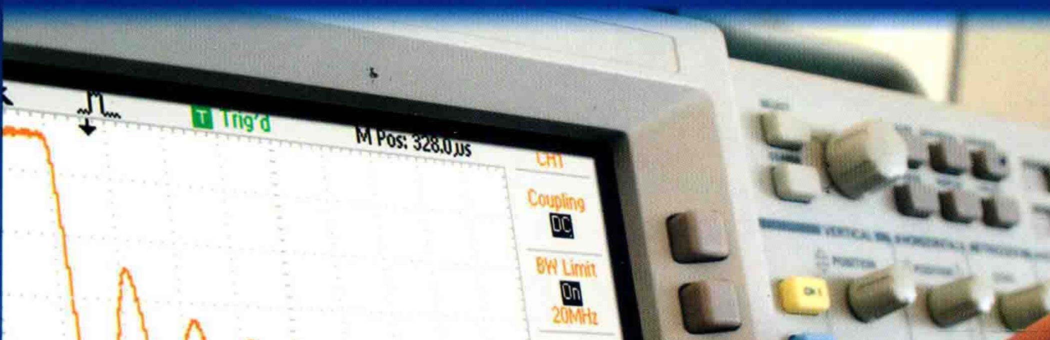
Numerical and Analytical Methods with MATLAB® for Electrical Engineers



William Bober • Andrew Stevens

CRC Press
Taylor & Francis Group

Numerical and Analytical Methods with MATLAB®
for Electrical Engineers

William Bober • Andrew Stevens

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
Version Date: 20120409

International Standard Book Number: 978-1-4398-5429-7 (Hardback)

Library of Congress Cataloging-in-Publication Data

Bober, William, 1930-
Numerical and analytical methods with MATLAB for electrical engineers / William Bober and Andrew Stevens.
p. cm. -- (Computational mechanics and applied analysis)
ISBN 978-1-4398-5429-7 (hardback)
1. Engineering mathematics--Data processing. 2. Numerical analysis--Data processing. 3. MATLAB. I. Stevens, Andrew, 1964- II. Title.

TA345.B63 2012
515.0285'53--dc23 2012005856

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Numerical and Analytical Methods with MATLAB® for Electrical Engineers

CRC Series in

COMPUTATIONAL MECHANICS and APPLIED ANALYSIS

Series Editor: J.N. Reddy
Texas A&M University

Published Titles

ADVANCED THERMODYNAMICS ENGINEERING, Second Edition
Kalyan Annamalai, Ishwar K. Puri, and Miland Jog

APPLIED FUNCTIONAL ANALYSIS
J. Tinsley Oden and Leszek F. Demkowicz

COMBUSTION SCIENCE AND ENGINEERING
Kalyan Annamalai and Ishwar K. Puri

CONTINUUM MECHANICS FOR ENGINEERS, Third Edition
Thomas Mase, Ronald Smelser, and George E. Mase

DYNAMICS IN ENGINEERING PRACTICE, Tenth Edition
Dara W. Childs

EXACT SOLUTIONS FOR BUCKLING OF STRUCTURAL MEMBERS
C.M. Wang, C.Y. Wang, and J.N. Reddy

THE FINITE ELEMENT METHOD IN HEAT TRANSFER AND FLUID DYNAMICS, Third Edition
J.N. Reddy and D.K. Gartling

MECHANICS OF LAMINATED COMPOSITE PLATES AND SHELLS: THEORY AND ANALYSIS, Second Edition
J.N. Reddy

MICROMECHANICAL ANALYSIS AND MULTI-SCALE MODELING USING THE VORONOI CELL FINITE ELEMENT METHOD
Somnath Ghosh

NUMERICAL AND ANALYTICAL METHODS WITH MATLAB®
William Bober, Chi-Tay Tsai, and Oren Masory

NUMERICAL AND ANALYTICAL METHODS WITH MATLAB® FOR ELECTRICAL ENGINEERS
William Bober and Andrew Stevens

PRACTICAL ANALYSIS OF COMPOSITE LAMINATES
J.N. Reddy and Antonio Miravete

SOLVING ORDINARY AND PARTIAL BOUNDARY VALUE PROBLEMS IN SCIENCE and ENGINEERING
Karel Rektorys

STRESSES IN BEAMS, PLATES, AND SHELLS, Third Edition
Ansel C. Ugural

Preface

I have been teaching two courses in computer applications for engineers at Florida Atlantic University (FAU) for many years. The first course is usually taken in the student's sophomore year; the second course is usually taken in the student's junior or senior year. Both computer classes are run as lecture-laboratory courses, and the MATLAB® software program is used in both courses. To familiarize students with engineering-type problems, approximately six or seven projects are assigned during the semester. Students have, depending on the difficulty of the project, either one week or two weeks to complete each project. I believe that the best source for students to complete the assigned projects in either course is this textbook.

This book has its origin in a previous textbook, *Numerical and Analytical Methods with MATLAB®* by William Bober, Chi-Tay Tsai, and Oren Masory, also published by CRC Press. The previous book was primarily oriented toward mechanical engineering students. I and Jonathan Plant of CRC Press envisioned that a similar text would fill a need in electrical engineering curricula; as a result, I enlisted Dr. Andrew Stevens to replace the projects in the existing textbook with those oriented toward electrical engineering students. This new textbook retains the philosophy of teaching that exists in the original textbook.

The advantage of using the MATLAB software program over other packages is that it contains built-in functions that numerically solve systems of linear equations, systems of ordinary differential equations, roots of transcendental equations, integrals, statistical problems, optimization problems, signal-processing problems, and many other types of problems encountered in engineering. A student version of the MATLAB program is available at a reasonable cost. However, to students, these built-in functions are essentially black boxes. By combining a textbook on MATLAB with basic numerical and analytical analysis (although I am sure that MATLAB uses more sophisticated numerical techniques than are described in these textbooks), the mystery of what these black boxes might contain is somewhat alleviated. The text contains many sample MATLAB programs that should provide guidance to the student on completing the assigned projects. Many of the projects in this book are non-trivial and, I believe, will be good training for a graduating engineer entering industry or in an advanced degree program.

Furthermore, I believe that there is enough material in this textbook for two courses, especially if the courses are run as lecture-laboratory courses. The advantage of running these courses (especially the first course) as a lecture-laboratory course is that the instructor is in the computer laboratory to help the students debug their programs. This includes the sample programs as well as the projects.

The common core of the book is the introduction to the MATLAB programming environment in Chapter 2. Then, depending on the individual curriculum (but typically sophomore year), a course might proceed with mathematical techniques for matrix algebra, root finding, integration, and differential equations (Chapters 3 through 6). A more advanced course (perhaps in junior or senior year) might include transform techniques (Chapters 7 and 8) and advanced topics in curve fitting and optimization (Chapters 9 and 10). MATLAB's graphical design environment, Simulink®, is introduced in Chapter 11 and could be relocated elsewhere in the syllabus.

We have tried to make each chapter stand alone so that each may be rearranged based on the preference of the instructor. In many cases, we have used the resistor-inductor-capacitor (RLC) circuit as an example, and we have put the basic derivation of this circuit in Appendix A to minimize the chapter dependencies and facilitate reordering. In all cases, we have attempted to provide illustrative examples using modern topics in electrical engineering.

All chapters (except for Chapter 1) contain projects, and some also contain several exercises that are less difficult than the projects and might be assigned prior to a project assignment. All projects require the student to write a computer program, most requiring the use of MATLAB built-in functions and solvers.

Additional materials, including down loadable copies of all examples in the textbook, are available from the CRC press website:

http://www.crcpress.com/product/isbn/9781439854297

MATLAB and Simulink are registered trademarks of The MathWorks, Incorporated. For product information, please contact

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

William Bober
Florida Atlantic University
Department of Civil Engineering

Acknowledgments

We wish to thank Jonathan Plant of CRC Press for his confidence and encouragement in writing this textbook. In addition, we would like to express thanks to Chris LeGoff for help in preparing the artwork and to Patrick Farrell and Michael Rohan for their technical expertise.

We also wish to express our deep gratitude to Selma Bober, Theresa Stevens, and Emma Stevens for tolerating the many hours we spent on the preparation of the manuscript, the time that otherwise would have been devoted to our families.

About the Authors

William Bober, Ph.D., received his B.S. degree in civil engineering from the City College of New York (CCNY), his M.S. degree in engineering science from Pratt Institute, and his Ph.D. degree in engineering science and aerospace engineering from Purdue University. At Purdue University, he was on a Ford Foundation Fellowship; he was assigned to teach one engineering course each semester. After receiving his Ph.D., he went to work as an associate engineering physicist in the Applied Mechanics Department at Cornell Aeronautical Laboratory in Buffalo, New York. After leaving Cornell Labs, he was employed as an associate professor in the Department of Mechanical Engineering at the Rochester Institute of Technology (RIT) for the following twelve years. After leaving RIT, he obtained employment at Florida Atlantic University (FAU) in the Department of Mechanical Engineering. More recently, he transferred to the Department of Civil Engineering at FAU. While at RIT, he was the principal author of a textbook, *Fluid Mechanics*, published by John Wiley & Sons. He has written several papers for *The International Journal of Mechanical Engineering Education* (IJMEE) and more recently coauthored a textbook, *Numerical and Analytical Methods with MATLAB®*.

Andrew Stevens, Ph.D., P.E., received his bachelor's degree from Massachusetts Institute of Technology, his master's degree from the University of Pennsylvania, and his doctorate from Columbia University, all in electrical engineering. He did his Ph.D. thesis work at IBM Research in the area of integrated circuit design for high-speed optical networks. While at Columbia, he lectured a course in the core undergraduate curriculum and won the IEEE Solid-State Circuits Fellowship. He has held R&D positions at AT&T Bell Laboratories in the development of T-carrier multiplexer systems and at Argonne National Laboratory in the design of radiation-hardened integrated circuits for colliding beam detectors. Since 2001, he has been president of Electrical Science, an engineering consulting firm specializing in electrical hardware and software. He has published articles in several scientific journals and holds three patents in the areas of analog circuit design and computer user interfaces.

Contents

Chapter 1

Numerical Methods for Electrical Engineers

1.1 Introduction

All disciplines of science and engineering use numerical methods for the analysis of complex problems. However, electrical engineering particularly lends itself to computational solutions due to the highly mathematical nature of the field and its close relationship with computer science. It makes sense that engineers who design high-speed computers would also use the computers themselves to aid in the design (a process known as bootstrapping). In fact, the entire field of computer-aided design (CAD) is dedicated to the creation and improvement of software tools to enable the implementation of highly complex designs.

This book describes various methods and techniques for numerically solving a variety of common electrical engineering applications, including circuit design, electromagnetic field theory, and signal processing. Classical engineering curricula teach a variety of methods for solving these problems using techniques such as linear algebra, differential equations, transforms, vector calculus, and the like. However, in many cases, the search for a closed-form solution leads to extreme complexity, which can cause us to lose physical insight into the problem. In solving these same problems numerically, we will often revert to fundamental physical relations, such as the differential relationship between capacitor current and voltage or the electric field of a point charge. Often, a problem that seems intractable when solved symbolically can become trivial when solved numerically. And sometimes, the simpler numerical solution can be elusive because we are so used to thinking in terms of advanced calculus (a classic case of not being able to see the forest but for the trees).

1.2 Engineering Goals

Some fundamental goals in engineering include

- Design new products or improve existing ones
- Improve manufacturing efficiency
- Minimize cost, power consumption, and nonreturnable engineering (NRE) cost
- Maximize yield and return on investment (ROI)
- Minimize time to market

The engineer will frequently use the laws of physics and mathematics to achieve these goals.

Many electrical engineering processes involve expensive manufacturing steps that are both delicate and time consuming. For example, the fabrication of integrated circuits can involve thousands of manufacturing steps, including wafer preparation, mask creation, photolithography, diffusion and implantation, dicing, testing, packaging, and more. These steps can take weeks or months to perform at substantial expense and in clean rooms. Any design mistakes require repeating the process; thus, it is our job as designers to model and simulate designs as much as possible in advance of manufacture to eliminate flaws and minimize the iterations necessary to produce the final product.

Using integrated circuit design as an example, we might use computers for the

a. Design stage: Solve mathematical models of physical phenomena (e.g., predicting the behavior of PN junctions)
b. Testing stage: Store and analyze experimental data (e.g., comparing the laboratory-measured actual behavior of PN junctions to the prediction)
c. Manufacturing stage: Controlling machine operations to fabricate and test silicon wafers and dice

1.3 Programming Numerical Solutions

Physical phenomena are always described by a set of governing equations, and numerical methods can be used to solve the set of governing equations even in the absence of a closed-form solution. Numerical methods invariably involve the computer, and the computer performs arithmetic operations on discrete numbers in a defined sequence of steps. The sequence of steps is defined in the program. A useful solution is obtained if

a. The mathematical model accurately represents the physical phenomena; that is, the model has the correct governing equations.
b. The numerical method is accurate.
c. The numerical method is programmed correctly.

This text is mainly concerned with items (b) and (c).

The advantage of using the computer is that it can carry out many calculations in a fraction of a second; at the time of this writing, computer speeds are measured in teraflops (trillions of floating point operations per second). However, to leverage this power, we need to write a set of instructions, that is, a program. For the problems of interest in this book, the digital computer is only capable of performing arithmetic, logical, and graphical operations. Therefore, arithmetic procedures must be developed for solving differential equations, evaluating integrals, determining roots of an equation, solving a system of linear equations, and so on. The arithmetic procedure usually involves a set of algebraic equations. A computer solution for such problems involves developing a computer program that defines a step-by-step procedure for obtaining an answer to the problem of interest. The method of solution is called an *algorithm*. Depending on the particular problem, we might write our own algorithm, or as we shall see, we can also use the algorithms built into a package like MATLAB® to perform well-known algorithms such as the Runge-Kutta method for solving a set of ordinary differential equations or use Simpson's rule for evaluating an integral.

1.4 Why MATLAB?

MATLAB was originally written by Dr. Cleve Moler at University of New Mexico in the 1970s and was commercialized by MathWorks in the 1980s. It is a general-purpose numerical package that allows complex equations to be solved efficiently and subsequently generate tabular or graphical output. While there are many numerical packages available to electrical engineers, many are highly focused toward a particular application (e.g., SPICE for modeling electronic circuits). Also, MATLAB is not to be confused with CAD software for schematic capture, layout, or physical design, although this software often integrates with an accompanying numerical package.

Originally, MATLAB was a command-line program that ran on MS-DOS and UNIX hosts. As computers have evolved, so has MATLAB, and modern editions of the program run in windowed environments. As of the time of this writing, MATLAB R2011b runs natively on Microsoft Windows, Apple MacOS, and Linux. In this text, we assume that you are running MATLAB on your local machine in a Microsoft Windows environment. It should be straightforward for non-Windows users to translate the usage descriptions to their preferred environment. In any case, these differences are largely limited to the cosmetics and presentation of the program and not the MATLAB commands themselves. All versions of MATLAB (on any platform) use the same command set, and the Command Window on all platforms should behave identically.

MATLAB is offered with accompanying "toolboxes" at additional cost to the user. A wide variety of toolboxes are available in fields such as control systems, image processing, radio frequency (RF) design, signal processing, and more. However, in this text, we largely focus on fundamental numerical concepts and limit ourselves to basic MATLAB functionality without requiring the purchase of any additional toolboxes.

1.5 The MATLAB Programming Language

There are many methodologies for computer programming, but the tasks at hand boil down to the following:

a. Study the problem to be programmed.
b. List the algebraic equations to be used in the program based on the known physical phenomena and geometries of the problem.
c. Create a general design for the program flow and algorithms, perhaps by creating a flowchart or by writing high-level pseudocode to outline the main program modules.
d. Carry out a sample calculation by hand to prove the algorithm.
e. Write the program using the list of algebraic equations and the outline.
f. Debug the program by running it and fixing any syntax errors.
g. Test the program by running it using parameters with a known (or intuitive) solution.
h. Iterate over these steps to refine and further debug the algorithm and program flow.
i. If necessary, revise the program to obtain faster performance.

Experienced programmers might omit some of these steps (or do them in their head), but the overall process resembles any engineering project: design, create a prototype, test, and iterate the process until a satisfactory product is achieved.

MATLAB may be considered a programming or scripting language unto itself, but like every programming language, it has the following core components:

a. Data types, i.e., formats for storing numbers and text in the program (e.g., integers, double-precision floating point, strings, vectors, matrices)
b. Operators (e.g., commands for addition, multiplication, cosine, log)
c. Control flow directives for making decisions and performing iterative operations (e.g., `if`, `while`, `switch`)
d. Input/output ("I/O") commands for receiving input from a user or file and for generating output to a file or the screen (e.g., `fprintf`, `fscanf`, `plot`, `stem`, `surf`)

MATLAB borrows many constructs from other languages. For example, the `while`, `switch`, and `fprintf` commands are from the C programming language (or its descendents C++, Java, and Perl). However, there are some fundamental differences as well. For example, MATLAB stores *functions* (known in other languages as "subroutines") in separate files. The first entry in a *vector* (known in most other languages as an *array*) is indexed by the number 1 and not 0. However, the biggest difference is that all MATLAB variables are vectors, thus providing the ability to manipulate large amounts of data with a terse syntax and allowing for the