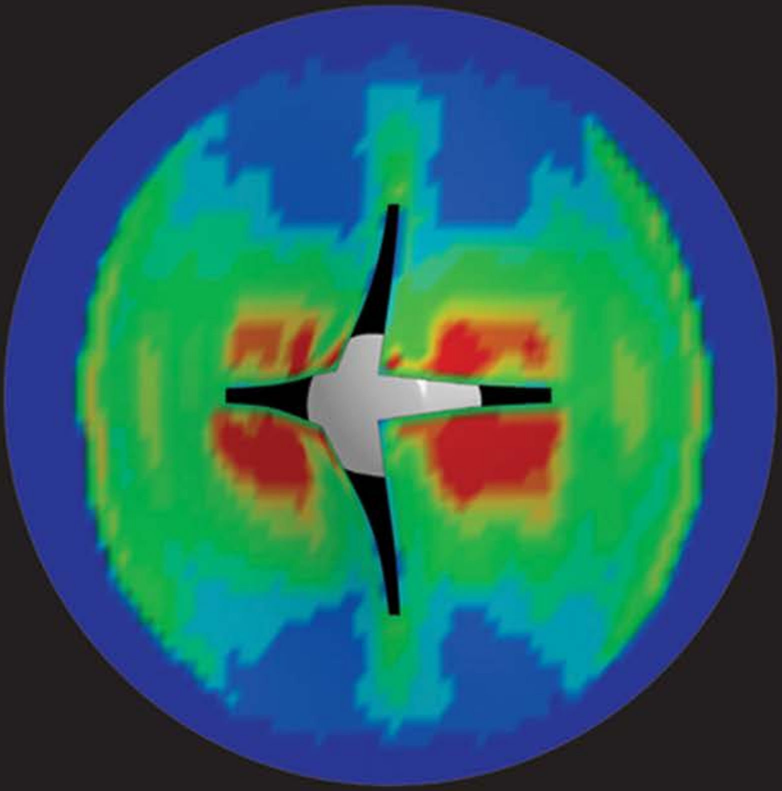
NUMERICAL METHODS FOR ENGINEERS AND SCIENTISTS

3rd Edition

An Introduction with Applications Using MATLAB®



AMOS GILAT | VISH SUBRAMANIAM

Numerical Methods for Engineers and Scientists

An Introduction with Applications using MATLAB®

Third Edition

Amos Gilat
Vish Subramaniam
Department of Mechanical Engineering
The Ohio State University

WILEY

Publisher:	Don Fowley
Acquisition Editor:	Linda Ratts
Editorial Assistant:	Hope Ellis
Cover Designer:	Wendy Lai
Associate Production Manager:	Joyce Poh

Cover Image: The image on the cover shows a numerical simulation of an aluminum projectile penetrating a composite plate. Courtesy of Dr. Kelly Carney, NASA Glenn Research Center, Cleveland, Ohio

This book was set in Times Roman by the authors. Printed and bound by R.R. Donnelley. The cover was printed by R.R. Donnelley. This book is printed on acid free paper.

Library of Congress Cataloging-in-Publication Data

Gilat, Amos.
Numerical methods for engineers and scientists : an introduction with applications using matlab / Amos Gilat, Vish Subramaniam, Department of Mechanical Engineering, the Ohio State University. -- Third edition.
pages cm
Includes bibliographical references and index.
ISBN 978-1-118-55493-7 (cloth)
1. MATLAB. 2. Numerical analysis--Data processing. 3. Engineering mathematics. I. Subramaniam, Vish. II. Title.
QA297.G49 2014
518.0285'53--dc23

2013016217

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Preface

This textbook is intended for a first course in numerical methods for students in engineering and science, typically taught in the second year of college. The book covers the fundamentals of numerical methods from an applied point of view. It explains the basic ideas behind the various methods and shows their usefulness for solving problems in engineering and science.

In the past, a numerical methods course was essentially mathematical, emphasizing numerical analysis and theory. More recently, due to the availability of powerful desktop computers and computing software that is both affordable and powerful, the content and nature of a first course in numerical methods for engineering and science students are changing. The emphasis is shifting more and more toward applications and toward implementing numerical methods with ready-to-use tools. In a typical course, students still learn the fundamentals of numerical methods. In addition, however, they learn computer programming (or improve their programming skills if they have already been introduced to programming), and use advanced software as a tool for solving problems. MATLAB is a good example of such software. It can be used by students to write their own programs, and can be used as a tool for solving problems using its built-in functions. One of the objectives of a course in numerical methods is to prepare students in science and engineering for future courses in their areas of specialization (and their future careers) where they will have to use computers for solving problems.

Main objectives of the book

To teach the fundamentals of numerical methods, with emphasis on the most essential methods.

To provide students with the opportunity to enhance their programming skills using the MATLAB environment to implement algorithms.

To teach the use of MATLAB as a tool (using its built-in functions) for solving problems in science and engineering, and for checking the results of any programs students write themselves.

Features/pedagogy of the book

- This book is written in simple, clear, and direct language. Frequently, bullets and a list of steps, rather than lengthy text, are used to list facts and details of a specific subject.
- Numerous illustrations are used for explaining the principles of the numerical methods.
- Many of the examples and end-of-chapter problems involve realistic problems in science and engineering.
- MATLAB is integrated within the text and in the examples. A light colored background is used when MATLAB syntax is displayed.
- Annotating comments that explain the commands are posted alongside the MATLAB syntax.
- MATLAB's built-in functions that are associated with the numerical methods are presented in detail.
- The homework problems at the end of the chapters are divided into three groups:

(*a*) ***Problems to be solved by hand:*** Problems related to improving understanding of numerical methods. In these problems the students are asked to answer questions related to the fundamentals of numerical methods, and to carry out a few steps of the numerical methods by hand.

(*b*) ***Problems to be programmed in MATLAB:*** Problems designed to provide the opportunity to improve programming skills. In these problems students are asked to use MATLAB to write computer programs (script files and user-defined functions) implementing various numerical methods.

(*c*) ***Problems in math, science, and engineering:*** Problems in science and engineering that have to be solved by using numerical methods. The objective is to train the students to use numerical methods for solving problems they can expect to see in future courses or in practice. Students are expected to use the programs that are presented in the book, programs that they write, and the built-in functions in MATLAB.

Organization of the book

Chapter 1: The first chapter gives a general introduction to numerical methods and to the way that computers store numbers and carry out numerical operations. It also includes a section on errors in numerical solutions and a section on computers and programming.

Chapter 2: The second chapter presents a review of fundamental mathematical concepts that are used in the following chapters that cover the numerical methods. It is intended to be used as a reminder, or a refresher, of concepts that the students are assumed (expected) to be familiar with from their first- and second-year mathematics courses. Since many of these topics are associated with various numerical methods, we feel that it is better to have the mathematical background gathered in one chapter (and easier to find when needed) rather than be dispersed throughout the book. Several of the topics that are covered in Chapter 2 and that are essential in the explanation of a numerical method are repeated in other chapters where the numerical methods are presented. Most instructors will probably choose not to cover Chapter 2 as one unit in the class, but will mention a topic when needed and refer the students to the chapter.

Chapters 3 through 11: These nine chapters present the various numerical methods in an order that is typically followed in a first course on numerical methods. These chapters follow the format explained next.

Organization of a typical chapter

An itemized list of the topics that are covered in the chapter is displayed below the title of the chapter. The list is divided into ***core*** and ***complementary*** topics. The ***core topics*** are the most essential topics related to the subject of the chapter. The ***complementary topics*** include more advanced topics. Obviously, a division of topics related to one subject into core and complementary is subjective. The intent is to help instructors in the design of their course when there is not enough time to cover all the topics. In practicality, the division can be ignored in courses where all the topics are covered.

The first section of the chapter provides a general background with illustrative examples of situations in the sciences and engineering where the methods described in the chapter are used. This section also explains the basic ideas behind the specific class of numerical methods that are described in the chapter. The following sections cover the core topics of the chapter. Next, a special section discusses the built-in functions in MATLAB that implement the numerical methods described in the chapter, and how they may be used to solve problems. The later sections of the chapter cover the complementary topics.

The order of topics

It is probably impossible to write a text book where all the topics follow an order that is agreed upon by all instructors. In the present book, the main subjects are in an order that is typical in a first course in numerical methods. Chapter 3 covers solution of nonlinear equations. It mostly deals with the solution of a single equation, which is a simple application of numerical methods. The chapter also includes, as a complementary topic, a section on the solution of a system of nonlinear equations. Chapter 4 deals with the solution of a system of linear equations. Next, Chapter 5 deals with eigenvalues and eigenvectors, and Chapter 6 covers curve fitting and interpolation. Chapter 7, which is new in the 3rd edition, covers an introduction to Fourier methods. Chapters 8 and 9 cover differentiation and integration, respectively. Finally, solution of ordinary differential equations (ODE) is presented in the last two chapters. Chapter 10 deals with the solution of initial-value problems (first-order, systems, and higher-order) and Chapter 11 considers boundary-value problems.

The order of some of the topics is dictated by the subjects themselves. For example, differentiation and integration need to be covered before ordinary differential equations. It is possible, however, to cover the other subjects in different order than presented in the book. The various chapters and sections in the book are written in a self-contained manner that make it easy for the instructor to cover the subjects in a different order, if desired.

MATLAB programs

This book contains many MATLAB programs. The programs are clearly identified as user-defined functions, or as script files. All the programs are listed in Appendix B. The programs, or the scripts, are written in a simple way that is easy to follow. The emphasis of these programs is on the basics and on how to program an algorithm of a specific numerical method. Obviously, the programs are not general, and do not cover all possible circumstances when executed. The programs are not written from the perspective of being shortest, fastest, or most efficient. Rather, they are written such that they are easy to follow. It is assumed that most of the students have only limited understanding of MATLAB and programming, and presenting MATLAB in this manner will advance their computing skills. More advanced users of MATLAB are encouraged to write more sophisticated and efficient programs and scripts, and compare their performance with the ones in the book.

Third edition

The main changes in the third edition are:

Fourier Methods: In response to many requests from professors that use the book in their courses, a new chapter (Chapter 7) on Fourier methods has been added to the book. The chapter covers Fourier series, discrete Fourier series, Discrete Fourier Transform, and an introduction to the Fast Fourier Transform (FFT) which are widely used in engineering for processing digital data.

Eignvalues and Eignvectors: This topic which was part of Chapter 4 (Solving a System of Linear Equations) in the first two editions of the book is now covered in a separate chapter.

MATLAB: The third edition of the book is updated to MATLAB R2012b. All the programs use anonymous functions and function handles are used for passing functions into functions. Appendix A has been updated to the current version of MATLAB.

Homework problems: About 50% of the problems have been added or changed. Most of the Chapters have 40 or more problems.

Support material

The following is available on the instructor companion site at www.wiley.com/college/gilat):

(*a*) for faculty who have adopted the text for use in their course, a fully worked solution manual, triple checked for accuracy.
(*b*) suggested course syllabi with suggested assignments to help quickly integrate the text into your course.
(*c*) conversion guides from other major numerical methods titles to show where each section of your current text is covered in this new text, helping you quickly convert from old to new.
(*d*) electronic versions of all the figures and tables from the text, for creating lecture slides and quizzes/exams based on images from the book.
(*e*) m-files of all the programs in the text.

Many people have assisted during the preparation of the first two editions of the book. We would like to thank the reviewers and users for the many comments and suggestions they have made.

Lawrence K. Agbezuge, *Rochester Institute of Technology*
David Alciatore, *Colorado State University*
Salame Amr, *Virginia State University*
John R. Cotton, *Virginia Polytechnic Institute and State University*
David Dux, *Purdue University*
Venkat Ganesan, *University of Texas-Austin*
Michael R. Gustafson II, *Duke University*
Alain Kassab, *University of Central Florida*
Tribikram Kundu, *University of Arizona*
Ronald A. Mann, *University of Louisville*
Peter O. Orono, *Indiana University Purdue University Indianapolis*
Charles Ritz, *California State Polytechnic University-Pomona*
Douglas E. Smith, *University of Missouri-Columbia*
Anatoliy Swishchuk, *University of Calgary*
Ronald F. Taylor, *Wright State University*
Brian Vick, *Virginia Polytechnic Institute and State University*
John Silzel, *Biola University*
James Guilkey, *University of Utah*

We would also like to thank Linda Ratts, acquisition editor, and Renata Marchione, editorial assistant, from Wiley. Special thanks to Professor Subramaniam's daughters, Sonya and Priya, for typing early drafts of some chapters and for proofreading them.

Our intention was to write a book that is useful to students and instructors alike. We would like to thank users of previous editions of the book who have sent us compliments and suggestions. We would appreciate any comments that will help to improve future editions.

Amos Gilat (gilat.1@osu.edu)
Vish Subramaniam (subramaniam.1@osu.edu)
Columbus, Ohio
June 2013

To Yaela, Taly, and Edan

To my parents, Dr. K. S. Venkateswaran & Seethalakshmy Venkateswaran,
and Deepa, Priya, and Sonya

Brief Table of Contents

Preface iii

Chapter 1 *Introduction* 1

Chapter 2 *Mathematical Background* 23

Chapter 3 *Solving Nonlinear Equations* 57

Chapter 4 *Solving a System of Linear Equations* 99

Chapter 5 *Eigenvalues and Eigenvectors* 165

Chapter 6 *Curve Fitting and Interpolation* 193

Chapter 7 *Fourier Methods* 251

Chapter 8 *Numerical Differentiation* 303

Chapter 9 *Numerical Integration* 341

Chapter 10 *Ordinary Differential Equations: Initial–Value Problems* 385

Chapter 11 *Ordinary Differential Equations: Boundary–Value Problems* 471

Appendix A *Introductory MATLAB* 509

Appendix B *MATLAB Programs* 547

Appendix C *Derivation of the Real Discrete Fourier Transform (DFT)* 551

Index 555

Contents

Preface iii

Chapter 1 *Introduction* 1

1.1 Background 1
1.2 Representation of Numbers on a Computer 4
1.3 Errors in Numerical Solutions 10
1.3.1 Round-Off Errors 10
1.3.2 Truncation Errors 13
1.3.3 Total Error 14
1.4 Computers and Programming 15
1.5 Problems 18

Chapter 2 *Mathematical Background* 23

2.1 Background 23
2.2 Concepts from Pre-Calculus and Calculus 24
2.3 Vectors 28
2.3.1 Operations with Vectors 30
2.4 Matrices and Linear Algebra 32
2.4.1 Operations with Matrices 33
2.4.2 Special Matrices 35
2.4.3 Inverse of a Matrix 36
2.4.4 Properties of Matrices 37
2.4.5 Determinant of a Matrix 37
2.4.6 Cramer's Rule and Solution of a System of Simultaneous Linear Equations 38
2.4.7 Norms 40
2.5 Ordinary Differential Equations (ODE) 41
2.6 Functions of Two or More Independent Variables 44
2.6.1 Definition of the Partial Derivative 44
2.6.2 Chain Rule 45
2.6.3 The Jacobian 46
2.7 Taylor Series Expansion of Functions 47
2.7.1 Taylor Series for a Function of One Variable 47
2.7.2 Taylor Series for a Function of Two Variables 49
2.8 Inner Product and Orthogonality 50
2.9 Problems 51

Chapter 3 *Solving Nonlinear Equations* 57

3.1 Background 57
3.2 Estimation of Errors in Numerical Solutions 59
3.3 Bisection Method 61
3.4 Regula Falsi Method 64
3.5 Newton's Method 66
3.6 Secant Method 71
3.7 Fixed-Point Iteration Method 74
3.8 Use of MATLAB Built-In Functions for Solving Nonlinear Equations 77
3.8.1 The `fzero` *Command* 78
3.8.2 The `roots` *Command* 79
3.9 Equations with Multiple Solutions 79
3.10 Systems of Nonlinear Equations 81
3.10.1 Newton's Method for Solving a System of Nonlinear Equations 82
3.10.2 Fixed-Point Iteration Method for Solving a System of Nonlinear Equations 86
3.11 Problems 88

Chapter 4 *Solving a System of Linear Equations* 99

4.1 Background 99
4.1.1 Overview of Numerical Methods for Solving a System of Linear Algebraic Equations 100
4.2 Gauss Elimination Method 102
4.2.1 Potential Difficulties When Applying the Gauss Elimination Method 110
4.3 Gauss Elimination with Pivoting 112
4.4 Gauss–Jordan Elimination Method 115
4.5 LU Decomposition Method 118
4.5.1 LU Decomposition Using the Gauss Elimination Procedure 120
4.5.2 LU Decomposition Using Crout's Method 121
4.5.3 LU Decomposition with Pivoting 128
4.6 Inverse of a Matrix 128
4.6.1 Calculating the Inverse with the LU Decomposition Method 129
4.6.2 Calculating the Inverse Using the Gauss–Jordan Method 131
4.7 Iterative Methods 132
4.7.1 Jacobi Iterative Method 133
4.7.2 Gauss-Seidel Iterative Method 133
4.8 Use of MATLAB Built-In Functions for Solving a System of Linear Equations 136
4.8.1 Solving a System of Equations Using MATLAB's Left and Right Division 136
4.8.2 Solving a System of Equations Using MATLAB's Inverse Operation 137
4.8.3 MATLAB's Built-In Function for LU Decomposition 138
4.8.4 Additional MATLAB Built-In Functions 139
4.9 Tridiagonal Systems of Equations 141

4.10 Error, Residual, Norms, and Condition Number 146
4.10.1 Error and Residual 146
4.10.2 Norms and Condition Number 148
4.11 Ill-Conditioned Systems 153
4.12 Problems 155

Chapter 5 *Eigenvalues and Eigenvectors* 165

5.1 Background 165
5.2 The Characteristic Equation 167
5.3 The Basic Power Method 167
5.4 The Inverse Power Method 172
5.5 The Shifted Power Method 173
5.6 The QR Factorization and Iteration Method 174
5.7 Use of MATLAB Built-In Functions for Determining Eigenvalues and Eigenvectors 184
5.8 Problems 186

Chapter 6 *Curve Fitting and Interpolation* 193

6.1 Background 193
6.2 Curve Fitting with a Linear Equation 195
6.2.1 Measuring How Good Is a Fit 195
6.2.2 Linear Least-Squares Regression 197
6.3 Curve Fitting with Nonlinear Equation by Writing the Equation in a Linear Form 201
6.4 Curve Fitting with Quadratic and Higher-Order Polynomials 205
6.5 Interpolation Using a Single Polynomial 210
6.5.1 Lagrange Interpolating Polynomials 212
6.5.2 Newton's Interpolating Polynomials 216
6.6 Piecewise (Spline) Interpolation 223
6.6.1 Linear Splines 223
6.6.2 Quadratic Splines 225
6.6.3 Cubic Splines 229
6.7 Use of MATLAB Built-In Functions for Curve Fitting and Interpolation 236
6.8 Curve Fitting with a Linear Combination of Nonlinear Functions 238
6.9 Problems 241

Chapter 7 *Fourier Methods* 251

7.1 Background 251
7.2 Approximating a Square Wave by a Series of Sine Functions 253
7.3 General (Infinite) Fourier Series 257
7.4 Complex Form of the Fourier Series 261

7.5 The Discrete Fourier Series and Discrete Fourier Transform **263**
7.6 Complex Discrete Fourier Transform **268**
7.7 Power (Energy) Spectrum **271**
7.8 Aliasing and Nyquist Frequency **272**
7.9 Alternative Forms of the Discrete Fourier Transform **278**
7.10 Use of MATLAB Built-In Functions for Calculating Discrete Fourier Transform **278**
7.11 Leakage and Windowing **284**
7.12 Bandwidth and Filters **286**
7.13 The Fast Fourier Transform (FFT) **289**
7.14 Problems **298**

Chapter 8 ***Numerical Differentiation*** **303**

8.1 Background **303**
8.2 Finite Difference Approximation of the Derivative **305**
8.3 Finite Difference Formulas Using Taylor Series Expansion **310**
8.3.1 Finite Difference Formulas of First Derivative **310**
8.3.2 Finite Difference Formulas for the Second Derivative **315**
8.4 Summary of Finite Difference Formulas for Numerical Differentiation **317**
8.5 Differentiation Formulas Using Lagrange Polynomials **319**
8.6 Differentiation Using Curve Fitting **320**
8.7 Use of MATLAB Built-In Functions for Numerical Differentiation **320**
8.8 Richardson's Extrapolation **322**
8.9 Error in Numerical Differentiation **325**
8.10 Numerical Partial Differentiation **327**
8.11 Problems **330**

Chapter 9 ***Numerical Integration*** **341**

9.1 Background **341**
9.1.1 Overview of Approaches in Numerical Integration **342**
9.2 Rectangle and Midpoint Methods **344**
9.3 Trapezoidal Method **346**
9.3.1 Composite Trapezoidal Method **347**
9.4 Simpson's Methods **350**
9.4.1 Simpson's 1/3 Method **350**
9.4.2 Simpson's 3/8 Method **353**
9.5 Gauss Quadrature **355**
9.6 Evaluation of Multiple Integrals **360**
9.7 Use of MATLAB Built-In Functions for Integration **362**
9.8 Estimation of Error in Numerical Integration **364**
9.9 Richardson's Extrapolation **366**

9.10 Romberg Integration 369
9.11 Improper Integrals 372
9.11.1 Integrals with Singularities 372
9.11.2 Integrals with Unbounded Limits 373
9.12 Problems 374

Chapter 10 *Ordinary Differential Equations: Initial-Value Problems* 385

10.1 Background 385
10.2 Euler's Methods 390
10.2.1 Euler's Explicit Method 390
10.2.2 Analysis of Truncation Error in Euler's Explicit Method 394
10.2.3 Euler's Implicit Method 398
10.3 Modified Euler's Method 401
10.4 Midpoint Method 404
10.5 Runge–Kutta Methods 405
10.5.1 Second-Order Runge–Kutta Methods 406
10.5.2 Third-Order Runge–Kutta Methods 410
10.5.3 Fourth-Order Runge–Kutta Methods 411
10.6 Multistep Methods 417
10.6.1 Adams–Bashforth Method 418
10.6.2 Adams–Moulton Method 419
10.7 Predictor–Corrector Methods 420
10.8 System of First-Order Ordinary Differential Equations 422
10.8.1 Solving a System of First-Order ODEs Using Euler's Explicit Method 424
10.8.2 Solving a System of First-Order ODEs Using Second-Order Runge–Kutta Method (Modified Euler Version) 424
10.8.3 Solving a System of First-Order ODEs Using the Classical Fourth-Order Runge–Kutta Method 431
10.9 Solving a Higher-Order Initial Value Problem 432
10.10 Use of MATLAB Built-In Functions for Solving Initial-Value Problems 437
10.10.1 Solving a Single First-Order ODE Using MATLAB 438
10.10.2 Solving a System of First-Order ODEs Using MATLAB 444
10.11 Local Truncation Error in Second-Order Range–Kutta Method 447
10.12 Step Size for Desired Accuracy 448
10.13 Stability 452
10.14 Stiff Ordinary Differential Equations 454
10.15 Problems 457

Chapter 11 ***Ordinary Differential Equations: Boundary-Value Problems*** **471**

11.1 Background **471**
11.2 The Shooting Method **474**
11.3 Finite Difference Method **482**
11.4 Use of MATLAB Built-In Functions for Solving Boundary Value Problems **492**
11.5 Error and Stability in Numerical Solution of Boundary Value Problems **497**
11.6 Problems **499**

Appendix A ***Introductory MATLAB*** **509**

A.1 Background **509**
A.2 Starting with MATLAB **509**
A.3 Arrays **514**
A.4 Mathematical Operations with Arrays **519**
A.5 Script Files **524**
A.6 Plotting **526**
A.7 User-Defined Functions and Function Files **528**
A.8 Anonymous Functions **530**
A.9 Function functions **532**
A.10 Subfunctions **535**
A.11 Programming in MATLAB **537**
A.11.1 Relational and Logical Operators **537**
A.11.2 Conditional Statements, if-else Structures **538**
A.11.3 Loops **541**
A.12 Problems **542**

Appendix B ***MATLAB Programs*** **547**

Appendix C ***Derivation of the Real Discrete Fourier Transform (DFT)*** **551**

C.1 Orthogonality of Sines and Cosines for Discrete Points **551**
C.2 Determination of the Real DFT **553**

Index **555**

Chapter 1

Introduction

Core Topics

Representation of numbers on a computer (1.2).

Errors in numerical solutions, round-off errors and truncation errors (1.3).

Computers and programming (1.4).

1.1 BACKGROUND

Numerical methods are mathematical techniques used for solving mathematical problems that cannot be solved or are difficult to solve analytically. An analytical solution is an exact answer in the form of a mathematical expression in terms of the variables associated with the problem that is being solved. A numerical solution is an approximate numerical value (a number) for the solution. Although numerical solutions are an approximation, they can be very accurate. In many numerical methods, the calculations are executed in an iterative manner until a desired accuracy is achieved.

For example, Fig. 1-1 shows a block of mass m being pulled by a force F applied at an angle θ. By applying equations of equilibrium, the relationship between the force and the angle is given by:

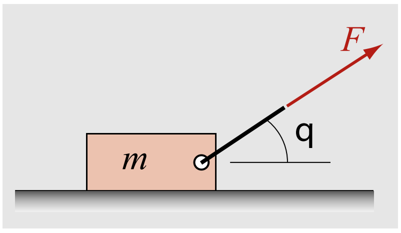


Figure 1-1: Motion of a block on a surface with friction.

$$F = \frac{\mu m g}{\cos\theta + \mu\sin\theta} \tag{1.1}$$

where μ is the friction coefficient and g is the acceleration due to gravity. For a given value of F, the angle that is required for moving the block can be determined by solving Eq. (1.1) for θ. Equation (1.1), however, cannot be solved analytically for θ. Using numerical methods, an approximate solution can be determined for specified accuracy. This means that when the numerical solution for θ is substituted back in Eq. (1.1), the value of F that is obtained from the expression on the right-hand side is not exactly equal to the given value of F, but is very close.

Numerical techniques for solving mathematical problems were developed and used hundreds and even thousands of years ago. Implementation of the numerical techniques was difficult since the calculations had to be carried out by hand or by use of simple mechanical computing devices, which limited the number of calculations that could be carried out, as well as their speed and accuracy. Today numerical methods are used with fast electronic digital computers that make it possible to execute many tedious and repetitive calculations that produce accurate (even though not exact) solutions in a very short time.

Solving a problem in science and engineering

The process of solving a problem in science and engineering is influenced by the tools (mathematical methods) that are available for solving the problem. The process can be divided into the following steps:

Problem statement

The problem statement defines the problem. It gives a description of the problem, lists the variables that are involved, and identifies the constraints in the form of boundary and/or the initial conditions.

Formulation of the solution

Formulation of the solution consists of the model (physical law or laws) that is used to represent the problem and the derivation of the governing equations that need to be solved. Examples of such laws are Newton's laws, conservations of mass, and the laws of thermodynamics. The models that are used (chosen) to solve the problem need to be consistent with the methods that are subsequently used for solving the equations. If analytical methods are expected to be used for the solution, the governing equations must be of a type that can be solved analytically. If needed, the formulation has to be simplified, such that the equations could be solved analytically. If numerical methods are used for the solution, the models and the equations can be more complicated. Even then, however, some limitations might exist. For example, if the formulation is such that a numerical solution requires a long computing time, the formulation might have to be simplified such that a solution is obtained in a reasonable time. An example is weather forecasting. The problem that is solved is large, and the numerical models that are used are very complicated. The numerical simulation of the weather, however, cannot outlast the period over which forecasting is needed.

Programming (of numerical solution)

If the problem is solved numerically, the numerical method that is used for the solution has to be selected. For every type of mathematical problem there are several (or many) numerical techniques that can be used. The techniques differ in accuracy, length of calculations, and difficulty in programming. Once a numerical method is selected, it is implemented in a computer program. The implementation consists of an ***algorithm***, which is a detailed plan that describes how to carry out the numerical method, and a computer program, which is a list of commands that allows the computer to execute the algorithm to find the solution.

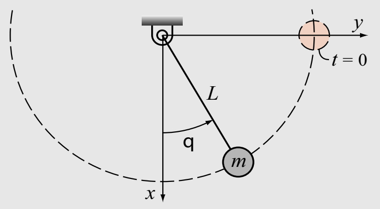
Interpretation of the solution

Since numerical solutions are an approximation (errors are addressed in Section 1.4), and since the computer program that executes the numerical method might have errors (or bugs), a numerical solution needs to be examined closely. This can be done in several ways, depending on the problem. For example, if the numerical method is used for solving a nonlinear algebraic equation, the validity of the solution can be verified by substituting the solution back in the equation. In more complicated problems, like a solution of a differential equation, the numerical solution can be compared with a known solution of a similar problem, or the problem can be solved several times using different boundary (or initial) conditions, and different numerical methods, and examining the subsequent differences in the solutions.

An illustration of the first two steps in the solution process of a problem is shown in Example 1-1.

Example 1-1: Problem formulation

Consider the following problem statement:
A pendulum of mass m is attached to a rigid rod of length L, as shown in the figure. The pendulum is displaced from the vertical position such that the angle between the rod and the x axis is θ_0, and then the pendulum is released from rest. Formulate the problem for determining the angle θ as a function of time, t, once the pendulum is released. In the formulation include a damping force that is proportional to the velocity of the pendulum.
Formulate the solution for two cases:
(a) $\theta_0 = 5°$, and (b) $\theta_0 = 90°$.

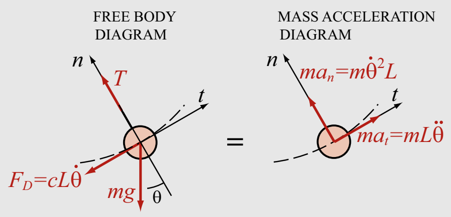


SOLUTION

Physical law

The physical law that is used for solving the problem is Newton's second law of mechanics, according to which, as the pendulum swings back and forth, the sum of the forces that are acting on the mass is equal to the mass times its acceleration.

$$\Sigma\bar{F} = m\bar{a} \tag{1.2}$$



Newton's Second Law

This can be visualized by drawing a free body diagram and a mass acceleration diagram, which are shown on the right. The constant c is the damping coefficient. It should be pointed out that the mass of the rod is neglected in the present solution.

Governing equation

The governing equation is derived by applying Newton's second law in the tangential direction:

$$\Sigma F_t = -cL\frac{d\theta}{dt} - mg\sin\theta = mL\frac{d^2\theta}{dt^2} \tag{1.3}$$

Equation (1.3), which is a second-order, nonlinear, ordinary differential equation, can be written in the form:

$$mL\frac{d^2\theta}{dt^2} + cL\frac{d\theta}{dt} + mg\sin\theta = 0 \tag{1.4}$$

The initial conditions are that when the motion of the pendulum starts ($t = 0$), the pendulum is at angle θ_0 and its velocity is zero (released from rest):

$$\theta(0) = \theta_0 \quad \text{and} \quad \left.\frac{d\theta}{dt}\right|_{t=0} = 0 \tag{1.5}$$

Method of solution

Equation (1.4) is a nonlinear equation and cannot be solved analytically. However, in part (*a*) the initial displacement of the pendulum is $\theta_0 = 5°$, and once the pendulum is released, the angle as the pendulum oscillates will be less than $5°$. For this case, Eq. (1.4) can be linearized by assuming that $\sin\theta \approx \theta$. With this approximation, the equation that has to be solved is linear and can be solved analytically:

$$mL\frac{d^2\theta}{dt^2} + cL\frac{d\theta}{dt} + mg\theta = 0 \tag{1.6}$$

with the initial conditions Eq. (1.5).

In part (*b*), the initial displacement of the pendulum is $\theta_0 = 90°$ and the equation has to be solved numerically. An actual numerical solution for this problem is shown in Example 8-8.

1.2 REPRESENTATION OF NUMBERS ON A COMPUTER

Decimal and binary representation

Numbers can be represented in various forms. The familiar decimal system (base 10) uses ten digits 0, 1, ..., 9. A number is written by a sequence of digits that correspond to multiples of powers of 10. As shown in Fig. 1-2, the first digit to the left of the decimal point corre-

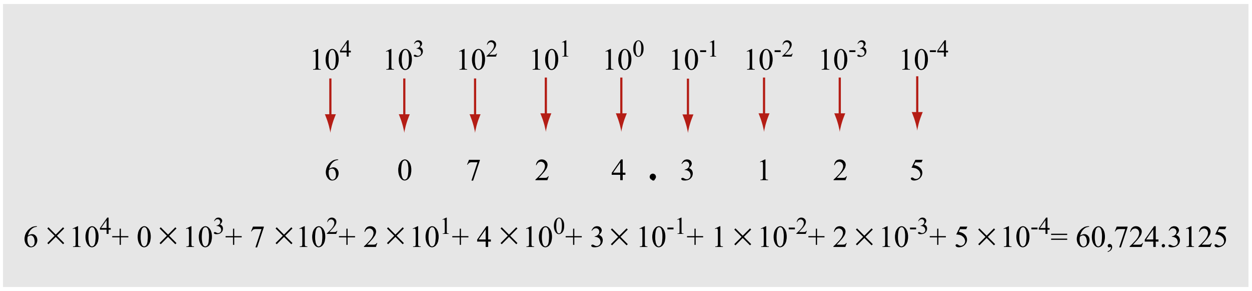

10^4 10^3 10^2 10^1 10^0 10^{-1} 10^{-2} 10^{-3} 10^{-4}
6 0 7 2 4 . 3 1 2 5
$6\times10^4 + 0\times10^3 + 7\times10^2 + 2\times10^1 + 4\times10^0 + 3\times10^{-1} + 1\times10^{-2} + 2\times10^{-3} + 5\times10^{-4} = 60{,}724.3125$

Figure 1-2: Representation of the number 60,724.3125 in the decimal system (base 10).

Base 10	Base 2			
	2^3	2^2	2^1	2^0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

Figure 1-3: Representation of numbers in decimal and binary forms.

sponds to 10^0. The digit next to it on the left corresponds to 10^1, the next digit to the left to 10^2, and so on. In the same way, the first digit to the right of the decimal point corresponds to 10^{-1}, the next digit to the right to 10^{-2}, and so on.

In general, however, a number can be represented using other bases. A form that can be easily implemented in computers is the binary (base 2) system. In the binary system, a number is represented by using the two digits 0 and 1. A number is then written as a sequence of zeros and ones that correspond to multiples of powers of 2. The first digit to the left of the decimal point corresponds to 2^0. The digit next to it on the left corresponds to 2^1, the next digit to the left to 2^2, and so on. In the same way, the first digit to the right of the decimal point corresponds to 2^{-1}, the next digit to the right to 2^{-2}, and so on. The first ten digits $1, 2, 3, \ldots, 10$ in base 10 and their representation in base 2 are shown in Fig. 1-3. The representation of the number 19.625 in the binary system is shown in Fig. 1-4.

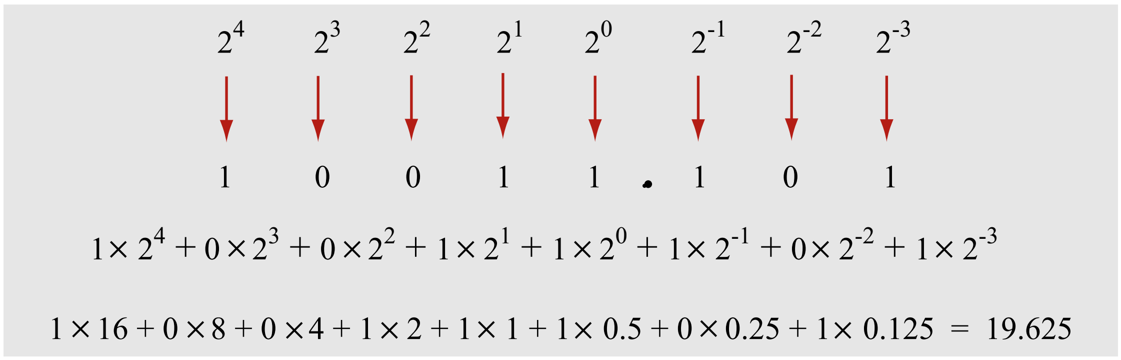

2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}
1 0 0 1 1 . 1 0 1
$1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$
$1\times 16 + 0\times 8 + 0\times 4 + 1\times 2 + 1\times 1 + 1\times 0.5 + 0\times 0.25 + 1\times 0.125 = 19.625$

Figure 1-4: Representation of the number 19.625 in the binary system (base 2).

Another example is shown in Fig. 1-5, where the number 60,724.3125 is written in binary form.

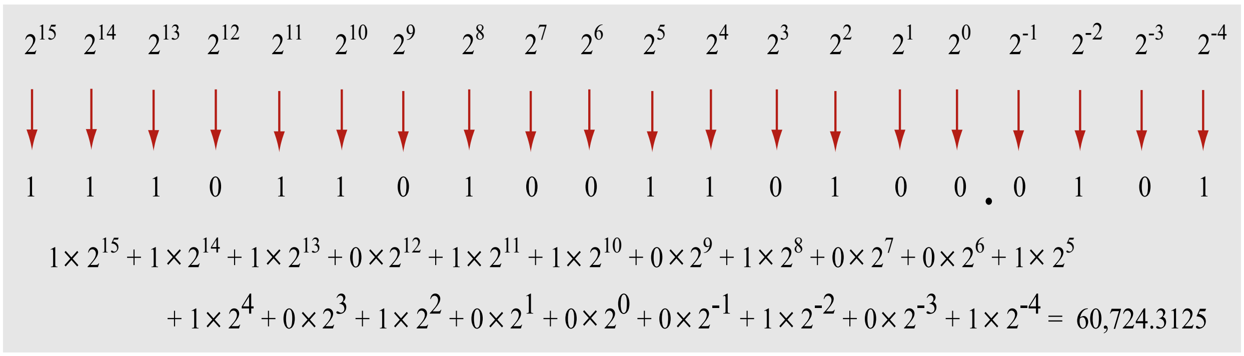

2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4}
1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 . 0 1 0 1
$1\times 2^{15} + 1\times 2^{14} + 1\times 2^{13} + 0\times 2^{12} + 1\times 2^{11} + 1\times 2^{10} + 0\times 2^9 + 1\times 2^8 + 0\times 2^7 + 0\times 2^6 + 1\times 2^5$
$+ 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 0\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4} = 60{,}724.3125$

Figure 1-5: Representation of the number 60,724.3125 in the binary system (base 2).

Computers store and process numbers in binary (base 2) form. Each binary digit (one or zero) is called a ***bit*** (for binary digit). Binary arithmetic is used by computers because modern transistors can be used as extremely fast switches. Therefore, a network of these may be used to represent strings of numbers with the "1" referring to the switch being in the "on" position and "0" referring to the "off" position. Various operations are then performed on these sequences of ones and zeros.

Floating point representation

To accommodate large and small numbers, real numbers are written in floating point representation. Decimal floating point representation (also called scientific notation) has the form:

$$d.dddddd \times 10^{p} \tag{1.7}$$

One digit is written to the left of the decimal point, and the rest of the significant digits are written to the right of the decimal point. The number *0.dddddd* is called the ***mantissa***. Two examples are:

$$6519.23 \quad \text{written as} \quad 6.51923 \times 10^{3}$$

$$0.00000391 \quad \text{written as} \quad 3.91 \times 10^{-6}$$

The power of 10, p, represents the number's order of magnitude, provided the preceding number is smaller than 5. Otherwise, the number is said to be of the order of $p+1$. Thus, the number 3.91×10^{-6} is of the order of 10^{-6}, $O(10^{-6})$, and the number 6.51923×10^{3} is of the order of 10^{4} (written as $O(10^{4})$).

Binary floating point representation has the form:

$$1.bbbbbb \times 2^{bbb} \qquad (b \text{ is a decimal digit}) \tag{1.8}$$

In this form, the mantissa is $.bbbbbb$, and the power of 2 is called the ***exponent***. Both the mantissa and the exponent are written in a binary form. The form in Eq. (1.8) is obtained by normalizing the number (when it is written in the decimal form) with respect to the largest power of 2 that is smaller than the number itself. For example, to write the number 50 in binary floating point representation, the number is divided (and multiplied) by $2^{5} = 32$ (which is the largest power of 2 that is smaller than 50):

$$50 = \frac{50}{2^{5}} \times 2^{5} = 1.5625 \times 2^{5} \quad \text{Binary floating point form: } 1.1001 \times 2^{101}$$

Two more examples are:

$$1344 = \frac{1344}{2^{10}} \times 2^{10} = 1.3125 \times 2^{10} \quad \text{Binary floating point form: } 1.0101 \times 2^{1010}$$

$$0.3125 = \frac{0.3125}{2^{-2}} \times 2^{-2} = 1.25 \times 2^{-2} \quad \text{Binary floating point form: } 1.01 \times 2^{-10}$$

Storing a number in computer memory

Once in binary floating point representation, the number is stored in the computer. The computer stores the values of the exponent and the mantissa separately, while the leading 1 in front of the decimal point is not stored. As already mentioned, a bit is a binary digit. The memory in the computer is organized in ***bytes***, where each byte is 8 bits. According to the IEEE[1]-754 standard (1985), computers store numbers and carry out calculations in ***single precision***[2] or in ***double precision***.[3] In single precision, the numbers are stored in a string of 32 bits (4 bytes), and in double precision in a string of 64 bits (8 bytes). In both cases the first bit stores the sign (0 corresponds to + and 1 corresponds to –) of the number. The next 8 bits in single precision (11 bits in double precision) are used for storing the exponent. The following 23 bits in single precision (52 bits in double precision) are used for storing the mantissa. This is illustrated for double precision in Fig. 1-6.

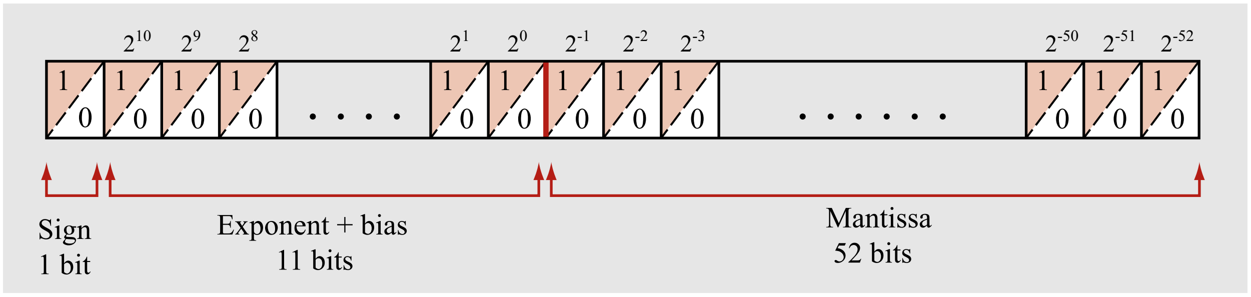


Figure 1-6: Storing in double precision a number written in binary floating point representation.

The value of the mantissa is entered as is in a binary form. The value of the exponent is entered with a bias. A bias means that a constant is added to the value of the exponent. The bias is introduced in order to avoid using one of the bits for the sign of the exponent (since the exponent can be positive or negative). In binary notation, the largest number that can be written with 11 bits is 2047 (when all 11 digits are 1). The bias that is used is 1023, which means that if, for example, the exponent is 4, then the value that is stored is $4 + 1023 = 1027$. Thus, the

1. IEEE stands for the Institute of Electrical and Electronics Engineers.
2. ***Precision*** refers to the number of significant digits of a real number that can be stored on a computer. For example, the number 1/3 = 0.333333... can be represented on a computer only in a chopped or rounded form with a finite number of binary digits, since the amount of memory where these bits are held is finite. The more digits to the right-hand side of the decimal point that are stored, the more ***precise*** is the representation of the real number on the computer.
3. This is somewhat of a misnomer. The precision in a double-precision number is not really doubled compared to a single-precision number. Rather, the "double" in double precision refers to the fact that twice as many binary digits (64 versus 32) are used to represent a real number than in the case of a single-precision representation.

smallest exponent that can be stored by the computer is -1023, and the largest is 1024 (which will be stored as 2047). However, the smallest and largest values of the exponent plus bias are reserved for zero and infinity (`Inf`) or not-a-number (`NaN`) due to invalid mathematical operation. The 11 bits for the exponent plus bias store values between -1023 and 1024. If the exponent plus bias and mantissa are both zero, then the number actually stored is 0. If the exponent plus bias is 2047 the number stored is `Inf` if the mantissa is zero, and it is `NaN` if the mantissa is not zero. In single precision, 8 bits are allocated to the value of the exponent and the bias is 127.

As an example, consider storing of the number 22.5 in double precision according to the IEEE-754 standard. First, the number is normalized: $\frac{22.5}{2^4}2^4 = 1.40625 \times 2^4$. In double precision, the exponent with the bias is $4 + 1023 = 1027$, which is stored in binary form as 10000000011. The mantissa is 0.40625, which is stored in binary form as .01101000....000. The storage of the number is illustrated in Fig. 1-7.

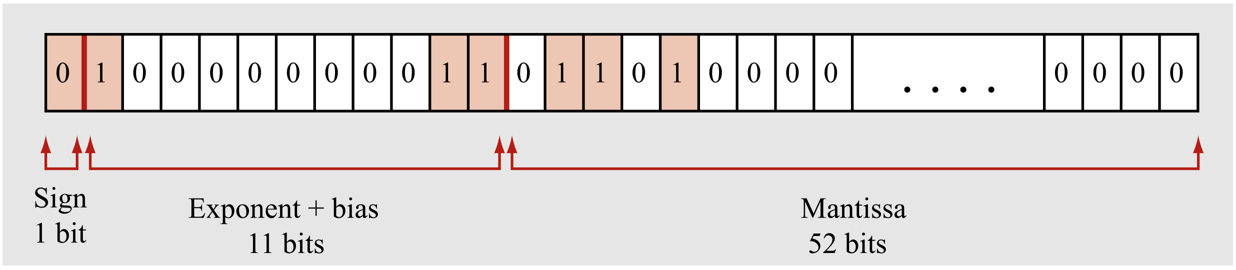


Figure 1-7: Storing the number 22.5 in double precision according to the IEEE-754 standard.

Additional notes

- The smallest positive number that can be expressed in double precision is:

$$2^{-1022} \approx 2.2 \times 10^{-308}$$

  This means that there is a (small) gap between zero and the smallest number that can be stored on the computer. Attempts to define a number in this gap causes an ***underflow*** error. (In the same way, the closest negative number to zero is -2.2×10^{-308}.)
- The largest positive number that can be expressed in double precision is approximately:

$$2^{1024} \approx 1.8 \times 10^{308}$$

  Attempts to define a larger number causes ***overflow*** error. (The same applies to numbers smaller than -2^{1024}.)

The range of numbers that can be represented in double precision is shown in Fig. 1-8.

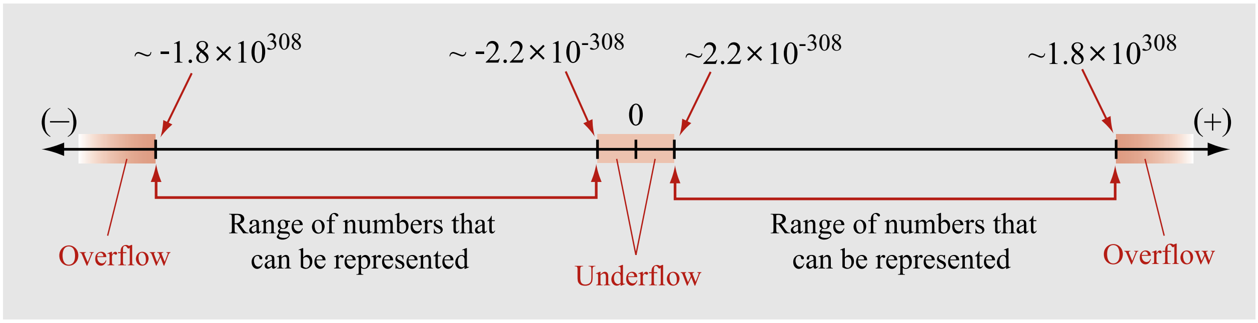


Figure 1-8: Range of numbers that can be represented in double precision.

- Since a finite number of bits is used, not every number can be accurately written in binary form. In other words, only a finite number of exact values in decimal format can be stored in binary form. For example, the number 0.1 cannot be represented exactly in finite binary format when single precision is used. To be written in binary floating point representation, 0.1 is normalized: $0.1 = 1.6 \times 2^{-4}$. The exponent -4 (with a bias) can be stored exactly, but the mantissa 0.6 cannot be written exactly in a binary format that uses 23 bits. In addition, irrational numbers cannot be represented exactly in any format. This means that, in many cases, exact values are approximated. The errors that are introduced are small in one step, but when many operations are executed, the errors can grow to such an extent that the final answer is affected. These errors, as well as other errors, are discussed in the next section.
- The interval between numbers that can be represented depends on their magnitude. In double precision, the smallest value of the mantissa that can be stored is $2^{-52} \approx 2.22 \times 10^{-16}$. This is also the smallest possible difference in the mantissa between two numbers. The magnitude of the real number that is associated with this mantissa, however, depends on the exponent. For numbers of the order of 1, the smallest difference between two numbers that can be represented in double precision is then 2.22×10^{-16}. This value is also defined as the ***machine epsilon*** in double precision. In MATLAB this value is assigned to the predefined variable `eps`. As shown below, when the name of the variable `eps` is typed (Command Window), the assigned value is displayed.

```
>> eps
ans =
    2.220446049250313e-016
```