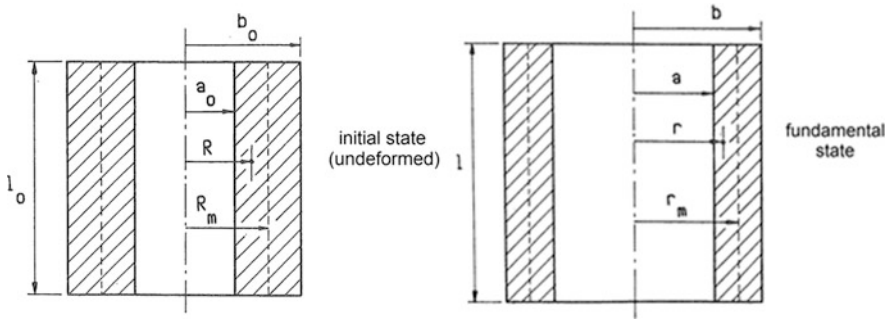


Fig. 7.1 Basic Axially-Symmetric Deformation

Basic Deformation

In the following calculations, the derivatives with respect to the reference coordinates are frequently needed. They are denoted as

$$(\cdot)' = \frac{\partial}{\partial R}(\cdot), \quad (\cdot)^* = \frac{\partial}{\partial \Phi}(\cdot), \quad (\cdot)^+ = \frac{\partial}{\partial Z}(\cdot). \tag{7.107}$$

The material points move only in the radial and axial directions. However, deformations are only functions of the radial coordinate (Fig. 7.1), i.e.,

$$\begin{aligned} &\phi = \Phi, \quad (\cdot)^* = 0 \\ &z^+ = \frac{l}{l_o}, \quad \dot{z}^+ = \frac{\dot{l}}{l_o} \end{aligned}, \tag{7.108}$$

where

$$l(t_o) = l_o. \tag{7.109}$$

Consequently, plane cross sections remain plane and the circular cylindrical lateral surfaces remain circular cylindrical. Since the deformation contains no rotation, the strain rate tensor in the current state can be directly obtained as

$$d^i_k = L^i_{.k} = v^i\Big|_k = \begin{bmatrix} \frac{\dot{r}'}{r'} & 0 & 0 \\ 0 & \frac{\dot{r}}{r} & 0 \\ 0 & 0 & \frac{\dot{l}}{l} \end{bmatrix}, \tag{7.110}$$

where according to Eq. (7.108)

$$\frac{\dot{z}^+}{z^+} = \frac{\dot{l}}{l}. \tag{7.111}$$

The assumption of incompressibility results in the condition

$$d_s^s = \frac{\dot{r}'}{r'} + \frac{\dot{r}}{r} + \frac{\dot{l}}{l} = 0, \tag{7.112}$$

the integration of which leads to the relationship

$$R^2 = \frac{l}{l_o}\left(r^2 - C_1\right), \tag{7.113}$$

where the initial conditions at $t = t_o$ are given by

$$r\left(t_o\right) = R, \tag{7.114}$$

$$r'\left(t_o\right) = 1.0, \tag{7.115}$$

The integration constant C_1 in (7.113) as well as the homogeneous axial strain are purely geometrical and independent of the loading or the material properties as long as the incompressibility condition is preserved. The integration constant C_1 in (7.113) is still a function of time [15, 28], the value of which together with the homogeneous axial strain determine the state of deformation of the thick-walled cylinder completely independent of the loading or the material properties.

The components of the strain rate tensor are obtained from

$$d_1^1 = \frac{1}{2r^2}\dot{C}_1 - \frac{1}{2}\left(1 + \frac{C_1}{r^2}\right)\frac{\dot{l}}{l}, \tag{7.116}$$

$$d_2^2 = \frac{1}{2r^2}\dot{C}_1 - \frac{1}{2}\left(1 - \frac{C_1}{r^2}\right)\frac{\dot{l}}{l}, \tag{7.117}$$

$$d_3^3 = \frac{\dot{l}}{l}, \tag{7.118}$$

and as functions of the reference coordinate R

$$d_1^1 = -\frac{1}{2}\frac{\dot{l}}{l} - \frac{1}{2}\left(\frac{1}{R^2 + C_1\left(\frac{l}{l_o}\right)}\right)\left(C_1\left(\frac{l}{l_o}\right)\right)^{\cdot}, \tag{7.119}$$

$$d_2^2 = -\frac{1}{2}\frac{\dot{l}}{l} + \frac{1}{2}\left(\frac{1}{R^2 + C_1\left(\frac{l}{l_o}\right)}\right)\left(C_1\left(\frac{l}{l_o}\right)\right)^{\cdot}, \tag{7.120}$$

$$d_3^3 = \frac{\dot{l}}{l}. \tag{7.121}$$

Monotonic deformation allows for the time integration of the components of the strain rate tensor to represent the components of the Hencky (logarithmic) strain tensor

$$\varepsilon_1^1 = -\frac{1}{2}\ln\left(\frac{l}{l_o}\right) + \frac{1}{2}\ln\left(1 - \frac{C_1}{r^2}\right), \tag{7.122}$$

$$\varepsilon_2^2 = -\frac{1}{2}\ln\left(\frac{l}{l_o}\right) - \frac{1}{2}\ln\left(1 - \frac{C_1}{r^2}\right), \tag{7.123}$$

$$\varepsilon_3^3 = \ln\left(\frac{l}{l_o}\right), \tag{7.124}$$

and with respect to material coordinates

$$\varepsilon_1^1 = -\frac{1}{2}\ln\left(\frac{l}{l_o}\right) + \frac{1}{2}\ln\left(\frac{R^2 l_o}{R^2 l_o + C_1 l}\right), \tag{7.125}$$

$$\varepsilon_2^2 = -\frac{1}{2}\ln\left(\frac{l}{l_o}\right) - \frac{1}{2}\ln\left(\frac{R^2 l_o}{R^2 l_o + C_1 l}\right), \tag{7.126}$$

$$\varepsilon_3^3 = \ln\left(\frac{l}{l_o}\right). \tag{7.127}$$

The strains and strain rates are functions of the radial locations r and R, independent of the cylinder geometry. The inner radius in the deformed state is termed as "a" and the outer radius as "b." The radius of middle surface is

$$r_m = \frac{a+b}{2}. \tag{7.128}$$

In the initial state ($t = t_o$)

$$a\,(t_o) = a_o, \tag{7.129}$$

$$b\,(t_o) = b_o, \tag{7.130}$$

$$r_m\,(t_o) = R_m. \tag{7.131}$$

The following geometric expression provides the relationship between the geometric variables in the initial state and those in the current state

$$\frac{b_o}{a_o} = \left[\frac{(b/a)^2 - L_1}{1 - L_1}\right]^{\frac{1}{2}}, \tag{7.132}$$

$$\frac{l_o}{R_m} = \frac{l}{r_m}\left[\frac{1 + (b/a)}{1 + (b_o/a_o)}\right]\frac{(l_o/l)^{1.5}}{\left[1 - \left(C_1/a^2\right)\right]^{0.5}}. \tag{7.133}$$

The following dimensionless parameter is introduced.

$$L_1 = \frac{C_1}{a^2}. \tag{7.134}$$

The parameter L_1 is identified as the "Radial Loading Parameter" and will be used as a measure for the radial deformation.

7.4.2 Stress–Strain Analysis

The material is assumed to be a homogeneous and isotropic elastic–plastic continuous medium, with isotropic plastic hardening. Small-elastic and large-plastic constant-volume deformation is also assumed ignoring time, temperature, and inertia effects. Stresses and strains are uniquely defined in the current state and are functions of the radial location. A theoretical solution is generally based on substituting the constitutive law into the equilibrium equations, maintaining geometric compatibility and satisfying boundary conditions. The cylinder is loaded by hydrostatic pressure and axial force simultaneously. The adopted constitutive law is based on applying the von Mises yield criterion in association with its normality rule.

Stress Distribution in the Elastic Range

Due to elastic incompressibility, the Hooke's law for small elastic deformation becomes

$$t_j^i = 2G\varepsilon_j^i, \tag{7.135}$$

a linear relationship between the components of the Hencky strain tensor (7.16)–(7.18) and that of the stress deviator. Accordingly, the stress differences become

$$\sigma_2^2 - \sigma_1^1 = -2G \ln\left(1 - \frac{C_1}{r^2}\right), \tag{7.136}$$

$$\sigma_2^2 - \sigma_3^3 = -G\left[3 \ln\left(\frac{l}{l_o}\right) + \ln\left(1 - \frac{C_1}{r^2}\right)\right], \tag{7.137}$$

$$\sigma_3^3 - \sigma_1^1 = G\left[3 \ln\left(\frac{l}{l_o}\right) - \ln\left(1 - \frac{C_1}{r^2}\right)\right], \tag{7.138}$$

and in terms of R

$$\sigma_2^2 - \sigma_1^1 = -2G \ln\left(\frac{R^2 l_o}{R^2 l_o + C_1 l}\right), \tag{7.139}$$

$$\sigma_2^2 - \sigma_3^3 = -G\left[3 \ln\left(\frac{l}{l_o}\right) + \ln\left(\frac{R^2 l_o}{R^2 l_o + C_1 l}\right)\right], \tag{7.140}$$

$$\sigma_3^3 - \sigma_1^1 = G\left[3 \ln\left(\frac{l}{l_o}\right) - \ln\left(\frac{R^2 l_o}{R^2 l_o + C_1 l}\right)\right]. \tag{7.141}$$

Stresses in the Elastic–Plastic Range

Due to the complexity arising from applying a constitutive law based on the von Mises yield criterion, an iterative algorithm was needed to be developed for calculating the stress distribution throughout the cylinder wall. The main approach is based on assuming small incompressible elastic but large plastic deformations together with almost-proportional loading. Stresses are calculated as functions of the radial position.

Due to applying a nonlinear constitutive law based on the von Mises yield criterion with the normality rule, the state of stress depends not only on the state of strain but also on the state of the strain rate. Consequently, calculating the stresses by classical integration of the incremental material law can only be carried out with substantially complex numerical calculations. An approximation method was therefore needed to be developed for the numerical calculation of the stresses in the current state utilizing an iterative approach to the solution.

For the Hencky strain tensor $\underset{\sim}{\varepsilon}$

$$\dot{\underset{\sim}{\varepsilon}} = \underset{\sim}{D}, \tag{7.142}$$

and accordingly

$$\underset{\sim}{\varepsilon} = \int_{t_o}^{t} \underset{\sim}{D}\, \mathrm{d}\tau \quad \text{if } \underset{\sim}{\varepsilon}\Big|_{t_o} = \underset{\sim}{0}. \tag{7.143}$$

The equivalent stress and strain rates are scalar quantities defined by

$$\overline{\sigma} = \left[\frac{3}{2}\mathrm{Tr}\left(\underset{\sim}{T}^2\right)\right]^{\frac{1}{2}}, \tag{7.144}$$

$$\dot{\overline{\varepsilon}} = \left[\frac{2}{3}\mathrm{Tr}\left(\underset{\sim}{D}^2\right)\right]^{\frac{1}{2}}, \tag{7.145}$$

and in component notation

$$\overline{\sigma} = \left[\frac{3}{2}\left(\left\{t_1^1\right\}^2 + \left\{t_2^2\right\}^2 + \left\{t_3^3\right\}^2\right)\right]^{\frac{1}{2}}, \tag{7.146}$$

$$\dot{\overline{\varepsilon}} = \left[\frac{2}{3}\left(\left\{\dot{\varepsilon}_1^1\right\}^2 + \left\{\dot{\varepsilon}_2^2\right\}^2 + \left\{\dot{\varepsilon}_3^3\right\}^2\right)\right]^{\frac{1}{2}}, \tag{7.147}$$

where $\underset{\sim}{T}$ is the Cauchy stress deviator and $\underset{\sim}{D}$ is the strain rate tensor.

It should be noted that for incompressible material behavior, the Cauchy stress tensor and the symmetric Kirchhoff stress tensor are identical. The value of the equivalent strain is obtained by the integration

$$\overline{\varepsilon} = \int_{t_o}^{t} \dot{\overline{\varepsilon}}\, \mathrm{d}\tau. \tag{7.148}$$

It is generally known, however, that

$$\overline{\varepsilon} \neq \left[\frac{2}{3}\mathrm{Tr}\left(\underset{\sim}{\varepsilon}^2\right)\right]^{\frac{1}{2}}. \tag{7.149}$$

A unique relationship between $\overline{\varepsilon}$ and $\overline{\sigma}$, which describes the isotropic material behavior at monotonic loading, is directly obtainable from a tension or compression test. As an approximation, the above-mentioned Ramberg–Osgood law is adopted.

Dividing the strain rate tensor into elastic and plastic components results in

$$\underset{\sim}{D} = \underset{\sim}{D}_{\mathrm{e}} + \underset{\sim}{D}_{\mathrm{p}}, \tag{7.150}$$

where according to Sect. 7.3

$$\underset{\sim}{D}_{\mathrm{e}} = \frac{1}{2G}\overset{\circ}{\underset{\sim}{T}}, \tag{7.151}$$

$$\underset{\sim}{D}_{\mathrm{p}} = \delta \frac{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)} \underset{\sim}{T} = \delta \left[\frac{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}_{\mathrm{e}}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)} + \frac{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}_{\mathrm{p}}\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)} \right] \underset{\sim}{T}. \tag{7.152}$$

7.4.3 The Iterative Solution

The components of the stress deviator are calculated iteratively. A carefully selected approximation in the first iteration can lead to a very accurate solution in the second iteration. An approximation only in the elastic part of the deformation leads to a small deviation from the exact solution. In the second iteration, the deviation becomes negligible. The stress and strain distributions are calculated, and the applied loads are obtained for prescribed states of large deformation.

First Iteration

As a first approximation, the elastic part of the strain rate tensor is obtained as

$$\underset{\sim}{D}_{\mathrm{e}} = \frac{3}{2} \frac{\overline{\varepsilon}_{\mathrm{e}}}{\overline{\sigma}} \underset{\sim}{\dot{T}}, \tag{7.153}$$

where the modulus of elasticity E, due to the assumed elastic incompressibility, is given by

$$E = 3G = \frac{\overline{\sigma}}{\overline{\varepsilon}_{\mathrm{e}}}. \tag{7.154}$$

Integration of (7.153) leads to

$$\underset{\sim}{\varepsilon}_{\mathrm{e}} = \frac{3}{2} \frac{\overline{\varepsilon}_{\mathrm{e}}}{\overline{\sigma}} \underset{\sim}{T}. \tag{7.155}$$

In the special case of proportional deformation, Eq. (7.149) does not apply. Accordingly, is valid along the loading path that

$$\underset{\sim}{D}_{\mathrm{e}} = \frac{\dot{\overline{\varepsilon}}_{\mathrm{e}}}{\overline{\varepsilon}_{\mathrm{e}}} \underset{\sim}{\varepsilon}_{\mathrm{e}} \tag{7.156}$$

and

$$\overline{\varepsilon}_{\mathrm{e}} = \left[\frac{2}{3} \mathrm{Tr}\left(\underset{\sim}{\varepsilon}_{\mathrm{e}}^2 \right) \right]^{\frac{1}{2}}. \tag{7.157}$$

For incompressible elastic–plastic material behavior and proportional deformation, a simple linear material law, which can also describe rigid plastic material behavior, is obtained after short calculation.

$$\underset{\sim}{D} = \left[\frac{\mathrm{Tr}\left(\underset{\sim}{D}^2\right)}{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)}\right]^{\frac{1}{2}} \underset{\sim}{T}. \tag{7.158}$$

The hardening parameter δ is substituted according to Eq. (7.88). The elastic part of the strain rate tensor becomes approximate if the loading path deviates from proportionality. However, Eq. (7.158) is exact for proportional loading.

Second Iteration

Rearranging the material law given by Eq. (7.89) leads directly to

$$\underset{\sim}{T} = \frac{1}{\delta}\frac{\mathrm{Tr}\left(\underset{\sim}{T}^2\right)}{\mathrm{Tr}\left(\underset{\sim}{T}\,\underset{\sim}{D}\right)}\left[\underset{\sim}{D} - \frac{1}{2G}\overset{\circ}{\underset{\sim}{T}}\right]. \tag{7.159}$$

The expression for the first iteration, Eq. (7.158) is reduced to

$$\underset{\sim}{T} = \frac{2}{3}\frac{\overline{\sigma}}{\dot{\overline{\varepsilon}}}\underset{\sim}{D} = \overset{(1)}{\underset{\sim}{T}}. \tag{7.160}$$

The material time derivative of Eq. (7.160) results in

$$\overset{\circ}{\underset{\sim}{T}} = \frac{2}{3}\left[\underset{\sim}{D}E_{\mathrm{t}} + \overline{\sigma}\,\overset{\cdot}{\overline{\left[\frac{\underset{\sim}{D}}{\dot{\overline{\varepsilon}}}\right]}}\right]. \tag{7.161}$$

The second term in Eq. (7.161) is dependent on the rate of change of the strain rate tensor and vanishes for proportional deformation. Substituting Eq. (7.161) back into the material law (7.159) results in the second approximation as a function of the first approximation

$$\overset{(2)}{\underset{\sim}{T}} = \overset{(1)}{\underset{\sim}{T}} - \frac{2}{9G\delta}\left(\overline{\sigma}^2\frac{1}{\dot{\overline{\varepsilon}}}\overset{\cdot}{\overline{\left[\frac{\underset{\sim}{D}}{\dot{\overline{\varepsilon}}}\right]}}\right). \tag{7.162}$$

The second term in Eq. (7.162) can be only calculated if a predefined load path is specified.

For small deformation, the approximation from the first iteration is very accurate as the deviation from proportionality remains small. However, improvement through the second iteration becomes impossible. At the elastic–plastic interface, the second term in (7.162) becomes very large because the hardening parameter δ becomes zero in the adopted material model. The effect of this term can be reduced in the numerical calculation, e.g., through multiplication with a factor η, defined by

$$\eta = \left(\frac{\overline{\varepsilon}_{\mathrm{p}}}{\overline{\varepsilon}} \right)^2 \tag{7.163}$$

so that

$$\overset{(2)}{\underset{\sim}{T}} = \overset{(1)}{\underset{\sim}{T}} - \frac{2}{9G\delta} \left(\overline{\sigma}^2 \frac{1}{\dot{\overline{\varepsilon}}} \overline{\left[\frac{\dot{\underset{\sim}{D}}}{\dot{\overline{\varepsilon}}} \right]} \right) .\eta. \tag{7.164}$$

The factor η ensures that for small plastic deformation in the elastic–plastic transition zone, the solution after the first iteration remains valid. The errors remain small when the deviation from the proportionality in the transition zone remains small. Through the second iteration, only very small changes occur in the values of the equivalent stress because the elastic and the plastic strain tensors are almost affine to each other, i.e., the deviations between the directions of their vectors (in a nine-dimensional space) are minimal.

7.4.4 Description of the Load Path

The three-dimensional state of stress in an elastic–plastic solid deformed according to the von Mises yield criterion depends on both the states of strain and strain rate. Therefore, adopting the von Mises yield criterion in the analysis makes it necessary to prescribe a load path in order to be able to determine the stress distribution through the cylinder.

The assumption of a predefined, e.g., exactly proportional relation between the applied pressure and axial force leads to an excessive numerical effort in calculating the nonhomogeneous state of stress. Consequently, an iterative, incremental solution becomes necessary. Every calculation step of such a solution corresponds to a deformation increment, followed by calculating the stress distribution. Numerical integration of the applied external loads is then necessary. It should be then checked whether the resulting load increments satisfy the assumed load path. The assumed deformation step must be then improved iteratively, until the corresponding load increments, which are obtained again through repeated numerical integration, coincide with the assumed loading path.

Instead of this approach, an iterative method is introduced, in which the description of the loading path is indirectly expressed through the prescription of the deformation path, which implies an implicit relation between the applied loads. Such an approach is only possible when assuming incompressible elastic and plastic deformation, and it does not represent an explicit definition of the loading path.

The control of a predefined loading path in an actual experimental test is also very problematic and implies usually strong approximations. If one or both load component reaches its maximum value, it becomes practically impossible to keep satisfying a specified relationship between the increments of the loading components. As an example of the suggested approach, a loading path that leads to a proportional loading path in the case of rigid-plastic closed thin-walled tubes is described below. For an arbitrarily chosen material point, having the radius $r = m$ in the current configuration, the dimensionless geometrical parameter γ is defined by

$$\gamma = \frac{m}{a}. \tag{7.165}$$

This material point is located at the position "$R = m_o$" in the reference (initial) configuration. It should be noted, however, that

$$\gamma_o = \frac{m_o}{a_o} \neq \gamma. \tag{7.166}$$

The integrated peripheral strain at the radius m can be obtained directly from (7.123) as

$$\varepsilon_{2m} = \varepsilon_2|_{(r=m)} = \ln\left[\left(1 - \frac{C_1}{m^2}\right)\mathrm{e}^{\varepsilon_3}\right]^{-\frac{1}{2}}. \tag{7.167}$$

As a simplification, that is still to be justified, a geometric relation is introduced.

$$\dot{\varepsilon}_3 = \frac{4\alpha}{3\mathrm{e}^{2\varepsilon_{2m}} - 2\alpha}\,\dot{\varepsilon}_{2m}, \tag{7.168}$$

$$\dot{\varepsilon}_{2m} = \left(\frac{3}{4\alpha}\mathrm{e}^{2\varepsilon_{2m}} - \frac{1}{2}\right)\dot{\varepsilon}_3. \tag{7.169}$$

Equations (7.168) and (7.169) are assumed to remain satisfied along the loading path at the selected position $r = m$, where the parameter α remains constant. The above-mentioned relationship leads to an exact proportional relationship between the applied internal pressure and the externally applied axial force for a rigid-plastic, closed thin-walled tube made of a von Mises material when m is selected to be the middle radius. In other cases, e.g., when analyzing thick-walled tubes or considering materials with large elastic deformation, the suggested loading path represents an approximation to the assumption of the proportional loading. The

selection of the radius m, where the relationships (7.167) to (7.169) apply, generally defines the load path to a particular load combination, and consequently the degree of its proximity to load proportionality. The influence of this stress history on the stress and strain distributions in the current state can be obtained by calculating the results for different values of the parameter m.

Integrating (7.168) results in

$$\varepsilon_3 = \ln\left(\frac{3\mathrm{e}^{2\varepsilon_{2m}} - 2\alpha}{3 - 2\alpha}\right) - 2\varepsilon_{2m}. \tag{7.170}$$

The initial conditions are utilized here to determine the integration constants. The value of α is directly obtained from

$$\alpha = \frac{3m^2}{2C_1}\left(1 - \mathrm{e}^{-\varepsilon_3}\right). \tag{7.171}$$

Consequently, the components of the velocity field are related by the expression

$$\dot{\varepsilon}_2 = \frac{m^2}{r^2}\left(\frac{\dot{\varepsilon}_3}{2} - \dot{\varepsilon}_m\right) - \frac{\dot{\varepsilon}_3}{2}. \tag{7.172}$$

7.4.5 Calculating the Current State

The second term of Eqs. (7.162) and (7.164), which is needed for the second iteration, is obtained after short calculation as

$$\overline{\left[\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}}\right]}^{\cdot} = -\frac{3}{8}\frac{\left[1 + 2\left(\frac{\dot{\varepsilon}_3}{\dot{\varepsilon}_2}\right)\right]}{\left[1 + \left(\frac{\dot{\varepsilon}_3}{\dot{\varepsilon}_2}\right) + \left(\frac{\dot{\varepsilon}_3}{\dot{\varepsilon}_2}\right)^2\right]^2}\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_2}\right)\overline{\left[\frac{\dot{\varepsilon}_3}{\dot{\varepsilon}_2}\right]}^{\cdot}, \tag{7.173}$$

$$\overline{\left[\frac{\dot{\varepsilon}_3}{\dot{\varepsilon}}\right]}^{\cdot} = -\frac{3}{8}\frac{\left[1 + 2\left(\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}_3}\right)\right]}{\left[1 + \left(\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}_3}\right) + \left(\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}_3}\right)^2\right]^2}\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_3}\right)\overline{\left[\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}_3}\right]}^{\cdot}. \tag{7.174}$$

The third component can be obtained from the incompressibility condition. Thus, a further material time derivative of the components of the velocity field is necessary, which is obtained after some calculations.

$$\left(\frac{\dot{\varepsilon}_2}{\dot{\varepsilon}_3}\right)^{\cdot} = A_2\dot{\varepsilon}_2 + A_3\dot{\varepsilon}_3, \tag{7.175}$$

with

$$A_2 = 2A_o M, \tag{7.176}$$

$$A_3 = A_o M \frac{\left(r^2 - C_1\right)}{r^2} - \frac{M}{r^2}\gamma^2, \tag{7.177}$$

with the abbreviations

$$A_o = 2\left(1 - \gamma_o^2\right) + \frac{m^2\gamma_o^2}{m^2 - C_1} - \frac{m^2}{r^2}\left(1 - \frac{r_o^2}{a_o^2}\right), \tag{7.178}$$

$$M = \frac{1}{2} + (\dot{\varepsilon}_2/\dot{\varepsilon}_3)_m. \tag{7.179}$$

The time derivative of the inverse of the left side of (7.175) is obtained from

$$\dot{\overline{(\dot{\varepsilon}_3/\dot{\varepsilon}_2)}} = (\dot{\varepsilon}_3/\dot{\varepsilon}_2)^2 \dot{\overline{(\dot{\varepsilon}_2/\dot{\varepsilon}_3)}}. \tag{7.180}$$

The desired expression for Eqs. (7.162) and (7.164) can be obtained by substituting Eq. (7.180) into Eqs. (7.173) and (7.174), so that the stress tensor and consequently the stress differences can be determined.

7.4.6 Equilibrium Condition

The stress components are obtained by utilizing the condition of radial equilibrium.

$$\frac{\partial \sigma_1^1}{\partial r} + \frac{1}{r}\left(\sigma_1^1 - \sigma_2^2\right) = 0. \tag{7.181}$$

Integration leads directly to

$$\sigma_1^1 = \int_a^r \left(\sigma_2^2 - \sigma_1^1\right)\frac{\mathrm{d}r}{r} - p. \tag{7.182}$$

In Eq. (7.182), p denotes the internal pressure. The integration constants are obtained from the boundary conditions at the inner and outer surfaces. The values of the other stress components can be calculated from the expressions of the stress differences.

7.4.7 External Loads

The external loads can be obtained from stress distribution in the tube wall. The expression for the internal pressure can be obtained from the boundary conditions in conjunction with Eq. (7.182).

$$p = \int_a^b \left(\sigma_2^2 - \sigma_1^1\right) \frac{\mathrm{d}r}{r}. \tag{7.183}$$

An expression for the axial force is obtained by integrating the axial stresses over the tube cross section.

$$N = 2\pi \int_a^b \sigma_3^3 r \, \mathrm{d}r. \tag{7.184}$$

Taking into consideration the axial effect of core pressure on the core area πa^2, the expression for the externally applied axial force F is obtained.

$$F = 2\pi \int_a^b \sigma_3^3 r \, \mathrm{d}r - \pi a^2 p. \tag{7.185}$$

Since the stresses can be obtained only numerically, a closed solution for calculating the external loads (7.183) and (7.185) is not possible. A numerical integration is therefore necessary.

7.5 Conclusion

Large elastic–plastic deformation of continuous media has been described and analyzed, with emphasis on the description of deformation in curvilinear coordinate systems. Large strain and strain rate tensors have been analyzed. Different definitions of the stress tensor have also been discussed. The application to the discussed theory is applied to the large elastic–plastic deformation of thick-walled cylinders loaded both radially and axially.

The tensor-based analysis is built on the continuum theory. Both tensor and component notations are adopted. The analysis assumes the material to be a homogeneous and isotropic continuous medium. Material-independent fundamentals are first discussed in detail, including strain, deformation, and velocity gradients. Different forms of the stress tensor in different space- or material-based coordinate systems are presented and discussed. The stress and strain definitions are used to derive the equilibrium condition and constitutive laws for elastic–plastic materials.

The presented definitions have also been used to develop a novel approach, recently published by the current author, for calculating the behavior of radially

and axially loaded thick-walled elastic–plastic cylinders with nonlinear hardening. An associative constitutive law is adopted, which is based on applying the von Mises yield criterion in association with the normality rule. The deformation of the cylinder is assumed to remain axially symmetric, i.e., deformation is independent of the axial position. The solution is capable of accurately providing continuous distribution of stress and strain gradients throughout the cylinder. It can, therefore, be used to investigate the instability and bifurcation of the corresponding thick-walled cylinders and to establish the basis for further research on the safety of pressurized thick-walled cylindrical structures.

This investigation sheds more light and provides valuable information for the safety design of extremely loaded pressure vessels. The presented approach also establishes the basis for further research in the stability investigations and bifurcation analysis of thick-walled pressure vessels.

References

1. Takla, M. (2018). Instability and axisymmetric bifurcation of elastic-plastic thick-walled cylindrical pressure vessels. *International Journal of Pressure Vessels and Piping, 59C*, 73–83.
2. Takla, M. (2018). Theoretical and numerical investigation of the elastic-plastic behavior of thick-walled cylinders. In L. Dai & R. N. Jazar (Eds.), *Nonlinear approaches in engineering application* (Vol. 5,. Chapter 13, pp. 381–402). Cham: Springer.
3. Lame, M. G. (1852). *Lecons sur la Theorie Mathematique de l'Elasticite des Corps Solides*. Paris: Imprimeur-Libraire.
4. Turner, L. B. (1909). The stresses in a thick hollow cylinder subjected to internal pressure. *Transactions of the Cambridge Philosophical Society, 21*, 377–396.
5. Belayev, N. M., & Sinitskij, A. K. (1938). Stresses and deformations in thick-walled cylinders on the plastic-elastic state of material. *Bull. Acad. Sci. U.R.S.S.*
6. Sokolovskij, V. V. (1955). *Theorie der Plastizität*. Berlin: VEB Verlag Technik.
7. MacGregor, C. W., Coffin, L. F. J., & Fisher, J. C. (1948). Partially plastic thick-walled tubes. *Journal of The Franklin Institute, 245*, 135–158.
8. Hodge, P. G., & White, G. N. (1950). A quantitative comparison of flow and deformation theories of plasticity. *Journal of Applied Mechanics, 17*, 180–184.
9. Gurtin, M. E. (1981). *An introduction to continuum mechanics*. New York: Academic Press.
10. Holzapfel, G. A. (2000). *Nonlinear solid mechanics*. New York: John Wiley.
11. Malvern, L. E. (1967). *Introduction to the mechanics of a continuous medium*. Upper Saddle River: Prentice Hall.
12. Ogden, R. W. (1997). *Non-linear elastic deformations*. New York: Dover.
13. Truesdell, C., & Toupin, R. (1965). The classical field theories. In *Handbuch der Physik, Bd. III* (Vol. 1). Berlin: Springer.
14. Truesdell, C. (1966). *The elements of continuum mechanics*. Berlin: Springer.
15. Celep, Z., (1971). *Beitrag zur Theorie dicker Kreiszylinder- und Kugelschalen unter innerem Druck*. Diss. TU Hannover.
16. Fischer, B. (1977). *Zur zyklischen, elastoplastischen Beanspruchung eines ickwandigen Zylinders bei endlichen Verzerrungen*. Mitteilungen aus dem Institut für Mechanik Nr. 9, Ruhr-Universität Bochum.
17. Oeynhausen, H. (1981). *Verzweigungslasten elastoplastisch deformierter, dickwandiger Kreiszylinder unter Innendruck und Axialkraft*. Mitt. Institut f. Mechanik, No. 29, RU Bochum.

18. Imaninejad, M., & Subhash, G. (2005). Proportional loading of thick-walled cylinders. *International Journal of Pressure Vessels and Piping, 82*, 129–135.
19. Bruhns, O. T. (1974). *Einige Bemerkungnen zur Bestimmung von Verzweigungslasten elasto-plastisch derformierter Kontinua*. Mitteilung Nr. 74-9 des Inst. f. konstr. Ing.bau, Ruhr-Univ. Bochum.
20. Lehmann, T. (1972). Einige Bemerkungen zu einer allgemeinen Klasse von Stoffgesetzen für große elasto-plastische Formänderungen. *Ingenieur Archiv, 41*, 297–310.
21. Lehmann, T. (1972). Zu einigen nicht-linearen Stoffgesetzen für plastische Formänderungen. *Rheologica Acta, 11*, 4–12.
22. Takla, M. (2018). Bifurcation of elastic-plastic thick-walled cylindrical structures. *International Journal of Mechanical Sciences, 141C*, 303–315.
23. Takla, M. (2018). Insight into elastic-plastic bifurcation of pressurized cylinders: Transition between bulging and necking; the line of catastrophic failure. *International Journal of Mechanical Sciences, 148*, 73–83.
24. Takla, M. (2019). Non-symmetric bifurcation and collapse of elastic-plastic thick-walled cylinders under combined radial and axial loading. *Marine Structures, 64*(2019), 246–262.
25. Chadwick, P. (1999). *Continuum Mechanics* (2nd ed.). Mineola: Dover.
26. Lehmann, T. (1972). On large elastic-plastic deformations. In A. Sawczuk (Ed.), *Foundations of plasticity*. Leyden: Noordhoff International Publishing.
27. Lehmann, T. (1975). *Elemente der Mechanik, Bd. II*. Braunschweig: Vieweg.
28. Bruhns, O. T. (1978). Some remarks on the stability of extremely loaded pressure vessels. Proceedings of the International Symposium. Mechanics of Inelastic Media and Structures, Warsaw.

Chapter 8
Big Data Modeling Approaches for Engineering Applications

Bryn Crawford, Hamid Khayyam, Abbas S. Milani, and Reza N. Jazar

8.1 Introduction

Engineering is intrinsically a field in which the application of science and mathematics is utilized to solve problems in pursuit of the design, operation, maintenance, and other faculties of systems in complex systems. Many of these systems contain nonlinear interactions and as such, require tools of varying robustness and power to describe them. Forecasting of future states or designing such systems is very costly, time consuming, and computationally intensive, due to finite project timelines and technical constraints within industry. Modeling can be effectively employed as an inexpensive and powerful tool to address these issues in pursuit of engineering objectives. Many computation methods are used to achieve the modeling of such systems. This technique can lower the cost of achieving desired goals, by reducing the number of experiments and even increase safety or reliability, by forecasting the events. This can be achieved with the results of laboratory tests or industrial data [1, 2].

Broadly speaking, there are two main categories of modeling [3]: deterministic and heuristic. Deterministic models are based on physical systems and are based on prior knowledge, giving insight into those physical systems, which are analytical and require a symbolic representation of the system in terms of predictive inputs

B. Crawford · A. S. Milani (✉)
School of Engineering, University of British Columbia (UBC), Kelowna, BC, Canada
e-mail: abbas.milani@ubc.ca

H. Khayyam
School of Engineering, RMIT University, Melbourne, VIC, Australia

R. N. Jazar
Xiamen University of Technology, Xiamen, China

School of Engineering, RMIT University, Bundoora, VIC, Australia

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_8

that determine the desired system output. Heuristic modeling is applied to complex systems, to which applying deterministic modeling approaches would require infeasible time, resources, and data volumes. Although the development of heuristic models may require less effort and is simpler, there are additional challenges in effectively using them, such as the size and quality of a dataset, combined with the complexity of the system being described through the modeling process. Many tasks may be achieved using modeling, including regression, classification, and clustering. With the advent of larger datasets being available in the modern day, this has offered a paradigm shift in the way that modeling approaches are examined and applied.

Modern advances in information and communication technologies have provided tools for organizations to archive and utilize unprecedented amounts of data of various types. This is the case for many different industries, including healthcare, manufacturing, satellite imaging, finance, and social media [4]. As an example of the volume of data available to modern corporations that have deployed the supporting infrastructure, Hewlett-Packard (HP) generates over one trillion events per day [5]. For organizations such as this, it is expected that daily data production will exceed 2.5 exabyte by 2020, which is a 44-fold increase from those seen in 2010 [6], illustrating the rapid changes in this area, which are occurring across the industrial landscape. "Big data" is a relatively amorphous term used to describe the rise in data volumes that are difficult to capture, store, manage, process, and analyze, using traditional database methods and tools [7, 8]. The new reality of big data has and shall continue to have profound implications on modeling, as new and highly valuable information can be extracted for decision-making.

This ever-increasing volume of data has led to a realization that we have entered the era of "big data," which is considered to be a major catalyst for innovation and an increase in competitiveness and productivity [9]. For example, Lund [10] states that due to increases in operational efficiency, retail and manufacturing industries worldwide are expected to generate an increased gross domestic product (GDP) of $325 billion. Similarly, healthcare and government services are expected to make comparable productivity gains in the order of $285 billion, by 2020. It was also projected that big data may increase healthcare productivity by more than $300 billion for the US government and $250 billion in the administration sector for the European Union [9]. There is a wide range of applications for big data, such as identifying irregular and suspicious activities, using financial transactions, log files, network traffic analysis, and other data sources [5]. Benefits of big data have been demonstrated beyond the field of business, including insurance, politics, defense, sciences, engineering, and many other fields. Additionally, a survey by the Economist Intelligence Unit illustrated that organizations that applied big data analytics have improved in their performance over the past 3 years by 26%, and they expect it will improve by 41% over the next 3 years. The ability of organizations to analyze big data will become a key basis for increased competition and innovation, underpinning new waves of productivity growth, and consumer surplus, on the proviso that the right policies and enablers are in place [9]. Hence, it is critical to

understand the potential challenges of big datasets, in selecting strategies to process and model the data, to obtain that direct business value.

As has been highlighted, there are many opportunities available in the use of big data, along with many challenges. Some of the challenges, and subsequent methods of modeling to extract value from big data, can be characterized under four primary banners. These are known as "the four V's" and are widely cited in the literature [4, 11]. They are volume, variety, velocity, and veracity.

Volume, often considered to be the primary characteristic of big data, refers to the absolute size of the dataset being considered. For example, 10 TB of data can be generated per 30 min from the operation of a single jet engine [12], sourced from many sensors (e.g., strain, temperature, pressure). Hence, for a commercial airline, many petabytes of data can be generated each day and considering that in the order of 25,000 flights may be operated per year, this data volume quickly becomes an issue. Classical computing architectures, modern single-unit hardware performance, data platforms, and modeling algorithms all find challenge in data of such great volumes.

Variety in big datasets also provides additional challenges. Given the great diversity of data sources, including sensors, images, video feeds, financial transactions, location data, text documents and others, reconciling these sources into unified modeling strategies is not straightforward [13]. When considering so many different types of data, big data modeling strategies typically address three distinct types of data: structures data, semi-structured data, and unstructured data. Structured data is often numerical in nature and can be easily categorized in a relational database. Unstructured data can be presented as a text file (e.g., PDF file) which is informal and does not have defined structure. Semi-structured data is largely unstructured, yet does contain some tags to separate semantic elements and has the capacity to enforce some hierarchies in the data [14], such as e-mails.

Velocity describes the speed of the dataset being used, considering actions of creation, transmission, processing, and ingestion [14]. This factor can be thought of as the challenge of training and evaluating a predictive model (which typically takes significant time with the volumes associated with big data), in the context of new and relevant data constantly being made available. IBM [15] illustrates the velocity of big data with case studies of the analysis of 500 million daily call detail records in real time to predict customer call synopses for marketing and advertising, or evaluating five million daily commercial trade events in an attempt to identify fraud.

Veracity refers to the credibility, or inversely the uncertainty, of data and its suitability of use in predictive modeling activities [16]. Due to the lack of control of many data collected in big data activities, such as user reviews on social media networks, or third-party sensor data, the accuracy and completeness of the dataset must be considered [4]. Not only is the impact of this factor in modeling and analysis difficult to quantify, but the impact on subsequent decision-making presents additional and far-reaching challenges. Given the explosion of the number of data sources available, achieving trust in the data sources parallel to this is becoming increasingly important [15].

Each application of big data has a different blend of challenges associated with volume, variety, velocity, and veracity. Highly varied data, combined with the context of different tasks to be achieved for each application, it can be seen that a broad range of solutions are needed to achieve the goals of modeling and decision-making. Machine learning (ML) methods present a capability for modeling highly nonlinear, complex systems. In pursuit of modeling such complex systems, ML is often seen as the most effective and only viable approach. ML algorithms have been used in a wide range of engineering applications, such as plant control systems [17], root cause analysis [18], nondestructive testing [19], material modeling [20], risk analysis [21], robotics [22], structural health monitoring [23], and many others, which demand the inherent learning and nonlinear capabilities of these models. Often, a perceived drawback of these methods is that large datasets are required to perform successful model training and converge toward solutions with acceptably low errors. Big data of course counteracts this notion, where very large volumes for training become accessible. However, while the availability of big data may seem like a solution, often the explosion of this data introduces new challenges to be considered. In the context of the challenges of volume, variety, velocity, and veracity, scalability becomes an additional logistical challenge for applying ML models for extracting value [24]. For example, or instance, most traditional machine learning algorithms are designed to be trained and implemented on datasets to be completely loaded into memory [25]. For big data with natures previously described, this approach is not feasible.

In the context of machine learning, new approaches are required for modeling big data. Broadly, there are six categories of approaches used, which can be used individually or in conjunction with each other, to generally address these issues of scalability. These are applied to big data problems to allow for the feasible application of ML for the effective modeling and capturing of nonlinear features within the complex data, for more powerful and usable predictions. This chapter presents an overview of some classical ML methods that have proven successful in modeling complex systems, especially in engineering. Following this, a summary of each of the six categories of approaches for modeling of big data is presented, including the major methods of each. This chapter aims to provide a summary of these methods, such that readers can formulate their own strategies for modeling big data.

In this chapter, a review of classical machine learning methods will be provided in Sect. 8.2, including a selection of clustering, classification, and regression methods. Sections 8.3–8.8 will detail the six approaches for applying scalable machine learning solutions to big data. Specifically, Sect. 8.3 will present Representation Learning methods for data reduction and other uses; Sect. 8.4 will focus on deep learning for capturing highly nonlinear behavior; Sect. 8.5 will present distributed and parallel learning that addresses the challenges of volumes associated with big data; Sect. 8.6 will provide detail on transfer learning for cross-domain and cross-task learning activities; Sect. 8.7 will present active learning for the efficient labeling of data points in supervised learning contexts; and Sect. 8.8 will present kernel-based learning for leveraging higher expressive power in modeling activities. Concluding remarks are provided in the final section.

8.2 Classical Machine Learning Methods

In the context of processing big data, machine learning (ML) is an approach to construct models using big data, for generating predictions and supporting decision-making activities. ML models are able to capture a very high degree of nonlinearity associated with complex systems, oftentimes resulting in black box solutions. As big data tends to have very large and disparate feature spaces, ML has found great success in modeling in this area. There are many different approaches to machine learning, depending on scope of problem, nature of dataset, desired outcomes, and other factors. Tasks to be completed by modeling include regression, classification, and clustering. Regression as a task involves the mapping of inputs to a continuous output(s), classification involves mapping inputs to a discreet or class-based output(s), and clustering identifies and leverages spatial relationships within a dataset to discover patterns or perform data reduction. The modeling task selected depends on the task to be performed on the real system being modeled. For example, in the context of machine learning-based computer vision, image segmentation that identifies different objects of regions in an image, is a classification task, whereby the membership of each pixel to a given class is determined by the learning-driven model. Similarly, for a modeling exercise seeking to predict the continuously mapped outcomes of a manufacturing process (e.g., temperature, viscosity) using input variable values (e.g., flow rate, concentration), regression is the task to be performed. For users seeking to discover the degree of correlation or other descriptors between variables in a system, such as complex environmental systems, clustering is the task to be implemented.

Generally, there are two major categories of machine learning: supervised learning and unsupervised learning (however, semi-supervised learning is also considered a smaller category). Supervised learning involves preparing a model through a training process, in which it is required to make predictions and is corrected when those predictions are wrong; the training process continues until the model achieves a desired level of accuracy on the training data. Unsupervised learning typically focuses on exploratory analysis and dimensionality reduction methods, such clustering used in Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), k-Means, deep learning, for understanding the structure of data. Supervised learning typically performed in the context of classification (mapping input feature spaces to discreet output labels) or regression (mapping input feature spaces to continuous outputs). Examples of supervised learning algorithms are ensemble methods, artificial neural networks, regularization methods, rule systems, explicit regression techniques, Bayesian methods, decision trees, clustering, and instance-based methods. However, it should be noted that many of the methods mentioned can be applied to both classification and regression problems, whereby their categorization is driven by their ubiquity in industry. Some of these methods have been presented below, in the context of clustering, classification, and regression as target activities.

8.2.1 Clustering

Clustering is a common machine learning method typically used for exploratory analyses of very large datasets with a high dimensionality of the input space, commonly seen in the fields of genetics, finance, biological system, and others. This can be in support of dimensionality reduction activities for further analysis and processing, identifying spatial patterns for appraisal by human users, and highlighting major contributing factors toward trends or hidden patterns within the data. They are particularly important in the studying highly complex systems, where a numerical or analytical model is not necessarily possible or feasible, such as a mix of continuous and discontinuous data, furry or incomplete data, and other challenges.

Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised learning procedure to achieve dimensionality reduction in very large datasets through feature extraction. Compared to dimensionality reduction techniques that employ feature reduction, PCA encodes the entire q-dimensional input space into a set of principal component vectors, which convolutes and captures the entirety of the information present in the dataset. Then, a subset of k vectors is selected to preserve the majority of information according to a variance maximization criterion.

The generation of a set of principal vectors $P \subset \mathbb{R}^q$ enforces that a core assumption of the linear model used for analysis is satisfied, which requires that all input variables to be independent of each other. This is a major advantage for users who cannot otherwise satisfy this condition for linear modeling approaches. Similarly, PCA has a tendency to minimize the loss of information typical in the dimensionality reduction process, depending on the signal and noise models used [26]. Conversely, a disadvantage of PCA is that the input independent variables to the model are less interpretable for users.

The process for applying PCA to a dataset is to create a column-based centered and standardized matrix from the input space $X \rightarrow Z$, where the centralizing operation is:

$$Z_{i,j} = X_{i,j} - \frac{\sum_{j=1}^{q} X_j}{i} \tag{8.1}$$

where i is the row index, j is the column index, and q is the dimensionality of the input space (number of input variables). This matrix Z is used to evaluate the symmetric, positive semidefinite covariance matrix and decompose it into the following:

$$Z^{\mathrm{T}}Z = PDP^{-1} \tag{8.2}$$

where P encodes the eigenvectors and D encodes the eigenvalues of the centered and standardizes dataset. Using the resulting sorted matrix $Z^{*} = ZP^{*}$, based on the magnitude of the corresponding eigenvalues (a measure of their contribution) then allows for analysis using the proportion of variance method, to determine a thresholding characteristic as a trade-off between the amount of information retained and the resulting k-dimensions wished to be used for further analysis and modeling.

A vector of weighting factors $w_{(k)} = (w_1, \ \ldots \ , w_p)_{(k)}$ is applied to the retained eigenvectors in order to maximize the variance of the model. This is achieved by starting with applying the following procedure to the first component, which must satisfy:

$$w_{(1)} = \underset{\|w\| = 1}{\arg\max} \left\{ \sum_i \left(x_{(i)}.w \right)^2 \right\} \tag{8.3}$$

The subsequent kth component is determined by first subtracting the first $k - 1$ component above from matrix Z:

$$\hat{Z}_k = Z - \sum_s^{k-1} Z w_s w_s{}^{\mathrm{T}} \tag{8.4}$$

The weights vector that yields the maximum variance is given by:

$$w_{(k)} = \underset{\|w\| = 1}{\arg\max} \left\{ \left\| \hat{Z}_k w \right\|^2 \right\} \tag{8.5}$$

PCA has found extensive use in many fields, including biological systems, environmental modeling, composite materials, genetics, information technology, and others, in which dimensionality reduction of datasets with great volume is necessary.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Density-based spatial clustering of applications with noise (DBSCAN) is a commonly implemented clustering algorithm that has found widespread adoption across many industries. The task of clustering works on the premise of identifying regions with a high density of observations as likely candidates for applying discreet cluster membership, while regions of comparatively sparsely distributed observations are considered outliers and treated as noise [27]. The implementation of DBSCAN is simple and robust, yet without employing strategies to assist with volume scalability,

model complexity can become infeasible. As the algorithm runtime is mostly sensitive to the querying of each point, without an accelerated indexing structure or if the data is degenerate, the runtime complexity is of order $O(n^2)$.

There are two primary parameters that define the DBSCAN model, which are ε (the maximum radius of the neighborhood from cluster core points p_k) and minPts (the minimum number of points within a neighborhood to define dense-region cluster). For iterative purposes in selecting a value of minPts $\geq D + 1$, where D is the dimensionality of the input space, is common to start with. A disadvantage of the algorithm is that it is sensitive to the combination of minPts, ε selected. Using these parameters, all points in the dataset are labeled as either core points, reachable points, or outlier points. Figure 8.1 illustrates the application of DBSCAN in $\mathbb{R}^2$, with a single cluster identified from a dataset composed of the three classes of points:

- **Core points**: For a maximum neighborhood radius ε, there must be |minPts| locally for the point to be labeled as a core point (depicted in blue in Fig. 8.1).
- **Reachable points**: These points are within radius ε of at least one other point, but do not satisfy the |minPts| constraint (depicted in orange in Fig. 8.1).
- **Outliers**: There are no other points in the dataset spatially within radius ε and implicitly the constraint |minPts| is also not satisfied (depicted in red in Fig. 8.1).

DBSCAN has found use in fields, such as astronomy, biological sciences, signal processing, knowledge discovery, protein classification, and others.

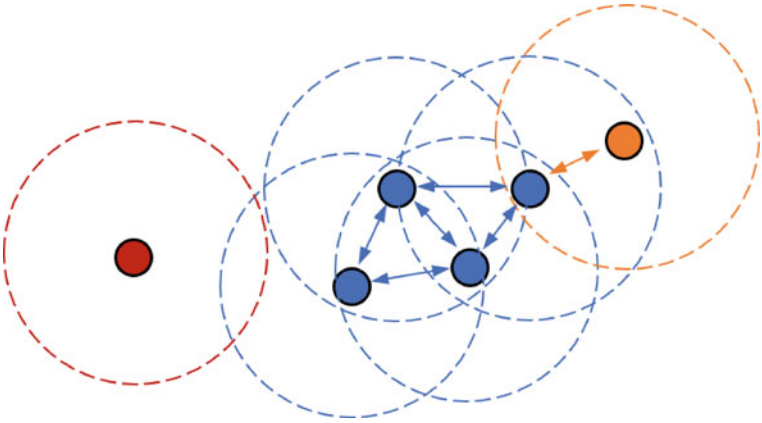


Fig. 8.1 A set of core points, reachable points, and an outlier point in an $\mathbb{R}^2$ dataset, labeled based on the model parameters ε (radius of the circles concentric to dataset points) and |minPts| (used to label points as either one of the three types, core points, reachable points, or outliers)

8.2.2 Classification

Classification is a very commonly employed machine learning method, used for the prediction of categorical class labels. The categories can be discrete or unordered in nature, where the models apply predictions on the form of these labels, to new and unlabeled examples, based upon observations in the training set. Hence, models (also known as classifiers) are constructed and trained to perform this classification task [28]. The aim of a classification activity is to predict the class label y' for an unseen pattern in the complex system x', by constructing a model f as a predictor. For a paired set of input space parameters and output class labels, given by $\{(x_1, y_1) \ \ldots \ (x_N, y_N)\}$, the applied learning algorithm determines q-dimensional hidden patterns through the training process and applies predictions to new data in a test set. The dimensionality of the input and output spaces are given by $X = \{x\}_{i=1}^{N} \subset R^q$ and $Y = \{y\}_{i=1}^{N} \subset R$, respectively.

Support Vector Machines

Support Vector Machines (SVM) is a relatively new supervised learning algorithm developed by Vapnik [29]. Generally, SVM models are categorized into four classes, identified by the task and type of error function used. These are Classification SVM Type 1 (C-SVM classification), Classification SVM Type 2 (ν-SVM classification), Regression SVM Type 1 (ε-SVM regression), and Regression SVM Type 2 (ν-SVM regression; [30]).

SVM exhibits good capabilities for fitting complex and nonlinear datasets in the context of learning algorithms. The core functionality of the algorithm is to define decision boundaries within the dataspace, with the objective of minimizing structural risk within the dataspace. This approach avoids global minimum problems and global minimum points can successfully be reached. SVM can be used for both classification and regression, although the latter is referred to Support Vector Regression (SVR). SVM generates hyperplanes within the dataspace that form the boundaries between different applicable class labels to member data points [31]. The optimal hyperplanes are determined by minimizing an objective error function, through an iterative training algorithm. Conversely, this can be described as maximizing the distance between the hyperplanes and the training points as members of the various classes. Figure 8.2 represents this in $\mathbb{R}^2$ space for visual simplicity.

An SVM hyperplane boundary is defined by a q-dimensional vector ($w \in \mathbb{R}^q$) and a bias (spatial offset) b, which constructs the hyperplane (w, b) within $\mathbb{R}^q$ as defined below:

$$W^{\mathrm{T}} Z + b = 0 \quad Z \in \mathbb{R}^q \tag{8.6}$$

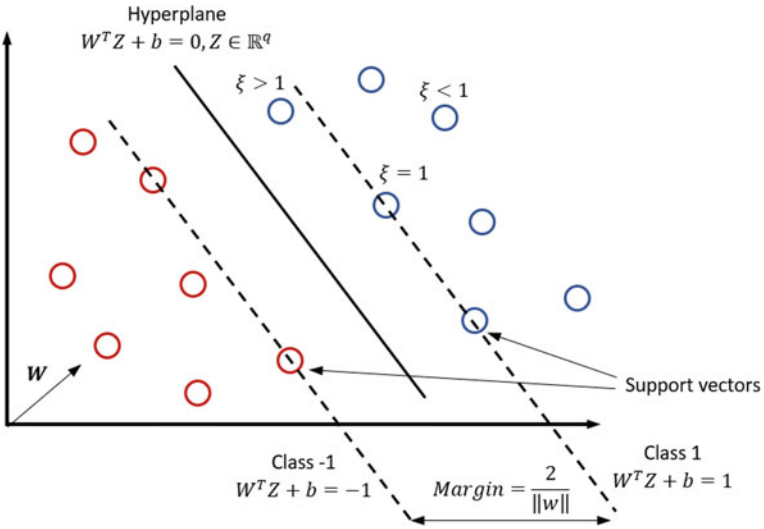


Fig. 8.2 A separating hyperplane placed between regions of training points from two classes, with the supporting vectors for each. The margin between the points and the plane is maximized, to reduce the risk of misclassification in predictive modeling

As these hyperplanes are in essence the decision boundaries for classification, this space $\mathbb{R}^q$ is divided into two regions $\hat{\mathbf{z}}_+$ and $\hat{\mathbf{z}}_-$ by (w, b) that defines the regions for classification. For binary classes, these two regions are defined according to the below criteria. Additionally, Fig. 8.2 demonstrates how a separating hyperplane, the location of which is determined by maximizing the margin between training points of the two classes, minimizes structural error in the classification exercise.

$$\hat{\mathbf{z}}_+ = \left\{ Z \in \mathbb{R}^q : W^{\mathrm{T}} Z + b < 0 \right\} \tag{8.7}$$

$$\hat{\mathbf{z}}_- = \left\{ Z \in \mathbb{R}^q : W^{\mathrm{T}} Z + b > 0 \right\} \tag{8.8}$$

With this model, only linear plane boundaries are generated. As many engineering problems consider highly nonlinear, complex systems, this alone may be inadequate to generate an accurate predictive model. To address this shortcoming, nonlinear transformations can be applied to the decision space ($Z = \phi(x)$: $\mathbb{R}^s \rightarrow \mathbb{R}^q$), where curved boundaries in s-dimensional space can be applied instead. A decision value, $d(x)$, is then used to perform the separation of regions within $\mathbb{R}^s$:

$$d(x) = T W \phi(x) + b \tag{8.9}$$

Using the nonlinear transformation function $\phi(x)$, the two classes X_- and X_+ can be predicted via modification of Eqs. 8.7 and 8.8:

$$\hat{C}_+ = \left\{ Z \in \mathbb{R}^q : W^{\mathrm{T}}\phi(x) + b < 0 \right\} \tag{8.10}$$

$$\hat{C}_- = \left\{ Z \in \mathbb{R}^q : W^{\mathrm{T}}\phi(x) + b > 0 \right\} \tag{8.11}$$

In order to have high-quality predictive model, the selected SVM parameters must provide a good approximation of the boundary space, B. This boundary space separates the previously defined sets, $\hat{C}_+$ and $\hat{C}_-$, defined by application of the decision value $d(x)$ to the nonlinear transformation. The SVM boundary B can approximate the limit hypersurface (by obtaining the global minimum of the problem):

$$\hat{B} = \{z : dz = 0\} \tag{8.12}$$

The optimal values of $\mathbf{w}$ and b in pursuit of this problem are determined by using a training set of points D_N, which contains the associated input space values x_i and class labels c_i:

$$D = \{(x_i, c_i)\} \text{ with } c_i = c_i(x), i = 1, 2, \ldots, N \tag{8.13}$$

Ultimately, the optimization problem to be solved is as follows, where the Euclidian distance between the training points and the constructed hyperplane is to be minimized:

$$\arg\min_{w,b,\xi} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{8.14}$$

Subject to the constraint $c_i(w^{\mathrm{T}}\phi(x_i) + b) \geq 1 - \xi_i = \xi_i \geq 0$, where $i = 1, \ldots, N$.

The minimum value of $\mathbf{w}$ is described as the classification margin. This is the maximum distance between the generated boundaries from the model, and the separated classes of training points, which, as previously described, provides the minimized structural risk. The classification error from the training set applied to the model is described by $x_i \in \hat{C}_-$ and $x_i \notin C_-$, or $x_i \in \hat{C}_+$ and $x_i \notin C_+$ (where C and $\hat{C}$ are the class spaces generated without and with the decision value $d(x)$, respectively). This is given by the nonzero value of $\mathbf{w}$ and the second term in Eq. 8.12. As can be seen, this yields a scenario of trading off between two objectives: maximizing the classification margin and minimizing the misclassification error. This leads to the selection of model hyperparameter C being critical in finding an optimal solution in trading off between these objectives. Generally, choosing a low

value of C leads to a lower misclassification error (as it is the coefficient to that term) and increases the classification margin. Conversely, a high value of C will result in the opposite conditions. Mathematically, this is relevant in the context of trading off between high bias and high variance within the model and is reflected in the application of test or cross-validation sets.

A weighted sum can be applied to the transformation $\phi(x_i)$ of the training set, to determine the optimal value of **w** and provides greater user control over the model, given by:

$$w = \sum_{i=1}^{N} c_i \alpha_i \phi\left(x_i\right) \tag{8.15}$$

The magnitude of the Lagrange multipliers $\alpha_i \geq 0$, used to find the local minimum of the function subject to the inequality constraint shown associated with Eq. 8.14, defines the weighting coefficients for each term shown above. By combining the above equation and the decision value $d(x)$ (Eq. 8.9), the kernel-based SVM is as follows:

$$d(x) = \sum_{i=1}^{N} c_i \alpha_i k\left(x, x_i\right) + b \tag{8.16}$$

where the kernel function is $k(x, y) = \phi T(x)\phi(y)$ and the sign of $c_i = \pm 1$.

Generally, the Lagrange multipliers $\alpha_i > 0$ hold true for only a subset of the N training data points, which are close to the boundary and require this approach to solve the optimization problem. This subset, denoted as $V \subset D$, is support vectors. As the summation of the optimization problem subject to constraint (Eq. 8.14) is limited to less than N terms, the definition is:

$$d(x) = \sum\nolimits_{(x_i c_i) \in V} c_i \alpha_i k\left(x, x_i\right) + b \tag{8.17}$$

In this context, selection of the value of C has a significant impact on the size of support vector set. This is because as C trades off between minimizing the classification errors and maximizing the classification margin, there will be cases where certain values of C results in a greater number of points being close enough to the boundary and requires the Lagrange multipliers, increasing the size of V. This means that a high value of C yields a higher-dimensional V and hence boundary with greater curvature, whereas a lower value of C lowers the number of support vectors and effectively smooths the surface of the decision boundary. It should be noted that many different kernel functions can be used for $k(x, y)$, which is another hyperparameter for user control of the model. The most commonly used kernel function is the radial basis function kernel based on Euclidian distance, defined as:

$$k\left(x, y\right) = \mathrm{e}^{-\gamma \|x-y\|^2} \tag{8.18}$$

SVM has been used in many fields of engineering applications, including image classification, object recognition, fraud detection, anomaly detection, and text categorization [32].

Decision Tree

Decision trees use a tree-like graphical model, similar in structure to a flowchart, for exploring a decision space with associated consequences. Similar to many of the machine learning models presented in this chapter, they can be used for classification or regression tasks, but in this case, they are more widely applied to the former. The tree model is constructed of many nodes and branches, which embody the flow of class labels and consequences. There are three types of nodes: decision nodes, chance nodes, and end nodes. Ultimately, a decision tree is used as a decision support tool that yields expected values (or utilities) of competing alternatives within the decision space. The three types of nodes build the model with respect to assigning class labels to input vector spaces and probabilities for associated outcomes, which are generally linearly summed to calculate the expected values. The branches in the model as results from the nodes then act as inputs to subsequent child nodes. This computationally simple approach allows for very large dimensional data to be processed using decision trees.

When constructing a decision tree, the root node is selected as the input attribute that in its own right provides the best classification result for the dataset, where the second-best predictive attribute is used to split the data into two or more subcategories. Subsequent nodes follow this same pattern, where remaining attributes are selected to classify and split the data. With different combinations and permutations to this process, the tree grows further from the root node, generating increasing numbers of branches at each step. The algorithms used for generating decision trees select the attribute used at each node, based on classification potential. A popular example for classification of decision trees is the chi-squared automatic interaction detection (CHAID) algorithm, which uses the chi-squared statistical test to determine if a partition is statistically significant when building the tree. Approaches such as this aim to remove branches that act as outliers or noise, which would only result in an overall poorer estimation of the expected value. This is performed to improve the accuracy of correct classifications on new data that the model is not trained on. These methods are collectively referred to as pruning, where branches are statistically tested and removed [28]. A visual example of a simple decision tree is provided in Fig. 8.3.

When constructing a decision tree algorithm, three key inputs are needed. The first is the training dataset (referred to as the data partition, D), which includes the input space values and associated class labels for the training process. The attribute list, containing the candidate attribute list, is also required. Finally, an attribute selection method must be defined. This is the method by which the optimal split of the dataset into separate classes is calculated and typically is a statistical, nonparametric method. The use of the statistical descriptor Gini index

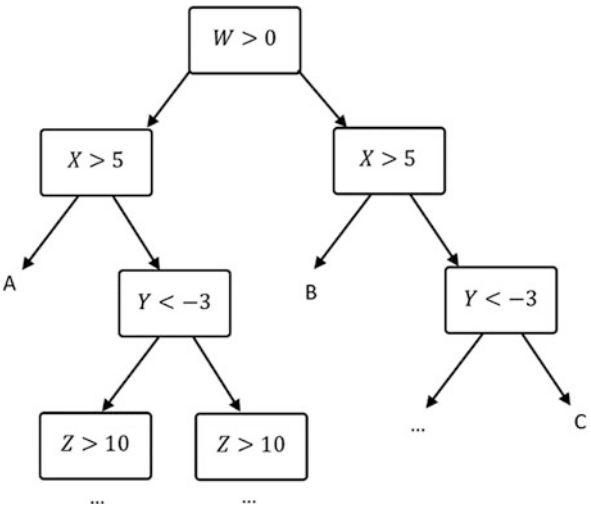


Fig. 8.3 Decision trees progressively partition the dataspace through decision nodes (leaves) and are used to calculate the expected value or utility from a set of given decisions

is a commonly used method, which results in binary classification decision trees. It is defined as double the integral between the model receiver operating characteristic (ROC) curve and diagonal ($y = x$; the decision boundary that results in a "50/50" correct classification scenario).

Decision tree algorithms have been successfully used in many different applications, including financial analysis, project risk management, manufacturing engineering, astronomy, and many others [33].

Bayesian Approaches

Bayesian approaches to classification tasks construct probabilistic models using training datasets. The learned distribution of the factors, based on probability and set theory, are leveraged to assign probabilities to new data examples for classification. The two most commonly used approaches are Naïve Bayes and Bayesian Belief Network, which are predicated on the same general probabilistic model. The latter generally has a greater degree of model fidelity, as the model builder has the ability to encode their own expertise into the model through knowledge engineering of the causal relationships [34].

Naïve Bayesian

Naïve Bayesian classification is a very simple and computationally efficient classification method that has been widely used in many industries, including engineering applications. The method is referred to as "naïve," as the causality model assumes

only direct influence between inputs and outputs and does not embed inter-factor causality.

To use the Bayesian approach for classification tasks applied to a sample of data without any labels, X, a hypothesis H_0 is made such that X is a member of a class, C. The probabilistic task in this exercise is to determine the posterior probability $P(H|X)$, where the confidence that X is associated with H_0 is given. To calculate $P(H|X)$ using conditional probability theory, $P(H)$, $P(X)$, and $P(X|H)$ are used as components in the Bayesian theorem as follows [35]:

$$P\left(H|X\right) = \frac{P\left(X|H\right) \cdot P(H)}{P(X)} \tag{8.19}$$

In this training exercise, we consider a set of m samples $S = \{S_1, S_2, \ldots, S_m\}$, where each sample itself is an n-dimensional feature vector $\{X_1, X_2, \ldots, X_n\}$. Each given value within the feature vector, X_i, associated with attributes $\{A_1, A_2, \ldots, A_n\}$, respectively. A set of k classes is similarly defined for each sample: $C_1, C_2, \ldots, C_k$. The aim of the model is to predict the class for a new and unlabeled sample S_j, given its n-dimensional feature vector. In order to achieve this, $P(C_i|X)$, $i = 1, 2, \ldots, k$ is calculated for each class and the highest conditional probability is chosen as the final result. Equation 8.19 is hence modified to incorporate class labels as the hypothesis of distribution membership and is given by:

$$P\left(C_i|X\right) = \frac{P\left(x|C_i\right) \cdot P\left(C_i\right)}{P(X)} \tag{8.20}$$

To maximize the value of $P(C_i|X)$, the numerator on the right-hand side of the equation $P(X|C_i) \cdot P(C_i)$ should be similarly maximized, as $P(X)$ is a constant value for all classes. $P(C_i)$ is calculated as follows:

$$P\left(C_i\right) = \frac{\text{cardinality of class } C_i}{m} \tag{8.21}$$

where m is the total number of training samples used. As conditional independence is assumed between all attributes, $P(C_i|X)$ can be explicitly calculated as follows:

$$P\left(C_i|X\right) = \prod_{t=1}^{n} P\left(X_t|C_i\right) \tag{8.22}$$

where X_t are attribute values for the sample data X. The conditional probabilities $P(X_t|C_i)$ can be estimated using the training dataset, as a direct calculation using the attributes column vector. Different density function models can be assumed to calculate $P(X_t|C_i)$, such as Gaussian, Gamma, Lognormal, and Poisson distributions. The relevant shape and hyperparameters of these models are estimated using the training data. Gaussian distributions are most commonly assumed for $P(X_t|C_i)$, which is calculated as:

$$P\left(X_t|C_i\right) = \frac{1}{\sqrt{2\pi\sigma_i{}^2}} \mathrm{e}^{-\frac{\left(x_t-\mu_i\right)^2}{2\sigma_i{}^2}} \tag{8.23}$$

where μ is the mean of the data ($-\infty < \mu < +\infty$) and σ is the standard deviation ($\sigma > 0$) [36].

Naïve Bayes models have found application in a variety of fields, such as root cause analysis, combined diagnostic and prognostic queries in manufacturing systems, text classification, anomaly and fraud detection, and many others within the areas of engineering and beyond.

Bayesian Belief Network (BBN)

Bayesian Belief Networks (BBNs) are an extension of Naïve Bayesian networks, where a directed acyclic graph (DAG) is used to connect the nodes of the network, denoting the probabilistic conditional dependencies between them [37]. Hence, there is model structure encoded in BBNs that allows for more accurate representation of the data. This structure can either be manually built by an expert user, or can be statistically inferred through the training dataset, by use of maximum expected likelihood algorithms, including Tree Augmented Naïve Bayes (TAN) classifier or k-dependence Bayesian classifiers. BBNs offer robust models in the context of incomplete data, or high degrees of uncertainty or inaccuracy [34].

Considering a subset of the system DAG, x_1 and x_2 are parent nodes in the network and y is a child node. x_1 is a parent of y if a link connects x_1 to y, in that direction of the causality model. The variables themselves are defined by mutually exclusive states (discreet) and their relationships through the DAG are defined by conditional probabilities (continuous). The application of Bayes Theorem for n mutually exclusive hypotheses ($i = 1, 2, \ldots, n$) determines these values according to Janssens et al. [38]:

$$p\left(H_i|E\right) = \frac{p\left(E|H_i\right) p\left(H_i\right)}{\sum_{i=1}^{n} p\left(E|H_i\right) p\left(H_i\right)} \tag{8.24}$$

where $p(H_i|E)$ is the posterior conditional probability for the nth hypothesis H ($i = 1, 2, \ldots, n$). This is calculated using the training data as the evidence E. $p(H_i)$ is the prior probability, or belief, of the hypothesis and $p(E|H_i)$ shows conditional probability between these two. When the hypothesis H_i is true, the total probability is given by the dominator of Eq. 8.24. This approach easily allows for updating, based on new evidence. The prior probability is the assigned likelihood before any evidence is collected, based on expert opinion, and subsequently updated using experiential evidence. The posterior probability is the likelihood assigned after the observations are collected. This equation is used in BBN.

BBN modeling has been applied to many fields, including manufacturing engineering, process root cause analysis, bioinformatics and genetics, accident modeling, risk analysis in the chemical engineering industry, fault diagnosis, pattern recognition, and knowledge engineering [34].

Ensemble Methods

Ensemble methods involve the training of multiple classifier models and combine them together to generate an improved global model with a higher performance. The component classifiers can be the same types of models (e.g., all decision trees—the Random Forest algorithm) with different shapes and hyperparameters, a mixture of different models (e.g., BBN, SVM, or others), or a combination. Bagging, boosting, and random forests are methods used to amalgamate the results of the consistent methods to generate the final prediction [39]. For categorical targets, different ensemble rules can be applied, including majority voting (collaborative generation), highest probability wins, or highest mean probability wins (adversarial generation) for the final prediction. Ensemble methods have proven to be especially suitable in cases when choosing the computational model is not straightforward, which can be a significant concern when modeling highly nonlinear systems. As the use of multiple models, especially if there is significant diversity between them, helps in reducing overfitting. Techniques such as bagging also particularly assist in reducing this problem. However, the obvious drawback of the ensemble methods approach for modeling is that the solution does not scale well for large datasets, as many computationally expensive methods may be embedded within this unified model. When ensemble methods are applied to regression modeling tasks, the voting process is instead based upon statistical values that aggregate the component models, such as mean or median values [40] (Fig. 8.4).

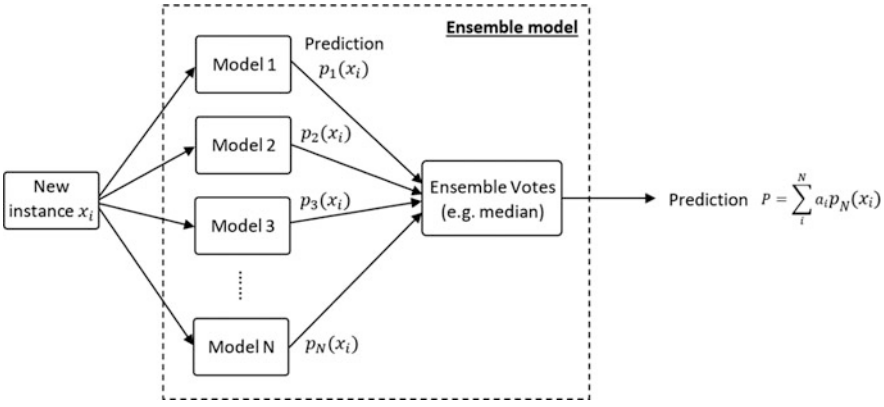


Fig. 8.4 The general architecture for ensemble methods is to generate multiple learning algorithms that form a unified model and predict class labels through an aggregation process such as voting

Ensemble methods have been used in many applications, including network intrusion detection, remote sensing, speech recognition, object identification, and others.

Random Forest

Random Forest (RF) is an ensemble method that uses decision trees for modeling [41]. RF encapsulates the core capabilities of decision trees and hence can be used for classification and regression tasks alike. Each decision tree in the model is built using bootstrap sampling, or sampling with replacement [42]. This approach adds an element of randomness to the modeling process and allows for a broad search of the decision space, without explicitly needing to calculate it in its entirety. Whereas in decision tree modeling, the split applied to the data partition at each node is determined using the most suitable variable according to the selected statistical measure for the model, RF models instead splits the data at each node by randomly selecting a predictor. This creates a wide array of diverse decision trees that contribute toward the voting process. By taking this direction in the modeling process, only two parameters need to be defined: the total number of trees in the forest and the number of variables to be considered for random selection when splitting at each node. The final prediction is given by:

$$P\left(c|v\right) = \sum_{t=1}^{T} P_t\left(c_i|v\right) \tag{8.25}$$

where t_i is the tree number, T is the total number of trees, and c_i is the predicted category.

RF models have shown to be user-friendly due to the property of having only two parameters and harnesses the general advantages of ensemble methods, namely, reducing overfitting and yielding high model performance. Figure 8.5 provides a visual demonstration of the decision tree construction, expected likelihood estimation, and voting process for the final category prediction.

Despite being a relatively new algorithm, RF has been applied to many areas with excellent results. This includes wind power generation, fault detection, remote sensing, chemical processing in the minerals industry, as well as others.

Adaptive Boosting

The boosting method is applied to classification problems, in order to improve the accuracy of classification models by reducing both bias and variance, which in turn converts weakly performing learning algorithms into strong ones. The approach with boosting is to provide weights to the algorithms that are typically determined from the accuracy of the weak learner in the group. Through this process, future

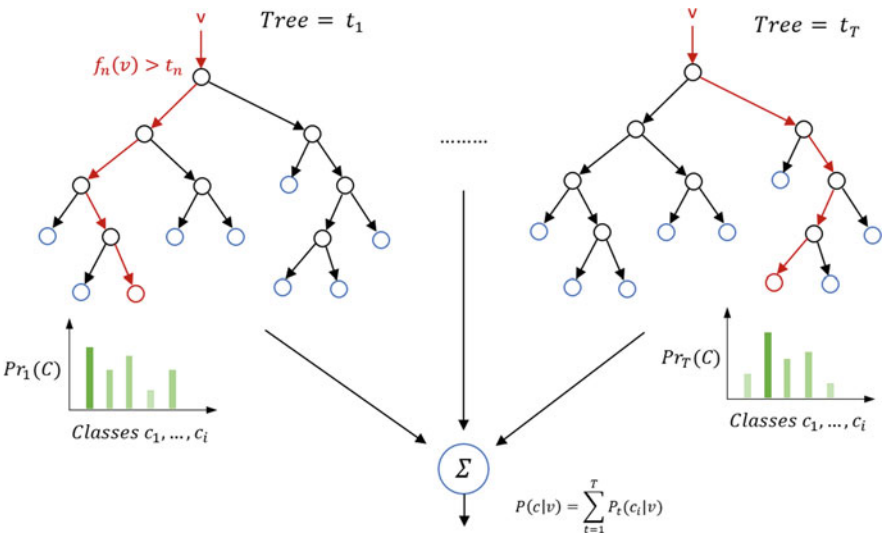


Fig. 8.5 The RF methods randomly construct a range of decision trees using the bootstrap sampling method. These trees each make classification predictions and vote on the final prediction made for the overall model

weak learners provide greater learning value on the training examples that previously weak learners misclassified. This is achieved by adding weight to examples that have been historically misclassified in the training process, whereas those correctly classified are given less weight. As this approach essentially filters out a few, important examples for training from the total dataset, a large amount of training data is needed to successfully run a boosting model. Boosting can be applied to both classification and regression tasks, yet classification is typically more common.

The adaptive boosting method is the most common boosting method, where the algorithm has the capability to adapt to the weak learners in the ensemble. Misclassified examples are provided with a higher weighting factor in the learning process in subsequent iterations [43, 44]. This reduces the total size of the dataset needed to effectively learn and produce high accuracy predictions. Similarly, for problems with very high-dimensional feature spaces that would require very long durations to converge toward an acceptable solution, this approach helps reduce the number of examples needed and hence training time.

For a given set of m training samples, with feature labels X and labels y, $(x_1, y_1), \ldots, (x_m, y_m)$; $x_i \in X$, $y_i \in \{-1, +1\}$, initial weights are applied uniformly to all samples $D_1(i) = \frac{1}{m}$. From here, the learning process begins according to the generic adaptive boosting algorithm below and based upon the error associated with misclassified examples ε_j, the updated weights $D_{t+1}(i)$ are calculated and applied iteratively as follows:

$$\text{find } h_t = \arg\min_{h_j \in H}, \varepsilon_j = \sum_{i=1}^{m} D_t(i) y_i \neq h_j(x_i) \tag{8.26}$$

If $\varepsilon_t \geq \frac{1}{2}$, then the iterative process stops, else:

$$\text{Set } \alpha_t = \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{8.27}$$

$$\text{Assign } D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{z_t} \tag{8.28}$$

where z_t is the normalization factor related to the total number of samples in the training set.

This process yields the following classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{8.29}$$

Given the benefits of boosting, combined with the ability to reduce training time for computationally expensive problems, this method is widely used in many application areas. This includes computer vision, image retrieval, facial recognition, and many more.

8.2.3 Regression Techniques

The approach to regression problems is similar to that of classification, whereby a set of inputs are to be mapped to an output. However, in the case of regression, the output to be predicted is a continuous numerical value, rather than a discreet category of class value. Hence, the aim of regression modeling in machine learning is to find patterns for the prediction of a dependent variable y, based on a set of independent variables $x_1, \ldots, x_k$.

Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) have found widespread adoption as predictive tools in many industries in recent years, from control systems to material science [45]. It is considered one of the hot topics of modeling, with new examples of successful applications emerging in the literature every day. For datasets with nonlinear relationships between input and output feature spaces, ANNs can offer

a modeling solution that meets a trade-off of needs between accuracy, training speed, prediction speed, and scalability for high data volumes, for both classification and regression tasks [46]. With the use of network elements such as softmax functions (generalized logistic function) or a one-vs-all approach (output $h_\Theta \in R^C$), classification tasks can be performed, although more generally ANNs are built and used for regression prediction. The network is constructed from neurons, positioned in layers that hold and compute values, as well as synapses, that connect the neurons and pass values through the network. The neurons in each layer work in parallel to each other. ANNs are comprised of three primary elements: an input layer, hidden layers, and an output layer [47]. Figure 8.6 shows the general architecture for ANN and how calculated data is passed throughout the network in parallel layers.

The input layer is the value in the input feature space $x_i \in X \in \mathbb{R}^d$, which is passed into the first hidden layer, composed of linear summation units and nonlinear activation functions, which in turn passes values on to subsequent hidden layers. The final hidden layer passes calculated values into an output layer, which combined these signals into a prediction, using the heavily transformed initial input layer values. The weightings applied to the linear summation units are randomly initialized to begin with, but are iteratively modified through a training process using a training dataset. This training process is performed to modify the weights, such that the ANN is able to make accurate predictions on both the training dataset, but also new inputs (e.g., a test or cross-validation set). Hence, a good ANN will produce low learning errors and low prediction errors. The selection of a suitable learning algorithm is important in developing a good ANN. Together with the number of hidden layers and the number of neurons per layer, these form key hyperparameters for users to determine. The learning algorithms employ the gradient descent process, which uses backpropagation, to calculate the gradients needed to update the weights and converge the ANN toward a solution that yields good predictions, as per:

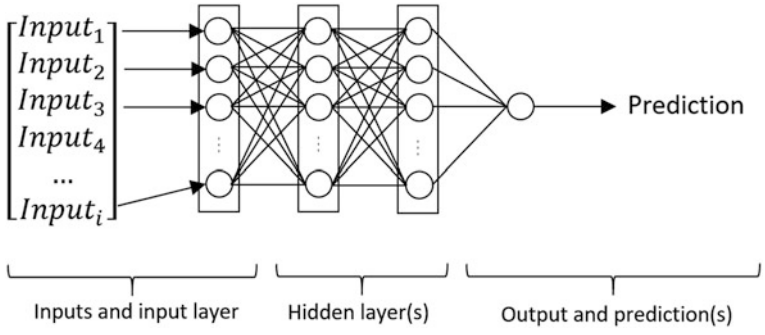


Fig. 8.6 ANNs are comprised of three components: an input layer that injects the input space into the network, hidden layers with linear weights (W_{li}) and nonlinear activation functions at each hidden layer node, and an output layer that produces a prediction

$$a_j{}^{(n)} = g\left(\sum_{k=1}^{m} \Theta_{ij}^{(n)} x_k\right) \tag{8.30}$$

where a is the activation function output from neuron j in hidden layer n, m is the size of the input space, k is the index of the input from the previous layer being summed, and Θ is the matrix of weights controlling the function mapping from layer to layer. As can be seen from this approach, the heavy nonlinear convolution of the input space into the ANN makes the model highly uninterpretable and "black-box" in nature, yielding little or no insight for the user.

Some examples of algorithms for ANN include Scaled Conjugate Gradient (SCG), Levenberg-Marquardt (LM), and Bayesian Regularization (BR; [48, 49]). These different methods pose advantages for different contexts, such as that SCG requires less memory and provides good generalizations for noise datasets, and as such may be useful for datasets that pose volume and veracity constraints. For SCG, the training process stops with respect to adaptive weight minimization or regularization mechanisms. On the other hand, BR takes more time to complete training. LM has been shown to have good convergence properties and take less time for training, but takes more memory [49, 50] and hence may be more suitable within a stream processing context with lower data volumes. The stability of the learning process for LM has shown to be greater, with fewer oscillations toward the converged solution.

Artificial neural networks have been rapidly applied and shown success in many engineering and particularly manufacturing areas, including material modeling, remote sensing, thermal-chemical-mechanical manufacturing problems, image processing, machining, and many others.

Support Vector Regression (SVR)

Support Vector Regression builds on the fundamentals of SVM as outlined in previous sections, but features extended functionality for regression tasks. Specifically, the data is projected into a higher-dimensional feature space and is fitted to a linear function with the minimum complexity for the feature space. This is achieved through the use of kernel functions, such as the commonly applied polynomial sigmoid, Radial Basis Function (RBF), and Gaussian kernels. The latter has found widespread adoption due to the excellent performance of the function, stemming from its highly nonlinear nature, and fewer parameters to be adjusted, which speeds up the training process [51]. Through this process, a linear regression problem can be generated to solve a nonlinear regression problem [52, 53].

The use of a transformation function $\varphi(x)$ on a training dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ determines a suitable function that nonlinearly connects inputs x with outputs y. This means that the relationship between x and y is inherently linear in the new feature space.

$$f(x) = \sum_{i=1}^{n} w\varphi(x) + b \tag{8.31}$$

where w and b describe as linear hyperplanes that can be fit to the training dataset and used as the decision boundary for future predictions. This is achieved in the same general context of SVM, where the structural risk of the model must be minimized. This is implemented as follows:

$$R_{\text{emp}} = \frac{1}{n} \sum_{i=1}^{\text{n}} L_{\varepsilon} \left(y_i, f\left(x_i\right)\right) \tag{8.32}$$

where L_{ε} is the ε-insensitive loss function:

$$L_{\varepsilon}\left(y, f(x)\right) = \begin{cases} 0, & \text{if } |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon, & \text{Otherwise} \end{cases} \tag{8.33}$$

As with general SVM, the expected risk associated with the linear regression operation, using the ε-insensitive loss function, is minimized by using $\|w\|^2$ to decrease the model complexity:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \left(\xi_i + \xi_i{}^*\right) \tag{8.34}$$

Subject to the following constraints:

$$\begin{cases} w\phi\left(x_i\right) + b - y_i \leq \varepsilon + \xi_i, \\ y_i - w\phi\left(x_i\right) - b \leq \varepsilon + \xi_i{}^*, \\ \xi_i, \xi_i{}^* \geq 0, \qquad i = 1, 2, \ldots, n \end{cases} \tag{8.35}$$

where $\xi_i, \xi_i{}^*, i = 1, 2, \ldots, n$ are the nonnegative slack variables, which show the difference between the true value of a training data point and the estimated solution $f(x)$.

Additionally, this optimization problem can be converted to a Lagrangian dual problem and can be solved using the following expression:

$$f(x) = \sum_{i=1}^{n} \left(a_i{}^* - a_i\right) K\left(x_i, x\right) + b \tag{8.36}$$

Subject to the following constraints:

$$0 \leq a_i{}^* \leq C, 0 \leq a_i \leq C \tag{8.37}$$

where a_i and a_i^* are the Lagrange multipliers obtained from the dual problem and inherently $a_i^* \neq a_i$. The inner product of the nonlinear transformation functions $\varphi(x_i)$ and $\varphi(x_j)$ yields $k(x_i, x_j)$. , which is the kernel function.

As has been previously stated, many different kernel functions may be used. The following is the RBF as one of these examples:

$$k(x, z) = \exp\left(\frac{\|x - z\|^2}{2\gamma^2}\right) \tag{8.38}$$

where γ must be set carefully selected to ensure a good solution, in addition to model hyperparameters ε and C shown in Eqs. 8.33 and 8.34, respectively [54]. As with SVM, poor selection of these parameters will lead to a poor model, resulting from high bias or high variance [55]. This trade-off is managed by parameter C, in providing greater weight toward either minimizing the regression prediction error or maximizing the margin, in this optimization problem. Similarly, parameter ε manages the number of support vectors in the model and for an RBF kernel as shown above, γ must be selected pertinent to that model. There are no common set of rules needed to determine these parameters and instead, users must rely on past experiences and mathematical insight in guiding their decisions [56].

SVR has been successfully implemented in many modeling applications, including signal processing, aerospace, petroleum sciences, environment and urban systems, manufacturing, and many more.

8.3 Representation Learning

Representation learning (RL) is a method of constructing classifiers or other predictors from unstructured data, which is useful in dimensionality reduction tasks. RL aims to determine a reasonably sized learned and lower dimensionality representation of an original dataset, capturing a vast number of input configurations, such that the challenges of volume, velocity, and variety of big data can be addressed. Many classical machine learning algorithms require that the entire dataset be loaded into local memory, such that the training process can be effectively performed. For data volumes and other characteristics previously discussed, this is not feasible and as such, representation learning offers a solution in facilitating significant improvements in computational and statistical efficiency. The lower-dimensional and latent representations of vertices within the dataset seeks to retain core information for the statistical processes of machine learning methods to succeed in modeling the full data, including topological structure, vertex content, and other embedded information [57]. Following this, modeling and analytical tasks can be easily applied at this new scale, through the application of conventional vector-based machine learning algorithms, such as ANN, with respect to the new representation feature space. Classification and regression tasks can subsequently

be performed using the outcome of an RL exercise; however, due to the inherent loss of information associated with dimensionality reduction, it is more commonly and meaningfully applied to classification. Figure 8.7 illustrates a visually intuitive example of how a highly feature-rich dataset can be represented in a new input space, where such different raw inputs may seem spatially irreconcilable at first glance.

RL is further subcategorized into the goal-oriented strategies for implementation of feature selection, feature extraction, and distance metric learning. Both semi-supervised and unsupervised tasks can be achieved with RL, depending on the availability of any labeled data to guide the deeper structural learning capabilities. From an algorithmic perspective, there are five categories in which RL can be categorized. These are deep learning-based methods, random walk-based methods, matrix factorization-based methods, edge-based modeling methods, and hybrid methods.

This strategy has found significant successes in the fields of communication networks by detecting community structures [58], modeling urban dynamics [59], biological sciences by inferring interactions between proteins for the facilitation of new treatments for diseases [60], natural language processing (NLP) [61], speech recognition, and intelligent vehicle systems.

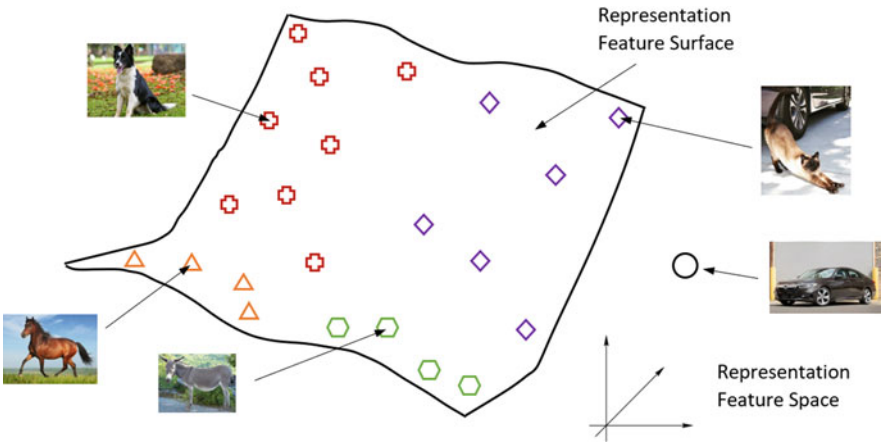


Fig. 8.7 Representation learning uses data reduction to extract lower-dimensional features of input data, to perform machine learning. A visual representation of this in this figure shows how learned images of dogs, cats, horses, and donkeys may share common feature spaces (e.g., colors and edges), but are clustered in different regions on the feature surface, within the feature space. This allows for successful classification at a lower computational cost, with an acceptable trade-off in minimally higher error rates. Images of new objects, such as cars, can be described in this feature space, but due to the high difference between it and the trained examples, it will likely not near the surface

8.3.1 Deep Learning-Based Methods

Deep learning-based methods are employed in problems with a very high degree of nonlinearity in the dataset, which can be learned from stacked, deep learning networks. Yet, in order to achieve this, computational time for these models is typically high [62]. As a measure of similarity between vertexes within a dataset, typical deep learning network architectures are designed for $V \in \mathbb{R}^d$, $d = \{1, 2, 3\}$ Euclidean structured datasets.

Structural Deep Network Embedding (SDNE) is an example of a deep learning-based algorithm for RL. This algorithm is applied to semi-supervised problems, where a semi-supervised deep encoder model is initially generated in order to determine the nonlinearity of the data network structures. In the unsupervised component of the algorithm, the vertex representations that preserve the second-order proximities are learned by reconstructing the vertex adjacent matrix representations, which is $|V|$-dimensional. The objective function to be minimized for this task is as follows:

$$\mathcal{L}_{2\text{nd}} = \sum_{i=1}^{|V|} \left\| \left(r_{v_i}^{(0)} - \hat{r}_{v_i}^{(0)} \right) \odot b_i \right\| \tag{8.39}$$

where $r_{v_i}^{(0)} = S_{i:}$ is the representation of the input dataspace, $\hat{r}_{v_i}^{(0)}$ is the lower dimensional output representation space, and b_i is a weighting factor that penalizes the fitting error on the nonzero elements of S. Following this, the supervised component of the algorithm then determines the first-order proximities within the embedding space, between the connected vertices. The objective function to be minimized for this task is as follows:

$$\mathcal{L}_{1\text{st}} = \sum_{i,j=1}^{|V|} S_{i,j} \left\| \left(r_{v_i}^{K} - \hat{r}_{v_i}^{K} \right) \right\|_2^2 \tag{8.40}$$

where $r_{v_i}^{K}$ is the representation of the vertex v_i within the Kth layer of the deep network and K is the depth of the network (number of hidden layers). Between these two objective functions, they are in turn combined into the following multi-objective optimization function:

$$\mathcal{L} = \mathcal{L}_{2\text{nd}} + \alpha \mathcal{L}_{1\text{st}} + \nu \mathcal{L}_{\text{reg}} \tag{8.41}$$

where $\mathcal{L}_{\text{reg}}$ is a regularization term that penalizes model complexity and reduced overfitting of the model to the dataset.

8.3.2 Random Walk-Based Methods

Random walk-based methods leverage the stochastic sampling of the dataspace through random walk algorithms, to build insight into the structural relationships between vertices. This is achieved by performing truncated random walks on the dataset, which will inherently capture the connections between vertices within sequences. This collection of sequences can then be used as an input for probabilistic analysis in building a structural model, via the frequency of occurrence of vertex-context pairs to measure the structural distance between them. This approach to obtaining structural information is efficient, but given that the random walks are truncated, only local structure is typically captured. Figure 8.8 provides insight into how a random walk is applied to a data structure network, in order to extract this information.

Examples of RL random walk algorithms include DeepWalk, node2vec, TriDNR, struct2vec, and others.

8.3.3 Matrix Factorization-Based Methods

Matrix factorization-based methods aim to capture a representation of the connections between vertices of a dataset within a matrix, to then obtain the embedding space by factorizing that matrix. This approach assumes that the majority of the variance of the original dataset is only affected by a small subset of latent factors, which are those to be determined through the factorization process. This approach has proven effective in learning informative vertex representations from the original dataset, but it suffers from scalability issues. Namely, factorization operations performed on datasets with the numbers of rows and columns typical of big data (in the order of millions or more), this problem can become computationally infeasible and must be kept in mind when designing the model [63]. To achieve this result,

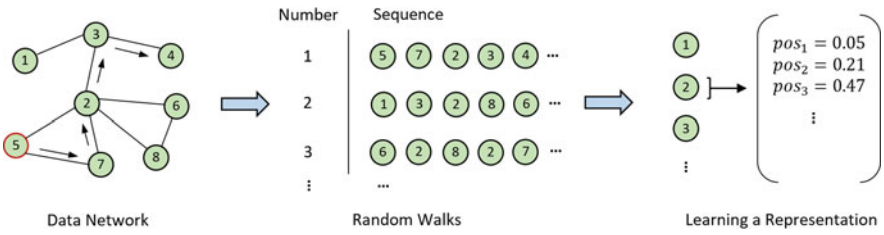


Fig. 8.8 An illustrated example of a random walk applied to Representation Learning. In this visual example, a series of truncated random walk samples of desired lengths are performed to obtain sequences of connected vertices. One is shown, starting at node 5 (red outline). After collecting a given set of sequences, analyses of this set can be performed, such as describing the probability of the position a given node in a random walk (node 2 is shown here)

many different types of matrices can be used as a means of preserving the data structure, including the modularity matrix, the k-step transition probability matrix, and the vertex-context matrix.

The High-Order Proximity Preserved Embedding (HOPE) is an example of a matrix factorization-based method to RL for big data. The application for this algorithm is for directed networks and it learns the asymmetric high-order proximity of those networks, by building vertex representations. Specifically, in asymmetric networks, the transitivity between vertices is important to capture. To this end, the HOPE algorithm learns two embedding vectors as follows:

$$U^{\mathrm{s}}, U^{\mathrm{t}} \in \mathbb{R}^{|V| \times d} \tag{8.42}$$

where U^s and U^t are referred to as the source and target embedding vectors, respectively. The vector embeddings are learned via the matrix factorization problem as follows:

$$\min_{U_{\mathrm{s}}, U_{\mathrm{t}}} \left\| S - U^{\mathrm{s}}.U^{\mathrm{t}^T} \right\|_F^2 \tag{8.43}$$

where S is the high-order proximity matrix. This minimization problem seeks to reduce the error between the embedding vectors and the proximity matrix to obtain an accurate representation space.

Other matrix factorization-based algorithms for RL include GraphWave, TADW, MMDW, DMF, LANE, and others.

8.3.4 Edge-Based Modeling Methods

Edge-based modeling methods use vertex–vertex connections within a dataset to directly learn vertex representations. By taking this approach, these methods are typically more efficient compared to the other RL methods presented, but they also have the disadvantage of not being able to reliably capture global network structure. This is due to the limitation that they only consider observable vertex connectivity information and do not generate new information for analysis in a transformed representation dataspace.

Large-scale Information Network Embedding (LINE) is an example of an edge-based modeling method. The vertex representations of the dataset are learned by explicitly modeling both the first-order and second-order proximity, using the observable vertex connectivity [64]. Two objective functions are formed to preserve the first- and second-order proximity information, each of which is minimized respectively. The first objective function, for the first-order proximity, is as follows:

$$O_1 = d\left(\hat{p}_1\left(.,.\right), p_1\left(.,.\right)\right) \tag{8.44}$$

where $p_1(v_i, v_j)$ is a vertex pair in the joint distribution modeled within the embedded latent, lower-dimensional space, $\hat{p}_1(.,.)$ is the empirical distance between the datapoints (in the original dataset), and $d(.,.)$ is the distance between the two. The distance metric used may be Euclidian or otherwise.

Following the preservation of distance-based information of the vertices, the second-order proximity objective seeks to preserve the structural information of the dataset. This is achieved by minimizing the following function:

$$O_2 = \sum_{v_i \in V} \lambda_i d\left(\hat{p}_2(.|v_i), p_2(.|v_i)\right) \tag{8.45}$$

where $p_2(.|v_i)$ is the modeled conditional distribution of the context for the vertices within the latent representation space with respect to point $v_i \in V$, $\hat{p}_2(.|v_i)$ is the conditional distribution for the empirical dataset and λ_i is the prestige of the vertex v_i.

By minimizing both of these objective functions, two different vertex representations are learned independently by each, capturing the first- and second-order proximity, respectively. These are then combined into a single, final vertex representation.

Examples of other edge-based modeling algorithms include TLINE, pRBM, GraphGAN, and others.

8.4 Deep Learning

Deep learning applies supervised and/or unsupervised learning strategies in deep architectures, in order to automatically find hierarchical representations of datasets. This strategy has the potential to capture highly complicated, nonlinear statistical patterns from very large feature spaces typical of big data, which are hierarchically organized by the networks. Deep learning architectures are characterized by multi-layering of commonly shallow algorithms, such as Artificial Neural Networks (ANNs), to include many processing layers [65]. Often, there are additional strategies implemented to handle various aspects of the learning process to this scale. Figure 8.9 provides a visual example of how multiple strategies may be used in concert to achieve this goal, such as Representation Learning and Deep Learning. Convolutional Neural Networks (CNNs) and Deep Belief Networks (DBNs) are examples of common deep learning architectures. Classification and regression tasks can both be achieved using DL strategies. For example, image segmentation classifies regions of pixels as members of given object classes (e.g., road, car, sign, and pedestrian in a road scene), whereas the direct modeling of a complex system with respect to physical and continuous inputs and outputs (e.g., a manufacturing process) can use a regression approach.

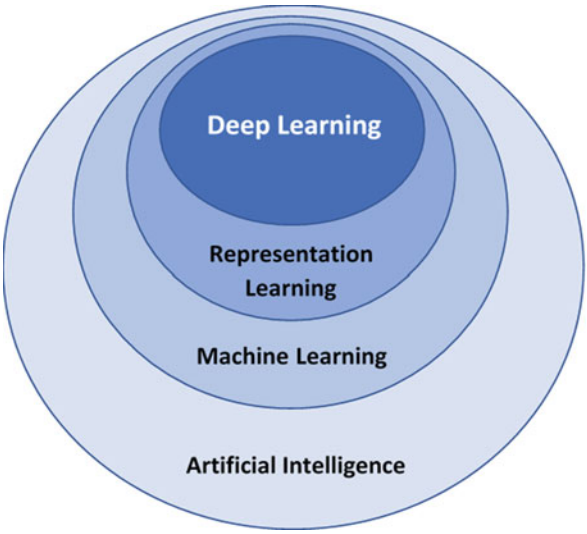


Fig. 8.9 Deep learning may be layered together with additional strategies, in order to deal with the large volumes of big data, while also capturing the highly nonlinear behavior of a complex system. Here, we use combine RL in order to perform dimensionality reduction and DL to learn the system, with both architectures further embedded in the generalized machine learning domain of supervised or unsupervised learning, as well as artificial intelligence principles

Deep learning has found extensive applications in many fields, including computer vision [66], speech recognition [67], NLP [68], intrusion prediction [69], and energy modeling [70].

8.4.1 Deep Neural Networks

Deep learning has commonly been applied as an extension to the capabilities of ANNs, with more complex training structures and architectures to capture increasingly nonlinear characteristics of target datasets. A typical ANN is composed of three network elements, being an input layer, subsequent hidden layers, and an output layer. Deep learning approaches applied to ANN models can greatly extend the number of hidden layers, which may have many embedded hidden layers with other auxiliary functions to perform convolution operations or nonlinear transformations of the data through the network. Given the ability to develop highly nonlinear models through deep learning, the balance between high bias and high variance is tipped more toward the potential for overfitting. Hence, additional methods can be incorporated into the learning process to reduce this, such as dropout, where neurons within the network are randomly disregarded for a given iteration in the training process. Dropout also provides additional robustness and capability for deep learning, but allowing training directly on raw, unnormalized data [70].

Additional challenges often need to be addressed when implementing deep neural networks to big data. One common challenge, especially for computer vision tasks, is the volume of the data being trained on. Considering an image of

1024 × 1024-pixel resolution and three-color channels (RGB), the input space is in excess of three million data points. A direct implementation of Perceptron ANNs would be computationally infeasible, as a vectorized implementation of this requires enormous matrices to be handled, limited by memory and computational speeds, especially for the analysis of live-streams of video-based images. To circumvent these challenges of scale, convolutional neural networks (CNNs) have been developed. Prior to the data being fed into a deep neural network for predictions, a series of convolutions and nonlinear activations (e.g., ReLU) capture high-dimensional nonlinear features, followed by pooling, which acts as a data reduction technique that preserves the captured features. This step is iterated until the input space into the deep neural network is small enough to be computationally feasible and large enough to successfully model the data. Figure 8.10 shows how this process generally works.

The nonlinear relationship between two connected layers in a deep neural network, h_l and h_{l+1}, is defined by the following function:

$$h_{l+1} = g\left(Wh_l + b\right) \tag{8.46}$$

where b is the bias vector fed into layer h_{l+1}, matrix W is the set of model parameters that act as linear weights applied to the outputs of layer h_l fed into layer h_{l+1}, and g is the activation that provides the nonlinear capabilities into the network (ReLU, sigmoid, tanh, or others). In order to construct a deep ANN, in order to model a given input space $y = f(u)$, the above function is stacked in series in a network of depth L according to the following set of equations via vectorized implementation:

$$h_1 = g_1\left(W_1 u + b_1\right) \tag{8.47}$$

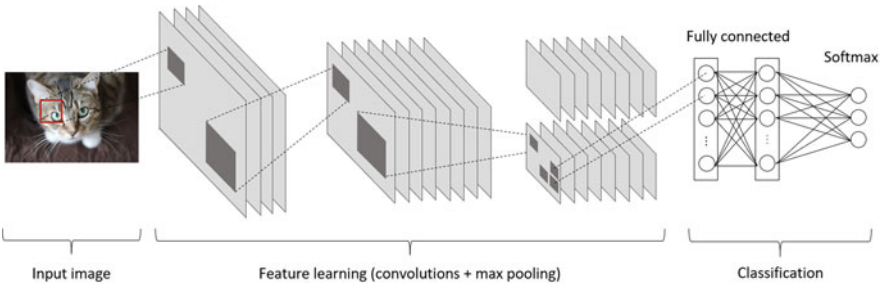


Fig. 8.10 Deep learning allows for capturing highly complex, nonlinear features in very large feature spaces, such as images. The use of a CNN in this figure shows how an input image, which would be computationally infeasible to feed into a Perceptron ANN (1024 × 1024 pixels have over a million features in its input space), can be dimensionally reduced through a series of pooling actions, performs feature learning through its nonlinear activation functions (e.g., ReLU), and is then fed into a smaller Perceptron for classification

$$h_2 = g_2 \left(W_2 h_1 + b_2 \right) \tag{8.48}$$

$$\ldots$$

$$y = g_L \left(W_L h_{L-1} + b_L \right) \tag{8.49}$$

For this set of equations, the goal of the learning task is to minimize the error function $\varepsilon(y^n, \tau^n)$, which calculates the difference between the inputs and outputs of the training dataset $\{u^n, \tau^n\}_{n=1}^{N}$ and the model predictions $y^n = f(u^n)$. This is achieved by applying the process of gradient descent to the network, in order to iteratively converge the set of network model parameters $\theta = \{W_1, \ \ldots \ , W_L, b_1, \ \ldots \ , b_L\}$ toward a model that minimizes the sum of the errors, calculated through the process of backpropagation [71]. The objective function to be minimized is as follows:

$$\min_\theta \left[J = \sum_{n=1}^{N} \varepsilon \left(y^n, \tau^n \right) \right] \tag{8.50}$$

With the addition of other operations, such as convolutions and pooling in CNNs, these may be further added into the process of gradient descent throughout the entire network. This understandably adds greater computational complexity. However, due to the significant advancement of graphical processing units (GPUs) in recent years, which contain architectures that harnesses parallelized computing, has allowed for the successful scaling of these methods through vectorized implementation.

More scalable vectorized implementations of deep learning ANN architectures, combined with powerful modern computing hardware solutions have opened opportunities to deployment in a broad array of problems, such as computer vision in the automotive industry, where on-board platforms are able to perform regression and classification tasks in situ. This offers applications in scenarios that require fast modeling and decision-making, such as for self-driving cars. Additionally, software libraries, such as Google® TensorFlow built on the Python language, or modules within Mathworks® Matlab software, facilitate the deployment of deep learning methods for users to easily implement. These factors act to reduce the cost of entry and shortening the development cycle for engineering problems, which can be leveraged by large firms and start-ups alike. Further, the open-source nature of the machine learning industry has led to aggressive strides in developing robust predictive models that are in turn widely employed by a variety of end users. Tools such as this provide solutions for users to address challenges of big data associated with volume and velocity, in particular, under the context of highly nonlinear image classification problems.

Image classification and object recognition are problems with many potential applications, including computer vision for automotive self-driving. Given the

highly nonlinear nature of the problem, deep learning networks have been employed as a method to address this complexity. The deep architectures of CNNs have the capability to both perform data reduction on the high-bandwidth feature space of input images (covered in Sect. 8.3), as well as make predictions on the class of objects in an image. There are many hyperparameters to be selected in defining a network to complete object classification. This includes the size, depth, and quantity of convolution and max pooling layers, learning rate for the learning process, the optimization algorithm used, applicable regularization terms to penalize overfitting or high variance of the network, the size of the mini-batch of inputs used for training, the architecture of the fully connected layers at the end of the network (i.e., number of layers and number of neurons per layer), among others. Hence, the possible number of models that can be trained for this task is great, such that finding the optimal solution is not trivial. The VGG-16 and VGG-19 CNNs, developed by the Visual Geometry Group at Oxford University, are CNN models developed for object classification tasks [72]. The depths of these networks are 16-layers and 19-layers, respectively, where the description of VGG-16 is given in Fig. 8.11.

VGG-16 and VGG-19 have shown excellent predictive capability for image recognition. Simonyan and Zisserman [72] have shown the networks are capable of a top five classification error on a test set of 7.4 and 7.3%, respectively, when using the hand-labeled ImageNet database for the Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) challenge. The success of this model has led to its direct integration into many programming libraries, including TensorFlow and Matlab. This has allowed for the rapid deployment of the model for end users. For example, below is a sample of code that can be used in Matlab to generate a semantic segmentation network (composed of an image classification network and up-sampling portion), by leveraging the VGG-16 CNN. The below function of Matlab allows for the construction of segmentation network layers (lgraph), pre-initialized with the layers and respective weights from VGG-16 as a pretrained model.

```
lgraph = segnetLayers(imageSize,numClasses,'vgg16');
```

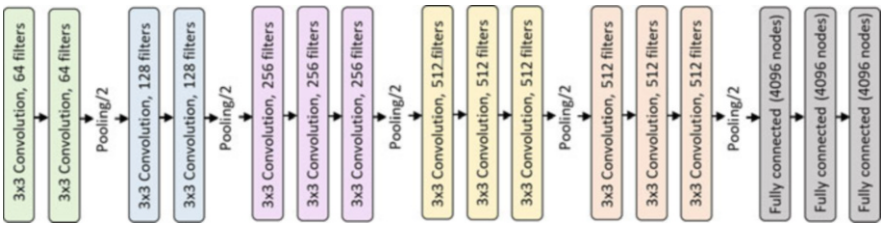


Fig. 8.11 The architecture of the VGG-16 CNN for object recognition. The deep learning CNN utilizes 13 3 × 3 convolutional layers, combined with pooling after the second, fourth, seventh, tenth, and thirteenth convolutional layers. The dimensionality of the feature space is progressively reduced, with the introduction of more filters deeper into the network (beginning with 64 and finally at 512 per layer). Two fully connected layers with 4096 neurons each are connected to the 16th and final layer, comprised of a softmax activation, to make the classification prediction

Following this, the training parameters can be specified for the network with a few simple commands. This includes the optimization algorithm used, which in the below code is specified as stochastic gradient descent with momentum (SGDM).

```
options = trainingOptions('sgdm', ...
'Momentum', 0.9, ...
'InitialLearnRate', 1e-2, ...
'L2Regularization', 0.0005, ...
'MaxEpochs', 120,...
'MiniBatchSize', 4, ...
'Shuffle', 'every-epoch', ...
'Verbose', false,...
'Plots','training-progress');
```

Once the model has been constructed and training parameters set, the model can be trained and then evaluated. As a method of evaluation, labeled segmented images from the test set, considered as the ground truth, can be compared with the predictions from the model. Example code for use in Matlab is given below, in which case image number 400 is extracted from the labeled image set for the comparison. In this case, a figure is generated that plots both image outputs IB and CB, for comparison in predictive accuracy, where the former is the human-labelled ground truth and the latter is the output prediction from the deep learning model, respectively.

```
pic_num = 400;
I = readimage(imds, pic_num);
Ib = readimage(pxds, pic_num);
IB = labeloverlay(I, Ib, 'Colormap', cmap, 'Transparency',0.8);
figure
% Show the results of the semantic segmentation
C = semanticseg(I, net);
CB = labeloverlay(I, C, 'Colormap', cmap, 'Transparency',0.8);
figure
imshowpair(IB,CB,'montage')
HelperFunctions.pixelLabelColorbar(cmap, classes);
title('Ground Truth vs Predicted')
```

Using the above code on an image set, the result shown in Fig. 8.12 can be generated. This network has been used to make classification-based segmentation predictions for regions in the image including bicycles, pedestrians, vehicles, roads, road signs, and other features that may be deemed relevant to autonomous driving problems. This model can be used to make such predictions in the context of stream processing, where data volume and velocity would be major concerns for classical machine learning architectures.

Beyond the application example above, deep neural networks have found many applications in fields ranging from computer vision, robotics, pattern recognition, chemical process modeling, and many others.

Fig. 8.12 Image segmentation performed on a photograph of a street, performed in Mathworks® Matlab software, comparing the ground truth from the labeled test set (left) and the prediction from the model (right). The segmentation is achieved through use of a pretrained VGG-16 CNN, for capturing high-level features through the deep learning architecture of the network and making classification predictions using a classical ANN appended to the feature-learning layers of the CNN (modified from Mathworks [73])

8.4.2 Deep Belief Network

Deep belief networks (DBNs) are an extension of functionality from deep neural networks, where these models use probabilistic methods in the learning process to reconstruct the inputs, generally within an unsupervised training context. Further training of a DBN using labeled data can yield a model for the purpose of classification or regression. It achieves this through the implementation of a series of stacked generative stochastic ANNs to learn a probability distribution over a set of given inputs. Different learning algorithms have been developed in the literature to this end, such as the efficient gradient-based contrastive divergence [74], which is what helped propel DBNs into prominence for a wide range of applications.

Contrast divergence applies an approximation to the maximum likelihood method for iteratively updating the learned weights of the connected network, to find the solution via the generative energy-based model. For a given layer in the DBN, the weights are updated using the following log likelihood calculation:

$$w_{ij}\,(t+1) = w_{ij} + \eta \frac{\partial \log\left(p(v)\right)}{\partial w_{ij}} \tag{8.51}$$

where $p(v) = \frac{1}{Z}\sum_{h} \mathrm{e}^{-E(v,h)}$ is the probability vector from the previous layer of the network (based on energy function $E(v,h)$ applied to the network), (v,h) are the states of the visible and hidden units in the network, and η is the learning rate at which the weights are shifted toward the converging solution.

At each Kth layer of the network, the outputs from the $(K-1)$th layer are initialized as the inputs and used to update the subsequent hidden units in the network, then reconstruct the input units, as follows:

$$p\left(h_j = 1|V\right) = g\left(b_j + \sum_i v_i w_{ij}\right) \tag{8.52}$$

$$p\left(v_i = 1|H\right) = g\left(a_j + \sum_j h_j w_{ij}\right) \tag{8.53}$$

where b_j and a_j are the bias of hidden unit h_j and input unit v_i, respectively. Additionally, g is the nonlinear sigmoid activation function to capture the system complexities within the model. These update steps can be continued until some threshold of intra-layer convergence is achieved. Following this, the weights are updated according to Eq. 8.51, where the magnitude of the weight change is proportional to the difference between the averages of the training data and model distributions:

$$\Delta w_{ij} \propto \left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{model}} \tag{8.54}$$

The model training process is continued until convergence toward a solution, determined by a stopping criterion, is met. Further units are added to the DBN, creating a deeper network, where each layer is trained using the above procedure.

DBNs have been used for applications including dimensionality reduction, feature learning, classification and others, used in applications including drug discovery, signal processing, and many other fields.

8.5 Distributed and Parallel Learning

Distributed and parallel learning (D&PL), as the name implies, used a distributed network of parallel processing peers, able to allocate learning processes to each, for the efficient scaling of machine learning algorithms to vast volumes of data. This addresses the technical limitations of classical machine learning frameworks, which typically require entire datasets to be loaded into local memory and also processed locally, which is irreconcilable with the realities of big data volumes. Data can be partitioned in preparation for D&PL either horizontally (by instances within the training set) or vertically (by features within the training set). The majority of partitions are made horizontally, as this is the most natural and suitable choice for most applications. Mixed fragmentation is an alternate approach where datasets are partitioned into subsets of instances and features to be stored and processed across different sites. Commercial platforms such as Apache Hadoop and Google MapReduce have been built to provide this service to users and are commonly employed as solutions.

The majority of D&PL algorithms leverage ensemble methods for collating individually models trained from independent datasets to ensure successful final models according to time and accuracy objectives [75]. The ensemble methods primarily seek to combine the outputs of the constituent models, rather than integrating the models themselves, as much information and model value are lost in the translation between models. This is especially the case for models that use different learning metrics (e.g., rule based and distance based). The ensemble approach provides an array of advantages to the otherwise challenging context of big data, such as the reduction of individual training biases within isolated models by ensemble processes, or the inherently scalable nature. Additionally, corollary benefits for this approach can be realized, such as robustness to security as a consequence of fragmenting datasets across multiple sites. There are many different architectures for D&PL. Although both classification and regression tasks can leverage D&PL model architectures, classification has found greater application due to ease of implementation. Material presented in this section will hence be written from the perspective of classification modeling. Figure 8.13 shows the architecture for distributed machine learning using Google's TensorFlow library on Apache's Spark platform, as a means of training local models that are aggregated into a unified model.

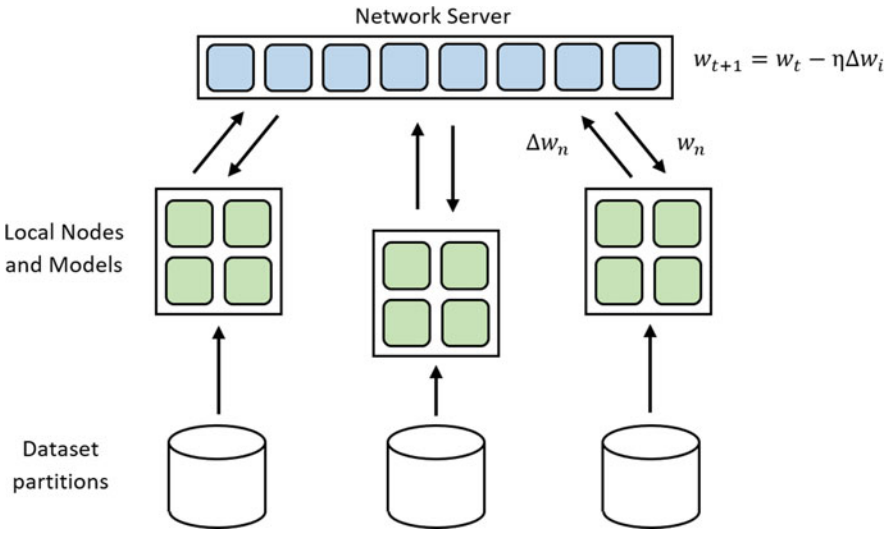


Fig. 8.13 Distributed and parallel learning systems aim to train models over many nodes and aggregate the results from the partially sampled full datasets (as inputs to training) of each to converge toward a high-accuracy model faster. Google's TensorFlow combined with Apache's Spark provides a method to achieve this, where local model replicas are trained using partitions of a global dataset (data shards) and a parameter server is used to update global model parameters by surveying the model replicas

There have been many examples of the successful implementation of D&PL, largely in the data and computer science communities that operate on the scales of volume and velocity that require such an approach, such as information retrieval in learned search processes. Additionally, the high bandwidth of data in some computer vision applications has found solutions with D&PL, as well as NLP. Other major applications include search engines [76], database management [77], and data mining [78].

Distributed machine learning algorithms are generally categorized into decision rules, stacked generalization, meta-learning, knowledge probing, distributed passing votes, effective stacking and distributed boosting. A brief description of each is provided as follows.

8.5.1 Decision Rules

The use of decision rules addresses the challenges of making statistical inferences of a large and fragmented dataset, in the assignment of predictions to individual instances. If this were to be performed directly on the total dataset, high-order statistical parameters from the joint probability density function could be extracted for the direct computation of the posterior probability functions as per Bayesian theory:

$$x \rightarrow c_j \text{ if } P\left(c_j|y_1, \ldots, y_n\right) = \max_k P\left(c_K|y_1, \ldots, y_n\right) \tag{8.55}$$

where $y_i \in \{y_1, \ \ldots \ , y_N\}$ is the set of N outputs and hence N model predictions to aggregate, for which instance x should be assigned to class c_j.

In the case of distributed processing, the full information from the joint probability distribution needed is not available and thus, the rules for applying classes to the instances must be expressed in terms of decisions generated from individual models [79]. The posterior probability can be estimated as $P(c_j|\, x) = y_i$ from model y_i, when measures of belief are available. This concept is what allows for efficient combination rules. Some of the prevalent rules used in the literature include the product rule, sum rule, max rule, min rule, median rule, and majority voting. For example, the sum rule for mapping instance x to class c_j is given by:

$$x \rightarrow c_j \text{ if } \sum_{i=1}^{N} y_{i_j}(x) = \max_{k\,=\,1}^{C} \sum_{i=1}^{N} y_{i_k}(x) \tag{8.56}$$

where the maximum value of the linear sums of model probabilistic predictions emerges as the final class to be applied to the instance.

8.5.2 Stacked Generalization

Stacked generalization seeks to combine multiple predictive models, by learning that each model output correlates with the true class of each instance. This is achieved by inferring the biases of the models from the use of an independent test dataset. To obtain this extra layer of information, each node in the distributed network is provided with a model to train using a test and validation dataset. All of the trained models are then shared with all nodes, so that they are locally stored on each. Having generated a set of N models across all network nodes, a meta-level training set can be generated. For the y_i, $i \in \{1, \ldots, N\}$ models, there will be hence N predictions for the output of instance x, but only one true class for the instance, class(x). This forms the meta-level training set:

$$[y_1, y_2 \ldots, y_{N-1}, y_N; \text{class}(x)] \tag{8.57}$$

Following, this set is then provided to a single node in the network to generate a global model that fits the aggregated outputs from each model, into a single prediction for class(x). Using this approach, for each new data point to be predicted, it must first be fed through each model y_i, $i \in \{1, \ldots, N\}$ to form a meta-level instance, in turn fed into the global model to generate the final prediction (Fig. 8.14).

8.5.3 Meta-Learning

Meta-learning builds upon the core concept of stacked generalization, where a meta-level training set is generated, composed of model predictions for the class of instance x, for training a global model that compensates for the individual biases of each constituent model. Different meta-level training sets can be defined that drive the learning outcome for the global classifier, which is where the extra functionality arises from the general approach of stacked generalization. There are three types of meta-learning strategies for defining this meta-level training set, combiner strategies, arbiter strategies, and hybrid strategies that blend the previous two.

Combiner strategies use a composition rule, in order to determine the content of the instances used in the meta-level training set. The meta-class approach, similar to stacked generalization, defines a training set composed of the individual model outputs y_i, as well as the true class of instance x, into $[y_1, y_2 \ldots, y_{N-1}, y_N; \text{class}(x)]$. The meta-class-attribute approach also adds the instance attributes to this set, adding further information for the global model to potentially provide a better overall prediction, given by $[y_1, y_2 \ldots, y_{N-1}, y_N; x; \text{class}(x)]$. Further, the meta-class-binary approach provides the output predictions from each model y_i rather than as a class c_j prediction, but as a one-vs-all vector with binary entries, where the

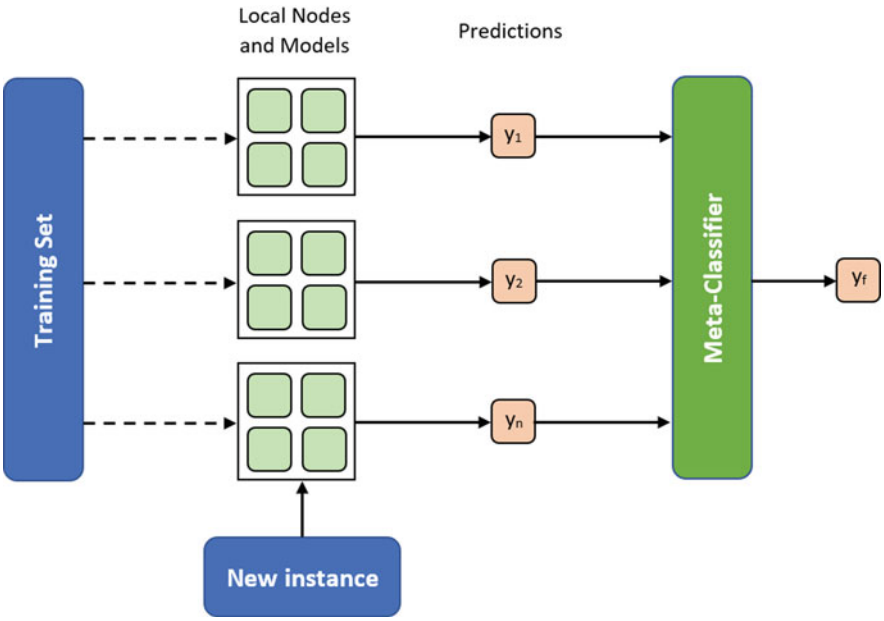


Fig. 8.14 Stacked generalization uses a meta-classifier to make a global prediction, using the aggregated predictions of individual node-based models within a computational network

actual predicted class is denoted by a 1 and all other elements are 0. This is denoted as $\left[y_{1_{1,\ldots,C}}(x), y_{2_{1,\ldots,C}}(x) \ldots, y_{N-1_{1,\ldots,C}}(x), y_{N_{1,\ldots,C}}(x); x; \text{class}(x)\right]$. These methods provide different levels of insight into the prediction characteristics of each model, for the global model to be trained upon.

Arbiter strategies define selection rules, which are applied to the validation dataset to create a subset of instances that will contain the meta-level set. This means that the chosen subset from the validation set is fed into the constituent models and that set of outputs is used to define the meta-level training set for the global model. Two schemes are used to this end, these being the meta-different and meta-different-incorrect schemes. Meta-different select instances from the validation set that yields different outputs from models y_i, to form the meta-set M_d, such that:

$$M_d = \{x | y_1(x) \neq y_2(x) \vee y_1(x) \neq y_3(x) \vee \cdots \vee y_{n-1}(x) \neq y_n(x)\} \tag{8.58}$$

This set specifically choses examples where the constituent models disagree with each other, which is more useful for training the global model to compensate for biases, rather than including instances where the models agree. By contrast, meta-different-incorrect combines instances with the model outputs that agree, but the agreed class is not the true class of x, such that:

$$M = M_d \cup \{x | y_1(x) = y_2(x) = \cdots = y_n(x) \wedge \text{class}(x) \neq y_1(x)\} \tag{8.59}$$

Having defined the meta-level training set from any of the combiner or arbiter strategies, or a hybrid approach of both, the set is sent to a single node in the network and trains the global model. In the case of combiner methods, the constituent models all produce predictions for a given instance, to generate a meta-level instance, which is fed into the global model for the final prediction. However, the arbiter methods define the final prediction as the majority vote for the class of instance x, with the global model providing the tie-breaking vote when necessary.

8.5.4 Knowledge Probing

Knowledge probing as a D&PL method is born from the realized limitation of typical stacked generalization methods, whereby the definition of a meta-level training set only provides insight into the broad statistical combination of predictions from constituent models. In short, this means that the meta-level set does not provide insight into each model's understanding of the data, which is instead embedded deeper within the black box itself. To this end, a descriptive model is built in knowledge probing, learning from an unseen set of data points, and the corresponding set of predictions.

Once the training set has been used to train a model at each node, the models are distributed to all other nodes in the network. Following this, probing instances are defined by using the inputs x from the validation set, combined with labeled classes that are the desired class $d(x)$ for each instance. The value of the desired class is determined by application of a decision rule to the collection of outputs from the constituent models, such as majority voting. Hence, the probing instances can be defined as:

$$[x; d(x)] \tag{8.60}$$

The probing instances are sent to a single node in the network, which builds the probing set from the instances, in order to train a global model. Once the global model has been established, any new instances sent to the node will simply determine the final prediction.

8.5.5 Distributed Passing Votes

Distributed passing votes implement one of two strategies, importance-based sampling (Ivote) and random-based sampling (Rvote), to generate subsets of data for training models. To begin with, Ivote shall be examined. Ivote uses sampling with replacement to generate each subset for training, such that each is more likely to contain instances that are incorrectly predicted by the ensemble of models up that point in the process. This means that the process is iterative, whereby each

subsequent model in the process relies on the output of the ensemble of previous models. The out-of-bag error is used in the sampling process to ensure that each model is tested using instances not from its own training set. This generates a training set that is able to obtain a suitable prediction for the generalization error.

For each node in the network, once the first subset of data for training is generated, it is used to train the model at that node. Following this, the out-of-bag error is calculated as follows:

$$e(k) = p \times e\,(k-1) + (1-p) \times r(k) \tag{8.61}$$

where p is the p-value selected for the training process ($p = 0.75$ is recommended by Brieman [80]), $e(k-1)$ is the error calculated from the previous model in the ensemble, k is the total number of models in the ensemble at this point in the iterative process, and $r(k)$ is the error rate of the kth classifier on this first subset of data. Following this, the test set is built by estimating the probability of selecting a correctly predicted instance, where those that were incorrectly predicted are assumed to have not been in the training set for this ith round of training. The probability is defined by:

$$\Pr(k) = \frac{e(k)}{1 - e(k)} \tag{8.62}$$

Using this metric, once z instances have been selected for this subset, a new model is trained on the subsequent subset. The process is repeated to produce a desired number of models. The Rvote process is similar to that for Ivote described above, except that each subset is selected by bootstrap sampling and every instance has the same probability of being selected for inclusion, irrespective of whether it was likely in the previous training set or not.

Both the Ivote and Rvote processes generate a great number of prediction models at each node in the network through this iterative process. The global model to determine final predictions combines the outputs of these constituent models, typically by a voting mechanism.

8.5.6 Effective Stacking

Effective stacking is an extension of stacked generalization that aims to provide better scalability for applying machine learning models to datasets with very high volumes and dimensionality. This need arises, because in the stacked generalization approach, the number of inputs to the meta-level training set for the global prediction model is a function of the number of classes being predicted, as well as the number of nodes and models in the network. Hence, for very large datasets with high dimensionality, the number of classes and the number of models can also become large and itself introduce scalability issues. Additionally, the global prediction

model is trained only using a subset of the total available data volume, in the form of a validation set of instances independent to those used to train the aggregate models at each node. Effective stacking addresses these problems by averaging the aggregate model predictions from other node to create a set that acts as the training set at a local node. Additionally, this set is used as a validation set for the global model. The meta-level dataset will hence be of the form:

$$\left[\frac{1}{N}\sum_{i=1}^{N} y_{i_1}(x), \frac{1}{N}\sum_{i=1}^{N} y_{i_2}(x), \ldots, \frac{1}{N}\sum_{i=1}^{N} y_{i_C}(x); \text{class}(x)\right] \tag{8.63}$$

Using this approach means that the meta-level dataset does not increase in size for a given number of C classes to be predicted, as the outputs from the N local models are simply averaged. From here, N global classifiers are trained, one at each node in the network, which represents the combined knowledge of all models in the network, other than the one local to the node, in terms of the local subset of data used. Final model predictions are made using the N individual predictions, typically using the sum rule.

8.5.7 Distributed Boosting

Distributed boosting applies the Adaptive Boosting method described in Sect. 8.2.2.4.2 to the context of D&PL. This process is iterative, through rounds $t \in \{1, \ldots, T\}$ of updating a model, where each round uses a different sampled training set of distribution D_t. At each iteration, this set contains more instances with a higher applied weighting for incorrect predictions, as per the boosting method. At each iteration, a model computes a hypothesis h_t, all of which are used to determine a final hypothesis h_f. While the rounds of boosting take place, node j in the network retains a locally stored distribution of instances $\Delta_{j,t}$ and local weights apply to the instances $w_{j,t}$. Combined, these reflect the prediction accuracy of the node, where instances with high weights are known to not be predicted accurately. As the iterations progress, by slowly merging the data subsets from the distributed nodes in the network, the global distribution is emulated for better final, global prediction accuracy.

The following method is used to implement the distributed boosting strategy, where every step in the method is performed at each node in the network. Node $j \in \{1, \ldots, N\}$ is provided with the boosted set of instances $S_j = \left\{\left(x_{j,1}, y_{j,1}\right), \ldots, \left(x_{j,m_j}, y_{j,m_j}\right)\right\}, x_{j,i} \in X_j$, each of which is labeled $y_{j,i} \in Y_j = \{1, \ldots, C\}$. Let $B_j = (i, y_j) : i = 1, \ldots, m, y \neq y_{j,i}$.

1. We begin by initializing the subset distribution $\Delta_{j,1}$ over the instances by $\Delta_{j,1} = \frac{1}{n}$.
2. A temporary version of the global distribution for the $t = 1$ iteration, $D_{j,1}$, is generated by initializing the jth interval within $D_{j,1}$:

$$\left[\sum_{p=1}^{j-1} m_p + 1, \sum_{p=1}^{j} m_p\right] \tag{8.64}$$

With values $\frac{1}{m_j}$.

3. $D_{j,1}$ is normalized with a normalization factor.
4. The following steps are performed for each iteration of the learning process $t = 1, 2, \ldots, T$:
   (a) A model $L_{j,t}$ is trained from instances drawn from the distribution $D_{j,1}$.
   (b) This model is broadcast to and locally stored at each other node in the network.
   (c) An ensemble model ($E_{j,t}$) is built to combine all the predictions from the aggregate classifiers to compute the hypothesis $h_{j,t} : X \times Y \to [0, 1]$.
   (d) The loss function for the hypothesis $h_{j,t}$ is calculated using the following equation:

$$\epsilon_{j,t} = \frac{1}{2} \sum_{(i,y)\in B_j} \Delta_{j,t}(i, y)\left(1 - h_{j,t}\left(x_{j,i}, y_{j,i}\right) + h_{j,t}\left(x_{j,i}, y_j\right)\right) \tag{8.65}$$

   (e) Following, the term $V_{j,t}$ is calculated:

$$V_{j,t} = \sum_{(i,y_j)\in B_j} w_{j,t}(i, y_i) \tag{8.66}$$

   where $w_{j,t} = \frac{1}{2}\frac{1-h_{j,t}(x_{j,i},y_j)+h_{j,t}(x_{j,i},y_j)}{\mathrm{acc}_j^p}$, $p \in 0, 1, 2$

   (f) $V_{j,t}$ is broadcast to all other nodes in the network. Given that this involves the combination of all weight vectors $w_{j,t}$, this may require a very large transfer of information volume, depending on the size of the distributed datasets. In order to reduce the data volume and hence time for broadcasting, the summed values from each node can instead be used, at the cost of lost model fidelity.
   (g) A weight vector $U_{j,t}$ is defined, where the weight factor $w_{j,t}$ is the jth interval $\left[\sum_{p=1}^{j-1} m_p + 1, \sum_{p=1}^{j} m_p\right]$. The values in the qth interval $q \in 1, \ldots, N, q \neq j$ are set to $\frac{V_{q,t}}{m_q}$

(h) Finally, the global distribution is updated such that:

$$D_{j,t+1}(i, y_i) = \frac{D_{j,t}(i, y_i)}{Z_{j,t}} \beta_{j,t}{}^{U_{j,t}(i,y_i)} \tag{8.67}$$

where $Z_{j,t}$ is a normalization constant for the distribution.

By following the above procedure, each node in the network retains $D_{j,t}$ (the locally approximated version of the global distribution) and $\Delta_{j,1}$ (the local distribution), for each iteration, t. Hence, the instance samples for boosting are collected from $D_{j,t}$, in order to emulate the process as if it were sampling from the true global, centralized distribution. The boosting process is supported by the weight vectors $w_{j,t}$ to train using instances that are more difficult to learn.

8.6 Transfer Learning

Transfer learning leverages the heterogeneity typical of big data, stemming from broad varieties of big data, to apply pretrained machine learning models to different domains, tasks, and data distributions. This is, in particular, driven by needs stemming from velocity characteristics of big data, where the training of new models from the ground up can take significant resources and time. By identifying a collection of domains, tasks, and data distributions with similarities, a wide range of models can be trained using transfer learning in a more efficient manner. This approach can be used for regression, classification, or clustering-based learning. Figure 8.15 shows a typical set of training performance curves, for ML models with and without transfer learning. The model with transfer learning both begins at a higher performance and converges toward an optimal solution faster. This increase in training and predictive performance can be attributed to the fact that a carefully selected older, template model can contain parameters that do not have to be changed much, in order to converge towards an optimal solution. Conversely, initialising a model with random numbers or all zeros, can instead lead to a larger volume of data being needed to complete the training process [81].

Transfer learning has been successfully applied in many fields, including image-based style transfer [82], the construction of informative priors, large-scale document classification [83], classification of cross-domain text [84], network localization [85], artificial intelligence agents in video games [86], and NLP [87].

Depending on similarities between the domains, tasks, and distributions of the originally trained model and the target application, inductive learning, transductive learning, and unsupervised transfer learning can be employed. The latter is currently an active area of new and challenging research.

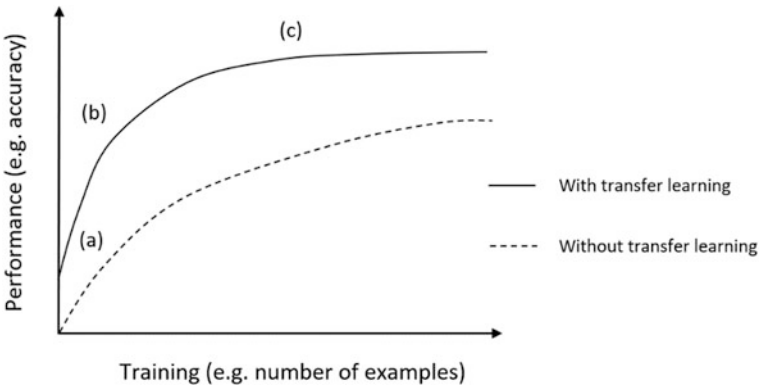


Fig. 8.15 Transfer learning takes pretrained models on one task, and applies it to a new task, to speed up the training process for the new task (i.e., less time, fewer training examples needed). Transfer learning provides three advantages for the training process, as depicted above: (a) a higher initial performance of the model, (b) a greater rate of performance improvement, and (c) a higher asymptote on the performance curve that is reached within a faster training timeframe

8.6.1 Inductive Learning

Transductive transfer learning aims to improve the performance of a target predictive model $f_T(.)$, by leveraging the knowledge in a source domain and learning task, $\mathcal{D}_S$ and $\mathcal{T}_S$, respectively. This is in pursuit of application to a target domain $\mathcal{D}_T$ and target task $\mathcal{T}_T$, where $\mathcal{T}_S \neq \mathcal{T}_T$. To this end, some labeled datapoints within the target domain are required to provide the mechanism to induce the target predictive model. The two primary contexts for this approach are: when labeled data in the source domain is also available, or when instead unlabeled data in the source domain is available. This approach allows for the transfer of knowledge via use of instances, feature representations (supervised or unsupervised), parameters or relational knowledge from the source to the target domain.

TrAdaBoost is a variant of the Adaptive Boosting method, applied to inductive learning problems, to use instances from a source domain for the training of the target predictive model. This approach is useful, as it helps sort the data to be more relevant to the target task. The algorithm iteratively applies weightings the source domain data, reducing the influence of the irrelevant source data on the training process, while increasing the influence of the good source data. Additionally, the incorrectly predicted instances from the target domain are given the same reweighting in the normal AdaBoost algorithm and do not need the extra attention required of the source domain instances.

Feature representation transfer is an approach to inductive transfer learning, where good feature space representations are learned to maximize prediction model accuracy. This can be performed in the context of supervised or unsupervised learning. For cases where the source domain contains sufficient volumes of labeled

data, supervised learning methods can be used. This involves learning a latent representation space that is commonly shared across tasks in the source and target domains. Common features in this space are learned via the following minimization-based objective function:

$$\underset{A,\,U}{\arg\min} \sum_{t\in[T,S]} \sum_{i=1}^{n_t} L\left(y_{t_i}, \left\langle a_t, U^{\mathrm{T}} x_{t_i}\right\rangle\right) + \gamma \|A\|_{2,1}^2, U \in O^d \tag{8.68}$$

where S and T are the source and target tasks, respectively. U is mapping function to transform the data into the lower-dimensional representation space and the (r, p)-norm of the matrix A is given by:

$$\|A\|_p^r = \left(\sum_{i=1}^{d} \left\|a^i\right\|_r^p\right)^{\frac{1}{p}} \tag{8.69}$$

Conversely, an unsupervised approach to learning feature representations uses the method of sparse coding to learn high-level features. To achieve this, higher-level bias vectors are learned using the following minimization-based objective function:

$$\min_{a,\,b} \sum_i \left\| x_{S_i} - \sum_j a^j{}_{S_i} b_j \right\|_2^2 + \beta \left\| a_{S_i} \right\|_1 \tag{8.70}$$

Such that the constraint $\|b_j\|_2 \leq 1,\ \forall\ j \in 1, \ldots, s$ is satisfied, where $b = \{b_1, b_2, \ldots, b_S\}$ are the bias vectors, $a^j{}_{S_i}$ are the new representations of the vectors for a given instance x_{S_i}. β is a coefficient that provides trade-off between the regularization and feature space reconstruction terms. Once the bias vectors are learned, a second optimization problem can be then solved to define the embedded higher-level features:

$$a^*{}_{T_i} = \underset{a_{\mathrm{T}_i}}{\arg\min} \left\| x_{T_i} - \sum_j a^j{}_{T_i} b_j \right\|_2^2 + \beta \left\| a_{T_i} \right\|_1 \tag{8.71}$$

Using this result, $a^*{}_{T_i}$ can be subjected to discriminative algorithms, with labeled instances, for either classification or regression tasks in the target domain.

Parameter learning in the context of inductive transfer learning assumes that models between the source and target domains share parameters or hyperparameter prior distributions. Regularization and Bayesian frameworks from multitask learning, which aim to learn source and target tasks simultaneously, are leveraged to this end. However, in transfer learning, the single aim is to improve predictive performance in the target domain, simply by utilizing the source domain data. This means that the weights of the loss function can be different, for different domains.

Hence, to achieve better performance in the target domain, larger weights may be applied to this loss function. The use of hierarchical Bayesian [88], Gaussian Process [89], and SVM [90] frameworks has been employed for the transfer of prior distributions, usually via an inter-task covariance matrix, for inductive parameter transfer learning.

8.6.2 Transductive Learning

Transductive transfer learning is performed in the context of the same source and target tasks being performed, with different domains, where the aim is to improve the predictive capabilities of a target model $f_T(.)$. The function is applied within $\mathcal{D}_T$ and the knowledge is garnered from source domain $\mathcal{D}_S$ and source task $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$.

In the use of instances across tasks to improve the model $f_T(\cdot)$'s predictions, an optimal model is sought for the target domain, which effectively minimizes the expected risk. This is given by:

$$\theta^* = \underset{\theta \in \Theta}{\arg\min} \sum_{(x,y) \in D_T} P(D_T)\, l\,(x,\, y,\, \theta) \tag{8.72}$$

where the loss function is given by $l(x, y, \theta)$ and depends on the parameter θ. However, with an absence of labeled data in the target domain, a model cannot be trained and hence $P(D_T)$ is intractable in the above equation. If $P(D_T) = P(D_S)$, then the substitution may simply be made, otherwise a model with the ability for high generalization needs to be trained. Here, the transductive transfer learning problem is solved by estimating the following for each instance:

$$\frac{P_T\left(x_{T_i},\, y_{T_i}\right)}{P_S\left(x_{S_i},\, y_{S_i}\right)} = \frac{P\left(x_{S_i}\right)}{P\left(x_{T_i}\right)} \tag{8.73}$$

where the left-hand side of the above expression are the weights corresponding to the penalty values applied to each instance $\left(x_{S_i},\, y_{S_i}\right)$ within the source domain, a predictive model can be learned for the target domain. The right-hand side of the above expression can be estimated through various means, such as a kernel-mean matching (KMM) algorithm.

Feature representation learning in transductive transfer learning is conducted in unsupervised regimes. This can be achieved through the use of a structural correspondence learning (SCL) algorithm, which extracts some relevant features from unlabeled data in the target domain to assist in reducing the difference between both domains and maximize value. SCL defines a set of m domain-specific pivot features on the data from both the source and target domains. Following, these

features are removed from the data and each is treated as a new label vector. To this end, we first assume that a linear classifier can solve the problem, given by:

$$f_l(x) = \text{sgn}\left(w_l{}^{\text{T}}.x\right), l = 1, \ldots, m \tag{8.74}$$

where sgn() is the signum function, m is the number of pivot features, and w_l is the parameters vector for classifier l, which can be stacked into a matrix $W = [w_1 w_2 \ \ldots \ w_m]$. $W = UDV^{\text{T}}$ is solved using singular value decomposition, in order to obtain $\theta = U_{[1:h,:]}{}^{\text{T}}$, where h is number of shared features between the source and target domains. From here, discriminative algorithms can be employed to build models directly.

8.7 Active Learning

Active learning seeks to shift machine learning strategies for big data from large volumes of unlabeled data, which is time consuming and difficult, to that which uses labeled data. Hence, active learning is most broadly applicable to supervised and semi-supervised machine learning scenarios. Given that labeled data is expensive to obtain, active learning minimizes the cost associated with this activity, by identifying a subset of points in the original data distribution most critical in achieving a desired constraint of accuracy. Figure 8.16 provides a visually intuitive example of active learning. The process of active learning identifies key points in a dataset, which if labeled, can provide the greatest value in classifying the remainder of the dataset. Similar approaches are taken for regression modeling.

Fields such as medical image classification [91] and biological DNA identification have found particular success in applying active learning techniques. In the

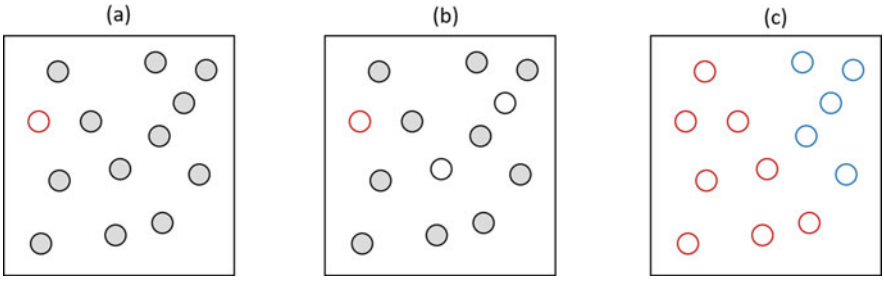


Fig. 8.16 The concept of active learning is intuitively illustrated in this figure with respect to a classification task. Here, in (**a**) a labeled instance belonging to class 1 (red) is shown, with the other points unknown and depicted in gray. Through active learning, two points are identified in (**b**), shown as white circles, which are deemed to provide the most statistically significant impact for the learning process once classified. With those key points labeled as class 1 (red) and class 2 (blue), other machine learning methods (e.g., *k*-Means) can be applied to more easily classify the remaining points into their respective classes

case of the latter, King et al. [92] demonstrated the ability to employ autonomous biological experimentation to discover the metabolic pathways for the yeast species *Saccharomyces cerevisiae*, which not only had the physical experiments conducted by a laboratory robot, but all experimental configurations were selected using an active learning approach. Other areas include NLP [93], experimental design [94], image processing [95], database searching [96], and network intrusion detection [97].

The goals of active learning are achieved by utilizing three different types of query strategies, namely, stream-based selective sampling, membership query synthesis, and pool-based sampling. For each strategy, the optimal selection of instances to be labeled can follow one of many frameworks. These include uncertainty sampling, query-by-committee, expected model change, expected error reduction, variance reduction, and density-weighted methods [98].

Membership query synthesis is the simplest type of active learning, whereby the learner requests labels for any given unlabeled instance. This is typically done assuming the learner model's queries are independent of each other and not part of an underlying distribution or pattern (i.e., it is random). For large datasets that describe highly complex and nonlinear system, this approach does not necessarily scale well. Additionally, given that many of these queries may not actually contain useful information to classify, especially when part of big datasets with limited veracity, stream-based and pool-based scenarios can be employed to address these limitations.

In the case of stream-based queries, the learner model samples instances one at a time from the dataset and on a case-by-case basis, decides whether a query should be submitted to have it labeled or not. Pool-based queries are similar in nature, although rather than one at a time, an entire collection of candidates is selected, of which each is evaluated and subsequently ranked before selecting the best query for labeling. Assuming that the distribution of the source dataset is nonuniform (in which case it will act akin to the membership query process), the queries will come from an underlying distribution and will provide value.

Different frameworks can be used in determining the relative value for querying a given instance in the learning process. The simplest method would be to define a minimum threshold for informativeness, where every point that meets that criteria (i.e., above the threshold) is submitted as a query. Alternatively, if two or more models are trained, comparisons between their ability to agree or disagree on predictions using labeled versus unlabeled data can identify regions in the feature space where additional labeled data will reconcile the models and provide greatest value. This region of uncertainty can be explicitly calculated, but it computationally expensive and must be reperformed after each new query is made, as the model hyperparameters will have been updated. For large datasets, statistical frameworks can be used on randomly selected instances to determine the comparative likelihood of increasing the value by labeling.

Uncertainty sampling is one statistical framework used to maximize the expected value of a submitted query. Methods such as least confidence, margin sampling, and entropy can be employed as measures of uncertainty. For example, least confidence uses the following equation to select the optimal instance for labeling, x:

$$x^*_{\mathrm{LC}} = \underset{x}{\arg\max}\, 1 - P_\theta\left(\hat{y}|x\right) \tag{8.75}$$

where $\hat{y}$ is the class label with the highest posterior probability and θ is the model. This expression can be intuitively understood as the model's belief that instance x being considered will be mislabeled, based on the posterior probability of each instance. Hence, the instance in which the model has least confidence in a correct prediction is selected. The selection criteria for margin sampling and entropy as measures of uncertainty are given as follows, respectively:

$$x^*_{\mathrm{M}} = \underset{x}{\arg\min}\, P_\theta\left(\hat{y}_1|x\right) - P_\theta\left(\hat{y}_2|x\right) \tag{8.76}$$

$$x^*_{\mathrm{E}} = \underset{x}{\arg\max} - \sum_i P_\theta\left(y_i|x\right) \log P_\theta\left(y_i|x\right) \tag{8.77}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the classes with the first and second highest probability of being selected as labels, respectively, and y_i has a range of all potential labels within the problem.

Compared to more random sampling methods, uncertainty sampling assists a learning model in converging toward a solution faster. Figure 8.17 illustrates this with two curves, where the uncertainty sampling methods harnesses greater value in the learning process, for each instance that is queried. For very large datasets with a high degree of embedded complexity, implementing this strategy can significantly reduce the total cost needed to achieve required model prediction accuracy via labeling instances.

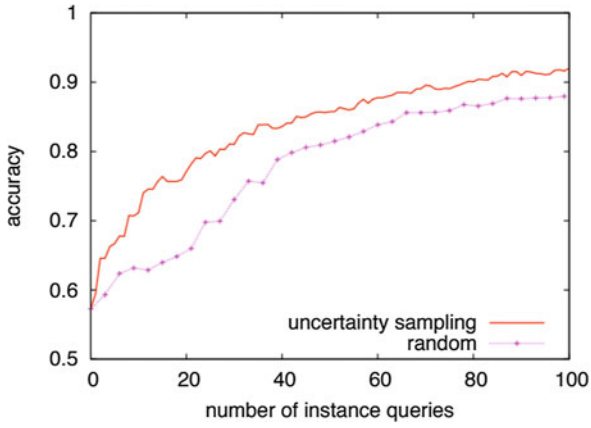


Fig. 8.17 A set of learning curves for the classification of the text classification of words baseball vs. hockey. The model that employed uncertainty sampling consistently showed better accuracy for a given number of queried instances, compared to the model that randomly queried instances [99]

Other frameworks in active learning exploit varying approaches in determining the highest value instance to query. For example, query-by-committee approaches define a set of competing set of models $\mathcal{C} = \left\{\theta^{(1)}, \ldots, \theta^{(C)}\right\}$, called a committee, which provide predictions on a set of instances being considered for querying. The instance that exhibits the greatest level of disagreement in the committee is selected as the final candidate for querying. There are different methods for measuring the level of disagreement in the committee, such as the vote entropy method as follows:

$$x^*_{\text{VE}} = \underset{x}{\arg\max} - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \tag{8.78}$$

where C is the number of models in the committee and $V(y_i)$ is the number of votes that each label receives from each member of the committee.

Other frameworks for selecting candidate instances for labeling are expected error reduction, variance reduction, and density weight methods.

8.8 Kernel-Based Learning

Kernel-based learning uses nonlinear kernel functions, embedded within classical and well-known linear statistical techniques (e.g., support vector machines) to project the input space of a dataset onto a higher-dimensional feature space. This allows for a higher expressive power, owing to the greater number of features, and the opportunity to perform multiple analyses with the same dataset [100]. However, this can be at the cost of higher computational complexity, especially if the model is not well designed for the task at hand. This is applicable for regression, classification, and clustering modeling tasks, as all methods ultimately seek to extract relational patterns from the input dataset.

At first glance, it may seem counterintuitive to create a higher dimensional representation for a big dataset. With a higher number of dimensions to represent the input space, not only does the explicit size of the dataset increases but the statistical principle, oft-referred to as the curse of dimensionality, may lead to the impression that the difficulty of estimation problems from the dataset will increase. With respect to the latter, an increase in the input space dimensions N, requires exponentially many patterns between variables to sample the space for a sufficient mathematical description, especially for the purposes of modeling and prediction. However, this is not always the case, as the selection of a suitable mapping between $\mathcal{X} \rightarrow \mathcal{F}$ can yield a set of simpler, such as linear, relationships in the new feature space $\mathcal{F}$. However, it can be seen that this becomes a trade-off for users in taking this approach, between minimizing the impact of increasing the size of the dataset and maximizing the computational efficiency realized from the discovery of simple class of decision rules in the higher dimensional feature space for modeling. Figure 8.18 provides visual insight into how this process can lead to simpler prediction models.

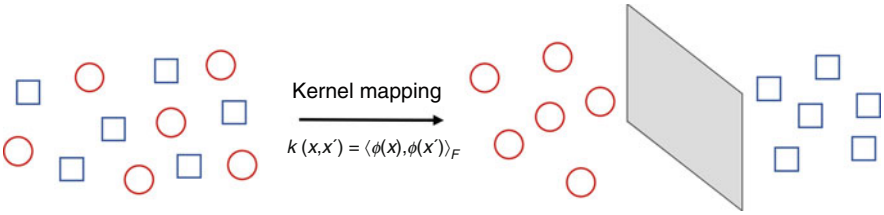


Fig. 8.18 Visual demonstration of the process of kernel-based learning. Data that may be represented in 2D feature space and is seemingly unclassifiable, can be projected into 3D space using their embedded nonlinearity, allowing for the much easier classification in this new feature space, using a computationally simpler linear hyperplane (now described in $\mathbb{R}^3$) as illustrated

The goal of kernel-based learning is to employ a kernel function, which maps the original dataset within feature space $\mathcal{X}$ to a space of higher dimension $\mathcal{F}$, via a nonlinear mapping Φ, as follows:

$$\Phi : \quad \mathcal{X} \to \mathcal{F}, x \mapsto \Phi(x) \tag{8.79}$$

This is achieved through the use of a kernel function, k, which performs the transformation without needing to know the mappings between $\mathcal{X} \to \mathcal{F}$ explicitly. This operation is expressed as:

$$k\left(x, x'\right) = \left\langle \Phi(x), \Phi\left(x'\right)\right\rangle_{\mathcal{F}}, \forall x, x \in \mathcal{X} \tag{8.80}$$

Polynomial and Gaussian kernels are the most widely used kernel functions and empirical evidence suggests that, although the selection of the best kernel for a given application is not formally defined, these two generally provide good results. A selection of some of the more common kernel functions includes the Gaussian radial basis function, polynomial, sigmoidal, inverse multiquadric, which are as follows, respectively:

$$\text{Gaussian radial basis} : k\left(x, y\right) = \exp\left(\frac{-\|x - y\|^2}{c}\right) \tag{8.81}$$

$$\text{Polynomial} : ((x.y) + \theta)^d \tag{8.82}$$

$$\text{Sigmoidal} : \tanh\left(k(x.y) + \theta\right) \tag{8.83}$$

$$\text{Inverse multiquadric}: \frac{1}{\sqrt{\|x-y\|^2+c^2}} \tag{8.84}$$

Once the dataset has been mapped $\mathcal{X} \rightarrow \mathcal{F}$, typical learning algorithms can be applied within the new feature space for a faster and computationally feasible training process. Applications in which kernel-based learning offers advantages are usually those with inherently high degrees of complexity and are either intractable within $\mathcal{X}$ alone, or the computational time saved with the approach opens up successful opportunities for applications that would otherwise be infeasible. In the case of the former, adaptive multi-regression [101], as well as convexly constrained parameter and function estimation [102] have found success, whereas in the latter, online classification [103] has benefited from kernel-based learning.

Kernel-based learning has found many areas of applications, due to its ability to express information from highly complex datasets. This includes de-noising in signal processing [104], image classification [105], and biological engineering [106].

8.9 Conclusion

The availability and prevalence of big data is both a challenge and opportunity for the future of modeling. There are many difficulties that arise from handling such datasets, typically associated with the volume of the data, variety in the input space, velocity of the input data-streams, and required predictions, as well as the veracity of the data used for modeling. However, the benefits that can be leveraged from big data, namely the significant boots in productivity from better predictions and hence decision-making, are too great to be ignored. Hence, in order to harness the value of big data in the face of these challenges, six approaches have been presented. Combining these approaches with classical machine learning methods, effective and efficient models can be constructed for prediction and decision-making purposes. Many of the approaches require an empirical view to select optimal models and associated parameters and with such a broad range of tools available, this is no small task. Hence, the study of different techniques for big data is of great importance and has been reviewed here.

References

1. Dobre, T. G., & Sanchez Marcano, J. G. (2007). *Chemical engineering: Modelling, simulation and similitude*. Weinheim: Wiley-VCH Verlag GmbH & KGaA.
2. Rodrigues, A. E., & Minceva, M. (2005). Modelling and simulation in chemical engineering: Tools for process innovation. *Computers & Chemical Engineering, 29*, 1167–1183.

3. Rasmuson, A., Andersson, B., Olsson, L., & Andersson, R. (2014). *Mathematical modeling in chemical engineering*. New York: Cambridge University Press.
4. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications, 19*(2), 171–209.
5. Alvaro, A., Manadhata, P., & Rajan, S. (2013). *Big data analytics for security intelligence*. Cloud Security Alliance. Retrieved from https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf
6. McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., & Barton, D. (2012). Big data. The management revolution. *Harvard Business Review, 90*(10), 61–67.
7. Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems, 47*, 98–115.
8. IBM. (2015). *What is the big data analysis*. Retrieved from http://www-01.ibm.com/software/data/infosphere/hadoop/what-is-big-data-analytics.html
9. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). *Big data: The next frontier for innovation, competition, and productivity*. McKinsey & Company. Retrieved September 12, 2018, from https://bigdatawg.nist.gov/pdf/MGI_big_data_full_report.pdf
10. Lund, S. (2013). *Game changers: Five opportunities for US growth and renewal*. McKinsey & Company. Retrieved September 26, 2018, from https://www.mckinsey.com/~/media/McKinsey/Featured%20Insights/Americas/US%20game%20changers/MGI_US_game_changers_Executive_Summary_July_2013.ashx
11. Chen, C., & Zhang, C. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Journal of Information Sciences, 275*, 314–347.
12. Dijcks, J. P. (2012). Oracle: Big data for the enterprise. *Oracle White Paper*.
13. Abawajy, J. (2015). Comprehensive analysis of big data variety landscape. *International Journal of Parallel, Emergent and Distributed Systems, 30*(1), 5–14.
14. Minelli, M., Chambers, M., & Dhiraj, A. (2012). *Big data, big analytics: Emerging business intelligence and analytic trends for today's businesses*. Wiley.
15. IBM. (2015). *Bringing big data to the enterprise*. Retrieved from http://www-01.ibm.com/software/in/data/bigdata/
16. Sathi, A. (2012). *Big data analytics: Disruptive technologies for changing the game*. MC Press.
17. Khayyam, H., Naebe, M., Zabihi, O., Zamani, R., Atkiss, S., & Fox, B. (2015). Dynamic prediction models and optimization of Polyacrylonitrile (PAN) stabilization processes for production of carbon fiber. *IEEE Transactions on Industrial Informatics, 11*, 887–896.
18. Alag, S., Agogino, A., & Morjaria, M. (2001). A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Special Issue on AI in Equipment Service, 15*(4), 307–319.
19. Hernán, D., Loaiza, B. H., Caicedo, E., Ibarra-Castanedo, C., Bendada, A. H., & Maldague, X. (2009). Defect characterization in infrared non-destructive testing with learning machines. *Non-Destructive Testing and Evaluation International, 42*(7), 630–643.
20. Khayyam, H., Fakhrhoseini, S. M., Church, J. S., Milani, A. S., Bab-Hadiashar, A., & Jazar, R. N. (2017). Predictive modelling and optimization of carbon fiber mechanical properties through high temperature furnace. *Applied Thermal Engineering, 125*, 1539–1554.
21. Kabir, G., Sadiq, R., & Tesfamariam, S. (2016). A fuzzy Bayesian belief network for safety assessment of oil and gas pipelines. *Structure and Infrastructure Engineering, 12*(8), 874–889.
22. Madsen, A. L., & Kjærulff, U. B. (2007). *Applications of HUGIN to diagnosis and control of autonomous vehicles. Studies in fuzziness and soft computing*. Berlin: Springer.
23. Fernandes, H., Zhang, H., Figueiredo, A., Malheiros, F., Ignacio, L. H., Sfarra, S., & Guimaraes, G. (2018). Machine learning and infrared thermography for fiber orientation assessment on randomly-oriented strands parts. In *Sensors 2018* (pp. 288–306).

24. Hu, H., Wen, Y., Chua, T., & Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access, 2*, 652–687.
25. Chen, X. W., & Lin, X. (2014). Big data deep learning: Challenges and perspectives. *IEEE Access, 2*, 514–525.
26. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202.
27. Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery, 2*(2), 169–194.
28. Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques*. New York: Elsevier.
29. Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks, 10*, 988–999.
30. Abd, A. M., & Abd, S. M. (2017). Modelling the strength of lightweight foamed concrete using support vector machine (SVM). *Case Studies in Construction Materials, 6*, 8–15.
31. Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning, 46*, 131–159.
32. Cholette, M. E., Borghesani, P., Gialleonardo, E. D., & Braghin, F. (2017). Using support vector machines for the computationally efficient identification of acceptable design parameters in computer-aided engineering applications. *Expert Systems with Applications, 81*, 39–52.
33. Rokach, L., & Maimon, O. (2014). *Data mining with decision trees: Theory and applications*. World Scientific.
34. Pourret, O., Naim, P., & Marcot, B. (2008). *Bayesian networks: A practical guide to applications*. West Sussex: Wiley.
35. Shatovskaya, T., Repka, V., & Good, A. (2006). Application of the Bayesian networks in the informational modeling. In *2006 international conference—Modern problems of radio engineering, telecommunications, and computer science* (pp. 108–108).
36. Catal, C., Sevim, U., & Diri, B. (2011). Practical development of an eclipse-based software fault prediction tool using naïve Bayes algorithm. *Expert Systems with Applications, 38*(3), 2347–2353.
37. Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.
38. Janssens, D., Wets, G., Brijs, T., Vanhoof, K., Arentze, T., & Timmermans, H. (2006). Integrating Bayesian networks and decision trees in a sequential rule-based transportation model. *European Journal of Operational Research, 175*, 16–34.
39. Cosma, G., Brown, D., Archer, M., Khan, M., & Graham Pockley, A. (2017). A survey on computational intelligence approaches for predictive modeling in prostate cancer. *Expert Systems with Applications, 70*, 1–19.
40. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News, 2*(3), 18–22.
41. Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.
42. Datla, M. V. (2015). Bench marking of classification algorithms: Decision trees and random forests—A case study using R. In *2015 international conference on trends in automation, communications and computing technology (I-TACT-15)* (pp. 1–7).
43. Schapire, R. E. (2003). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*. Springer.
44. Zhang, C. X., Zhang, J. S., & Zhang, G. Y. (2008). An efficient modified boosting method for solving classification problems. *Journal of Computational and Applied Mathematics, 214*, 381–392.
45. Li, F., & Pengfei, L. (2013). The research survey of system identification method. In *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*.
46. Hunter, D., Yu, H., Pukish, M. S., III, Kolbusz, J., & Wilamowski, B. M. (2012). Selection of proper neural network sizes and architectures—A comparative study. *IEEE Transactions on Industrial Informatics, 8*, 228–240.

47. Davim, P. (2012). *Computational methods for optimizing manufacturing technology models and techniques*. Hershey: Engineering Science Reference.
48. Cherkassky, V., & Mulier, F. M. (2007). *Learning from data: Concepts, theory, and methods*. Chichester: Wiley.
49. Kermani, B. G., Schiffman, S. S., & Nagle, H. T. (2005). Performance of the Levenberg–Marquardt neural network training method in electronic nose applications. *Sensors and Actuators B: Chemical, 110*, 13–22.
50. Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks, 5*, 989–993.
51. Keerthi, S. S., & Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation, 15*, 1667–1689.
52. Gunn, S. R. (1998). *ISIS Technical Report—Support vector machines for classification and regression*. Retrieved July 8, 2018, from http://svms.org/tutorials/Gunn1998.pdf
53. Vapnik, V., Golowich, S. E., & Smola, A. (1996). Support vector method for function approximation, regression estimation, and signal processing. In *Advances in neural information processing systems* (Vol. 9).
54. Hu, W., Yan, L., Liu, K., & Wang, H. (2016). A short-term traffic flow forecasting method based on the hybrid PSO-SVR. *Neural Processing Letters, 43*, 155–172.
55. Cai, Z. J., Lu, S., & Zhang, X. B. (2009). Tourism demand forecasting by support vector regression and genetic algorithm. In *2nd IEEE international Conference on Computer science and information technology (ICCSIT) 2009* (pp. 144–146).
56. Vapnik, V. (1998). *Statistical learning theory*. Wiley.
57. Tu, W., & Sun, S. (2012). Cross-domain representation-learning framework with combination of class-separate and domain-merge objectives. In *Proceedings of the 1st International Workshop on Cross Domain Knowledge Discovery in Web and Social Network Mining, 2012* (pp. 18–25). Beijing.
58. Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences, 99*(12), 7821–7826.
59. Zhang, C., Zhang, K., Yuan, Q., Peng, H., Zheng, Y., Hanratty, T., Wang, S., & Han, J. (2017). Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. In *Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017* (pp. 361–370).
60. Natarajan, N., & Dhillon, I. S. (2014). Inductive matrix completion for predicting gene–disease associations. *Bioinformatics, 30*(12), 60–68.
61. Wang, S., Tang, J., Aggarwal, C., & Liu, H. (2016). Linked document embedding for classification. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management 2016* (pp. 115–124).
62. Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016* (pp. 1225–1234).
63. Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016* (pp. 1105–1114).
64. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web 2015* (pp. 1067–1077).
65. LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature, 521*, 436–444.
66. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference On Computer Vision and Pattern Recognition, 2005 (CVPR 2005)*, IEEE (Vol. 1. pp. 886–893).
67. Hinton, G., Deng, L., Yu, D., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Dahl, G., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine, 29*(6), 82–97.

68. Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *INTERSPEECH. ISCA* (pp. 437–440).
69. Suthaharan, S. (2013). Big data classification: Problems and challenges in network intrusion prediction with machine learning. In *ACM Sigmetrics: Big Data Analytics Workshop*. Pittsburgh: ACM.
70. Fan, C., Xiao, F., & Zhao, Y. (2017). A short-term building cooling load prediction method using deep learning algorithms. *Applied Energy, 195*, 222–233.
71. Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications, 83*, 187–205.
72. Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv technical report. Retrieved August 12, 2018, from http://arxiv.org/pdf/1409.1556
73. Mathworks. (2018). *Practical deep learning examples with MATLAB*. Retrieved August 14, 2018, from https://au.mathworks.com/campaigns/offers/deep-learning-examples-with-matlab.html
74. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507.
75. Peteiro-Barral, D., & Guijarro-Berdiñas, B. (2013). A survey of methods for distributed machine learning. *Progress in Artificial Intelligence, 2*, 1–11.
76. Giordana, A., & Neri, F. (1995). Search-intensive concept induction. *Evolutionary Computation, 3*(4), 375–416.
77. Tsoumakas, G., & Vlahavas, I. (2009). Distributed data mining. In J. Erickson (Ed.), *Database technologies: Concepts, methodologies, tools, and applications* (pp. 157–171). Hershey: IGI Global.
78. Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Cambridge: The MIT Press.
79. Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis in Machine Intelligence, 20*(3), 226–239.
80. Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning, 36*(1), 85–103.
81. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge Data Engineering, 22*(10), 1345–1359.
82. Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *ECCV 2016: European Conference on Computer Vision* (pp. 694–711).
83. Dai, W., Xue, G., Yang, Q. & Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proceedings of the 22rd AAAI Conference on Artificial Intelligence*, Vancouver, Canada (pp. 540–545).
84. Ling, X., Xue, G.-R., Dai, W., Jiang, Y., Yang, Q., & Yu, Y. (2008). Can Chinese web pages be classified with English data source? In *Proceedings of the 17th International Conference on World Wide Web* (pp. 969–978). Beijing: ACM
85. Pan, S. J., Kwok, J. T., Yang, Q., & Pan, J. J. (2007). Adaptive localization in a dynamic WiFi environment through multi-view learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence* (pp. 1108–1113). Vancouver.
86. Kuhlmann, G. & Stone, P. (2007). Graph-based domain mapping for transfer learning in general games. In *18th European Conference on Machine Learning*, ser. Lecture Notes in Computer Science (pp. 188–200). Warsaw: Springer.
87. Jiang, J., & Zhai, C. (2007). Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (pp. 264–271). Prague: Association for Computational Linguistics.
88. Lawrence, N. D., & Platt, J. C. (2004). Learning to learn with the informative vector machine. In *Proceedings of the 21st International Conference on Machine Learning*. Banff: ACM.

89. Schwaighofer, A., Tresp, V., & Yu, K. (2005). Learning Gaussian process kernels via hierarchical Bayes. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems* (pp. 1209–1216). Cambridge: MIT Press.
90. Evgeniou, T., & Pontil, M. (2004). Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 109–117). Seattle: ACM.
91. Hoi, S. C. H., Jin, R., Zhu, J., & Lyu, M. R. (2006). Batch mode active learning and its application to medical image classification. In *Proceedings of the International Conference on Machine Learning (ICML) 2006* (pp. 417–424).
92. King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Sparkes, A., Whelan, K. E., & Clare, A. (2009). The automation of science. *Science, 324*(5923), 85–89.
93. Berger, A. L., Della Pietra, V. J., & Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics, 22*(1), 39–71.
94. Flaherty, P., Jordan, M., & Arkin, A. (2006). Robust design of biological experiments. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)* (Vol. 18, pp. 363–370). MIT Press.
95. Grira, N., Crucianu, M., & Boujemaa, N. (2005). Active semi-supervised fuzzy clustering for image database categorization. In *Proceedings of the ACM Workshop on Multimedia Information Retrieval (MIR)* (pp. 9–16). ACM Press.
96. Hauptmann, A., Lin, W., Yan, R., Yang, J., & Chen, M. Y. (2006). Extreme video retrieval: Joint maximization of human and computer performance. In *Proceedings of the ACM Workshop on Multimedia Image Retrieval* (pp. 385–394). ACM Press.
97. Moskovitch, R., Nissim, N., Stopel, D., Feher, C., Englert, R., & Elovici. Y. (2007). Improving the detection of unknown computer worms activity using active learning. In *Proceedings of the German Conference on AI* (pp. 489–493). Springer Publishing.
98. Settles, B. (2010). *Active learning literature survey*. Madison: University of Wisconsin. Retrieved August 2, 2018, from http://burrsettles.com/pub/settles.activelearning.pdf.
99. Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 331–339). Morgan Kaufmann.
100. Müller, K. R., Mika, S., Rätsch, G., Tsuda, K., & Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks, 12*(2), 181–201.
101. Slavakis, K., Bouboulis, P., & Theodoridis, S. (2012). Adaptive multiregression in reproducing kernel Hilbert spaces: The multiaccess MIMO channel case. *IEEE Transactions on Neural Network Learning Systems, 23*(2), 260–276.
102. Theodoridis, S., Slavakis, K., & Yamada, I. (2011). Adaptive learning in a world of projections. *IEEE Signal Processing Magazine, 28*(1), 97–123.
103. Slavakis, K., Theodoridis, S., & Yamada, I. (2008). Online kernel-based classification using adaptive projection algorithms. *IEEE Transactions on Signal Processing, 56*(7), 2781–2796.
104. Mika, S., Schölkopf, B., Smola, A. J., Müller, K.-R., Scholz, M., & Rätsch, G. (1999). Kernel PCA and de-noising in feature spaces. In *Advances in Neural Information Processing Systems 11* (pp. 536–542). Cambridge: MIT Press.
105. LeCun, Y. A., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Müller, U. A., Säckinger, E., Simard, P. Y., & Vapnik, V. N. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. In *Neural networks: The statistical mechanics perspective* (pp. 261–276). Singapore: World Scientific Publishing.
106. Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., & Müller, K. R. (2000). Engineering support vector machine kernels that recognize translation initiation sites in DNA. *Bioinformatics, 16*, 799–807.

Chapter 9
Genetic Programming Approaches in Design and Optimization of Mechanical Engineering Applications

Hamid Khayyam, Ali Jamali, Hirad Assimi, and Reza N. Jazar

9.1 Introduction

Optimization in engineering can be defined as a methodology to making something as real, robust, functional, as it is possible. It is in the aspect of mathematical modeling to optimal (maximizing or minimizing) of the objective function without violating the constraint(s) as in Eq. (9.1).

The Mathematical Optimization Problem is given as follows:

$$\begin{aligned} &\text{Minimize function } f_0(x) \\ &\text{Subject to constraints } f_i(x) \le b_i \quad i = 1, \ldots, m \end{aligned} \tag{9.1}$$

- $x = (x_1, \ldots, x_n)$: optimization parameters to be addressed
- f_0: $R^n \to R$: objective function
- f_i: $R^n \to R$, $i = 1, \ldots, m$: constraint functions

Multiple solutions can be found for optimization problems, which are dependent on objective function [1]. As shown in Fig. 9.1, the objective function *f(x)* can *reach*

H. Khayyam (✉)
School of Engineering, RMIT University, Melbourne, VIC, Australia
e-mail: hamid.khayyam@rmit.edu.au

A. Jamali
Faculty of Mechanical Engineering, University of Guilan, Rasht, Iran

H. Assimi
Optimisation and Logistics, School of Computer Science, The University of Adelaide, Adelaide, Australia

R. N. Jazar
Xiamen University of Technology, Xiamen, China

School of Engineering, RMIT University, Bundoora, VIC, Australia

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_9

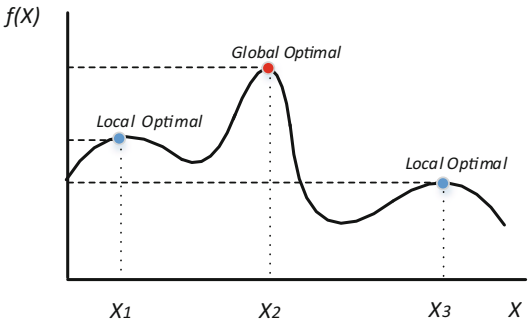


Fig. 9.1 Global and local optima

the highest value for x. The points x_1, x_2, and x_3 are all local maxima of the function, when the first order condition $f(x) = 0$ and is met. This is indicated by the horizontal tangents illustrated in Fig. 9.1. Any small increase or decrease in the value of x would decrease the corresponding function's value, as per: $f(x) \geq f(x \pm \varepsilon)|\varepsilon \to 0$.

However, since x_1 and x_3 are local (not global) optima and only point x_2 is a global optimum, as it provides the maximum overall output value for the total domain of the objective function $f(x)$.

It is often difficult to find the true solution to such a problem, as a local or the global optimum are difficult to differentiate in a mathematical search and the solution space is extremely complex.

9.1.1 Conventional Approaches

Conventional (traditional) approaches are operated using iterative search algorithms and they begin the process with a solution, such as a deterministic or arbitrarily selected value, by then improving according to some deterministic rule. These approaches can be applied to engineering optimization is highly sensitive to the nature of problem [1–3] such as:

Linear Programming (LP)

LP is applied to the linear objective function that models the problem at hand, and its constraints optimization problem. The general approach is to use the Simplex Algorithm, such that the first LP function is transformed into its canonical form or mid integer linear programming [4, 5]. Following this, a basic (initial) feasible solution of the LP is determined from the easier to manipulate canonical form and the initial solution is then moved to a basic, feasible solution which is both in the same neighborhood as the first solution, but also closer to the optimal solution. This is the case among all other adjacent basic feasible solutions, where the process is repeated until the true, global optimum solution is achieved.

Quadratic Programming (QP)

QP can find applications for optimization problems where the objective function is in a quadratic form and the constraints are linear (in-)equalities [6]. This solution approach uses a modified version of the aforementioned Simplex Algorithm.

Dynamic Programming (DP)

DP is an optimization based solution approach, which transforms a larger complex problem, into sequences of smaller and simpler problems to solve, in the context of the multistage nature of the optimization procedure [7]. To begin with, the most recent subproblem in the sequential queue, T, is solved. Following this, the optimal solution for the penultimate problem, $T-1$, is also determined, which similarly leads sequentially to the optimal solution for T. This process is repeated until all subproblems and hence, finally the total problem, are solved.

Convex Optimization (CO)

CO is a study the problem consisting the objective is a convex function if minimizing, or a concave function if maximizing and all the constraints are convex functions. More explicitly, a convex problem is of the form can be defined by [8]:

$$\text{Minimize subject to}: f_0(x)$$

$$f_i(x) \leq b_i, \quad i = 1, \ldots, m$$

Objective and constraint functions are convex:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

$$\text{if } \alpha + \beta = 1, \quad \alpha \geq 0, \quad \beta \geq 0 \tag{9.2}$$

Includes least-squares problems and linear programs as special cases.
In general, CO is used for solving some optimization problems if:

- No logical solution.
- Reliable and effective algorithms.
- Computation time (roughly) proportional to max [*n3, n2m, F*],

where F is cost of evaluating f_i's and their first and second derivatives.

In comparison with other optimization approaches, CO has a number of tricks for transforming problems into convex form for solving the complex problems:

Convex Hull (CH)

CH is integration of convexity, segment, and convex combination as expressed in Eqs. (9.3) to (9.5):

$$\text{A set } S \text{ is convex if } x \in S \text{ and } y \in S \text{ implies the segment } xy \in S \tag{9.3}$$

$$\text{The segment } xy \text{ is the set of all points of the form } \alpha x + \beta y \text{ with } \alpha \geq 0, \quad \beta \geq 0 \quad and \quad \alpha + \beta = 1 \tag{9.4}$$

A *convex combination* of points $x_1, \ldots, x_k$ is a sum of the form $\alpha_1 x_1 + \ldots + \alpha_k x_k$ with $\alpha_i \geq 0$ for all i and

$$\alpha_1 + \cdots + \alpha_k = 1 \tag{9.5}$$

CH of a set of planar points is the minimum convex polygon containing all the points. Each point x_i in the S has a weight (or coefficient) α_i, they are non-negative and used to calculate for the points as weighted average. For any selected weights (or coefficient), combination of the resulting convex is a point in the convex hull. The selected weights (or coefficients) in all possible ways form the whole convex hull. The convex hull can be written by following equation:

$$\text{Conv}(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i x_i \;\middle|\; (\forall i : \alpha_i \geq 0) \wedge \sum_{i=1}^{|S|} \alpha_i = 1 \right\}. \tag{9.6}$$

The entire CH containing the data can be broken up into triangles using the Delaunay triangulation method [9]. For example, the Delaunay triangulation results in the Convex Hull boundary shown in Fig. 9.2.

Stochastic Programming (SP)

SP is used when there is some uncertainty present in the data being incorporated into the objective function for solving the optimization problem [10]. Solution approaches can be included to improve the robustness of the process, such as recourse strategies, assumption of different scenarios and by performing sensitivity analyses.

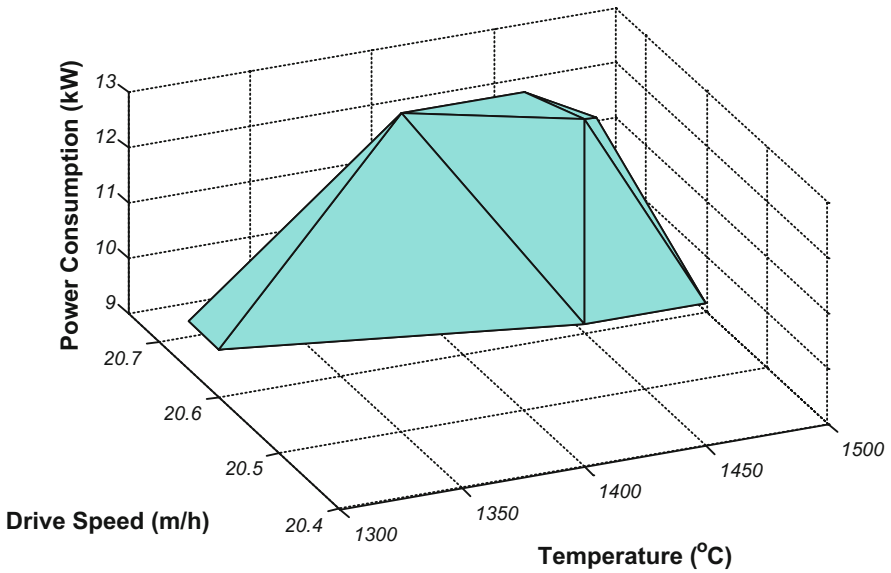


Fig. 9.2 An example of Delaunay triangulation results for power consumption by high temperature and drive speed [5]

Greedy Algorithms (GA)

GAs as functions always prefer the next step in the iterative solution process, which generates the greatest possible improvement toward the global optimum, yet does not assess or incorporate the consequences [11]. Given a current solution, which is suboptimal (not the global optimum), a greedy algorithm would conduct the search for a modified solution, that is within a certain neighborhood relative to the suboptimal solution, and then choose the best candidate among them. This approach is often referred to as hill climbing, by way of comparing the analogous actions of a mountaineer that chooses every step with efficiency and progress in mind. These algorithms are only focused on the "next" step in the iterative optimization process. Hence, it is possible for them to remain entrenched in many of the local optima present in complex function, especially if the initial values are not chosen well to speed the process. Hence, this approach requires smooth and less nonlinear solution spaces. Additionally, solving with monotonous objective functions makes the process simpler for generating good solutions. This is related to the general concept of gradient search and suffers from similar shortcomings.

Gradient Search (GS)

GS is used once the objective function $f(x)$ is in a form that is differentiable and strictly convex, where the solution can be achieved through the first order constraint $\partial f/\partial x = 0$. Given the proposed solution:

$$\nabla f\left(x'\right) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial y}{\partial x_n}\right) \quad \text{for} \quad x' = x \tag{9.7}$$

As mentioned earlier, conventional approaches are used only for mathematical problems that satisfy certain preconditions. The constraints must be expressed in certain formats and the objective function must be of a certain type, and so forth. Therefore, the applications are restricted to a rather limited set of problems. Many of the current conventional optimization approaches do suffer from a nontrivial sensitivity to the following problems:

- Difficulties in passing over (calculating a rejecting) local optimal solutions,
- Risks of divergence,
- Computational difficulties in handling multiple, complex constraints, or numerical difficulties associated to computing first or second order derivatives [12].

To overcome these problems, the heuristic approaches are comparative and superlative solutions and they have been used in a number of engineering optimization problems [13–15].

9.1.2 Heuristic Approaches

Heuristic Optimization (HO) methods have been used to a number of engineering applications, due to their capability of overcoming relatively common challenges present in automated design problems, control and mechanical engineering problems [16–20], including but not limited to uncertainty, nonlinearity, varied parameter types, and the local minima presence and/or discontinuities [21]. HO is a technique which can solve a problem more quickly in comparison with conventional approaches which are slower. Also, HO can find an estimated solution when conventional approaches fail to find an accurate solution. In other words, Ho can provide accuracy, optimality, extensiveness, or exactness with respect to the speed. However, applying heuristic optimization to a problem at hand can be computationally expensive because HO requires experienced data to develop itself. These kind of data are usually difficult to find, which lead to classify the evaluations in minor or major issues.

In this chapter, we define some heuristic approaches as technique for solving mathematical optimization problems [12, 22] and presenting the genetic programming as an intuitive approach in truss structure's design of mechanical engineering problem that can be interpreted and exploited intelligently to obtain a reasonable

solution. These approaches and techniques have been described in the following sections: Section 9.2 present the heuristic optimization techniques. Section 9.3 defines the Genetic programming as intuitive approach. Section 9.4 explains the truss structure's design and optimization of mechanical engineering problems. Section 9.5 presents some of the optimization applications in truss structure's mechanical design solved through genetic programming. Concluding remarks are provided in the last section. A large number of references have also been included.

9.2 Heuristic Optimization Techniques

9.2.1 Underlying Concepts

Heuristics approaches were first presented in 1945 by G. Polya [23] and then in 1978 [24] the approaches were introduced to solving of some science and engineering problems.

Compared to conventional optimization approaches, heuristic methods are designed such that better computational performance is possible, while at the expense of yielding a lower model accuracy.

However, the "rules of thumb" that are the underlying factors to conducting a heuristic method are typically very context sensitive. That is, they are often very specific as to the nature of the problem at hand. Further, as heuristic methods are techniques to solve problems based on the user's expertise, domain specific representations of the design space are used, and hence general heuristics can be defined for fundamental problem solving only, such as search methods [25].

The central feature common to all Heuristic Optimization (HO) methods is that they begin with a more or less arbitrary initial solution as a starting point, which is then followed with iterative processing to produce new solutions. This is done by some generation rule, which evaluates the new solutions, eventually serving the best solution evaluated during the search process to the user. The execution of the iterative search procedure is typically ended when improvements over a given number of iterations become insubstantial. Other metrics for stopping include When the solution at hand is good enough for the application as defined by a threshold; when the allowed CPU time, or other resource-based external limit has been reached; or when some internal parameter (as defined by the user) terminates the algorithm's execution. Another obvious stopping condition would be exhaustion of valid candidate solutions. This latter case is hardly ever realized in practice, due to the complexity and sheer size of the design space for such engineering systems.

Due to the fact that HO approaches may be substantially different in their underlying concepts, it is hard to find a method of generally classifying them. Yet, some central concepts have been highlighted to facilitate some comparisons

between methods. New heuristic approaches rapidly increasing, as well as variants, or combinations of others, the list that follows are far from exhaustive [1]:

First Generation Through Neighborhood Search New solutions can be generated by taking the current solution and modifying it by neighborhood search, or alternatively new solutions can be based on past experience. By employing such approaches, simple search strategies can be applied. For example, the use of a deterministic rule, random guesses in the design space, or potentially a combination of both.

New Generation To overcome the presence of local optima, HO approaches typically consider not only new solutions in the design space that lead to a short-term improvement, but also include a portion of solutions that are inferior to the best solution, by comparison. To enforce the requirement of convergence, inferior solutions could be included in cases where the known global optimum is close by, or alternatively might be provided a smaller "weight" in the process. Additionally, the solution(s) with the best performance in the search thus far may be reinforced, sometimes referred to as the elitist principle, or novel solutions may be ranked in order of preference/score and only the very best of those are kept for future might be a solution, among others. It is important to note that the underlying acceptance rules for this process can be deterministic or embody certain degrees of randomness.

Knowledge Past knowledge can be used to help the iterative search process to assist in reaching the global optimum faster. This is often referred to as the guided search approach. However, while incorporating any sort of past knowledge may have the potential to certainly reduce the search space, and simultaneously increase the speed of convergence, there is the added drawback that this might yield inferior solutions.

Limitations Given that the search algorithm is often very large for the types of problems considered, new and useful solutions can be discovered by searching in the neighborhood of an agent's current solution. Similarly, the complexity at a population-level considers this promising. Conversely, some approaches will instead explicitly exclude some neighborhoods or regions from the search process, in order to avoid features such as cyclic search paths, expending overt computation time on nominally irrelevant alternatives, as well as others.

Flexible Tolerance in Constraints There exist some methods, which are developed to address specific types of constraints and are therefore not good to be generalized for applications to other optimization problems. Other capabilities allow for the testing and ranking of different algorithms, which in turn may also affect the decision of which (optimal) approach to select for a given optimization problem.

Combinatorial Complex Problems HO can solve combinatorial problems to find an optimal solution from a finite set of possible solutions. For HO methods, generally speaking, the complexity of the problem largely depends on the costs for evaluating each candidate solution in the design space, the total number of iterations to be completed, and if applicable, the overall population size and costs associated with managing the population (i.e., for storage or compute tasks). Generally, the

total number of iterations, in addition to the size of the population, will increase for more complex problems (i.e., larger problem spaces). This will result in an increase in the computational costs required to complete the task. Yet, for HO methods, this is not uncommonly lower than for employing conventional (traditional) methods.

Convergence Rate A common metric to compare, and often rank, different algorithms, is the process time, as a resource. This relates to the aforementioned underlying concepts of HO. While computational "speed" may be a useful property of an algorithm in terms of practicality and logistics, conversely it can be considered as not meaningful when viewed as a lone criterion. This is because it does not holistically describe the effectiveness of the method, such as the ability to converge to local and global optima. Only a time limit can be imposed as a metric in this case.

Various Reliability Problem For some common heuristic methods, the exist proofs that show these methods will eventually converge toward the global optimum. However, this is under the caveats of sufficient computation time and appropriate choices of model parameters. In practice, it is sometimes necessary to accept a trade-off between these two objectives. As an optimization algorithm getting stuck in an objective function's local optimum, heuristic methods are hence commonly judged, or ranked, by their typical ratio of finding solutions in a global optimum, versus inferior solutions.

More details to the broadly defined categories of heuristic optimization is in Sect. 9.2.2.

9.2.2 Heuristic Approaches

Engineering optimization problems are mostly including different variables under complex constraints and extremely nonlinear. Modern heuristic optimization approached have been developed with an aim to achieve global search [26]. This section provides details of some of the most commonly used heuristic optimization approaches. These can be considered as typical examples for these types of approaches, which are also underpinned by differences as described in Sect. 9.2.1.

Simulated Annealing

Simulated Annealing (SA) works as a hill-climbing search approach, which is useful in finding global optima in the existence of large numbers of local optima. In 1980s, SA were established by using the Metropolis algorithm [27]. SA is a method for solving unconstrained and bound-constrained optimization problems. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy.

At each iteration of the simulated annealing algorithm, a new point is randomly generated. The distance of the new point from the current point, or the extent of the search, is based on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective, but also, with a certain probability, points that raise the objective. By accepting points that raise the objective, the algorithm avoids being trapped in local minima, and is able to explore globally for more possible solutions. An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to converge to a minimum. A basic simulated annealing pseudocode is in Table 9.1.

Tabu Search

The Tabu Search (TS) strategy was introduced by [28], as a search strategy that avoids returning to solutions already visited by maintaining a Tabu list, which stores successive approximations. Three main strategies in TS are: (1) forbidding is to control what arrives the Tabu list, (2) freeing is to check which exits the Tabu list, and (3) short-term that is to manage between the forbidding and freeing strategy to select trial solutions. Since the Tabu list is finite in length, at some point, after a number of steps, some solutions can be revisited. Adding a newly generated solution to a completed Tabu list is done by removing the oldest one from the list, based on a First In, First Out (FIFO) principle. New approximations can be generated in different ways. Some of the TS parameters are: local search procedure, neighborhood structure, aspiration conditions, form of Tabu moves, addition of a Tabu move, maximum size of Tabu list, stopping rule. A basic TS pseudocode presented in Table 9.2 uses the following procedure: at each step, a given number of new approximations are calculated within the neighborhood of the currently considered solution but considering as feasible only the ones which are not in the Tabu list. Among the new approximations the best one is chosen to replace the current solution, being also introduced in the Tabu list.

Table 9.1 A basic simulated annealing pseudocode

s = Generate_Initial_Solution()
$T = T\_0$
While termination conditions not met
s_1 = Pick_At_Random ($N(s)$)
If $f(s_1) < f(s)$
$s = s_1$
Else
Accept s_1 as new solution with probability $p(T,s_1,s)$
Endif
Update(T)
Endwhile

Table 9.2 A basic Tabu Search pseudocode

Step 1: Choose an initial solution i in S. Set $i^* = i$ and k=0.
Step 2: Set k=k+1 and generate a subset V^* of solution in N(i,k) such that either one of the Tabu conditions is violated or at least one of the aspiration conditions holds.
Step 3: Choose a best j in V^* and set i=j.
Step 4: If $f(i) < f(i^*)$ then set $i^* = i$.
Step 5: Update Tabu and aspiration conditions.
Step 6: If a stopping condition is met then stop. Else go to Step 2.

Table 9.3 A basic evolution strategy pseudocode

Procedure ES{
t = 0;
Initialize P(t);
Evaluate P(t);
While (Not Done)
{
P_p (t) = Select_Best(μ,P(t)) (Select μ Best);
P_c (t) = generate (λ, P_p (t)) (*Generate λ new individual*) ;
Evaluate (P_c (t));
P(t+1)= Select_Survivors(P_p (t) $\cup P_c$ (t));
t = t + 1;
}

Evolution Strategy

Evolution Strategy (ES) introduced in [29] as a part of evolutionary computation and developed further after 1970s. ES has two main generation parameters: number of parents μ, and number of offsprings creation λ notation by ES (μ, λ). These evolution from one generation to another controlled by mutation and each generation begin with a population of individuals. Each individual λ is evaluated to assign their fitness, then all individuals are ranked in descending order according to their finesses. Next, the first μ fittest individuals are nominated to create the parent population. Next, each of the μ parents will create by repeated mutation $k = \lambda/\mu$ offsprings. Finally, the new, mutated population will change the old one and the algorithm reiterates. A basic evolution strategy pseudocode is in Table 9.3.

Genetic Algorithms

The Genetic Algorithm (GA) is able to efficiently investigate large search spaces. It has two major requirements: Representation and Evaluation. GAs have been designed and developed by Holland [30] and later by Goldberg [31] and De Jong [32]. GAs are search methodologies which employ the specific mechanisms of genetic evolution such as selection, crossover, and mutation to evolve based on the

Table 9.4 A basic pseudocode of genetic algorithm

Step 1: Select an encoding schema
Step 2: Randomly initialize chromosome pool
Step 3: Evaluate each individual in the population
Step 4: Evaluate fitness of each individuals (fitness $= \frac{f_i}{\sum f_i}$) to breed new population
Step 5; Select individuals using Roulette wheel or tournament and Create new population from selected parents using crossover and mutation
Step 6: Replace old population with new population
Step 7: Repeat steps 3 - 6 for each generation

natural selection principle. Seven steps of a basic pseudocode of a genetic algorithm are presented in Table 9.4. GA uses the selection operator in each generation to choose a pair of individuals with respect to their fitness as parents to generate two offspring by the crossover operator. Next a pair of parent chromosomes select, and they enter the crossover step to generate two offspring. Crossover operator generates solutions that inherit good characteristics from both parents. Lately, mutation operator to create individuals by small changes in the genes to ensure the newness in the genetic material. Next, all offspring produced by crossover and mutation are put in a selection pool to choose from to form the new population. Finally, the initial population is replaced with the offspring population and the three steps of selection, crossover, and mutation for a next generation will be repeated. In order to avoid losing the best solution [25] a copy of the best individual from the current population and transfer it unchanged in the next generation will be made.

Particle Swarm

The Particle Swarm (PS) optimization method was first proposed in the 1990s, by Kennedy and Eberhart [33]. PS is a stochastic optimization method based on emulates swarming performance, and characteristics of swarms, of large groups of animals (birds or insects, etc.). Fundamentally, the PS method develops a population of agents, which individually move in the search space, leveraging cooperative strategies such as the interaction of the individual particles to exchange information. PS is basically a form of directed mutation. Any particle i has two parts: position X_i and its velocity V_i. The position of a particle i is calculated with respect to its prior position, denoted by X_i. An additional correction term proportional with its velocity $\varepsilon \cdot V_i$ is also included. Sequentially, a value for velocity is individually assigned to each particle in the systems and is calculated using four primary components: (1) the weighting or importance of the previous time-step value of velocity V_i; (2) the importance of the best "personal" (local optima based on partial information available to the agent) solution for particle i, X_i P; (3) the importance of the ideal local solution so far for informants of particle i, X_i L; and (4) the importance of the ideal global solution so far, as determined by the entire swarm at a macroscopic perspective, XG. These individual factors are all considered by being weighted

Table 9.5 A basic pseudocode of particle swarm

[x*] = PSO()
P = Particle_Initialization ();
For *i=1* to *it_max*
For each particle *p* in *P* do
fp = f(p);
If *fp* is better than *f(pBest)*
pBest = p;
end
end
gBest = best p in P;
For each particle *p* in *P* do
*v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);*
p=p+v
end
end

together (e.g., in a linear sum), denoted by a, b, and c. A basic Particle Swarm pseudocode is in Table 9.5.

Cuckoo Search

Cuckoo search (CS) method is inspired by modeling the reproduction procedure of cuckoo species in the nature. Cuckoo search is an optimization algorithm developed by Xin-she Yang and Suash Deb in 2009 [34]. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds of other species. The main steps of this algorithm are in a way that firstly each cuckoo lays an egg at a time and the egg is dumped in a random selected nest. Then, the best nests with high quality of eggs are opted for being processed at next generation. After that the egg is thrown away or the nest is abandoned, which results in building a new nest in a new location as presented in Table 9.6.

Differential Evolution

Differential Evolution (DE) was largely proposed as a new evolutionary algorithm by [35]. DE works based on the differences between randomly selected possible solutions. This approach uses the search space topography in the neighborhood of the current solution. In depends on the candidate solutions in a wide area and in a narrow area, mutations will have large amplitudes and will be of small importance respectively. In any generation, all current possible solutions will be considered as reference solutions different mechanisms of DE will be applied. Consequently, for an X_i, two different individuals X_{r1} and X_{r2}, other than X_i, will be randomly selected and a mutation will be used to X_i depends on the difference between X_{r1} and X_{r2}, to produce a mutant X_i. Then a crossover, based on the difference between current and

Table 9.6 A basic pseudocode of cuckoo search

Objective function $f(x)$, $x = (x_1, ..., x_d)^T$
Initial a population of n host nests $x_i (i = 1, 2, ..., n)$
While (t <MaxGeneration) or (stop criterion);
Get a cuckoo (say i) randomly by Lévy flights;
Evaluate its quality/fitness F_i
Choose a nest among n (say j) randomly
if ($F_i > F_j$),
Replace j by the new solution;
end
Abandon a fraction (pa) of worse nests
[and build new ones at new locations via Lévy flights];
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best;
end while

Table 9.7 A basic pseudocode of differential evolution

Initialization
Parameter limits should be defined
If not, parameter ranges should cover the suspected optimum
Generate the initial population (X(i, j, k))
For n=1: No. of Generation or Evaluation
For j=1: No. of Population
Choose two different individuals (X_{r1} and X_{r2})
$V = X_{best} + F * (X_{r1} - X_{r2})$ (F is mutant factor)
$U(n,k) = \begin{cases} X(n,k) \ if \ \text{rand} \leq CR \\ V(n,k) \ otherwise \end{cases}$ k=1: No. of Design Variables (CR is probability of crossover)
End
Replace X with the generated U if it is better
End

mutated solutions, will be applied to generate the new X_i. When its fitness function is better, Xi will change the reference solution with the best at any time (Table 9.7).

Genetic Programming

The Genetic Programming (GP) algorithm is a systematic method for getting computers to automatically solve a problem starting from a high-level statement of what needs to be done and some of the advantage are:

- It can find fit solutions in a very less time. (fit solutions are solutions which are good according to the defined heuristic)

- The random mutation guarantees to some extent that we see a wide range of solutions.
- Coding them is really easy compared to other algorithms which does the same job.

In next section GP will be explained in detailed with their applications.

9.3 Genetic Programming and Applications

9.3.1 Genetic Programming

Inspired by Darwin's natural selection concept in the mathematical platform, Genetic Programming (GP) is a meta-heuristic as well as evolutionary optimization approach capable of solving optimization problems [36–39]. GP, which is an improved version of the Genetic Algorithm (GA), was introduced by Koza in 1992 and so far has been used in many optimization problems such as the system identification, controller design, modeling, neural networks and circuit design [40–42]. The GP algorithm uses computer programs as members of its population and displays them with a tree structure. Like the real trees, these trees consist of branches and leaves, which in the algorithm's terminology, they are called as function and terminal, respectively. Each of these trees can represent a computer program. In order to form the initial population, the set of functions and terminals must first be defined for making the tree's elements. The set of functions can include mathematical, logical, conditional, loop, and arithmetic operators, while, the set of terminals can consist of the fid and random numbers, variables or program inputs. Also, two depths are considered for the tree as the initial and maximum depths. The initial depth is defined as the maximum depth the trees might have in the first generation and the maximum one is for controlling the highest depth the trees can meet during the program execution after the implementation of the genetic operators. Similar to the other evolutionary algorithms, the initial population are randomly created in the genetic programming. Koza specifies three techniques, namely grow, full, and what is referred to as "ramped-half-and-half" configuration, for creating random initial population [36]. In grow method, primary individual produced in such a way that they do not grow in length beyond the maximum depth/length as specified by the user (local stopping criterion). The full method is somewhat similar in process to the grow method, other than that the terminals are guaranteed to all have a user-defined depth. Generally, the "ramped-half-and-half" method is a combination of the full and grow approaches. It must be noted, the depth of a node is mathematically translated to the number of edges from the root of that specific node. The depth of the tree is considered as the depth of its deepest leaf. For example, the tree of Fig. 9.3 represents 5 sin 3x.

After the creation of the trees in the initial generation, the fitness value of each decoded tree evaluated based on described objective function. Since the first

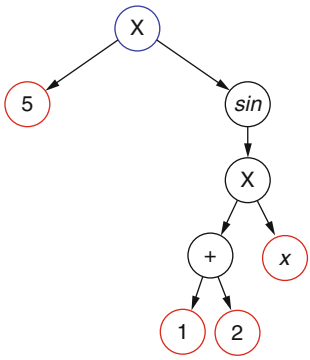


Fig. 9.3 Tree structure represents 5 sin 3*x*

generation trees are created in a randomized manner, the average fitness is weak in this initial generation, additionally the genetic operators should be used in order to improve the population fitness. The two main operators in the genetic programming algorithm are the crossover and mutation [36]. Crossover operator takes two parents, cuts them from a random point, and swaps the fragments to produce two offsprings. Mutation operator takes one parent, cuts it from a random point, and generates a new tree in the cut point to form the offsprings. It must be noted that each of the common selection methods such as tournament or roulette wheel can be used to select a random individual as an input for GP operators. Figure 9.4 depicts the (a) crossover and (b) mutation operations.

After applying the genetic operators on the population, all of the new and initial population trees were collected in a pool and then, based on one of the selection methods (tournament or roulette wheel), trees are selected to the number of initial population and transferred to the next generation. If the end condition of the genetic planning algorithm is not satisfied, the initial population will be replaced by the next generation's ones and this process will be repeated until the end of the algorithm. Finally, the best founded tree (from the fitness point of view) during the algorithm implementation is extracted as the optimum solution of the problem.

Like the other evolutionary algorithms, some important parameters must be determined by the user to control the genetic programming performance. The main control parameter is the size of the initial population, which represents the magnitude of the initial search in the search space. The other parameters are the initial and maximum depths, number of generations, probability of crossover and probability of mutation.

9.3.2 Applications of Genetic Programming in Optimization

GP has proved its efficiency to solve a great diversity of problems in a variety of domains such as, machine learning, pattern recognition, system identification, controller design, circuit design, etc. In this section, some applications of genetic

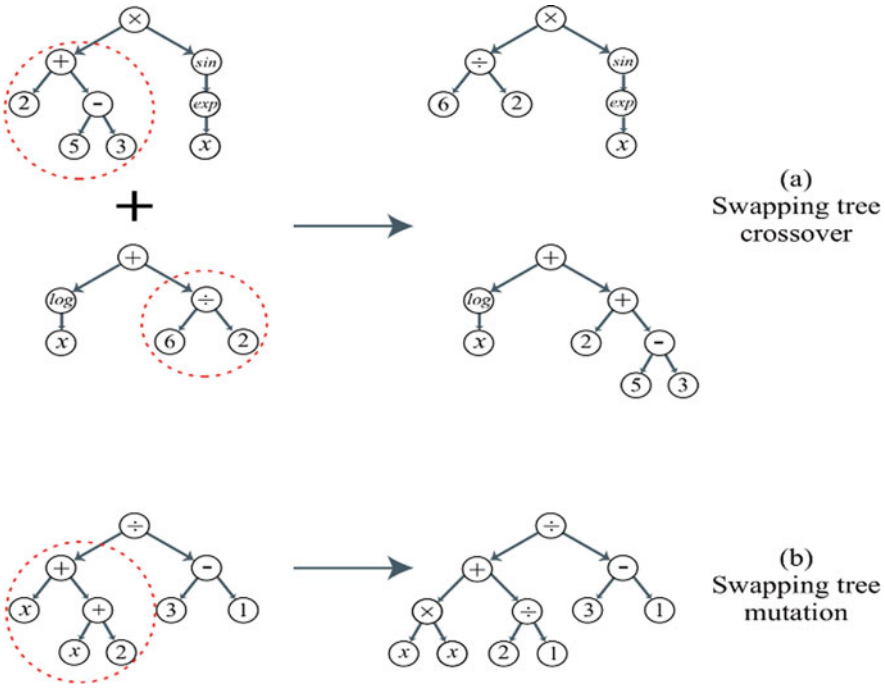


Fig. 9.4 Schematic of genetic operators

programming belong to the domain of controller design, system identification, and, truss optimization.

Applications of Genetic Programming in Automated Design of Controllers

The early comprehensive efforts in using GP as a controller design approach carried out by its inventor, John Koza. Koza well introduced genetic programming as a promising evolutionary approach for design of robust controllers for many types of industrial processes [39, 43–45]. He used GP methods for the automatic combination and utility of both topology-based information and parameter-tuning processes of different types of mathematical controllers in a parallel array for variety. Although it can be concluded that the results generated through this method by the author were interesting, however, due to computationally expensive nature of his approach, many researchers tried to use GP in more efficient scenarios, In this regards, Fukunaga et al. investigated application of simulation-based genetic programming for evolution of controllers to ultimately perform high-level motion-based tasks on a service robot [46]. GP is applied by Kobayashi et al. for the nonlinear systems output regulation [47]. They used only output information to design an optimal feedback controller based on inverse system modeling. Grosman

and Lewin used a new approach to automatically generate Lyapunov function for nonlinear system with maximum domains of attraction [48, 49]. They applied the proposed approach on the analysis of two independent dynamic systems are, namely, van der Pol's equation and a model of an exothermic, continuous stirred tank reactor. Chen and Lu employed GP approach to optimally derive the structure and parameters of a robust system controller for interval types of plants, which are based on the Kharitonov theorem [50]. Ibadulla et al. compared two symbolic regression methods of conventional genetic programming and a variational approach of genetic programming for control system synthesis of the two nonlinear systems [51]. Balandina stabilized nonlinear duffing oscillator, as a famous nonlinear control benchmark, by Cartesian Genetic Programming (CGP) which is program in a form of a directed graph structure [52]. Dracopoulos and Effraimidis adopted genetic programming into a complex, nonlinear helicopter control problem, for the purpose of providing hovering functionality. This problem is high dimensional in nature and illustrated its efficiency compared to other controller design methods, such as neuro-evolutionary [53]. Bourmistrova and Khantsis proposed a flight control system design optimization approach via genetic programming [54]. They developed a methodology for design a controller that satisfied the objectives of the sea-based recovery of a fixed-wing Unmanned Aerial Vehicle by ship. Maher and Mohamed proposed an enhanced genetic programming algorithm for optimal controller design concerning the initial population, the tree structure, genetic operations, and some other new nongenetic operators [55]. GP is employed by Alfaro-Cid to design a robust controller for a supply ship under environmental disturbances [56]. In this way, the current and desired state of the propulsion and heading dynamics of a supply ship are given as inputs, and controller generate the commanded forces required to maneuver the ship. Barate and Manzanera presented various automatic design of vision based obstacle avoidance controllers using GP [57]. Das et al. used GP to evolve tuning rules for reduced process parameters, which are based on a minimum time domain integral performance index [58]. Imae et al. proposed a differential GP for design of robust optimal controllers by solving Hamilton–Jacobi–Bellman (HJB), Hamilton–Jacobi–Isaacs (HJI) and Francis–Byrnes–Isidori (FBI) equations [59]. Kumar and Balasubramaniam suggested a reduced calculus optimal controller design approach for linear singular systems by solving the matrix Riccati differential equation [60]. Nallasamy and Ratnavelu reported an optimum controller design scheme for stochastic linear singular Takagi-Sugeno fuzzy delay systems with quadratic performance using GP [61]. Kumaresan and Ratnavelu introduced an optimal GP based controller design paradigm for stochastic linear quadratic singular neuro Takagi-Sugeno fuzzy systems with singular cost [61]. Mwaura proposed a new Gene Expression Programming (GEP) algorithm to automatically develop robot controllers and robot morphology [62]. GEP utilized two famous evolutionary algorithm, GA and GP, to precisely explore and exploit the feasible search domain.

Applications of Genetic Programming in Nonlinear System Identification

System identification and modeling of complex processes which use input–output data have always attracted many research efforts [63–67]. Generally, techniques for system identification are applied in a wide array of fields, in order to model and predict the behavior of very complex systems. This is done using the given input–output data for the problem. Extremely flexible tree representation of GP is useful to derive mathematical equations or complete models of nonlinear systems.

Willis et al. used GP to develop empirical models of steady-state and dynamic input–output of the chemical process [67]. Performance of the proposed approach is demonstrated by solving two typical processes: a vacuum distillation column and a twin screw cooking extruder. Togan and Baysec derived a robust model using GP for modeling and prediction of the torque and brake specific fuel consumption of a gasoline engine [68]. In this way, spark advance, throttle position, and engine speed are selected as input variables of model. Comparing the obtained model with those generated by artificial neural networks shown more simplicity and accuracy of obtained model. Madar et al. used GP to generate nonlinear input–output models of dynamical systems [69]. They applied Orthogonal Least Squares algorithm to estimate the contribution of the branches of the tree to the accuracy of the model. Saghafi and Arabloo proposed a novel compositional models based on GP to predict the gas condensate compressibility factor below dew point pressure using more than 1800 data series [70]. Cao et al. used GP embedded by genetic algorithm to discover and optimize the tree structure for modeling of nonlinear dynamic systems including systems of ordinary and higher-order differential equations for a number of practical example [71]. Hinchliffe proposed a multiple basis function GP (MBF-GP) algorithm and compared its performance with the standard algorithm [66]. The steady-state and dynamic modeling ability of MBF-GP and neural networks compared based on three nonlinear input–output data tables including, time series with/without time delay, and, Cooking Extruder. Jamali et al. proposed a new multi-objective genetic programming (MOGP) for Pareto optimal modeling of some complex nonlinear systems using some input–output data [42]. MOGP used a diversity preserving mechanism to get more uniformly Pareto front and a real number alteration operator to increase its exploration ability. In this way, two nonlinear mathematical models, each using different input–output datasets, for an explosive cutting process have been considered separately in a three-objective optimization processes. The pertinent conflicting objective functions that have been considered for such Pareto optimizations are namely, training error (TE), prediction error (PE), and the length of tree (complexity of the network) (TL) of the GP models. In another study, they used MUGP algorithm for robust Pareto modeling and prediction of complex nonlinear processes using some input–output data table [38]. Robust modeling consider uncertainties included in measured data to obtain more robust models. To achieve more robust model, the mean and standard deviation of training error and prediction error are considered in robust Pareto modeling.

Applications of Genetic Programming in Truss Optimization

The truss optimization searches for an optimal structure having the optimal stress distribution in the elements under the static and dynamic constraints. Trusses are generally investigated from the three perspectives for the optimization purpose. The topology optimization viewpoint looks for the best interconnection between the structure nodes and examines the presence or absence of the elements in the design space. Sizing optimization intends to optimize the cross-sectional areas for the truss members. Finally, the shape optimization seeks for optimal geometric nodal coordinates in the design space. In short, the combination of topology and size optimization of a truss, optimizes the cross-sectional dimensions of its elements while simultaneously utilizing new elements or detecting the current elements as redundant. In all of these viewpoints, the aim is to optimize one or more objective functions, including the structure's weight, volume, and displacement. Converse to this idea, the structure optimization problem is generally constrained to the static and dynamic constraints, including the maximum allowed stress in the structural elements, maximum allowable displacement of the structure's nods, natural frequency of the structure and buckling. Two viewpoints are usually considered for displaying the design space. The first one is named by convention as the ground structure, which specifies all the boundary conditions of the problem. On the other hand, it draws the structure with all the possible connections, specifies the loading and type of the supporting nodes [72]. Unlike the first viewpoint, the second one, which is the open space design, specifies only the boundary conditions including the loading and nodes and does not provide any information on the possible connections among the structure's nodes. As a result, design in an open space is carried out without prior knowledge.

The subject of trusses optimization has attracted many researchers in recent decades. Many conventional (traditional) optimization techniques have been implemented for optimizing the trusses such as the sensitivity analysis and approximate methods [73–75]. Although being powerful and effective for small structures, these methods were weak in solving more complex and non-convex problems [37, 76]. This is due to the inability for displaying the internodes' connections. After developing meta-heuristic techniques such as evolutionary algorithms, these methods have proved their ability to find an optimal global solution of the problems and many researchers have used these methods during the last two decades for solving the structural optimization problems [77]. Many studies have implemented the evolutionary algorithms in order to solve the structural optimization problems. One of the problems with the evolutionary algorithms is that they are computationally expensive compared to the conventional approaches. However, this issue does not matter with the development of computer systems and parallel processing methods.

Le et al. combined Electromagnetism and Firefly algorithms and proposed a new algorithm called Electromagnetism-like Firefly Algorithm (EFA) to solve discrete structural optimization [78]. The initial population in the proposed algorithm consisted of electrified fireflies having active or inactive behaviors based on objective function evaluation. In active mode, a firefly can fly randomly to any place while

in the inactive mode it only moves by the electromagnetic interactive forces. Population diversity in EFA increased by using a novel adaptive interactive force. Moreover, two new mechanisms are, namely, current-to-best and bidirectional-searching suggested to boost exploration–exploitation and local search abilities of the proposed algorithm, respectively—the capability of the new algorithm in finding global optimum design demonstrated by six popular truss optimization problems.

Prayogo et al. proposed a differential Big Bang–Big Crunch (DBB-BC) algorithm and employed it for optimum design of some construction-engineering problems such as a tubular column, 3-bar truss, 25-bar and 72-bar tower truss [79]. DBB-BC used a combination of BB-BC algorithm, differential evolution algorithms, and neighborhood search to get better performance in the point view of exploration and exploitation.

Discrete sizing optimization of a structure is studied by Gholizadeh et al. using two improved versions of Black hole and Multiverse algorithms [80]. They improved the mechanism of regenerating, the radius of the event horizon, and, updating equation in original algorithms to improve its deficiency in solving optimization problems including discrete design variables. Three benchmark problems including steel trusses, steel frames, and reinforced concrete frames are employed to clarify the efficiency of the proposed algorithm.

Cao et al. proposed an improved subspace harmony search (SHS) and employed a modified version of Deb-rule-constraints-handling for optimum design of truss structures with discrete cross sections [81]. They suggested three new rules for memory consideration, pitch adjustment, and, randomization of basic algorithm's operators to improve its search space exploration. To use information of the infeasible domain, they modified the Deb-rule-constraints-handling, and infeasible design with slight violations of constraints are considered as a feasible design. They compared search capability and computational efficiency of the proposed algorithm with other state-of-the-art evolutionary algorithms based on for weight minimization of truss structure problems.

Soh and Yang introduce a new tree expression for truss structure and applied it to the problem of simultaneous topology, shape and size optimization of trusses with static constraints [82]. They solved two well-known examples in truss optimization by proposed algorithms and obtained lighter trusses in comparison of conventional optimization methods. They also improved performance of his proposed algorithm by using fuzzy logic expert system to control the iteration process of GP.

Zheng et al. utilized linear GP to find the optimum nodal locations and member sizing of planer and space trusses [83]. Furthermore, a grammar-based form of GP has been employed to find fully optimized solutions made of elements with real-word cross sections. This approach trapped truss benchmark problems without clear boundary conditions.

Assimi et al. presented a genetic programming approach (SOGP) including an alteration operator and used it to optimize the mass of a truss with static constraints [76]. They demonstrated that owing to non-presence of mathematical operators in possible results (tree expressions), it is worthwhile to employ a modification operator to revise the tree components probabilistically and improved the global

search ability of the algorithm. This operator converts the real values presented in the tree by a predefined probability. It can be looked like as a classical one-point mutation operator; although, this operator can be characterized as a custom-defined one-point mutation because it incorporates the current value of the desired member in the process of mutating it to a new member. This work also depicted that GP could detect redundant truss nodes and members in the design space. They showed that GP could achieve lighter solutions with unique topologies than other state-of-the-art approaches.

Afterward, they suggested a GP combined with Nelder–Mead (HGP) and applied it to the problem of size optimization of trusses with both static and dynamic constraints for topology and size optimization which is a more complicated problem comparing with earlier problems [37]. The claimed that the alteration operator used in SOGP is dependable on a random variable and might not perform effectively in a more non-convex optimization problem and, GP may reach a local optimum. Hence, they combined the Nelder–Mead as an operator with GP to attain lighter trusses through the generations and strengthened GP abilities to obtain better results which improved the performance of the algorithm in comparison with other approaches such as SOGP [76]. It is worth mentioning that this study only observed the size and topology optimization of trusses under natural frequency constraints and static design limitations. Because of this matter, the design variables in test problems contain only the size of truss elements. Thus, truss optimization dealt with shape optimization (layout modification) stayed unchallenged. As a matter of fact, incorporation of shape optimization design variables into the optimal truss design problem makes the optimization search space more non-convex and probably may direct the algorithm to a false convergence.

Recently, Assimi et al. examined GP for truss optimization problem from the multi-objective point of view [84]. This approach will yield to provide a set of potential optimum truss designs, which helps the designers to choose the proper design based on the associated criteria. They claimed that HGP approach hybridized with Nelder–Mead as a local strategy is highly computational, and they proposed another approach to overcome the pitfall of no change in the real values composing the tree elements. This study incorporated an adaptive mutant factor to enhance the GP performance. This operator (which is motivated by the differential evolution algorithm) guides potential results to a new area in the search space based upon an adaptive mutant factor. This adaptive mutant factor aims to lead GP to explore more globally in early iterations and exploit more locally later.

9.4 Application of Genetic Programming in Mechanical Design of Truss Structures

This section defines the problem of truss optimization with discrete design variables. In this part, the objective function and constraints of the optimization problem

are presented. Following that, the algorithm of Discrete version of Structural Optimization by Genetic Programming (DSOGP) is described.

9.4.1 Problem Statement

The truss optimization problem is a nonlinear problem subjected to different constraints such as the kinematic stability, maximum allowable stress in the truss members, maximum allowable displacement in the truss nodes, critical buckling load or natural frequency constraints. These constraints can divide the search space of the optimal design of trusses into feasible and infeasible parts. This study considers discrete design variables for optimization problem besides the structural constraints. It means that the optimization problem is subjected to satisfy existing provisions in the design codes for constructional constraints to make it applicable. Finally, the final optimum design should be found in the feasible search space (without violating the design constraints) incorporating optimum discrete design variables.

The term of the ground structure has been used in the literature [85] to illustrate a truss problem in structural form. The ground structure represents the default layout of a truss. It defines the geometry of the design space including the position of nodes and how each node can be connected to other nodes by truss members. Moreover, the ground structure signifies the boundary conditions of the problem such as how external forces are exerted on the truss.

Every node in a truss can be categorized into two types: *essential* and *optional* nodes. Essential nodes must be present in each feasible solution including the final optimum design. These nodes either carry the loads or support the structure. Other nodes in the design space which are not essential are known as the type of optional [86].

Figure 9.5 depicts a ground structure of an example truss. This figure shows that the truss consists of six truss members incorporating four nodes. Nodes 1 and 2 support the truss, and node 3 carries the external load on it. Hence these three nodes are identified as essential nodes. This figure also presents the connectivity table among the nodes and how each node can be potentially connected. Interestingly, in this sample truss, all nodes are connected to all other nodes. The ground structure of this sample truss also defines the geometric position of nodes in the design space, which plays a key role in the determination of the length of each truss member.

In discrete size and topology optimization, we aim to minimize the weight of the truss. The weight of a truss is defined as the summation of the weight of each truss member. The weight of each member is dependent on the material density, length, and the size of the member. The optimization algorithm for this problem looks for the optimum cross-section areas to determine the weight of each truss member. Simultaneously, it is important to find how the nodes are connected in the final optimum design. In other words, the optimization algorithm aims to minimize the mass of the truss while it looks for the optimum size of each member and decides

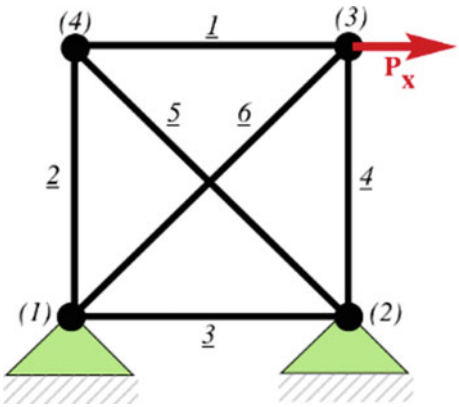


Fig. 9.5 The ground structure of a six-member truss

to keep or eliminate each truss member. It should be noted that different constraints have restricted the search space. For instance, the optimization algorithm should distribute the stress in the truss members and the displacement in truss nodes without violating these constraints. The discrete space of design variables also makes the feasible section of design space narrower. Equation (9.5) states the optimization problem formulation for sizing and topology optimization of trusses with discrete design variables.

$$\begin{aligned} &\text{Find } A = \{A_1, A_2, \ldots, A_m\} \\ &\text{To minimize } f = \sum_{i=1}^{m} \rho_i l_i A_i + P \\ &\text{Subjected to} \qquad \begin{aligned} &G1 \equiv \text{Truss has all essential nodes} \\ &G2 \equiv \text{Truss is kinematically stable} \\ &G3 \equiv \sigma_i \le \sigma_i^{\max} \quad i = 1, 2, \ldots, m \\ &G4 \equiv \delta_i \le \delta_j^{\max} \quad j = 1, 2, \ldots, n \\ &G5 \equiv A_i \in D \quad i = 1, 2, \ldots, m \end{aligned} \end{aligned} \tag{9.8}$$

where P defined as follows,

$$P = \begin{cases} 10^9 & \text{if } G1 \text{ is violated,} \\ 10^8 & \text{if } G2 \text{ is violated in the first step,} \\ 10^7 & \text{if } G2 \text{ is violated in the second step,} \\ 10^5 \left(\sum_{i=1}^{m} |\langle G3_i \rangle| + \sum_{j=1}^{n} \left|\langle G4_j \rangle\right| \right. & \text{otherwise} \end{cases} \tag{9.9}$$

where f denotes the objective function which is the sum of truss weight and its corresponding penalty (P) if a truss violates the constraints.

Definition of truss optimization problem constraints and how to tackle them are as follows which is adopted 710 from [86] and penalty values are presented in

Eq. (9.6). m and n denote the number of truss members and nodes. ρ_i, l_i, and A_i signify the material properties, length, and size of the ith truss member, respectively.

Constraint G1 checks if a truss includes all the essential nodes defined by the ground structure. If the truss violates this constraint, it results in a highly infeasible truss and the algorithm penalizes this truss with a considerable P with 10^9 added to its truss weight to ignore it with high probability in the evolution process.

Constraint $G2$ makes sure that the truss is kinematically stable. As this process is computationally expensive, it is suggested to be handled in two steps [86]. In step 1, the algorithm calculates the degree of freedom in the truss, if it returns a positive value, the truss is not static and is a mechanism and a penalty of 10^8 will be added to its weight. Otherwise, if the degree of freedom is a nonpositive value, the algorithm checks the positive definiteness in the stiffness matrix of the truss. If the truss violates the kinematic stability in this step, a penalty of 10^7 will be added to its weight.

Other constraints of the problem are the maximum allowable stress and maximum allowable displacement in the truss which form G3 and G4 constraints, respectively. These constraints check if the stress in truss members and displacement in truss nodes exceed the maximum allowable value. To handle this constraint, bracket operator penalty has been used. Bracket operator penalty is a prominent approach to handle constraints in one term and normalizes constraint values when their order of magnitude is different [87]. Hence, the bracket operator penalty for a constraint as $g_k(x) \leq g_k^{\text{allow}}(x)$ is defined as follows,

$$|\langle g_k(x)\rangle| = \frac{g_k(x)}{g_k^{\text{allow}}(x)} - 1 \leq 0 \tag{9.10}$$

So, if this term returns a negative value, the bracket operator penalty is zero; otherwise, it takes the absolute value of the above equation as the bracket operator penalty.

Therefore, G3 and G4 assure that if a truss violates maximum allowable stress and displacement in the truss, a corresponding penalty based on bracket penalty term which is commensurate with the degree of constraint violation is added to the weight of truss.

G5 is the fabricational constraint, which denotes that the size of each member should be chosen from a predefined set of discrete sizes (D). Because genetic programming generates its initial population based on this predefined set, this constraint is satisfied automatically.

9.4.2 Discrete Version of Structural Optimization by Genetic Programming

This section defines the proposed genetic programming approach for truss optimization with the discrete size of members by the algorithm of a Discrete version of Structural Optimization by Genetic Programming (DSOGP).

Initialization

Genetic programming requires a set of functions and terminals to generate its initial population.

Each genetic programming individual (as an expression tree) should represent a solution for the truss optimization problem in the search space. Therefore, each should represent a truss for this problem in the design space. Due to this, genetic programming elements in an expression tree should be the truss members and truss nodes.

Equation (9.8) defines the function and terminal sets for this purpose:

$$\begin{aligned} \text{Function set} &= \{A_1, A_2, \ldots, A_m\} \\ \text{Terminal set} &= \{N_1, N_2, \ldots, N_n\} \end{aligned} \tag{9.11}$$

N_q denotes the position of qth node in the design space with Cartesian coordinates (x_q, y_q, z_q). Therefore, the generated expression tree for an individual in the proposed approach includes the cross-sections sizes and nodes position in the design space. Interestingly, A_i has two meanings. First, A_i denotes a binary function, which takes two arguments (which are nodes) and connects these two. Moreover, A_i denotes the size of the labeled ith member in the tree. For instance, a sample tree expression is shown in Fig. 9.6; if we suppose $A_5(N_4, N_2) = 19.9$ in.^{2}, it means that there exists a member labeled as the fifth member in the truss connecting nodes 2 and 4 with the cross-section size of 19.9 in.^{2}

DSOGP takes the ground structure of the truss test problem as an input to find out the geometry and boundary conditions of the problem. Next, it generates a population employing the function and terminal sets stated in Eq. (9.8). DSOGP generates its initial population using the ramped half-and-half approach [88] and may contain feasible or infeasible individuals.

Decoding

It is essential for DSOGP to decode its population from the tree expression form to the truss form successfully. It should be noted that each expression tree should be mapped into one unambiguous truss form to maintain the consistency of the

algorithm. For this purpose, the output of a binary function present in the tree, namely A_i is defined with two rules as follows [82],

1. If A_i is connected with a branch from top left, the output of A_i is the element at its below right.
2. If A_i is connected with a branch from top right, the output of A_i is the element at its below left.

Decoding steps of the sample tree expression (depicted in Fig. 9.7) is as follows,

It is evident that this tree includes six binary functions (A_1, A_2, ... , A_6) and four terminals (N_1, N_2, ... , N_4). The decoding process should be started from the deepest level and the left side, as shown in Fig. 9.7.

Step 1. A_5 is connecting N_4 and N_2 in Fig. 9.6; so there exists a truss member labeled as the fifth member with size of "A_5" which connects nodes N_4 and N_2 in the design space.

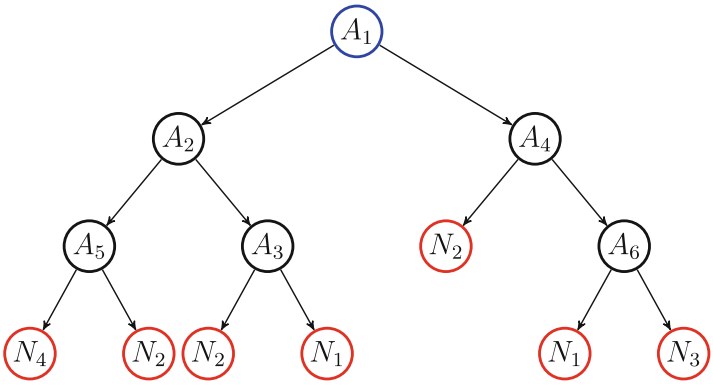


Fig. 9.6 A expression tree of a sample individual in DSOGP population

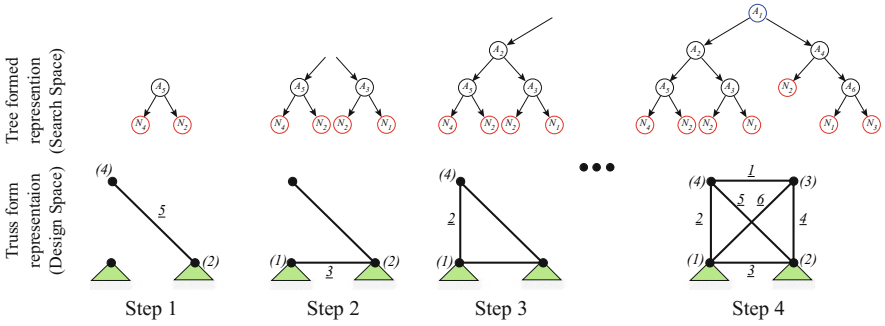


Fig. 9.7 Steps of mapping a tree expression into the truss form

Step 2. A_3 is connecting N_2 and N_1. Therefore, there exists a truss member labeled as the third member with size of "A_3" which connects nodes N_2 and N_1 in the design space.

Step 3. To decode the rest of the tree, the output of A_5 and A_3 should be determined. A_3 follows the first rule because it has a connection from the top left. Therefore, the output of A_3 is the terminal at its left which is N_1. Conversely, the output of A_5 is N_4. Hence, A_2 is taking A_5 and A_3 as arguments which their outputs are N_4 and N_1, respectively. Therefore, A_2 connects nodes N_4 and N_1 in the design space.

Step 4. Following the abovementioned steps for the right hand of the tree, the sample tree shown in Fig. 9.6 will be decoded into the sample truss shown in Fig. 9.7.

Fitness Evaluation

After the decoding steps, the competence of an individual in solving the optimization problem should be evaluated. In this study, fitness function defined in Eqs. (9.5) and (9.6) measures the weight of a truss and penalizes an individual if they violate the constraints. To evaluate the constraints such as kinematic stability in the second step, distribution of stress in truss members and displacement of truss nodes, finite element analysis has been done. This study uses OpenSEES [89] for the structural analysis. OpenSEES is a comprehensive framework which provides numerous types of elements, materials and numerical solvers to analyze the trusses obtained after decoding.

Genetic Operators and Selection

DSOGP assigns the fitness value to each expression tree in the population and performs the swap tree crossover and mutation to produce an offspring pool. Hence, DSOGP employs the tournament selection and selects the new population. Next, if the termination criterion is satisfied, DSOGP designates the best-found solution during the run as the ultimate optimum design. Otherwise, the current population is replaced with the new population and DSOGP reiterates the above steps till termination criteria are met.

9.5 A Case Study for Optimal Mechanical Design of Truss Structure

This section provides an example of Discrete version of Structural Optimization by Genetic Programming (DSOGP) algorithm for the optimal mechanical design

Table 9.8 Setting parameters for DSOGP

Parameters	Value
Population size	100
Probability of crossover	80%
Probability of mutation	10%
Tournament selection size	4
Termination criterion	Maximum generation
Maximum generation	500

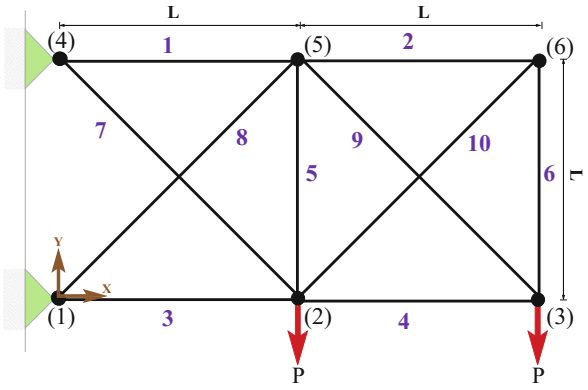


Fig. 9.8 The ground structure for ten-member truss problem

of truss with discrete design variables. The proposed approach has been applied to a planar truss optimization problem. The obtained results are compared with other state-of-art methods available in the literature and resulted in better solutions for size and topology optimization of trusses.

Because genetic programming is a stochastic algorithm, each problem has been tested for 30 times, the best results have been designated as the ultimate optimal design and reported in the corresponding tables and figures. Moreover, Table 9.8 lists the setting parameters used in DSOGP for this study.

Ten-member truss problem is a well-known benchmark for structural optimization. Figure 9.8 depicts the schematic of the ground structure of this truss. As the name of the benchmark suggests, it consists of ten members and six nodes. Nodes 1 and 4 support the truss, and nodes 2 and 3 carry the exerted forces on the truss. These external forces are downward loads with $P = 100$ kips. So, there exist four essential nodes in the problem with two optional nodes.

The material density and the modulus of the elasticity are defined as 0.1 pci and 30,000 ksi, respectively [90]. The truss is subject to maximum allowable displacement and stress with an absolute value of 2 in. for both directions and 25 ksi for both tension and compression, respectively.

The design variable set include $D = [1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50]$ (in.^{2}) which are 42 selected sizes selected

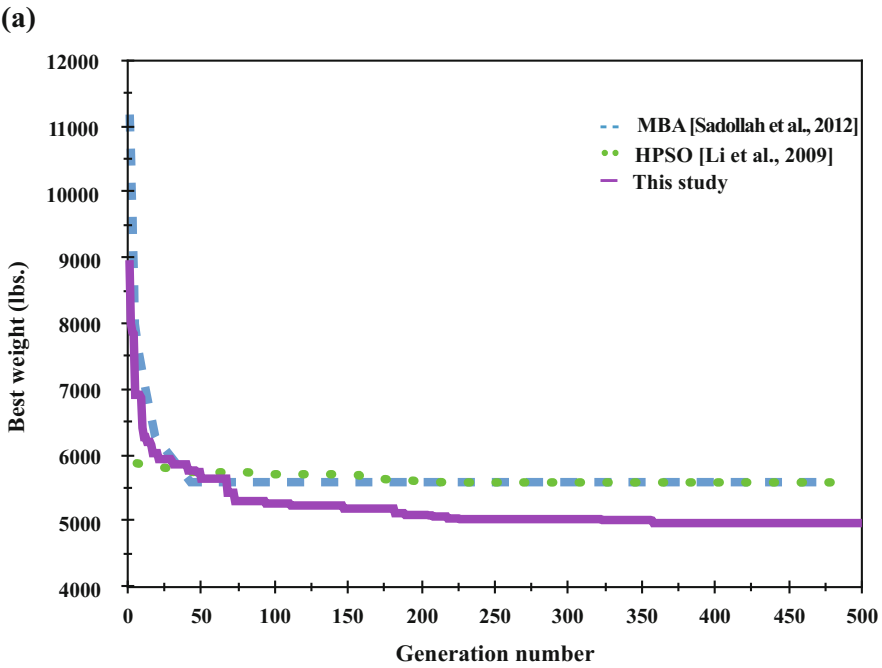


Fig. 9.9 The convergence plot for the best solution for ten-member truss problem

from the American Institute of Steel Construction (AISC) design code [91]. This case study has been studied previously by [90, 92–97].

Figure 9.9 illustrates the progress of the best-found solution obtained by the proposed algorithm during the evolution compared with the improvement of the best solution reported by [90, 93]. This figure states that DSOGP initially found a feasible solution with the weight of 8974.38 lbs. and evolved to the best-found solution with the weight of 4980.10 lbs after 358 generations. Despite, DSOGP required more iterations to converge to a solution, but this solution is remarkably lighter and better than the other two methods with 9.3% improvement.

Table 9.9 presents the obtained results compared with the other methods applied to this problem. Inspection of the data in this table states that DSOGP has found a unique layout incorporating six truss members into the ultimate design and identifying four truss members as redundant members. This unique topology has been shown in Fig. 9.10. It also demonstrates that even though the design variables are discrete, but DSOGP could find a solution with fewer elements leading to lighter truss design because it employs a tree-based expression to explore the search space.

Table 9.9 Comparison of optimal results for the ten-member truss

Size of members (in.^{2})	GA	HPSO	TLBO	MBA	aeDE	HHS	EFA	DSOGP
	[92]	[90]	[94]	[93]	[95]	[96]	[97]	Our study
A_1	33.50	30.00	33.50	30.00	33.50	33.50	33.50	30.00
A_2	1.62	1.62	1.62	1.62	1.62	1.62	1.62	–
A_3	22.00	22.90	22.90	22.90	22.90	22.90	22.90	22.00
A_4	15.50	13.50	14.20	16.90	14.20	14.20	14.20	13.90
A_5	1.62	1.62	1.62	1.62	1.62	1.62	1.62	–
A_6	1.62	1.62	1.62	1.62	1.62	1.62	1.62	–
A_7	14.20	7.97	7.97	7.97	7.97	7.97	7.97	7.22
A_8	19.90	26.50	22.90	22.90	22.90	22.90	22.90	22.00
A_9	19.90	22.00	22.00	22.90	22.00	22.00	22.00	22.00
A_{10}	2.62	1.80	1.62	1.62	1.62	1.62	1.62	–
Best weight (lbs)	5613.84	5531.98	5490.74	5507.75	5490.74	5490.74	5490.74	4980.10
Average weight (lbs)	N/A	N/A	5503.21	5527.30	5502.62	5493.49	5528.23	4983.51
Standard deviation (lbs)	N/A	N/A	20.33	11.38	20.78	10.46	18.37	11.76
Improvement (%)	11.29	9.98	9.30	9.58	9.30	9.30	9.30	–
Constraint violation	None	None	None	None	None	None	None	None

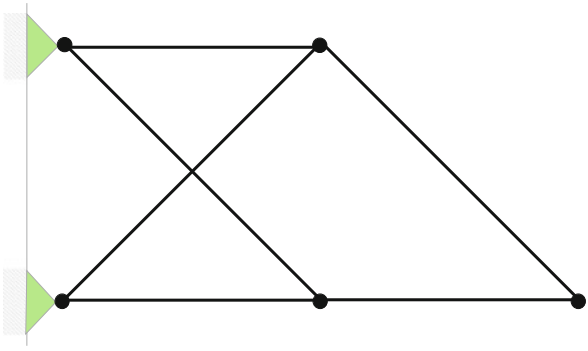


Fig. 9.10 The optimal topology obtained for ten-member truss problem

9.6 Conclusion

Engineering optimization problems are mostly involving with different variables under complex constraints and extremely nonlinear. These constraints might be

written as simple bounds, or as nonlinear relationships between them. This chapter provided basic knowledge of most popular conventional and heuristic optimization approaches, and how they are applied in common optimization problems in engineering applications. The heuristic optimization approaches have been devised and developed for solving a complex problem more quickly when conventional optimizations are too slow, or for finding an approximate solution when conventional approaches fail to find any exact solution. Their success is due largely to their most important features, namely the need of minimal additional knowledge on the optimization problem and a highly numerical robustness of algorithms. Among of the heuristic optimization approached, Genetic programming has been shown that it can be considered as shortcut optimization approaches to achieve by trading optimality, completeness, accuracy, or precision for speed solving of engineering problems. In order to optimal mechanical engineering design of truss with discrete design variables, genetic programming has been constructed. The results showed that the genetic programming uses high-level building blocks of variable length and their size and complexity could be changed during breeding. Genetic programming works well in a large number of different cases and has many potentials to solving applications of mechanical engineering optimization.

References

1. Maringer, D. (2006). *Portfolio management with heuristic optimization*. Berlin: Springer.
2. Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation, 76*, 60–68.
3. Nazari-Heris, M., Mohammadi-Ivatloo, B., & Gharehpetian, G. B. (2018). A comprehensive review of heuristic optimization algorithms for optimal combined heat and power dispatch from economic and environmental perspectives. *Renewable and Sustainable Energy Reviews, 81*, 2128–2143.
4. Hao Jiang, H., Sidney Fels, S., & Little, J. J. (2007). *A linear programming approach for multiple object tracking*. Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition.
5. Khayyam, H., Naebe, M., Bab-Hadiashar, A., Jamshidi, F., Li, Q., Atkiss, S., et al. (2015). Stochastic optimization models for energy management in carbonization process of carbon fiber production. *Applied Energy, 158*, 643–655.
6. Bertsekas, D. P. (1999). *Nonlinear programming*. Cambridge: MIT.
7. Bellman, R. (2003). *Dynamic programming*. New York: Dover.
8. Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.
9. Fang, T.-P., & Piegl, L. (1992). Algorithm for Delaunay triangulation and convex-hull computation using a sparse matrix. *Computer-Aided Design, 24*, 425–436.
10. Marti, K., & Kall, P. (1994). *Stochastic programming*. Berlin: Springer.
11. Goyal, A., Lu, W., & Lakshmanan, L. (2011). *CELF++: optimizing the greedy algorithm for influence maximization in social networks*. Paper presented at the Proceedings of the 20th international Hyderabad, India.

12. Fletcher, R. (2013). *Practical methods of optimization* (2nd ed.). New York: Wiley.
13. Khayyam, H., Naebe, M., Zabihi, O., Zamani, R., Atkiss, S., & Fox, B. (2015). Dynamic prediction models and optimization of polyacrylonitrile (PAN) stabilization processes for production of carbon fiber. *IEEE Transaction on Industrial Informatics, 11*, 887–896.
14. Khayyam, H., Naebe, M., Bab-Hadiashar, A., Jamshidi, F., Li, Q., & Atkiss, S. (2014). Stochastic optimization models for energy management in carbonizationprocess of carbon fiber production. *Applied Energy, 158*, 643–655.
15. Khayyam, H., & Bab-Hadiashar, A. (2014). Adaptive intelligent energy management system of plug-in hybrid electric vehicle. *Energy, 69*, 319–335.
16. Gen, M., & Cheng, R. (1999). *Genetic algorithms and engineering optimization*. New York: Wiley.
17. Vinther, K., Nielsen, R. J., Andersen, P., & Bendtsen, J. D. (2017). Optimization of interconnected absorption cycle heat pumps with micro-genetic algorithms. *Journal of Process Control, 53*, 26–36.
18. Golkarnarenji, G., Naebe, M., Badii, K., Milani, A. S., Jazar, R., & Khayyam, H. (2019). A machine learning case study with limited data for prediction of carbon fiber mechanical properties. *Computers in Industry, 105*, 123–132.
19. Golkarnarenji, G., Naebe, M., Church, J. S. S., Badii, K., Bab-Hadiashar, A., Atkiss, S., et al. (2017). Development of a predictive model for study of skin-core phenomenon in stabilization process of PAN precursor. *Journal of Industrial and Engineering Chemistry, 49*, 46–60.
20. Golkarnarenji, G., Naebe, M., Badii, K., Milani, A. S., Jazar, R. N., & Khayyam, H. (2018). Support vector regression modelling and optimization of energy consumption in carbon fiber production line. *Computers & Chemical Engineering, 109*, 276–288.
21. Matott, L. S., Bartelt-Hunt, S. L., Rabideau, A. J., & Fowler, K. R. (2006). Application of heuristic optimization techniques and algorithm tuning to multilayered sorptive barrier design. *Environmental Science and Technology, 40*, 6354–6360.
22. Golkarnarenji, G., Naebe, M., Badii, K., Milani, A. S., Jazar, R. N., & Khayyam, H. (2018). Production of low cost carbon-fiber through energy optimization of stabilization process. *Materials, 11*, 385.
23. Polya, G. (1945). *How to solve it*. Princeton, NJ: Princeton University Press.
24. Box, F. (1978). A heuristic technique for assigning frequencies to mobile radio nets. *IEEE Transactions on Vehicular Technology, 27*, 57–74.
25. Russell, S. J., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3rd ed.). London: Pearson.
26. Gavrilas, M. (2013). *Heuristic and metaheuristic optimization techniques with application to power systems*. Paper presented at the conference elected topics in mathematical methods and computational techniques in electrical engineering.
27. Kirkpatrick, C. D. G. S., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.
28. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13*, 533–549.
29. Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. *Technical Report Library Translation No. 1122*. Royal Aircraft Establishment, Farnborough.
30. Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
31. Jong, K. A. D. (1985). Genetic algorithms: A 10 year perspective. In *Proceedings of the 1st internal conference on GAs and their applications* (pp. 169–177).
32. Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.
33. Kennedy, R. E. J. (1995). *Particle swarm optimization*. Paper presented at the proceedings of IEEE internal conference on neural networks.
34. Yang, X. S., & Deb, S. (2009). *Engineering optimisation by Cuckoo search*. https://www.library.cornell.edu/.

35. Storm, K. P. R. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.
36. Koza, J. R. (1992). *Genetic programming—On the programming of computers by means of natural selection*. Cambridge: MIT.
37. Assimi, H., & Jamali, A. (2018). A hybrid algorithm coupling genetic programming and Nelder–Mead for topology and size optimization of trusses with static and dynamic constraints. *Expert Systems with Applications, 95*, 127–141.
38. Jamali, A., Khaleghi, E., Gholaminezhad, I., Nariman-zadeh, N., Gholaminia, B., & Jamal-Omidi, A. (2014). Multi-objective genetic programming approach for robust modeling of complex manufacturing processes having probabilistic uncertainty in experimental data. *Journal of Intelligent Manufacturing, 28*, 149–163.
39. Koza, J. R., Yu, J., Keane, M. A., & Mydlowec, W. (2000). *Evolution of a controller with a free variable using genetic programming*. Paper presented at the EuroGP 2000.
40. Poli, R., & Koza, J. (2014). *"Genetic Programming", Search methodologies*. New York: Springer US.
41. Gholaminezhad, I., Jamali, A., & Assimi, H. (2014). *Automated synthesis of optimal controller using multi-objective genetic programming for two-mass-spring system*. Paper presented at the 2nd RSI/ISM International Conference on Robotics and Mechatronics, ICRoM 2014.
42. Jamali, A., Khaleghi, E., Gholaminezhad, I., & Nariman-zadeh, N. (2014). Modelling and prediction of complex non-linear processes by using pareto multi-objective genetic programming. *International Journal of Systems Science, 47*, 1675–1688.
43. Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., & Lanaz, G. (2003). *Genetic programming IV: Routine human-competitive machine intelligence*. Dordrecht: Kluwer Academic.
44. Koza, J. R., Keane, M. A., Yu, J., Mydlowec, W., & Bennett, F. H. (2000). *Automatic synthesis of both the topology and parameters for a controller for a three-lag plant with a five-second delay using genetic programming*. Paper presented at the EvoWorkshops.
45. Koza, J. R., Keanne, M. A., Yu, J., Bennett, F. H., & Mydlowec, W. (2000). Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines, 1*, 121–164.
46. Fukunaga, A., Hiruma, H., Komiya, K., & Iba, H. (2012). Evolving controllers for highlevel applications on a service robot: A case study with exhibition visitor flow control. *Genetic Programming and Evolvable Machines, 13*, 239–263.
47. Kobayashi, T., Kimoto, T., & Imae, J. (2011). *Genetic programming based output regulation via optimal inversion for nonlinear systems*. Paper presented at the IEEE/SICE, International Symposium on System Integration.
48. Grosman, B., & Lewin, D. R. (2005). Automatic generation of lyapunov functions using genetic programming. *IFAC Proceedings, 38*, 75–80.
49. Grosman, B., & Lewin, D. R. (2006). Lyapunov-based stability analysis automated by genetic programming. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control* (pp. 766–771).
50. Chen, P., & Lu, Y. (2011). Automatic design of robust optimal controller for interval plants using genetic programming and Kharitonov Theorem. *International Journal of Computational Intelligence Systems, 4*, 826–836.
51. Ibadulla, S. I., Shmalko, E. Y., & Daurenbekov, K. K. (2017). The comparison of genetic programming and variational genetic programming for a control synthesis problem on the model "Predator-victim". *Procedia Computer Science, 103*, 155–161.
52. Balandina, G. I. (2017). Control system synthesis by means of cartesian genetic programming. *Procedia Computer Science, 103*, 176–182.
53. Dracopoulos, D. C., & Effraimidis, D. (2012). Genetic programming for generalised helicopter hovering control. *15th European Conference EuroGP, Málaga, Spain*.
54. Bourmistrova, A., & Khantsis, S. (2010). Genetic programming in application to flight control system design optimisation. In *New achievements in evolutionary computation*. Rijeka: InTech.

55. Maher, R. A., & Mohamed, M. J. (2013). An enhanced genetic programming algorithm for optimal controller design. *Intelligent Control and Automation, 4*, 94–101.
56. Alfaro-Cid, E., McGookin, E. W., Murray-Smith, D. J., & Fossen, T. I. (2008). Genetic programming for the automatic design of controllers for a surface ship. *IEEE Transactions on Intelligent Transportation Systems, 9*, 311–321.
57. Barate, R., & Manzanera, A. (2007). *Automatic design of vision-based obstacle avoidance controller using genetic programming*. Paper presented at the 8th International Conference on Artificial Evolution, Tours, France.
58. Das, S., Pan, I., Das, S., & Gupta, A. (2012). Improved model reduction and tuning of fractional-order PIλDμ controllers for analytical rule extraction with genetic programming. *ISA Transactions, 51*, 237–261.
59. Imae, J., Kikuchi, Y., Ohtsuki, N., Kobayashi, T., & Zhai, G. (2004). *Design of nonlinear control systems by means of differential genetic programming*. Paper presented at the 43rd IEEE Conference on Decision and Control.
60. Kumar, A. V. A., & Balasubramaniam, P. (2007). Optimal control for linear singular system using genetic programming. *Applied Mathematics and Computation, 192*, 78–89.
61. Nallasamy, K., & Ratnavelu, K. (2012). Optimal control for stochastic linear quadratic singular Takagi–Sugeno fuzzy delay system using genetic programming. *Applied Soft Computing, 12*, 2085–2090.
62. Mwaura, J., & Keedwell, E. (2015). Evolving robotic neuro-controllers using gene expression programming. In *2015 IEEE symposium series on computational intelligence* (pp. 1063–1072).
63. Porkhial, S., Salehpour, M., Ashraf, H., & Jamali, A. (2015). Modeling and prediction of geothermal reservoir temperature behavior using evolutionary design of neural networks. *Geothermics, 53*, 320–327.
64. Jamali, A., Ghamati, M., Ahmadi, B., & Nariman-zadeh, N. (2013). Probability of failure for uncertain control systems using neural networks and multi-objective uniform-diversity genetic algorithms (MUGA). *Engineering Applications of Artificial Intelligence, 26*, 714–723.
65. Jamali, A., Nariman-zadeh, N., Darvizeh, A., Masoumi, A., & Hamrang, S. (2009). Multi-objective evolutionary optimization of polynomial neural networks for modelling and prediction of explosive cutting process. *Engineering Applications of Artificial Intelligence, 22*, 676–687.
66. Hinchliffe, M. (2001). *Dynamic Modelling Using Genetic Programming*, PhD. Faculty of Engineering, University of Newcastle.
67. Willis, M., Hiden, H., Hinchliffe, M., McKay, B., & Barton, G. W. (1997). Systems modelling using genetic programming. *Computers & Chemical Engineering, 21*, S1161–S1166.
68. Togan, N., & Baysec, S. (2010). Genetic programming approach to predict torque and brake specific fuel consumption of a gasoline engine. *Journal of Applied Energy, 87*, 3401–3408.
69. Madar, J., Abonyi, J., & Szeifert, F. (2005). Genetic programming for the identification of nonlinear input-output models. *Industrial and Engineering Chemical Research, 44*, 3178–3186.
70. Saghafi, H., & Arabloo, M. (2018). Development of genetic programming (GP) models for gas condensate compressibility factor determination below dew point pressure. *Journal of Petroleum Science and Engineering, 171*, 890–904.
71. Cao, H., Kang, L., Chen, Y., & Yu, J. (2000). Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines, 1*, 309–337.
72. Dorn, W. S. (1964). Automatic design of optimal structure. *Journal de Mécanique, 3*, 25–52.
73. Topping, B. H. V. (1983). Shape optimization of skeletal structures: A review. *Journal of Structural Engineering, 109*, 1933–1951.
74. Schmit, L. A., & Miura, H. (1976). A new structural analysis/synthesis capability-ACCESS 1. *AIAA Journal, 14*, 661–671.
75. Haftka, R. T., & Gürdal, Z. (1992). *Elements of structural optimization*. Berlin: Springer.

76. Assimi, H., Jamali, A., & Nariman-zadeh, N. (2017). Sizing and topology optimization of truss structures using genetic programming. *Swarm and Evolutionary Computation, 37*, 90–103.
77. Kicinger, R., Arciszewski, T., & Jong, K. D. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures, 83*, 1943–1978.
78. Le, D. T., Bui, D.-K., Ngo, T. D., Nguyen, Q.-H., & Nguyen-Xuan, H. (2019). A novel hybrid method combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures. *Computers & Structures, 212*, 20–42.
79. Prayogo, D., Cheng, M.-Y., Wu, Y.-W., Herdany, A. A., & Prayogo, H. (2018). Differential Big Bang—Big Crunch algorithm for construction-engineering design optimization. *Automation in Construction, 85*, 290–304.
80. Gholizadeh, S., Navid, R., & Shojaei, E. (2018). Improved black hole and multiverse algorithms for discrete sizing optimization of planar structures. *Engineering Optimization*.
81. Cao, H., Qian, X., Asce, M., Zhou, Y. L., & Yang, H. (2018). Applicability of subspace harmony search hybrid with improved Deb rule in optimizing trusses. *Journal of Computing in Civil Engineering, 32*, 04018021.
82. Soh, C. K., & Yang, Y. (2000). Genetic programming-based approach for structural optimization. *Journal of Computing in Civil Engineering, 14*, 31–37.
83. Zheng, Q. Z., Querin, O. M., & Barton, D. C. (2006). Geometry and sizing optimisation of discrete structure using the genetic programming method. *Structural and Multidisciplinary Optimization, 31*, 452–461.
84. Assimi, H., Jamali, A., & Nariman-zadeh, N. (2018). Multi-objective sizing and topology optimization of truss structures using genetic programming based on a new adaptive mutant operator. *Neural Computing and Applications*, 1–21.
85. Dorn, W. S. (1964). Automatic design of optimal structures. *Journal de Mecanique, 3*, 25–52.
86. Deb, K. (2001). Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design, 37*, 447–465.
87. Deb, K., & Kalyanmoy, D. (2001). *Multi-objective optimization using evolutionary algorithms*. New York: Wiley.
88. Koza, J. R. (1992). *Genetic programming: On the programming of computer programs by natural selection*. Cambridge: MIT.
89. Mazzoni, S., McKenna, F., Scott, M. H., & Fenves, G. L. (2006). *OpenSees command language manual*. Richmond, CA: Pacific Earthquake Engineering Research (PEER) Center.
90. Li, L. J., Huang, Z. B., & Liu, F. (2009). A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers and Structures, 87*, 435–443.
91. Coello, C. A. C., Rudnick, M., & Christiansen, A. D. (1994). Using genetic algorithms for optimal design of trusses. In *Sixth International conference on tools with artificial intelligence, proceedings* (pp. 88–94).
92. Rajeev, S., & Krishnamoorthy, C. S. (1992). Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering, 118*, 1233–1250.
93. Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2012). Mine blast algorithm for optimization of truss structures with discrete variables. *Computers & Structures, 102-103*, 49–63.
94. Camp, C. V., & Farshchin, M. (2014). Design of space trusses using modified teaching-learning based optimization. *Engineering Structures, 62-63*, 87–97.
95. Ho-Huu, V., Nguyen-Thoi, T., Vo-Duy, T., & Nguyen-Trang, T. (2016). An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers & Structures, 165*, 59–75.
96. Cheng, M.-Y., Prayogo, D., Wu, Y.-W., & Lukito, M. M. (2016). A hybrid harmony search algorithm for discrete sizing optimization of truss structure. *Automation in Construction, 69*, 21–33.
97. Duc, T. L., Dac-Khuong, B., Tuan, D. N., Quoc-Hung, N., & Nguyen-Xuan, H. (2019). A novel hybrid method combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures. *Computers and Structures, 212*, 20–42.

Chapter 10
Optimization of Dynamic Response of Cantilever Beam by Genetic Algorithm

Javad Zolfaghari

10.1 Introduction

Structural design involves developing a structure that meets specific performance criteria. Such a design can usually be improved by using numeric optimization procedures. In the design of many important mechanical and structural systems, dynamic response must be considered. In order to obtain an optimal dynamic response, natural frequencies and mode shapes or other dynamic behaviours such as velocity of an existing model may have to be altered in a prescribed manner. Optimization methods for dynamic response of a system are not routine or well established, particularly when dealing with the velocity and acceleration response.

Since, in most low-frequency vibration problems, the response of the structure to dynamic excitation can be expressed as a function of its fundamental frequency and mode shape, a large amount of literature is related to the optimization of structure with frequency constraints. Accordingly, most of the papers in the field of optimization of dynamic response of structures deal with the optimum design problem in which objective or constraint quantities are associated with the vibration phenomena (e.g. natural frequency).

As a preliminary step in non-frequency optimization, maximizing the tip velocity response of a cantilever beam was studied in this chapter. The optimization investigated in this chapter has been delimited to adjusting the height of the beam while the other parameters are fixed. This classic example is chosen because it gives us a perspective to industrial problem with complexity in constructions and any accessibility to implicit objective function and constraints both equal and unequal. Hence, by developing a flexible method for such problems, we can obtain a precise solution for a number of unresolved problems, at the moment.

J. Zolfaghari (✉)
Independent Researcher, Tehran, Iran

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1_10

However, firstly, which zero-order optimization lead us to a concept for approaching a solution is analysed by using methods such as scaling which is the main factor of decreasing generality. Secondly, it is found that genetic algorithm could be a reliable method among others. Finally, efficient penalty function and powerful nonconventional method to achieve global optimum during generalization is presented.

10.2 Dynamic Response Analysis

A dynamic response analysis is the solution of the equation of motion. Using the finite-element formulation, the linear equation of motion in matrix notation may be generally expressed as

$$[M]\{\ddot{u}(t)\} + [C]\{\dot{u}(t)\} + [K]\{u(t)\} = \{F(t)\} \tag{10.1}$$

where

$[M]$ is the mass matrix
$[C]$ is the damping matrix
$[K]$ is the stiffness matrix
$\{F(t)\}$ is the vector of applied loads, a function of time
$\{u(t)\}$ is the nodal displacement vector, a function of time (unknown)
t is the time variable

If n is the number of active degrees of freedom (DOF) in the system, then $[M]$, $[C]$ and $[K]$ are of order n.

In the absence of damping and external load, the equation is reduced to

$$[M]\{\ddot{u}\} + [K]\{u\} = \{0\} \tag{10.2}$$

For a linear structure, the displacements due to free vibration are harmonic and of the form

$$\{u\} = \{\varphi\} \ \sin \ \omega t \tag{10.3}$$

where $\{\varnothing\}$ is an n-dimensional vector and the natural circular frequency of vibration (rad/s). Substituting Eqs. 10.3 into 10.2, one obtains the generalized eigenvalue problem

$$\left([K] - \omega^2 [M]\right)\{\varphi\} = \{0\}. \tag{10.4}$$

Static response can be viewed as a special case of dynamic response in which the accelerations and velocities are very low that the inertia and damping forces can be neglected.

One of the destructive phenomena in a structural system is the state of response in which the response amplitude of an undamped structure subject to a harmonic force excitation tends to infinity. Resonance occurs when the exciting frequency coincides with one of the natural frequencies of the system. In order to limit the response of structures to dynamic loads, the natural periods of system must be less than the threshold (critical period). Similarly, tall buildings are subject to resonance. Therefore, the natural frequency of a tall building should be less than the critical period at which wind vortices are shed to avoid resonance with the forces induced by vortex shedding.

Solving an eigenvalue problem for frequencies is usually an integral part of analysing the dynamic response of a structure. Finite-element method is often incorporated as a part of the optimal design process. One of the difficulties in these analyses is that the finite-element model changes during the optimization process and the design and analysis iterations become greatly complicated. Generally, one can classify the dynamic response of structures into two categories: deterministic (prescribed) and nondeterministic or 'random load'. Deterministic loading itself is divided into periodic and nonperiodic loading or

- *Harmonica steady-state input*
- *Transient response*

Three dynamic analyses, namely direct frequency response, modal frequency response and modal transient response, have been discussed in various references. Another classification has been made in step-by-step integration and modal analysis.

10.2.1 Modal Analysis

The process of determining the natural frequencies and mode shapes of a system is termed as modal analysis. Natural frequencies and mode shapes are obtained by solving the following equation:

$$\left([k] - \omega_i^2 [M]\right) \{\Psi\}_i = 0 \tag{10.5}$$

where

$[k]$ is the reduced stiffness matrix of the structure
$[M]$ is the reduced mass matrix of the structure
ω_i is the circular natural frequency of the mode i
$\{\psi\}_i$ is the reduce mode shape vector of mode i
$\{\psi\}_i$ is often normalized such that

$$\{\Psi\}_i^T [M] \{\Psi\}_i = [I] \tag{10.6}$$

Modal analysis provides the vibration characteristics of a structure. Usually a modal analysis is required for starting a more detailed dynamic analysis, such as a transient dynamic analysis. Modal analysis is also important in designing a structure for dynamic loading conditions.

10.2.2 The Eigenvectors

The displacement solutions of the equilibrium equation for free, undamped vibrations are defined as eigenvectors. The eigenvalues represent the natural frequencies of the system and the eigenvectors, the corresponding mode shapes.

There are n eigen solutions to equation $\left(\omega_1^2, \{\varphi_1\}\right), \left(\omega_2^2, \{\varphi_2\}\right), \left(\omega_3^2, \{\varphi_3\}\right), \ldots,$ $\left(\omega_n^2, \{\varphi_n\}\right)$. ω_i^2 are the eigenvalues, which represent the square of the natural circular frequency of the system, and $\{\varphi\}_i$ are the eigenvectors, which represent the corresponding mode shapes.

10.2.3 Number of Modes

It is evident that if all n modes are considered in the solution of the discretized equilibrium equations, then the solution will be exact. In this case, however, no reduction in the size of the problem is achieved.

Practically, however, all n modes need not to be considered because it is unlikely that all of them will contribute significantly to the response of the system. In most cases, a reasonably good approximation to the response can be obtained by considering only a fraction of the total number of modes.

10.2.4 Dynamic Analysis Methods

The numerical methods available to solve the equation for a transient dynamic analysis are classified into

- *Direct integration*
- *Mode superposition*

Direct Integration Method

In this method, the equilibrium equation is solved at discrete time points in a step-by-step manner. Moreover, a particular variation of displacements, velocities and accelerations is assumed between any two time points. At a given time point,

all equations are coupled and solved simultaneously. Three different approaches commonly used in a step-by-step integration procedure are categorized as follows: linear acceleration, **Newmark's β** and **Wilson's θ**.

Mode Superposition Technique

Mode superposition is one of the techniques used to determine the dynamic response of linear structures. Computation time is significantly saved when the mode superposition method is used because usually only the first few modes need to be considered for a good approximation to the solution. The accuracy of the solution, however, depends on the number of modes included in the analysis.

10.3 Optimization of Dynamic Response

The optimum design of structures for dynamic response usually involves more difficulties than the optimization of statically loaded structures. These difficulties are due to the fact that the displacements, velocities and accelerations are often related to the objective function and the constraints which, in turn, are, in general, *implicit, nonlinear or a black box* situation. The algorithm may query the value $f(x)$ for a point x, but it does not obtain gradient information, and, in particular, it cannot make any assumptions on the analytic form of $f(x)$ functions of the design variables.

Optimum design methods in dynamic response can be classified as matrix structural analysis and nonlinear optimization techniques [1]. Cost, volume (weight), maximum values of the displacement, velocity, acceleration, stress or displacement (during a prescribed time interval) are considered as the objectives for the optimum design of structures in dynamic response.

The first task in an optimization procedure is to identify the most applicable dynamic response in order to define cost functions and then begin to formulate the specific trial design. Once the problem formulation task is accomplished, the designer has to concentrate on the available solution techniques to find the optimum design.

The state of the system in dynamic analysis is time dependent and can be described by a state variable vector. Also, performance and failure constraints must be satisfied for all time in the entire interval. The designers might have to change the design criteria in order to obtain better solutions or convergence of the solutions. In this case, they may have to set up a new optimization procedure for a new design.

Although ongoing research in the area of analysing dynamic response and improving optimization procedures has had a significant growth over the past 20 years, the capability to perform dynamic response optimization is still in a very early phase. Most of the papers in the field of optimization of dynamic response of structures deal with the optimum design problems in which objective or constraint quantities associate with vibrational phenomena (e.g. natural frequencies).

Szyszkowski and King [2] and Grandhi and Venkayya [3] are amongst the numerous researchers who have treated frequencies in the objective function and constraints in their optimization procedure, respectively. Olhoff [4] determined the distribution of structural material of transversely vibrating beams or rotating circular shafts, such that maximum natural frequency or critical speeds are obtained. Instead of defining the natural frequency or critical speed in the objective functions, he solved an equivalent problem where the volume is minimized for a given natural frequency or critical speed. Niordson [5] found the best possible tapering of a simply supported beam, which for a given volume would have the highest possible value of the natural frequency for the lowest mode of lateral vibrations.

In structural optimization, Venkayya et al. [6], Khan and Willmert [7] and a few others applied the optimality criterion method to structural design problems of natural frequency constrains. Jan and Truman [8] considered the design of structural systems under multiple natural frequency constraints. A numerical procedure was developed by Turner [9] for minimum weight structures with specified natural frequencies. In that procedure, a finite-element idealization was employed, and the governing nonlinear equations were solved by an iterative procedure. He also employed Lagrange multipliers to introduce the free vibration equation as a constraint function. Later, Khan et al. [10] developed two optimality criterion techniques for the minimum weight design of mechanical systems subject to stress and natural frequency constraints. They applied the techniques, for instance, to a cantilever beam with concentrated mass, and they considered the area of each element as design variable. They concluded that optimality criterion techniques resulted in a substantial saving in computational times compared to standard nonlinear programming techniques. Recently, Truman and Petruska [11] applied optimality criteria (based on the Kuhn–Tucker condition) for an arbitrary dynamic loading to structural systems. In their work, the objective function is the weight of the structure and the constraints can be stresses, displacements and natural frequencies, as well as side constraints. The algorithm used consists of several steps that involve solving the system of ordinary differential equations [discretized equation of motion, Eq. (10.1)] using Newmark's methods and solving the system of equations to get the Lagrange multipliers.

Paeng and Arora [12] discussed the optimization of mechanical systems under dynamic loadings by using the multiplier methods. They defined an augmented function for the dynamic response optimization problem and developed the design sensitivity analysis for that function. Chahande and Arora [13] studied the multiplier method (also called the augmented Lagrangian method) for optimum design of mechanical and structural systems subjected to dynamic loads. In this work, three types of constraints have been considered: equality, inequality and explicit bounds. The basic idea of the method is to transform the constraint optimization problem into a sequence of unconstrained problems. The augmented function for the unconstrained problem was defined using the objective function and the constraint functions of the original problem and multipliers for constraints. The multiplier algorithm used has been successfully applied to several test problems.

It is interesting to note that the optimized shapes are not always intuitive. In Johnson et al. [14], for instance, the optimum thickness of a cantilever tube excited in torsion by a harmonically varying load is at its minimum at the root and has a large value at the tip. Again, in a study by Brach [15], the optimized shape of a cantilever beam for minimum tip deflection subjected to an impulse load at the tip requires a relatively large mass (42.4% of the total) directly under the load (at the tip).

10.4 Presentation of Classic Example

The cantilever beam selected in this chapter is a rectangular cross-sectional aluminium beam, with lumped mass on the tip. The beam is released from a deflected state equivalent to the deflection created by locating a 100 kg lumped mass at the tip. Through the dynamic analysis, a 30 kg lumped mass remained at the tip. The width of the beam is 0.5 m, the material of the beam has been assumed to be linearly elastic with a density of 2710 kg/m and a modulus of elasticity of 69 GPa. The structures here are 2-D elastic beams. For analysis purposes, a finite-element model of the beam has been defined. The software package ANSYS was employed to determine the dynamic response of the beam. The research by Abolbashari [16] showed the accuracy of dynamic response of cantilever beam.

The cantilever beam was modelled with 10 elements as shown in Fig. 10.1. The design variables are the heights of the elements (thickness of cantilever beam) which are varied to maximize the tip velocity. The lower bounds for the design variables are given in Table 10.1. The beam is subjected to both equality and inequality constraints. During the optimization procedure, the volume (0.098 m^3), width (0.5 m) and length (3.5 m) of the beam are held constant (equality constraint), and the maximum stress is checked, so that the allowable stress of 100 MPa, which represents the yield stress of annealed aluminium, is not exceeded (inequality constraint).

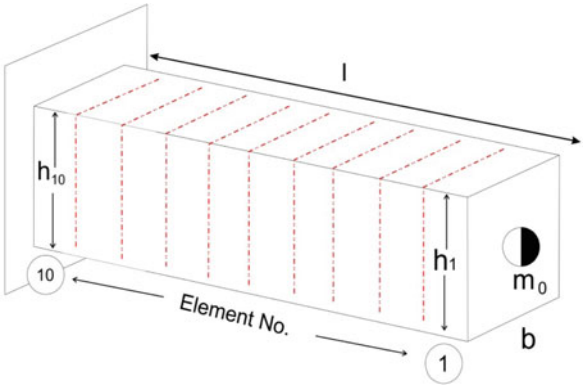


Fig. 10.1 Discretization of a cantilever beam

Table 10.1 Minimum acceptable element heights for the test problem when using ANSYS 4 4 for dynamic analysis

Element no.	Minimum height (mm)
1	8.5
2	13.1
3	16.2
4	16.2
5	16.2
6	16.2
7	16.2
8	16.2
9	16.2
10	16.2

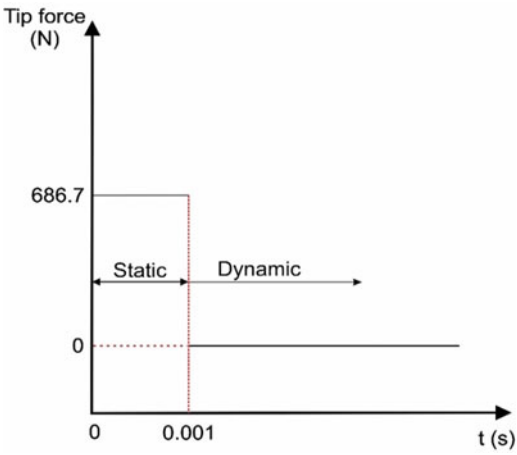


Fig. 10.2 Nongravitational loading history for a static analysis followed by a dynamic analysis

The loading procedure for all tests in this study is as follows, unless otherwise stated. The beams are subjected to both their weight and the weight of a lump which is located at the tip. First, a static analysis is required to calculate tip deflection. The lumped mass for static analysis is 100 kg [generates 981 (N) force]. Then, the lumped mass is reduced to 30 kg and the beams are released from the previously deflected position. The load history is plotted in Fig. 10.2.

Moreover, it is assumed that the load is applied in the vertical plane of symmetry. A typical loading system is shown in Fig. 10.3.

Mathematically, the problem may be expressed as

$$\text{maximize} \quad \dot{u}\left(l, t^*\right) \quad 0 \leq t^* \leq T \tag{10.7}$$

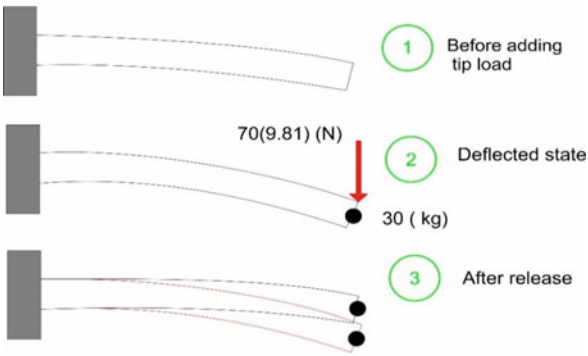


Fig. 10.3 Time history and deflection related to example

Subject to the following conditions

$$E\frac{\partial^2}{\partial x^2}\left[I(x)\frac{\partial^2 u}{\partial x^2}\right]+m(x)\frac{\partial^2 u}{\partial t^2}=p(x.t) \tag{10.8}$$

$$\begin{cases} u\,(0,t)=0 \\ u'\,(0,t)=0 \end{cases} \qquad \begin{cases} EIu''\,(l,t)=0 \\ EIu'''^{(l,t)}=m_0\,[g+\ddot{u}\,(l)] \end{cases} \qquad 0\le t\le T$$

$$u\,(x,0)=u^*(x)$$

$$\dot{u}\;\;(x,0)=0$$

$$\int_0^l m\;\;dx=M$$

$$\sigma_{\mathrm{m}}\,(x,t)\le\sigma_\alpha$$

$$I(x)\ge I_0(x)>0$$

$$\rho,l,E,M\;\;\text{are}\;\;\text{held}\;\;\text{constant}$$

where

$\dot{u}\ (l, t^*)$	is the tip velocity
l	is the length of the beam
ρ	is the density
x	is the position along the beam
E	is the Young's modulus of the beam material
$p(x.\ t)$	is the transverse external load
$I(.), I_0(.)$	are the cross-sectional area moments of inertia
$u(x, t)$	is a small transverse deflection of the beam
$m(x)$	is the mass per unit length of the beam
m_0	is the tip mass
g	is the gravitational acceleration
M	is the total mass of the beam
$u^*(x)$	is the initial deflection
T	is the total time in which the motion is considered
t^*	is the time when the maximum velocity occurs
$\sigma_{\mathrm{m}}(x, t)$	is the maximum normal stress
σ_α	is the allowable normal stress

10.5 Comparison of Zero-Order Numerical Optimization

The design space consists of the heights of the beam elements; in this case, h_i, $i = 1 - 10$. The beams are subject to both equality and inequality constraints. The following constraint sets have been studied:

1. Constant volume, width and length constraints with a limit constraint on stress.
2. Constant volume and width constraints with maximum stress subject to a limit constraint but no limitation on length.

In the above case, the stress constraint is

$$\sigma_{\max} \leq \sigma_{\mathrm{a}}, \tag{10.9}$$

where $\sigma_{\max}$ is the maximum stress within the beam elements and σ_{a} is the allowable stress. It should be noted that due to the numerical calculation difficulty, the design variables had to be bounded. These limit constraints are

$$h_{\mathrm{l}} \leq h_i \leq h_{\mathrm{u}}, \quad (i = 1, \ldots, n) \tag{10.10}$$

where h_{l} and h_{u} are the lower and upper limits of the heights, respectively.

The main intention in this work is the improvement of genetic algorithm method in the optimization of dynamic response of cantilever beam which is mentioned above.

According to the research by Abolbashari [16], like simple genetic algorithm (*SGA*), there is another zero-order numerical optimization method.

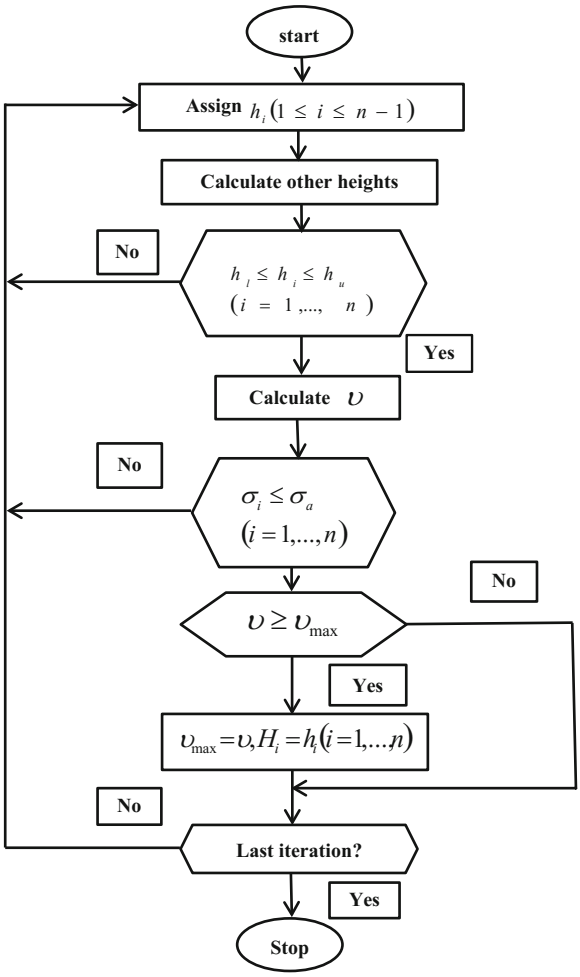
10.5.1 Modified Direction Set (MDS) Strategy

The simplest Direction Set (**Powell**'s) method [17] for multidimensional (n) problem is a search in one direction at a time along the coordinate directions to its minimum/maximum. The procedure of the simplest direction set begins with moving along the first direction to the minimum/maximum of the function, then moving from there along the second direction to the minimum/maximum of the function and so on. The procedure is repeated through the whole set of directions until the function stops decreasing/increasing. This cannot be followed exactly in the optimization of the beam because of the constant volume constraint. Therefore, the search in any directions has to be modified such that reduction of height in one element is accompanied by an increase in the height of the others. To handle this, a FORTRAN program is provided to calculate the height of the other elements when some of them are assigned a value by the user. In this program, the remainder of the volume (the total volume minus the volume of the candidate elements that have had their heights assigned by the designer) is calculated first and then this remaining volume is equally divided among the rest of the elements. The designer's choice of selecting the elements and assigning them a value plays a central role in this strategy. This choice has to be made based on the tip velocity (v, objective value). Thus, the direction search is manually modified by the designer in each iteration.

The above procedure is summarized in Fig. 10.4, where H_i, $i = 1, \ldots, n$ are the optimized heights. The search begins by decreasing the height of the first element (fixed to the wall) of a uniform beam and increasing the other heights to maintain a constant volume. The next step is to select a candidate element for decreasing/increasing the height in such a way that does not violate the design variable and stress constraints (Eqs. 10.9 and 10.10) and obtains a higher velocity response. One may keep decreasing the specific element height until a stop criterion (i.e. violating the constraints or obtaining a lower velocity response) occurs or may try to decrease the heights of more than one element simultaneously.

This strategy is nondeterministic, and the results may vary from one run to another. Since there are no exact footprints in each step of MDS strategy, and it also involves the engineering intuition of the designer in the design procedure, the MDS strategy seems to be inefficient; however in this study, it has been found to work fairly well (Fig. 10.5 and Table 10.2).

Fig. 10.4 Procedure of the modified direction strategy



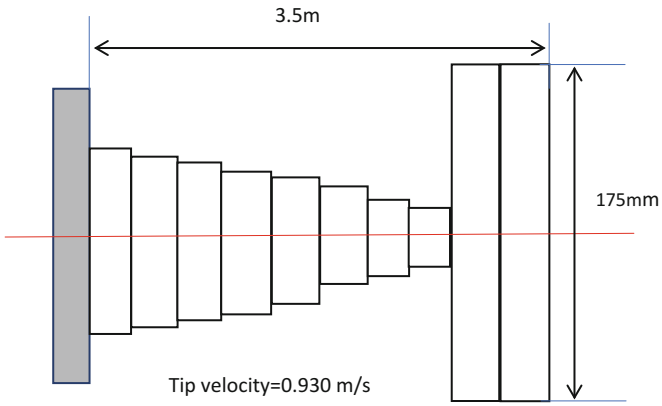


Fig. 10.5 Optimized solution for schematic side view and tip velocity by MDS method (scales are not accurate)

Table 10.2 Optimized heights and maximum stress of each element by MDS method

Element no.	Height (mm)	Stress (MPa)
1	175.4	0.20
2	175.4	0.5
3	16.3	99.04
4	19.5	99.95
5	22.5	99.30
6	25.2	99.35
7	27.7	99.59
8	30.2	99.49
9	32.7	99.76
10	35.2	99.47

10.5.2 Equally Assigned Height (EHA) Strategy

This strategy is exactly the same as the Modified Direction Set, except there is no choice for the designer to assign the element height and/or alter the element height for more than one element simultaneously. Rather, the designer has a choice to assign any value for the increment of altering the heights as it is illustrated below. In this strategy, the test begins with a uniform beam of height H_{ini}, the height of the element that carries the largest stress (i.e. the element fixed to the wall) is decreased by a specified increment, and the heights of the other elements are increased such that the total volume remains constant.

In this strategy, the volume that has been lost by decreasing the height of the candidate element is equally divided (the name Equally Assigned Height derives from this fact) among the other elements that have not been treated yet, and the new heights would then be calculated. This process continues until the stress in any element exceeds the upper limit of the constraint. Then the second last reduced height, which has not violated the constraints, is considered as the best height of this element. This best height is kept unchanged until the end of the process. It is evident that repeating the process for different values of increment increases the chance of obtaining a better solution but at the cost of running several times. The elements are treated in turn, and there is no reiteration for altering the heights of the elements after they have been optimized once. It is obvious that the last element (i.e. element at the free end) must remain untouched because there is no further element to carry the volume which has been taken off from the candidate element which in turn needs to be added to the other element/elements.

The reason for stopping the iteration at the end of the first procedure is that either the elements are at the highest allowable stress or the heights are at the lower limit. During the process, the best set of element heights (i.e. the set that results in a higher velocity response) is stored at a separate location; in that way, the algorithm reports the best set ($H_i, \quad i = 1, \ \ldots \ , n$) found during the whole process.

This strategy is a blind search from objective evaluation point of view because the velocity is not monitored, and no sensitivity analysis is performed in order to find the new search direction in the design space.

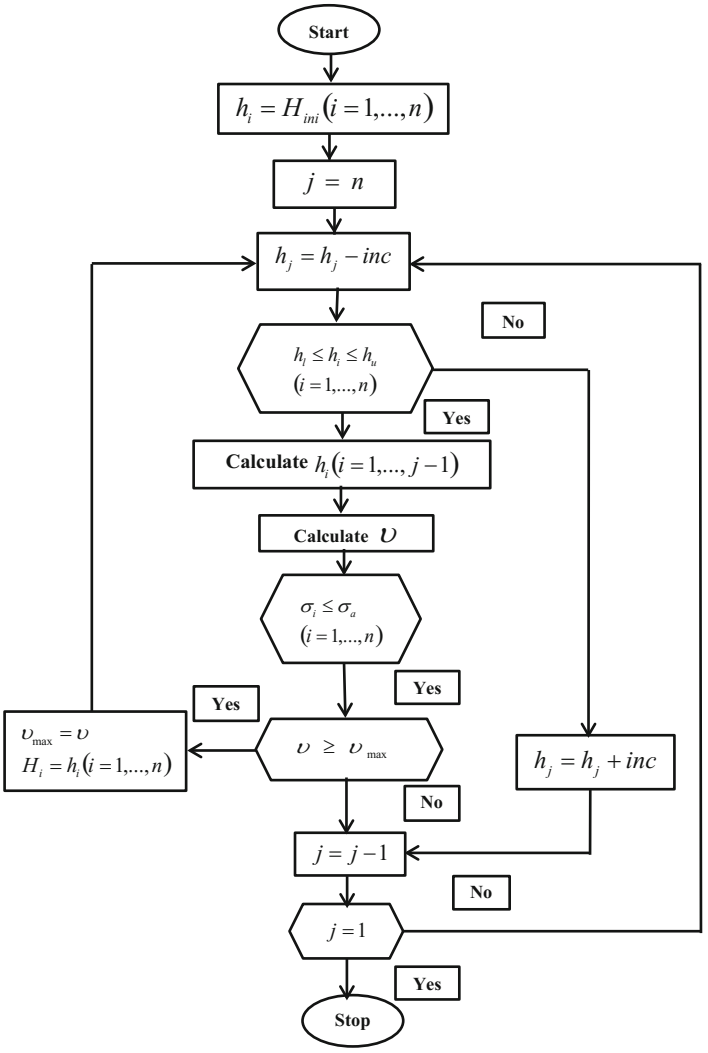


Fig. 10.6 Conceptual flowchart of the Equally Assigned Height strategy

This strategy is illustrated in Fig. 10.6; a program in 'TurboC^{++}' has been written to implement the strategy. Other functions of this program include writing ANSYS input files, running the ANSYS program and subsequently evaluating the output from ANSYS. From all potential solutions, the set which results in a higher velocity response is stored at a separate location; in that way, the algorithm would report the best set found during the whole process. Thus, smaller values of the increment do not necessarily result in a better solution because this strategy does not make use of the objective value information (i.e. maximum velocity) to improve the next step. This strategy could be judged as a fast and efficient strategy from a computing point of view (Fig. 10.7 and Table 10.3).

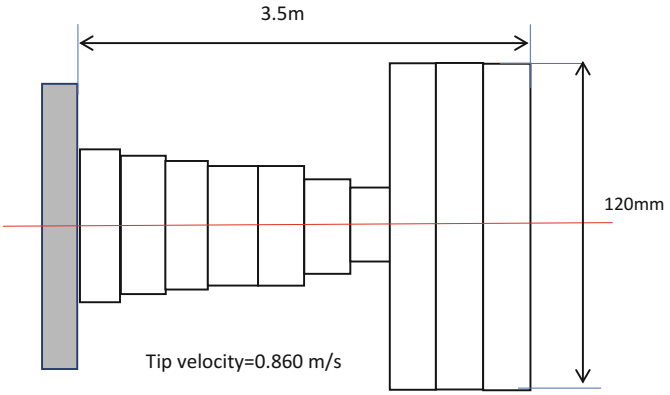


Fig. 10.7 Optimized solution for side view and tip velocity by EAH method

Table 10.3 Optimized heights and maximum stress of each element by EAH method

Element no.	Height (mm)	Stress (MPa)
1	120.8	0.35
2	120.8	0.87
3	120.8	1.60
4	19.0	95.58
5	23.9	81.54
6	27.8	76.58
7	27.8	93.68
8	29.6	98.37
9	32.9	94.47
10	36.5	89.13

10.5.3 Fully Stressed (FS) Strategy

The main idea of the Fully Stressed strategy is searching in the design space to find regions where each element carries the maximum allowable stress when the beam is released from a deflected state. Candidate elements for decreasing/increasing their heights in each trial are selected depending on the current state of their stresses. The height of the overstressed element (h_{nhi}) has been increased, and the height of the element having the lowest stress (h_{nlo}) has been decreased in such a way that the total volume remains constant. The process should be continued until no further step is possible (i.e. all elements have a maximum stress near the allowable stress). A conceptual flowchart of this strategy is presented in Fig. 10.8. It is evident that reiteration is not applicable in this technique because further iteration results in the violation of the stress constraint. Similar to the Equally Assigned Height techniques during the optimization process, the best set of element heights (i.e. the set that results in a higher velocity response) is stored at a separate location; in that way, the algorithm reports the best set ($H_i, \quad i = 1, \ \ldots \ , n$) found during the whole process (Fig. 10.9 and Table 10.4).

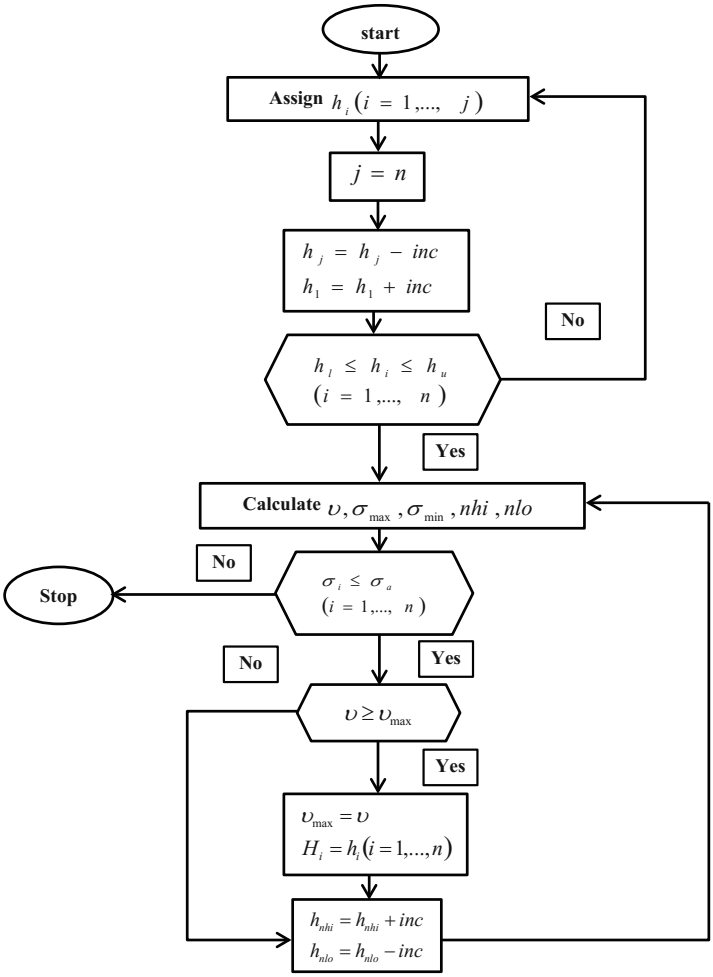


Fig. 10.8 Conceptual flowchart of the Fully Stressed strategy

It is obvious that the result depends on the size of increment which is used to alter the heights and the initial values of the design. This strategy does not make use of the objective value information (i.e. maximum velocity) to improve the next step. From all the potential solutions, the set results of the highest velocity response are recognized as the optimum. A second program in 'TurboC^{++}' has been written to implement this strategy and to run the ANSYS program to obtain stress and velocities.

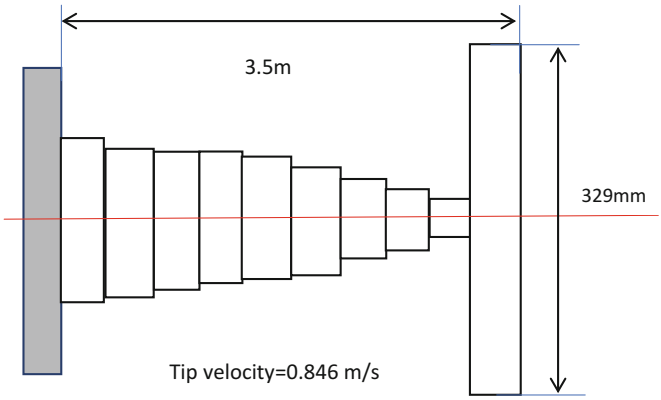


Fig. 10.9 Optimized solution for side view and tip velocity by FS method

Table 10.4 Optimized heights and maximum stress of each element by FS method

Element no.	Height (mm)	Stress (MPa)
1	329.8	0.07
2	13.4	99.52
3	17.2	98.91
4	20.8	93.93
5	23.7	94.17
6	26.0	96.94
7	28.9	94.25
8	30.5	99.86
9	33.8	95.07
10	35.7	97.76

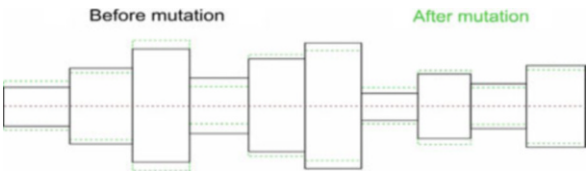


Fig. 10.10 Typical mutation of the beam element heights before scaling

10.5.4 Evolutionary Program (EP)

Evolutionary programs are the subject of growing research interest. Most of their problem-solving is based on the principles of evolution and heredity with a selection process based on the fitness of an individual and a set of operators, such as mutation. They imitate the principles of natural evolution in parameter optimization problems. The concept behind the mutation operation is the introduction of some variability into the design variables by small random changes as illustrated in Fig. 10.10.

Consider a set of individual element heights as given by a mutation of this set is a new set, where

$$\overline{h_i} = h_i + R\epsilon \tag{10.11}$$

and all $\boldsymbol{h_i}$ have been rescaled to keep a constant volume. R is a random number between 0 and 1, and e is usually selected as a small fraction, namely 0.001.

Evolution programs can be formulated in different ways for a given problem. Methods for generating the initial design, techniques for handling constraints and probabilities of applying different operators may differ from one formulation to another. However, a common principle governs all these formulations: an artefact or sample design is selected, it then undergoes some transformations, and during this evolution process, the individuals strive for survival based on the fitness or evaluation criteria [18]. Usually the best (i.e. fittest which is the highest velocity in this study) set of design variables is altered by small random changes (mutation process), and then its fitness is evaluated. A conceptual flowchart of an evolutionary program is presented in Figs. 10.11 and 10.12 and Table 10.5).

10.5.5 Genetic Algorithms (GAs)

Traditionally, numerical optimization techniques have been used to guide the search in the design space. More recently, expert system technology has been integrated into this process [19]. Both the approaches have relatively good features but tend to suffer in constrained optimization problems, where there are large, nonlinear spaces and no enough domain-dependent information to guide the search. In such cases, genetic algorithm is the best suited technique [20].

GAs are inspired by the basic mechanism of natural selection as described by Charles Darwin in *The Origin of Species*. Perhaps the lack of dependence of GAs on the gradient information and the fact that GAs use multiple starting points make them less susceptible to the pitfall of convergence to a local optimum. GAs perform a random search in the design space and are widely applicable global search algorithms that have been used in structural optimization as well as in other areas.

In some aspects, GAs are also different from usual zero-order methods; their search is parallel rather than sequential (i.e. they search from a population of points instead of a single point in the design space), they use probabilistic (nondeterministic) rules, and they can treat problems in which the design variables are continuous or discrete. Therefore, GAs are more robust than classical zero-order methods [21].

GAs use an iterative improvement technique. The population undergoes a sequence of transformations to simulate evolution. At each iteration, the individuals strive for survival: the relatively 'fit' individuals are more likely to reproduce, while the relatively 'unfit' ones are less likely to reproduce. One commonly used technique for doing this is a *roulette wheel* [20]. The roulette wheel selection returns the first individual whose fitness added to the fitness of the preceding population members is greater than or equal to a random number between 0 and total fitness (sum of the fitnesses of all members in the population). Although the selection procedure is random, the individual's chance of being selected is directly proportional to the fitness.

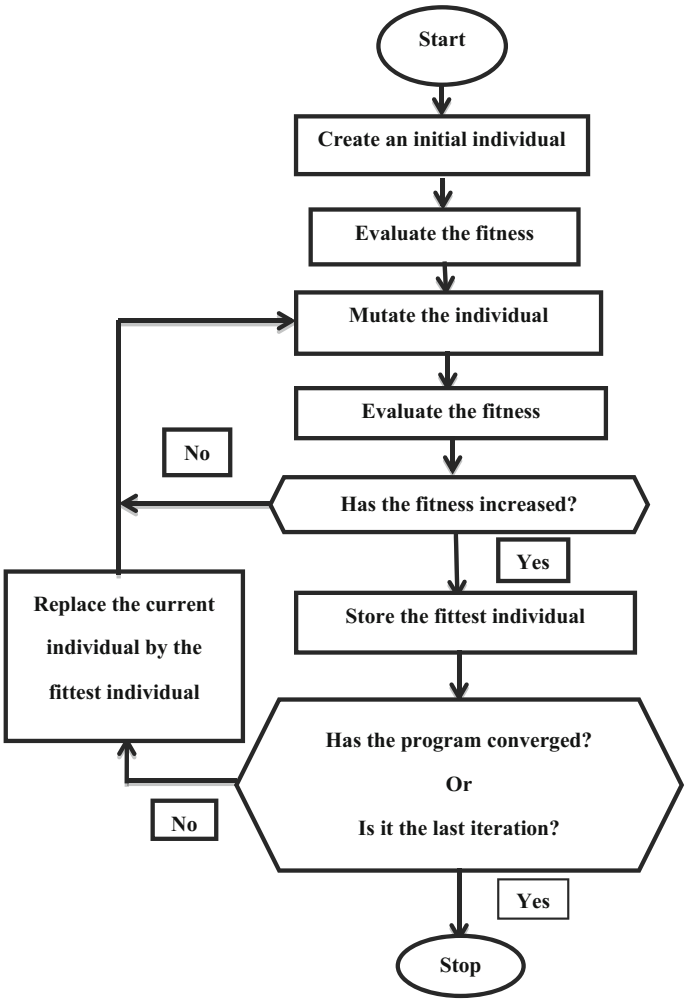


Fig. 10.11 A conceptual flowchart of an Evolutionary Program

After several generations, the program converges with the best individual representing the optimum solution. Stochastic processes are used to generate an initial population of individual designs. Then, a new population is formed by allowing the individuals to reproduce by means of crossover and mutation. A conceptual flowchart of a simple genetic algorithm is presented in Fig. 10.13.

Creation of Initial Population Firstly, a primary population must be randomly generated in MAX_POP number (the number of chromosomes). There are 10 elements in this example, so 10 × (MAX_POP) numbers will be created. Each chromosome (including 10 numbers as the height) should be individually scaled

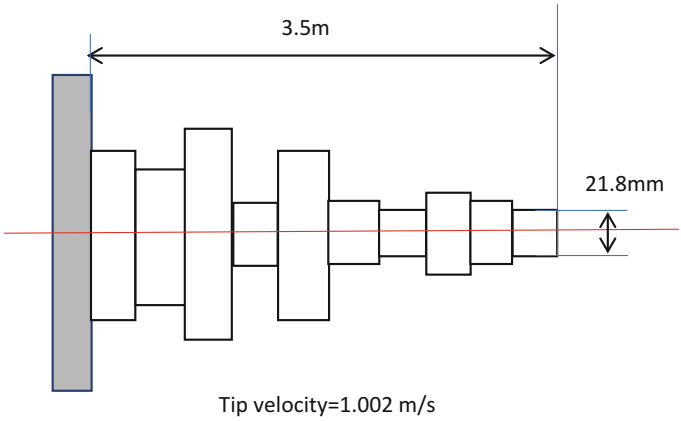


Fig. 10.12 Optimized solution for side view and tip velocity by EP method

Table 10.5 Optimized heights and maximum stress of each element by EP method

Element no.	Height (mm)	Stress (MPa)
1	21.6	9.16
2	25.4	14.21
3	38.5	9.85
4	16.6	74.82
5	20.1	68.64
6	102.8	3.40
7	46.0	22.19
8	130.9	3.48
9	55.2	24.45
10	102.9	8.64

with respect to the constant volume. The value generated for each element must not be less than the values presented in Table 10.1. It should be considered that this random production just occurs in the first generation. Hence, in this part of solution the equality constraint (constant volume) could be exerted.

```
int pp,counter, i;       / * dummy integer */
float oldh[NUMBER_ELEMENTS],volume,scale;
float pop[MAX_POP] [NUMBER_ELEMENTS];
for (pp = 0 ; pp < MAX_POP ; pp++)
{
volume =0;        / * Initialize volume total to 0 */
for(i=2; i<NUMBER_ELEMENTS+2 ; i++ )
{
oldh[i] = 0.056;
volume= volume + oldh[i]*ra*0.5*0.5;
}
scale=(.098/volume);    / * Scale volume of the beam to be 0.098 */
volume =0;
for(i = 2; i<NUMBER_ELEMENTS+2 ; i++ )
```

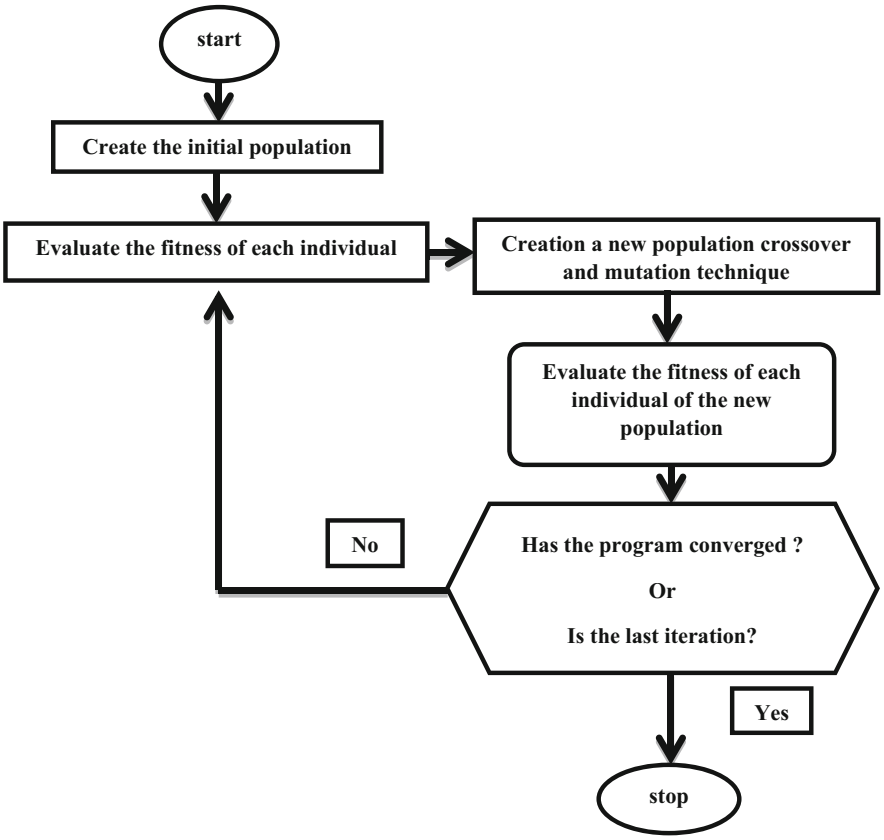


Fig. 10.13 A conceptual flowchart of Simple Genetic Algorithm

```
{
oldh[i] = oldh[i]*scale;
pop[pop][i]=oldh[i];
volume = volume + volume+oldh[i]*ra*0.5*0.5;
}
counter=0;              /* do-while loop for generation the
                            appropriate heights randomly */
do
   {
counter=counter+1;       / * Increment the counter */
volume = 0;                / *Initialize volume total to 0 */
for(i=2;i<NUMBER_ELEMENTS+2 ; i++)
{
h[i] = pop[pp] [i] + (100.0/(10.0+2.0*pp))*(float)rand()
       /RAND_MAX_;
volume = volume + h[i]*ra*0.5*0.5;
                         / * End of for loop with i index     */
volume=0;
```

```
for(i = 2;i< NUMBER_ELEMENTS+2 ; i++ )
{
   pop[pp][i]=h[i]*scale;
   volume=volume+pop[pp][i]*ra*0.5*0.5;
}
if(counter==100)
{
Printf("\n check the program, the No.Of loops is exceeded 100\n");
exit(0);
   }
}
while ( pop[pp] [2]<0.0085 II pop[pp] [3]<0.0131 II
pop[pp] [4]<0.0162II pop[pp] [5]<0.0162 II
pop[pp][6]<0.0162 II pop[pp] [7]<0.0162 II
pop[pp] [8]<0.0162 II pop[pp] [9]<0.0162 II
pop[pp] [10]<0.0162 II pop[pp][11]<0.0162 II
}          / * Associated with for loop with pp index */
```

Evaluation of the Fitness of Each Individual and Handling the Stress Constraint In this part, each chromosome information that is the values of 10 elements as the height which are scaled to reach the constant volume are inserted in *height.src*, and the maximum tip velocity as fitness is calculated by the system command in TurboC^{++} for entering ANSYS. This calculation is repeated MAX_POP times for each generation. First, the values of heights are stored in *can.dat*; next, this file is an input file to *ns6*, a program for static and dynamic analysis in ANSYS. Output static and dynamic answers are stored in *javab* and *je* files, respectively. In this part, these two results combine with each other and yield *jesol*, which comprises maximum tip velocity, maximum stress and static deflection. It is clear that these values are calculated for each chromosome and repeated MAX_POP times in this program.

```
Pr=fopen("d:\\proper.dat","r");
If((Pr=fopen("d:\\proper.dat","r"))==NULL)
Printf("Cannot open input file.\n);
      fscanf(Pr,"% S%d\ n",dumstr,&nd );
      fscanf(Pr,"% S%d\ n",dumstr,&velold );
      fscanf(Pr,"% S%d\ n",dumstr,&k );
      fscanf(Pr,"% S%d\ n",dumstr,&kk );
      fscanf(Pr,"% S%d\ n",dumstr,&biggest );
      fscanf(Pr,"% S%d\ n",dumstr,&generations );
      fscanf(Pr,"% S%d\ n",dumstr,&loops );
      fscanf(Pr,"% S%d\ n",dumstr,&fitness_max );
fclose(Pr);
nd=nd+1;
if(nd==1)
{float a2=0.0085,a3=0.0131,a=0.0162,b=0.4142;
Int i,k;
       for(i=0; i<MAX_POP;i++)
         {
                  Pop[i][2]=a2+(b-a2)*
                           (float)(rand()%RAND_MAX_)/RAND_MAX_;
                  Pop[i][3]=a3+(b-a3)*
                           (float)(rand()%RAND_MAX_)/RAND_MAX_;
       for (k=4; k<NUMBER_ELEMENTS+2;k++)
```

```
              {
                   Pop[i][k]=a+(b-a)*
                             (float)(rand()%RAND_MAX_)/RAND_MAX_;
               }
          }
                POP1=fopen("d:\\height.src","w");
                if ((POP1=fopen("d:\\height.src","w"))==NULL)
                printf("Can not open input file.\n");
for (pp=0;pp<MAX_POP; pp++)
               for(i=2;i<NUMBER_ELEMENTS+2;i++)
                  fprintf(POP1,"h %d=%f\n",i,pop[pp][i]);
fclose(POP1);
}
                Pr=fopen("d:\\proper.dat","w");
     fprintf(Pr,"nd=%d\n",nd);
     fprintf(Pr,"velold=%f\n",velold);
     fprintf(Pr,"k=%d\n",k);
     fprintf(Pr,"kk=%d\n",kk);
    fprintf(Pr,"biggest=%d\n",biggest);
    fprintf(Pr,"generations=%d\n",generations);
      fprintf(Pr,"loops=%d\n",loops);
      fprintf(Pr,"fitness_max=%f\n",fitness_max);
fclose(Pr);
}
        POP=fopen("d:\\height.src","r");
        if ((POP=fopen("d:\\height.src","r"))==NULL)
printf("Can not open input file.\n");
nd1=10*(loops);
              for(i=1;i<nd1+1;i++)
              fscanf(POP," %s%f\n",dumstr,&dumdob);
              for(i=2;i<12;i++)
              { fscanf(POP," %s%f\n",loops,i,&pop[loops][i]);
               Printf("h[%d][ %d]= %f\n",loops,i,pop[loops][i]);
               }
fclose(POP);
run_no=generations*MAX_POP+loops+1;
stmax=0;
volume=0;
    for(i=2;i<NUMBER_ELEMENTS+2;i++)
    {
    h[i]=pop[loops][i];
    volume=volume+h[i]*ra*0.5*0.5;
    }
    Printf("vol=%f",volume);
              OutFile=fopen(d:\\can","w");
              fprintf(OutFile,"/bach");
             for(i=2;i<12;i++)
             fprintf(OutFile,"\nh%d=%f",i,h[i]);
             fprintf(OutFile,"\nl=%f\n",l);
             fclose(OutFile);
             system("copy d:\\can+d:\\ns6 d:\\can.dat");
if(h[2]>0.0085 && h[3]>0.0131 && h[4]>0.0162 && h[5]>0.0162
      && h[6]>0.0162 && h[7]>0.0162 && h[8]>0.0162 &&
    h[9]>0.0162 && h[10]>0.0162 && h[11]>0.0162)
```

```
      {
     k=k+1
                system("d:\\ansys\\bin\\ansys-ed.exe
                          <d:\\can.dat> d:\\can.out");
                system("copy d:\\javab + d:\\je d:\\jesol");
          }
                     for(i=2;i<12;i++)
                     printf("\n The h%d is %f ",i,h[i]);
printf("\nloop=%d , vmax=%f \n",loops,velold);
```

Generally, structural design problems are required to conform to a number of inequality constraints related to stress, displacement, dimensional relationships and other parameters. In the genetic algorithm, these can be performed by using a penalty function approach. The idea of penalty approach is that a constraint violation is penalizing to deter the future use of that set of parameters. A penalty function must penalize the fitness (i.e. velocity) value obtained from a set of design variables that resulted in constraint violation. The fitness function used in this part of solution (with SGA) is as follows:

$$\text{fitness} = \begin{cases} \upsilon_{\max} & ; \text{if } \sigma_{\max} - \sigma_{\mathrm{a}} \leq 0.0 \\ \upsilon_{\max}.r.\left(\frac{\sigma_{\mathrm{a}}}{\sigma_{\max}}\right)^{n} & ; \text{if } \sigma_{\max} - \sigma_{\mathrm{a}} > 0.0 \end{cases} \tag{10.12}$$

where

$v_{\max}$ is the maximum tip velocity
$\sigma_{\max}$ is the maximum stress within the beam
σ_{a} is the maximum allowable stress
r and n are the coefficient and the power or penalty function

This formulation with $r = 0.95$ and n was rounded to work well for this test problem.

Generation of New Population Selection In each generation, the best set is selected as the first set of new population. The first 10% of the population were then selected for the next generation without any replacement or operation on them. This procedure may be done by shifting each member of the old 10% to a one higher location in the matrix. Figure 10.14 shows this procedure for three generations. Thus, after the number of generations equals 10% of the population size, the first 10% of population would be a collection of the best sets. The rest (i.e. 90%) of the population undergoes a sequence of transformation to simulate evaluation. At each iteration, the individuals strive for survival: the relatively 'fit' individuals are more likely to reproduce; one commonly used technique for doing this is a roulette wheel. The roulette wheel selection returns the first individual whose fitness added to the fitness of the preceding population numbers is greater than or equal to a random number between zero and total fitness (sum of the fitnesses of all members in the population). Although the selection procedure is random, the individual's chance of being selected is directly proportional to the fitness.

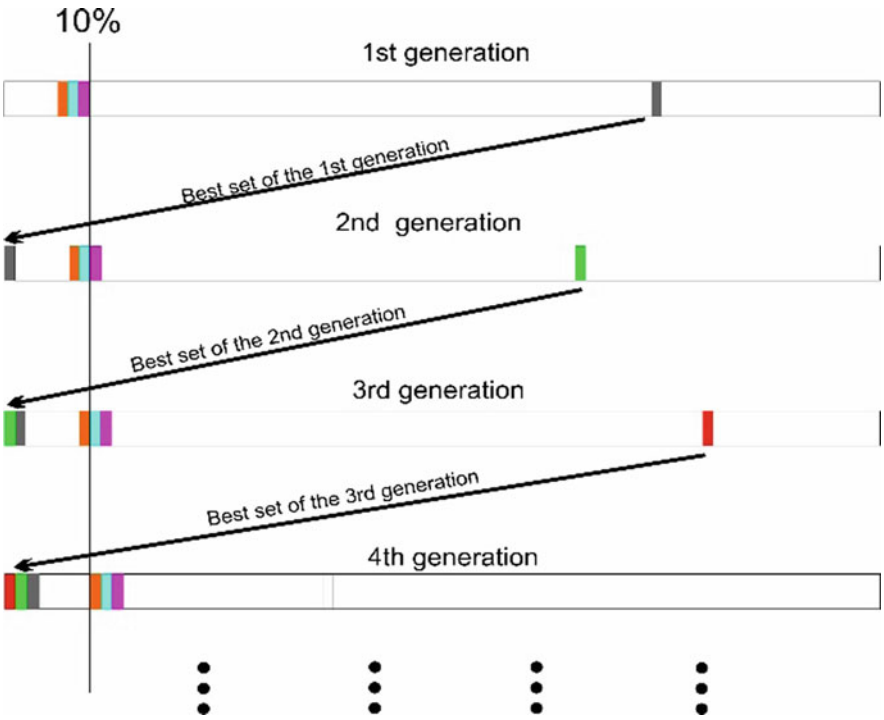


Fig. 10.14 Procedure of replacement of the best set of each generation to be the first set of the new population

```
void sum_ fitness(void)  /* Routine to sum up the fitnesses and
                            calculate the cumulative probability
                            for each chromosome       */
{
int  i;          /*   Lopp index        */
total_ fitness = 0;
for( i  = 0; < MAX_POP ; i + + )
{
total_fitness= total _fitness + fitness[ i];
}
printf(tot = %f\ n",total_fitness);
 for(i =0;< MAX_POP ;i + + )
{
P[i] = 0.;
q[i] = 0.;
}
P[0] = fitness[0]/total_fitness;
q[0] =p[0];
for( i = 1;i<  MAX_POP ; i + +)
{
P[i] = fitness[i]/total_ fitness;
q[i] = q[i-1]+ p[i];
```

```
    }
}
void select _new _pop (vood)        /*Routine to generate a  new
                                  population based on the roulette
                                          wheel technique*/
{
int i,j,k              /*        Loop indices       */
float test ;         /*  Variable for storing the random number*/
for( i = (0.1*MAX_POP + 1 );i < MAX_POP ; i + + )
{
     /*Generate a random No. between 0 and 1  and
      store as test variable     /*
test  = (float )  ( rand()%RAND_MAX_)/RAND_MAX_;
j=0;
for(i = 0 ; i < (0.1*MAX_POP) ; i + + )
{
for( K= 2 ; < NUMBER _ ELEMENTS + 2 ; K + + )
 new pop[ i + 1] [k] = pop [ i] [ k ];
}
for ( k = 2 ; k < NUMBER _ EIEMENTS + 2 ; K + + )
new _ pop [0] [k] = pop [ biggest] [k  ];
              /*while loop  to increment the
              individual numbers till the cumulative
              probability of the individual is less
              then the random number */
 while ( test > q[i] )
{j=j +1;}
for( k = 2 ; k < NUMBER _ ELEMENT + 2 ; K ++)
new _ pop [ i  ][ k ] = pop [j] [k] ;
    }
}
```

Crossover Crossover combines two randomly selected parent individuals (i.e. design variable sets) to form two new individuals by swapping the corresponding segments of the parent individuals. The crossover site is randomly selected. The purpose of the crossover operator is information exchange between different potential solutions. In the test problem, as can be seen in Fig. 10.1, there are 10 design variables, h_1,...,h_{10}, which represent the heights of the beam elements. As an example of one-point simple crossover between two individual sets, consider

$$\left(h_1^{\mathrm{p}}, \ldots, h_{10}^{\mathrm{p}}\right) \quad \text{and} \quad \left(h_1^{\mathrm{q}}, \ldots, h_{10}^{\mathrm{q}}\right) \tag{10.13}$$

the superscripts p and q denote individuals and subscripts represent element number. After crossover at the kth ($1 \leq k < 10$) position, the sets become

$$\left(h_1^{\mathrm{p}}, \ldots, h_k^{\mathrm{p}},\ h_{k+1}^{\mathrm{q}}, \ldots h_{10}^{\mathrm{q}}\right) \quad \text{and} \quad \left(h_1^{\mathrm{q}}, \ldots, h_k^{\mathrm{q}},\ h_{k+1}^{\mathrm{p}}, \ldots h_{10}^{\mathrm{p}}\right) \tag{10.14}$$

```
void cross_over(void)     /* Routine for performing the crossover
                          operation And scaling the new heights*/
{
int i,k;         /* Loop indices  */
```

```
float colume,scale,j;
float temp[20], h[NUMBER_ELEMENTS+2];
                    / * for loop for performing the
                    90% of the population */
for( i = (0.1*MAX POP) ; i < MAX_POP; i = i+2)
{
j = (NUMBER_ELEMENTS*(float)(rand() %RAND_MAX_)/RAND_MAX_)+2;
              / * generate a random No.   between 2 and
              NUMBER_ELEMENTS+2=12 */
for( k = 2; k < j; k++ )
{ temp[k] = new_pop[i] [k];
New_pop[i][k] = new_pop[i+l] [k];
new_pop[i+l] [k] = temp[k];
}
volume=0;
for(k=2; k<NUMBER_ELEMENTS+2;k++)
{ h[k]=new_pop[i][k];
volume= volume+h[k]*ra*0.5*0.5;
}
scale= (0.98/volume);
for(k=2;k<NUMBER_ELEMENTS+2;k++)
new_pop[i][k]=h[k]*scale;
}
}
```

Mutation In the genetic algorithms method, usually after crossover, a few namely 3–8% of the individuals mutate. The concept behind the mutation operation is the introduction of some variability into the design variables by small random changes. Consider a set of individual element heights given by

$$(h_1, \ldots h_i, \ldots, h_{10}) \tag{10.15}$$

Mutation of this set is a new set

$$\left(\overline{h_1}, \ldots, \overline{h_i}, \ldots, \overline{h_{10}}\right) \tag{10.16}$$

where

$$\overline{h_i} = h_i + R\varepsilon \tag{10.17}$$

R is a random number between 0 and 1 and ε is selected as a small fraction, namely 0.001. After exerting mutation operation, certainly the new chromosome will not be included in the volume of the mentioned problem (0.098 m^3). Hence, the resulted height of elements according to the mentioned volume must be scaled. Three percent mutation probability should be chosen in resolving this problem.

```
   void mutate(void)      / * Routine to perform the mutation
                          operation and scale the heights   */
int i,j,k;                / * Loop indices */
float volume, scale;
```

```
float h[NUMBER_ELEMENTS+2];
           / * for loop to mutate 90% of the population with
           the probability of PERCENT MUTATION    */
for( i = (0.1*MAX_POP); i < MAX_POP ; i ++ )
{
j = MAX_ POP * (   float)rand()/RAND_MAX_;
         /*Generate a random  No . 0<r<MAX_POP   */
if (j<(PERCENT_MUTATION*MAX_POP/100))
            /* if j is within the  PERCENT _ MUTATION
                          of  the population size      */
{
volume = 0;
for ( k = 2; NUMBER _ ELEMENTS + 2 ; K + + )
{
h[ k] = new _ pop[ i  ][k] + 0.001*(float) rand()/RAND _ MAX_;
volum = volume + h [k]*ra* 0.5*0.5;
}              /*End of  for with k index     */
 scale = (  .098/ volume);
 for ( k = 2 ; k <NUMBER_ ELEMENTS+2; K+ + )
new_pop[ i] [ k] = h [ k]* scale;
}                / * End  of  if loop of j index    */
}         /* End of loop  of i index       */
}
```

In accordance with the flowchart in Fig. 10.13, after some generations, the program will converge towards an optimum solution.

In Fig. 10.15 and Table 10.6, various solutions are presented in which the number of chromosome in each generation (MAX_POP) and the number of generations in each run are different.

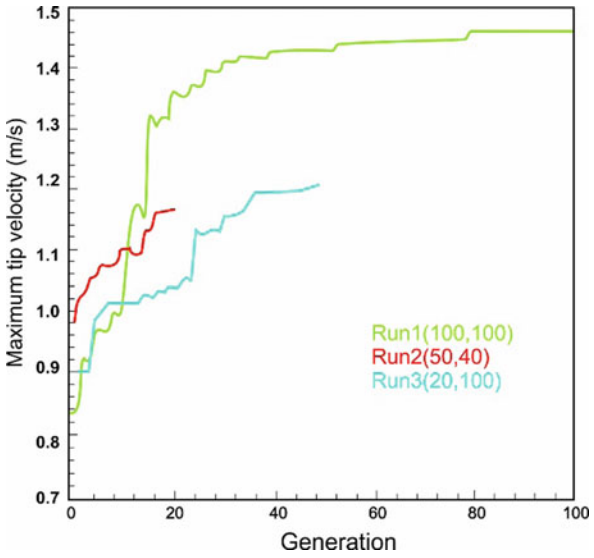


Fig. 10.15 Three runs of SGA program, with Run(gen_num,MAX_POP)

Table 10.6 Three different runs by SGA (different MAX_POP and gen_num for each run)

Run no.	MAX_POP	Generation	ANSYS run	Maximum tip velocity (m/s)
1	100	20	2000	1.1566
2	40	50	2000	1.2025
3	100	100	10,000	1.4562

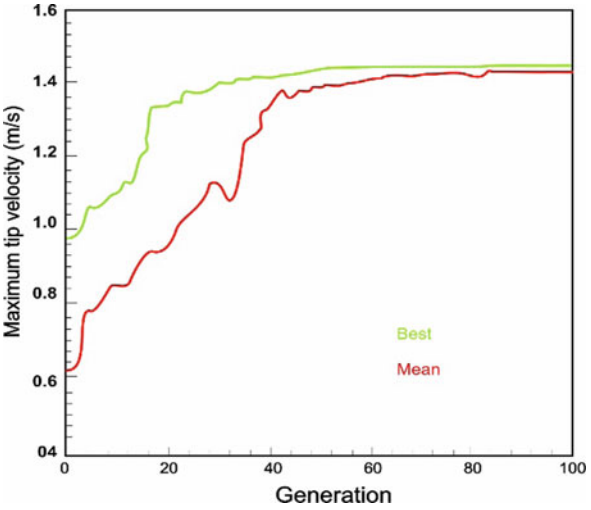


Fig. 10.16 Maximum tip velocity response of the best individual and mean velocity in the population using SGA

Table 10.7 Optimized heights and maximum stress of each element by SGA method

Elements no.	Height (mm)	Stress (MPa)
1	9.34	13.21
2	13.22	15.72
3	66.07	40.52
4	125.55	25.17
5	22.25	32.71
6	17.05	35.16
7	22.18	37.33
8	130.94	20.42
9	116.09	22.85
10	37.26	34.91

In Fig. 10.16 and Table 10.7, maximum tip velocity of the beam for the best chromosome and mean tip velocity of the beam for the existing chromosome in each generation are shown, when the number of population is 100 and the generation is 100, too. In this solution, the obtained maximum velocity for the best chromosome is 1.456 m/s and the maximum stress in it is 40.52 MPa which is less than the allowable quantity.

By running the SGA program several times, it would be seen that the best chromosome in each run is different. *This subject demonstrates that the solution will be trapped in local optimum by means of SGA method* (Fig. 10.17).

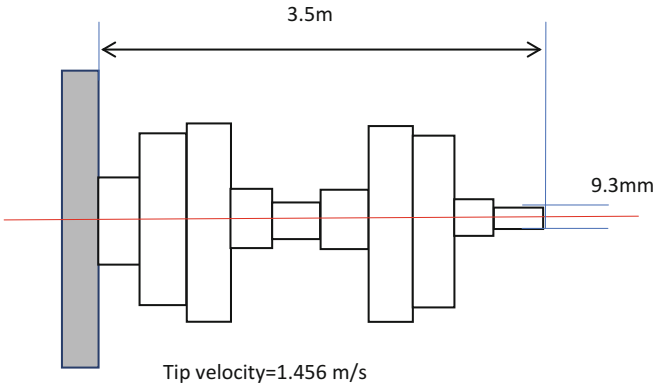


Fig. 10.17 Optimized solution for side view and tip velocity by SGA method

There are three weaknesses and limitations related to the mentioned solution obtained by using the simple genetic algorithm. They are:

1. In each part, by tuning the volume and scaling the height, the volume constraint would be exerted, resulting in the following limitations:
   (a) Generality of the solution of the problem may be damaged. That is the problem; there would be some constraints or a constraint more difficult than constant volume ones which cannot be resolved using scaling method of heights.
   (b) Searching in the whole space of solution could be limited to the one that the constant volume constraint is satisfied. It could be an essential factor in causing the following two main problems.
2. Low velocity is obtained because of using traditional crossover and mutation operators (the operators explained in the previous chapters, namely conventional operators).
3. There will be premature convergence and the solution is trapped in a local optimum because of the low power of crossover operators and traditional mutation operator in searching the solution space.

10.6 Joines and Houck's Method in Penalty Function

In order to overcome limitation 1 mentioned at the end of the previous chapter, the penalty function method should be proposed. Owing to imposing penalty function, instead of the restriction of remaining in search space based on constraints, the algorithms could be able to utilize the whole searching space (feasible as well as infeasible area). Figure 10.18 shows the concept of this issue. Simple

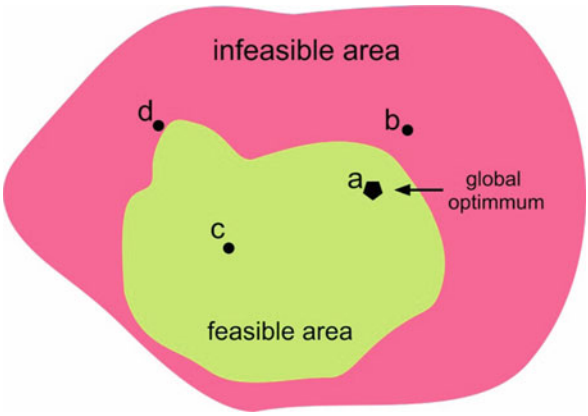


Fig. 10.18 Search space: feasible and infeasible area

comparison of two chromosomes (b) and (c) shows that chromosome (b) includes more information than chromosome (c) related to global optimum of chromosome (a). Without considering the whole space, probably the algorithm will be trapped in local optimum. Comparing the two chromosomes (b) and (d), the amount of penalty for (b) is greater than that of (d) but more efficient for rapidly achieving global optimum of (a). Hence, it has to be less penalized initially and then more penalized for increasing the number of generations.

Generally, in penalty function method, the constrained problems invert to nonconstrained ones. One of the most effective methods is presented by Joines and Houck [22].

Joines and **Houck** considered the following nonlinear programming problem:

$$\begin{array}{lll} \text{Max} & f(x) & \\ \text{S.T.} & g_i(x) \geq 0 & i = 1, 2, \ldots, m_1 \\ & h_i(x) = 0 & i = m_1 + 1, \ldots, m\ (= m_1 + m_2) \\ & x \in X & \end{array} \tag{10.18}$$

where f, g_1, g_2,... g_{m_1}, h_{m_1+1}, h_{m_1+2}... h_m are valued functions defined on E^n, X is a subset of E^n and x is an n dimensional real vector with components $x_1, x_2, \ldots, x_n$ that satisfy the restrictions and meanwhile minimize the function f. The function f is usually called the objective function. Each of the constraints $g_i(x) = 0$ is called an inequality constraint, and each of the constraints $h_i(x) = 0$ is called an equality constraint. The set X might typically include lower and upper bounds of the variables, which is usually called domain constraint. A vector $x \in X$ satisfying all the constraints is called a feasible solution to the problem; the collection of all such solutions forms the feasible region.

Penalty techniques transform the constrained problem into an unconstrained problem by penalizing infeasible solution. **Joines** and **Houck**'s method constructs the evaluation function with penalty term as follows:

$$\text{eval} = f(x) - p\,(t, x) \tag{10.19}$$

The penalty function is also constructed with equations (10.13), a variable penalty factor, and (10.14), a penalty for the violation of constrains, as follows:

$$p\,(t,x) = \rho_t^{\alpha} \sum_{i=1}^{m} d_i^{\beta}(x) \tag{10.20}$$

where $\boldsymbol{t}$ is the iteration of genetic algorithm and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are parameters used to adjust the scale of penalty value. The penalty term for single constraint $\boldsymbol{d_i(x)}$ and the variable penalty factor $\boldsymbol{\rho_t}$ are given as follows:

$$d_i(x) = \begin{cases} 0 & ;\text{ if } x \text{ is feasible} \\ |g_i(x)| & ;\text{ if otherwise for } 1 \le i \le m_1 \\ |h_i(x)| & ;\text{ if otherwise for } m_1 + 1 \le i \le m \end{cases} \tag{10.21}$$

$$\rho t = ct \tag{10.22}$$

where c is a constant. The penalty on infeasible chromosomes is increased along with the evolutionary process because the variable penalty factor ρ_t varies with the iteration of the genetic algorithm for Joines and Houck's method.

The experimental results of Joines and Houck's method indicated that the quality of the solution was very sensitive to the values of the three parameters. How to determine the variable penalty factor is problem-dependent, and it is necessary to design the component with a proper dynamic property suitable for a given problem because this component is constantly increased along with the growth of

$$h_1 + h_2 + \cdots + h_{10} = 0.56 \tag{10.23}$$

$$\text{fitness} = \begin{cases} \upsilon_{\max} - B & ;\text{ if } \sigma_{\max} - \sigma_{\mathrm{a}} \le 0.0 \\ (\upsilon_{\max} - B)\,.r.\left(\frac{\sigma_{\mathrm{a}}}{\sigma_{\max}}\right)^n & ;\text{ if } \sigma_{\max} - \sigma_{\mathrm{a}} > 0.0 \end{cases} \tag{10.24}$$

where

$$B = (ct)^{\alpha}\{|h_1 + h_2 + \cdots + h_{10} - 0.56|\}^{\beta} \tag{10.25}$$

Applying the above procedure in the simple genetic algorithm and supposing that $\alpha = 2$, $\beta = 1$ and $c = 0.2$, the results would be as follows: conventional operators cannot act as fast as possible to research all space dimensions, for in one of run algorithms, the solution in local optimum with $v = 0.149$ m/s and volume $= 0.231$ m^3 would be trapped and the algorithm stopped (Fig. 10.19). *Hence, the next step to solve these problems is to use nonconventional and powerful operators.*

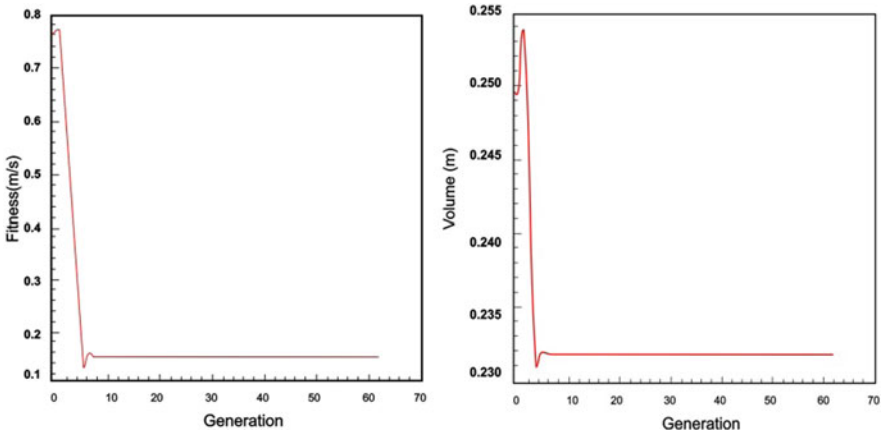


Fig. 10.19 Exerting penalty function in SGA

10.7 Application of Nonconventional Operator for Improving SGA Method

10.7.1 Fuzzy-Connective-Based Crossover Operators

The crossover operator plays a central role in the performance of genetic algorithms; it exploits the available information about the search space from the population. This operator has been highlighted as another key point for solving the premature convergence problem. Numerous investigations have been directed to find the optimal crossover rates and more powerful alternative crossovers which would allow suitable levels of exploitation to be established in the development of such crossover operators, which were attempted for the case of real-code GAs (RCGAs). Fuzzy-connective-based crossover (FCB-crossover) operators were presented for RCGAs based on the use of fuzzy connectives: t-norm, t-conorm, average function and generalized compensation operators. A set of offspring selection mechanisms (OSMS) was proposed which chooses the chromosomes (produced by the crossover) that will be the population members. Different exploration and exploitation degrees may be introduced with the FCB-crossover operators. The OSMs establish a relationship between these properties, so that they induce different diversity levels in the population, and therefore, the premature convergence problem may be eradicated.

To describe the FCB-crossover operators, we follow three steps: define gene combination functions, use these functions to define crossover operators between two chromosomes and apply the crossover operators to the individuals in the population, establishing the number and type of operators along with the OSM to be used.

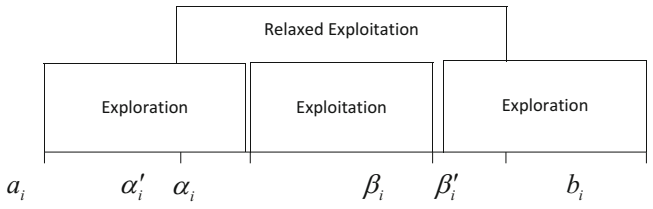


Fig. 10.20 Action interval for a gene

Gene Combination Function Let us consider $c_i^1, c_i^2 \in [a_i, b_i]$ two real genes to be combined and $\alpha_i = \min\left\{c_i^1, c_i^2\right\}$ and $\beta_i = \max\left\{c_i^1, c_i^2\right\}$. The action interval $[a_i, b_i]$ of these genes can be divided into three intervals $[a_i, \alpha_i]$, $[\alpha_i, \beta_i]$ and $[\beta_i, b_i]$. These intervals bound three regions to which the resultant genes of some combination of the former may belong. Moreover, considering a region $\left[\alpha_i^{'}, \beta_i^{'}\right]$ with $\alpha_i^{'} \leq \alpha_i$ and $\beta_i^{'} \geq \beta$ would seem reasonable. This is shown in Fig. 10.20.

The intervals described above could be classified as exploration or exploitation zones; the interval with both the genes being the extremes is an exploitation zone, the two intervals that remain on both the sides are exploration zones, and the region with extremes $\alpha_i^{'}$ and $\beta_i^{'}$ could be considered as a relaxed exploitation zone.

With regard to these intervals, four functions were proposed: **F, S, M** and **L** defined from $[a, b]$ * $[a, b]$ in $[a, b]$, $a, b \in R$, which fulfil

$$
\begin{aligned}
&(P1) \qquad \forall c, c' \in [a, b] \;\; F\left(c, c'\right) \leq \min\left\{c, c'\right\} \\
&(P2) \qquad \forall c, c' \in [a, b] \;\; S\left(c, c'\right) \geq \max\left\{c, c'\right\} \\
&(P3)\; \forall c, c' \in [a, b] \;\; \min\left\{c, c'\right\} \leq M\left(c, c'\right) \leq \max\left\{c, c'\right\}. \\
&(P4) \quad \forall c, c' \in [a, b] \;\; F\left(c, c'\right) \leq L\left(c, c'\right) \leq S\left(c, c'\right) \\
&(P5) \quad F, S, M, \text{and } L \text{ are monotone and non-decreasing}
\end{aligned}
$$

Each one of these functions allows us to combine two genes giving results belonging to each one of the aforementioned intervals. Therefore, each function will have different exploration or exploitation properties depending on the range being covered by it.

The fuzzy connectives t-norm, t-conorm, averaging function and generalized compensation operators may be used to build these functions; we may associate F to a t-norm, S to a t-conorm, M to an averaging operator and L to a generalized compensation operator. In order to do so, we need to transform the genes, that will be combined, from the interval $[a, b]$ to $[0, 1]$ and later result in $[a, b]$.

Complying with a set of fuzzy connectives, a set of functions associated with it is built as described below:

IF $c, c^{'} \in [a, b]$ then

$$
F\left(c, c'\right) = a + (b - a).T\left(s, s'\right) \tag{10.26}
$$

Table 10.8 Families of fuzzy connectives, from top to bottom, the Logical, Hamacher, Algebraic, and Einstein families are shown

t-norm	t-conorm	Averaging fun. ($0 \leq \lambda \leq 1$)	Gen. com Op.
$T_1(x,y) = \min(x,y)$	$G_1(x,y) = max(x,y)$	$P_1(x,y) = (1-\lambda)x + \lambda y$	$C_1 = T_1^{1-\lambda}.G_1^{\lambda}$
$T_2(x,y) = \frac{xy}{x+y\_xy}$	$G_2(x,y) = \frac{x+y-2xy}{1-xy}$	$P_2(x,y) = \frac{1}{\frac{y-y\lambda-xy+x\lambda}{xy}+1}$	$C_2 = P_2(T_2, G_2)$
$T_3(x,y) = xy$	$G_3(x,y) = x+y-xy$	$P_3(x,y) = x^{1-\lambda}y^{\lambda}$	$C_3 = P_3(T_3, G_3)$
$T_4(x,y) = \frac{xy}{1+(1-x)(1-y)}$	$G_4(x,y) = \frac{x+y}{1+xy}$	$P_4(x,y) = \frac{2}{1+\left(\frac{2-x}{x}\right)^{1-\lambda}\left(\frac{2-y}{y}\right)^{\lambda}}$	$C_4 = P_4(T_4, G_4)$

$$\boldsymbol{S}\left(\boldsymbol{c}, \boldsymbol{c}'\right) = \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{a}).\boldsymbol{G}\ \left(\boldsymbol{s}, \boldsymbol{s}'\right) \tag{10.27}$$

$$\boldsymbol{M}\left(\boldsymbol{c}, \boldsymbol{c}'\right) = \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{a}).\boldsymbol{B}\ \left(\boldsymbol{s}, \boldsymbol{s}'\right) \tag{10.28}$$

$$\boldsymbol{L}\left(\boldsymbol{c}, \boldsymbol{c}'\right) = \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{a}).\boldsymbol{C}\ \left(\boldsymbol{s}, \boldsymbol{s}'\right) \tag{10.29}$$

where

$$s = \frac{c-a}{b-a} \quad \text{and} \quad s' = \frac{c'-a}{b-a} \tag{10.30}$$

These operators have the properties of being continuous and nondecreasing and satisfy the respective properties $(P1) - (P5)$. The families of fuzzy connectives used are shown in Table 10.8.

Fuzzy connective families in Table (10.8) fulfil the following property:

$$(P6)\ T_4 \leq T_3 \leq T_2 \leq T_1 \leq P_J\ (J = 1, \ldots, 4) \leq G_1 \leq G_2 \leq G_3 \leq G_4$$

FCB-Crossover Let us assume that $O \in \{F, S, M, L\}$ and $K_1 = \left(k_1^1, \ldots, k_n^1\right)$ and $K_2 = \left(k_1^2, \ldots, k_n^2\right)$ are two chromosomes that have been selected to apply the crossover operator to them. We can generate the chromosome $H = (h_1, \ldots, h_2, \ldots, h_3)$ as

$$h_i = O\left(k_i^1, k_i^2\right), \quad i = 1, \ldots, n \tag{10.31}$$

This operator applies the same F, S, M or L function for all the genes in the chromosomes to crossover. For this reason, they will be called F-crossover, S-crossover, M-crossover and L-crossover when the F, S, M and L functions are applied, respectively. It should be emphasized that these crossover operators have different properties: the F-crossover and S-crossover operators show exploration, the M-crossover operator shows exploitation and the L-crossover operator shows relaxed exploitation.

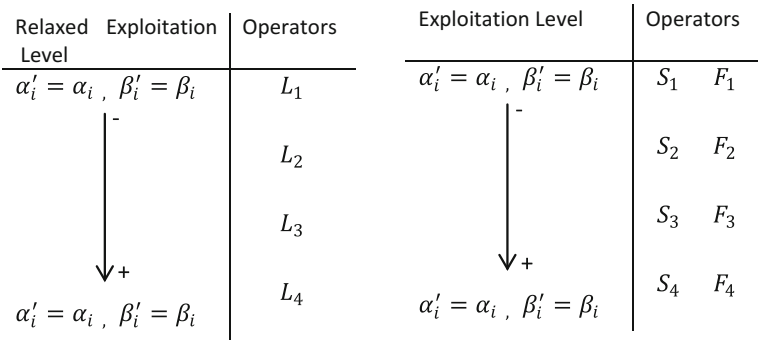


Fig. 10.21 Exploration and relaxed exploitation levels

According to the property (*P*6) of the families of fuzzy connectives in Table 10.8, we can see that the degree to which each crossover operator shows its related property will depend on the fuzzy connective on which it is based. Thus, we dispose F_j-crossover and S_j-crossover operators with different exploration levels; the F_4-crossover and S_4-crossover show the maximum exploration, whereas the F_1-crossover and the S_1-crossover represent the minimum exploration. These operators give results between the extremes of the exploration domain. With respect to the L_j-crossover, the level of relaxed exploitation directly depends on the T_j, G_j and P_j used for its definition. Figure 10.21 shows the behaviour of the interval definition for the crossover operators.

Application of the *FCB*-Crossover By using the previously proposed crossover operators, different application strategies can be built which are differentiated according to how they carry out the following two steps:

1. Generation of offspring using the different defined operators.
2. Selection of offspring resulting from the crossover which will form part of the population.

Herrera et al. presented four proposals [23], which are shown in Table 10.9, where we specify the strategy name, the *FCB*-crossover operators applied, the number of chromosomes in the population that should undergo crossover and finally the type of *OSM* and the way in which it introduces diversity (measured in three diversity levels: strong, high and weak).

The experiments developed by Herrera et al [23] on different test functions showed, in general, the suitability of using the logical *FCB*-crossovers and the *OSM* that choose the two best elements from a set of four where the exploitation and exploration properties are equitably assigned. Another important conclusion was that by using F-crossover and S-crossover with F and S functions distant from the minimum and maximum (logical F and S function), respectively, the population diversity levels obtained are greater, which agrees with the properties shown in Fig. 10.21. This result suggested the definition of the dynamic *FCB*-crossovers in [23],

Table 10.9 Application strategies (N_c is the number of chromosomes that should undergo crossover, N is the population size and P_c is the crossover probability)

Statergy	FCB − crossover	N_c	OMS	Diversity
STI	$F-, S-$ *and* $M-$ *crossover*	$^2/_3 . P_c . N$	All three offspring	Strong
*ST*2	$F-, S-$ *and tow* $M-$ *crossovers*	$P_c . N$	The two most promising	Weak
*ST*3	$F-, S-, M$ *and* $L-$ *crossover*	$P_c . N$	The two most promising	Weak
*ST*4	$F-, S-, M$ *and* $L-$ *crossover*	$^1/_2 . P_c . N$	All three offspring	High

which extend the use of the *FCB*-crossover operators in order to follow the principle 'to protect the exploration in the initial stages and the exploitation later'.

```
void cross_over(void)     /*Routine for performing the
                            fuzzy_based crossover
                               Operation with st1+st4 strategy*/
{      int generations;
           Int i j = 2,k;              /*Loop indices       */
float volume, scale, landa,n,a=0.0162,a2=0.0085,
          a3=0.0131,b=0.4142;
float
temp[20],s1[NUMBER_ELEMENTS+2],s2[NUMBER_ELEMENTS+2],
          t[NUMBER_ELEMETS],g[NUMBER_ELEMENTS];
         float H[MAX_POP][NUMBER_ELEMENTS+2];
if(generations<=10)
{
    for(i=(0.1*MAX_POP);i<13;i=i+2)
If(generatins<=10)
{
for(i=(0.1*MAX_POP);i<13;i=i+2)
{landa=(float)(rand0%RAND_MAX_)/RAND_MAX_;
s1[2]=(new_pop[i][2]-a2)/(b-a2);
s2[2]=(new_pop[i+1][2]-a2)/(b-a2);
t[2]=s1[2]*s2[2];//(s1[2]+s2[2]-s1[2]*s2[2]);
g[2]=(s1[2]+s2[2]-s1[2]*s2[2]);//(1.0-s1[2]*s2[2]);
s1[3]=(new_pop[i][3]-a3)/(b-a3);
s2[3]=(new_pop[i+1][3]-a3)/(b-a3);
t[3]=s1[3]*s2[3];//(s1[3]+s2[3]-s1[3]*s2[3]);
g[3]=(s1[3]+s2[3]-s1[3]*s2[3]);//(1.0-s1[3]*s2[3]);
for(k=4:k<NUMBER ELEMENTS+2;k++)
{
s1[k]=(new_pop[i][k]-a)/(b-a);
s2[k]=(new_pop[i+1][k]-a)/(b-a);
t[k]=s1[k]*s2[k];//(s1[k]+s2[k]-s1[k]*s2[k]);
g [k]=(s1[k]+s2[k]-s1[k]*s2[k]);//(1.0-s1[k]*s2[k]);
}
n=1-landa;
H[j][2]=a2+(b-a2)*t[2];
H[j+1][2]=a2+(b-a2)*g[2];
```

```
H[ j+2][2] =a2+(b-a2)*(pow(s1[2],n)*pow(s2[2],landa));
H[j][3] =a3+(b-a3)*t[3];
H[j+1][3]=a3+(b-a3)*g[3 ];
H[j+2][3]=a3+(b-a3)*(pow(s1[3],n)*pow(s2[3],landa));
for(k=4;k<NUMBER_ELEMENTS+2;k++)
{
H[j][k]=a+(b-a)*t[k];
H[j+1][k]=a+(b-a)*g[k];
H[j+2][k]=a+(b-a)*(pow(s1[k],n)*pow(s2[k],landa));
}
j=j+3;
{
for(i=2;i<MAX_POP;i++)
}
    for (k=2;k<NUMBER_ELEMENTS+2;k++)
}
new_pop[i][k]=H[i][k];
{
{
{
If(generation>10)
}
for(i=(0.1*MAX_POP);i<11;i=i+2)
}landa= )fioat)(rand0%RAND_MAX)/RAND_MAX_;
s1[2]=(new_pop[i][2]-a2)/(b-a2);
s2[2]=(new_pop[i+1][2]-a2)/(b-a2);
t[2]=s1[2]*s2[2];//(s1[2]+s2[2]-s1[2]*s2[2]);
g[2]=(s1[2]+s2[2]-s1[2]*s2[2]);//(1.0-s1[2]*s2[2]);
s1[3]=(new_pop[i][3]-a3)/(b-a3);
s2[3]=(new_pop[i+1][3]-a3)/(b-a3);
t[3]=s1[3]*s2[3];//(s1[3]+s2[3]-s1[3]*s2[3]);
g[3]=(s1[3]+s2[3]-s1[3]*s2[3]);//(1.0s1[3]*s2[3]);
for(k=4;k<NUMBER_ELEMENTS+2;k++)
}
s1[k]=(new_pop[i][k]-a)/(b-a);
s2[k]=(new_pop[i+1][k]-a)/(b-a);
t[k]=s1[k]*s2[k];//(s1[k]+s2[k]-s1[k]*s2[k]);
g[k]=(s1[k]+s2[k]-s1[k]*s2[k])//1.0-s1[k]*s2[k]);
}
H[j][2]=a2+(b_a2)*t[2];
H[j+1][2]=a2+(b_a2)*g[2];
H[j+2][2]=a2+(b-a2)*(pow(s1[2],n)*pow(s2[2],landa));
H[j+3][2]=a2+(b-a2)*(pow(t[2],n*pow(g[2],landa));
H[j][3]=a3+(b-a3)*t[3];
H[j+1][3]=a3+(b-a3)*g[3];
H(j+2)[3]=a3+(b-a3)*(pow(s1[3],n)*pow(s2[3],landa));
H[j+3][3]=a3+(b-a3)*(pow(t[3],n)*pow(g[3],landa));
for(k=4);k<NUMBER_ELEMENTS+2;k++)
}
H[j][k]=a+(b-a)*t[k];
H[j+1][k]=a+(b-a)*g[k];
H[j+2][3]=a3+(b-a3)*(pow(s1[3],n)*pow(s2[3],landa));
H[j+3][3]=a3+(b-a3)*(pow(t[3],n)*pow(g[3],landa));
for(k=4;k<NUMBER_ELEMENTS+2;k++)
```

```
}
H[j][k]=a+(b-a)*t[k];
H[j+1][k]=a+(b-a)*g[k];
H[j+2][k]=a+(b-a)*(pow(s1[k],n)*pow(s2[k],landa));
H[j+3][k]=a+(b-a)*(pow(t[k],n)*pow(g[k],landa));
{
j=j+4;
{
for(i=2;i<MAX_POP;i++)
}
       for(k=2;k<NUMBER_ELEMENTS+2;k++)
       }
                    new_pop[i][k]=H[i][k];
 {
}                  / *Loop indices*/
/*for loop for performing the 90 percent of the population */
for(i=(0.1*MAX_POP);i<MAX_POP; i=i+2)
}
j=(NUMBER_ELEMENTS*(float)(rand() %RAND_MAX_  )/RAND_MAX_)+2;
*/Generate a random No. Between 2 and NUMBER_ELEMENTS+2 */
for(k=2;k<j;k++)
} temp[k]=new_pop[i][k];
new_pop[i][k]=new_pop[i+1][k];
new_pop[i+1][k]=temp[k];
{
volume=0;
for(k=2;k<NUMBER_ELEMENTS+2;k++)
   {h[k]=new_pop[i][k];
     volume=    volume+h[k]*ra*0.5*0.5;
   {
                      {
}
```

10.7.2 Dynamic Mutation Operator

This operator also called nonuniform mutation is given by Janilow and Michalewicz [24]. It is designed for fine-tuning capabilities aimed at achieving high precision. For a given parent x, *if* the element X_k of it is selected for mutation, the resulting offspring is $\boldsymbol{x}' = \left[x_1, \ldots, x'_k, \ldots, x_n\right]$, where x'_k is randomly selected from the following two possible choices:

$$x'_k = x_k + \Delta\left(t, x_k^U - x_k\right) \quad or \quad x'_k = x_k + \Delta\left(t, x_k - x_k^L\right) \tag{10.32}$$

The function $\Delta(t, y)$ returns a value in the range $[0, y]$ such that the value of $\Delta(t, y)$ approaches 0 as t increases (t is the generation number). This property causes the operator to search the space uniformly initially (when t is small) and locally at later stages. The function $\Delta(t, y)$ is given as follows:

$$\Delta(t, y) = y.r.\left(1 - \frac{t}{T}\right)^{b} \tag{10.33}$$

where $\boldsymbol{r}$ is a random number from $[\mathbf{0}, \mathbf{1}]$, $\boldsymbol{T}$ is the maximal generation number and $\boldsymbol{b}$ is a parameter determining the degree of nonuniformity. It is possible for the operator to generate an offspring which is not feasible. In such a case, we can reduce the value of random number.

Regarding the mentioned subjects and considering the presented principle to reach the ultimate solution, the 10 first generation should use $\boldsymbol{ST_1}$ strategy and the rest $\boldsymbol{ST_4}$ strategy. Of course, this method used fuzzy operators ($\boldsymbol{C_2}, \boldsymbol{P_2}, \boldsymbol{G_2T_2}$). Hence, by applying Joines and Houck's method and using nonconventional operators (FCB-Crossover) and dynamic mutation with coefficient $b = 1$, the ultimate solution will be obtained.

```
Void mutate(void)         /*Routine for dynamic mutation*/
}
Int i,j,k,1,m,generations,s,          /*Loop indices      */
Fioat volume,scale;
Fioat  h[NUMBER_ELEMENTS +2],HU=0.4142,HL=0.0162;
Fioat H[NUMBER_ELEMENTS*MAX_POP];
            /*for Loop to mutate 90% of th  Population
            with the probability of
            RERCENT_MUTATION   */
For(i=0;i<MAX_POP;i++)
}
For (k=2;k<NUMBER ELEMENTS+ 2;k++)
}H[i*NUMBER_ELEMENTS+k-2]=new_pop[i][k];}
{
K=6;       /*k=(fioat)PERCENT_MUTATION/100.0)
                 *NUMBER_ELEMENTS*MAX_POP;*/
For (i=0;i<k;i++)
}
Radumize0;
J=random(NUMBER_EIEMENTS*MAX_POP);
S=random(2);
If(s==0)
H[j]=H[j]+0.01*((fioat)(rand0%RAND_MAX)/RAND_MAX_)
      *(HU_H[j])*(1.0-
)fioat)generation/(fioat)gen_num);
if(s==1)
H[j]=H[j]-0.01*((fioat)(rand0%RAND_MAX)/RAND_MAX_)
      *(H[j]-HL)*(1.0-
)fioat)generation/(fioat)gen_num);
If(H[j]<0.0162)
H[j]=0.016201;
If(H[j]>0.4142)
H[j]=0.4142;
m=j%10+2
l=  )j+2-m)//NUMBER_EIEMENTS;
new_pop[l][m]=H[j];
{
{
```

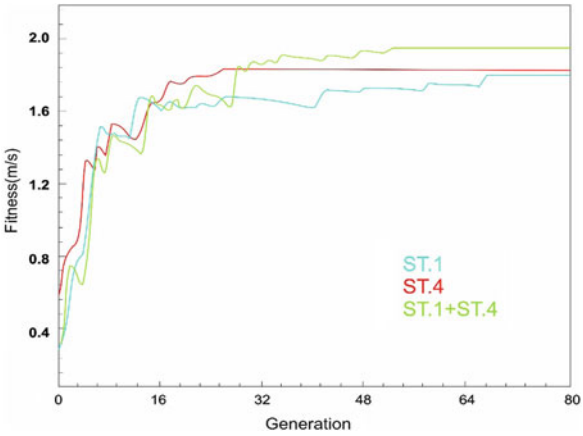


Fig. 10.22 Graphic results of ST1, ST4 and ST1 + ST4

In Figs. 10.22 and 10.23 and Table 10.10, the outcomes according to ST1 (exerted on whole generation), ST4 (exerted on whole generation) and ST1 + ST4 (exerted ST1 on 10 first generation and ST4 on the rest) show improvement in results (maximum tip velocity). In Table 10.10, the complementary information about the best solutions by ST1, ST4 and ST1 + ST4 is presented.

The ultimate solution is presented in Figs. 10.24 and 10.25 and Table 10.11. It would be said that in this procedure, the number of population is 20, as seen in Fig. 10.22, to converge at generation 80 and to obtain the answer. For the ultimate solution of the particular case presented, the artificial selection replaced the natural selection. It is interesting in which ultimate solution by running several times of program, the acquired results are the same for all heights and maximum tip velocity. It could be demonstrate that we reach to global optimum.

Artificial Selection As shown previously, the concepts of genetic algorithms are inspired from nature. The operators existing in nature are very simple and take place gradually, spending huge amounts of time. Therefore, the evolution procedure took place in a million years. Hence, previously, nonconventional operators are utilized. Now, for the case example, artificial selection will be exerted instead of natural selection, which is roulette wheel. To reach the global optimum rapidly, the best chromosomes are directly selected and placed on the next pool instead of random selection. The selected chromosomes are arranged based on the value of fitness of each one of them.

```
  Void select_ new_pop (void)    /*Routine to generate a
                                        new population
                                  Based on the artificial
                                        selection */
{
int s=1,counter= 2,i,k,l,counter1,ll,counter2;
float fmax;
int j , generations;
for(i=0;i<(0.1*MAX_POP-1);i++)
```

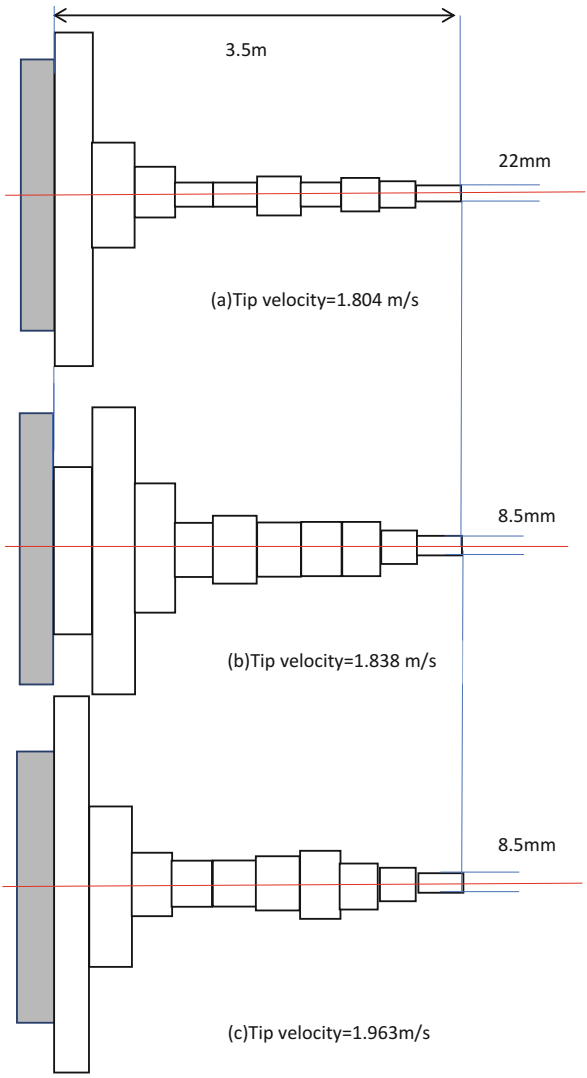


Fig. 10.23 Side view of beam acquired with (**a**) ST1, (**b**) ST4 and (**c**) ST1 + ST4

```
{
for(k=2;k<NUMBER_ELEMENTS+2;k++)
New_pop[i+1][k]=pop[i][k];
}
for(k=2;k<NUMBER_ELEMENTS+2;k++)
New_pop[0][k]=pop[biggest][k];
for(l=2;l<MAX_POP;l++)
{
fmax=0.000001;
counter1=-1;
for(i=0;i<MAX_POP,i++)
```

Table 10.10 Results obtained from the strategies ST1, ST4 and ST1 + ST4

Element no.	Height (mm) by ST1	Height (mm) by ST4	Height (mm) by ST1 + ST4
1	22.07	8.522	8.576
2	16.93	13.45	13.19
3	18.34	16.24	16.57
4	16.61	16.26	32.59
5	25.76	16.30	25.93
6	17.15	22.65	16.35
7	17.66	18.69	17.85
8	22.56	65.88	21.31
9	93.75	263.33	101.16
10	309.11	118.59	306.44
Maximum tip velocity	1.804	1.838	1.963
No. of generations	80	80	80
No. of population	20	20	20
Volume (m^3)	0.09799	0.09799	0.09800
Max. stress	80	83	93

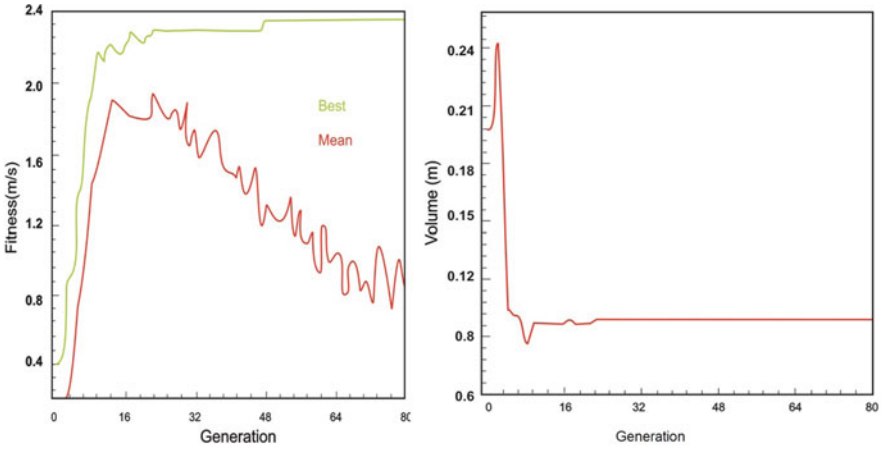


Fig. 10.24 Graphical variation for best chromosome in ultimate solution

```
{
if(fmax<fitness[i])
{
 fmax=fitness[i];
         Counter 1=i;
 }
}
If(counter 1>=0)
```

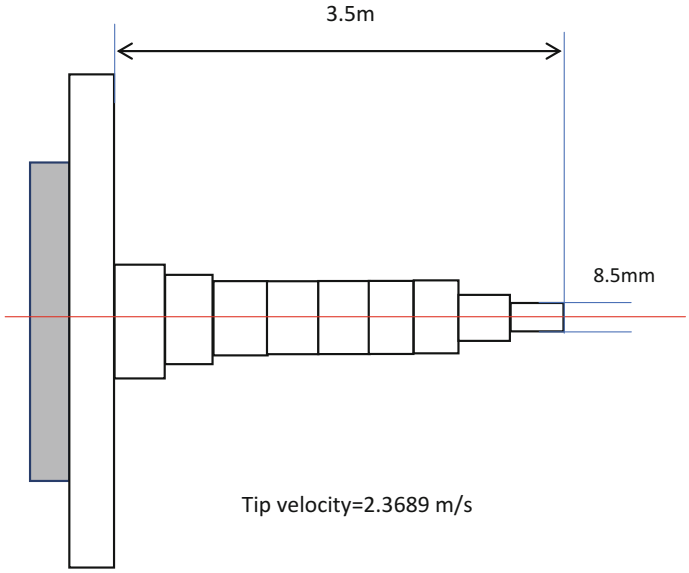


Fig. 10.25 Side view of the beam with maximum tip velocity by ultimate solution

Table 10.11 Ultimate solution ST1 + ST4 strategy for dynamic mutation and artificial selection

Element no.	Height (mm)
1	8.511
2	13.104
3	16.212
4	16.201
5	16.223
6	16.430
7	16.858
8	19.534
9	27.242
10	409.686
Maximum tip velocity	2.3689 m/s
Number of generation	80
Number of population	20
Volume (m^3)	0.09800
Maximum stress (MPa)	99

```
{
fitness[counter 1]=0.0;
for(k=2;k<NUMBER_ELEMENTS+2;K++)
new_pop[counter][k]=pop[counter 1][k];
Counter=counter+1;
}
If(counter1<0)
```

```
{  ll=l-2;
counter2=counter-2;
for(k=2;k<NUMBER_ELEMENTS+2;K++)
new _pop[l][k]=new_pop[11-s*counter2+2][k];
if(1==(s+1)*counter2+1)
{s=s+1;}
    }
}
```

10.8 Conclusion

Considering the solution of problem in this topic with SGA, it would be seen in each run program that various values of velocity and heights are obtained. But the ultimate solution in each run could be almost the same in terms of velocity and heights. While comparing the results obtained from SGA and the ultimate solution, it was found that the time taken to reach the optimum solution is more for SGA. Moreover, it would be seen that in the SGA, the solution is trapped in local optimum, whereas the result acquired by means of ultimate solution method is reliable and the most likely will be a global optimum.

Eventually, the powerful and generalized method represented in this work is an efficient and flexible one that could be used in solution of other mechanical dynamic response optimization problems by revising or altering various parameters which exist in all operations which are presented.

References

1. Yamakawa, H. (1984). Optimum structural designs for dynamic response. In R. H. Atrek, K. M. Gallagher, & O. C. Zinkiewicz (Eds.), *New directions in optimum structural* (pp. 245–266). Chichester: Wiley.
2. Szyszkowski, W., & King, J. M. (1992). Optimization of frequencies spectrum in vibration of flexible structures. In *Fourth ALAA/USAF/OAI symposium on multidisciplinary analysis and optimization, number AIAA-92-4775-CP, Cleveland, OH, September 21-23* (pp. 695–700).
3. Grandhi, R. V., & Venkayya, V. B. (1988). Structural optimization with frequency constraints. *ALAA Journal, 26*(7), 858–866.
4. Olhoff, N. (1980). Optimization of transverse vibrating beams and rotating shafts. In E. J. Haug & J. Cea (Eds.), *"Optimization of distributed parameter structures", Iowa City, Iowa, USA, 1981. Proceedings of the NATO Advanced Study Institute on Optimization of Distributed Parameter Structural Systems, May 20-June 4* (pp. 177–199). Alphen aan den Rijn: Sijthff & Nordhoff.
5. Niordson, F. I. (1963). On the optimal design of a vibrating beam. *Quarterly Journal of Applied Mechanics, 23*(1).
6. Venkayya, V. B., Khot, N. S., Tischler, V. A., & Tailar, R. F. (1971). Design of optimum structures for dynamic loads. In *Proceedings of the third conference on matrix methods in structural mechanics* (pp. 619–658). Athens, OH: Wright Patterson Air Force Base, Air Force Flight Dynamic Lab, University of Ohio.

7. Khan, M. R., & Willmert, K. D. (1981). An efficient optimality criterion method for natural frequency constrained structures. *Journal of Computers and Structures, 14*(5-6), 501–507.
8. Jan, C. T., & Truman, K. Z. Optimal design with multiple frequency constraints. In S. Hernandez & C. A. Brebbia (Eds.), *Optimization of structural systems*.
9. Turner, M. J. (1967). Design of minimum mass structures with specified natural frequencies. *ALAA Journal, 5*(3), 406–412.
10. Khan, M. R., Thornton, W. A., & Willmert, K. D. (1975). Optimality criterion techniques applied to mechanical design. *Journal of Mechanical Design, 100*, 319–327.
11. Truman, K., & Peruska, D. (1991). Optimum design of dynamically excited structural systems using time history analysis. In S. Hernandez & C. A. Brebbia (Eds.), *Optimization of structural systems and industrial applications* (pp. 197–220). Boston: Computational Mechanics.
12. Paeng, J. K., & Arora, J. S. (1989). Dynamic response optimization of mechanical systems with multiplier methods. *Journal of Mechanics Transmissions, and Automation in Design, 111*, 73–80.
13. Chahande, A. I., & Arora, J. S. (1993). Development of a multiplier method for dynamic response optimization problems. *Structural Optimization, 6*, 69–78.
14. Johnson, E. H., Moore, G. J., & Izadpanah, K. (1992). Example of dynamic response optimization using MSC/NASTRAN. In *Fourth ALAA/USAF/NASA/OAI symposium on multi-disciplinary analysis and optimization number AIAA-92-4814-CP, Cleveland, OH, September 21-23* (pp. 959–967).
15. Brach, R. M. (1968). Optimum design of beams for sudden loading. *Journal of Engineering Mechanics, 94*(EM6), 1395–1407.
16. Abolbashari, M. H. (1995). *A numerical approach for optimization of velocity response of structure*, Ph.D. Dissertation, University of Saskachcnan.
17. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing* (2nd ed.). New York: Cambridge University Press.
18. Michalewcz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Berlin: Springer.
19. Tong, S. S. (1986). Coupling artificial intelligence and numerical computation for engineering design. *ALAA 24th Aerospace Science Meeting*, number ALAA—86-0242, Reno, Nevada.
20. Powell, D. J., Skolnick, M. M., & Tong, S. S. (1991). "Inter digitation: A hybrid technique for engineering design optimization employing genetic algorithms", expert systems, and numerical optimization. In L. Dais (Ed.), *Handbook of genetic algorithms, Chapter 20* (pp. 312–331). NewYork: Van Nostrand Reinhold.
21. Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley.
22. Joines, J., & Houck, C. (1994). On the use of non-stationary penalty function to solve non-linear constrained optimization problem with Gas. In *Proceeding of the first IEEE conference on evolution computation* (pp. 579–584). Orlando, FL: IEEE.
23. Herrera, F., Lozano, M., & Verdegay, J. L. *Fuzzy Connective Base Crossover operators to model genetic algorithms population diversity. Technical Report DECSAI-95110*. University of Granada, Department of Computer Science and Artificial Intelligence University of Granada, Granada, Spain.
24. Janillow, C., & Michalewics, Z. (1991). An experimental comparison of binary and floating point representation in genetic algorithm. In *Proceeding of the 4th International Conference on Genetic Algorithms*, (pp. 31–36). San Mateo, CA: Morgan Kaufmann Publishers.

Index

R. N. Jazar, L. Dai (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-030-18963-1