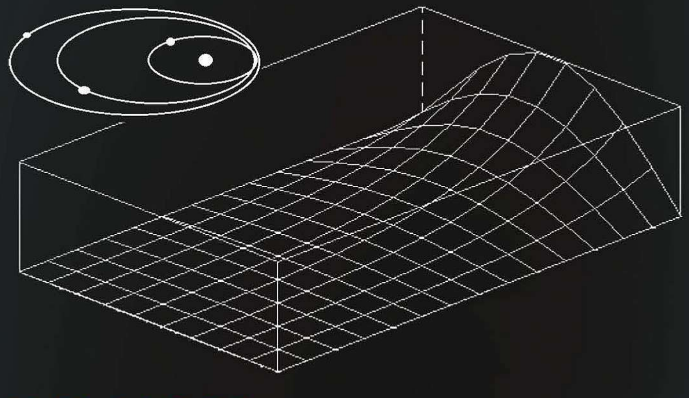
COMPUTER MODELS OF PROCESS DYNAMICS

FROM NEWTON TO ENERGY FIELDS



OLIS RUBIN

IEEE PRESS

WILEY

Computer Models of Process Dynamics

Computer Models of Process Dynamics

From Newton to Energy Fields

Olis Rubin
Brooklyn, Pretoria, South Africa

IEEE PRESS
WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data Applied for

ISBN: 9781119885658 [Hardback]

Cover Design: Wiley
Cover Images: © zhengshun tang/Getty Images; Courtesy of Olis Rubin

Set in 9.5/12.5pt STIXTwoText by Straive, Pondicherry, India

To Judy
She brought beauty, goodness and love in my life.
We give thanks for all the blessings that we have received, together with our children

Contents

Preface

"It is unworthy for excellent men to lose hours like slaves in the labor of calculation which could safely be relegated to anyone else if machines were used"

Gottfried Wilhelm Leibniz (1685)

My first job was with a large electrical engineering enterprise where I had been moved to a division that designed and installed heavy duty motor drives. Our division had just acquired an analog computer that was barely large enough to simulate these machines. I was lucky to be chosen to work with my friend Gopal to see what we could achieve with this "new-fangled contraption." Our first project was to investigate the performance of a Ward-Leonard (generator–motor) set that drove a huge mining machine. The computer gave us a new "window to the world," through which we could see what was happening inside the motor. All that we had learned as undergraduates now fell into place. Our basic training had considered the steady state operation of electric motors, while the lectures on dynamic behavior were entirely mathematical. We could now construct a computer model that allowed us to see the hardware, the differential equations, and the transient behavior of shafts and other variables. This is what a computer model can do for us!

Leibniz would have been elated if he could have foreseen how far machines could take us, beyond just saving us laborious calculations. Digital computers now provide us with platforms that make the physical modeling process almost effortless, leaving us free to think about more important questions. This book is written to show how a computer can be programmed to simulate motion in general. We will not include the growing market for new user interfaces, such as virtual reality or networking. These fields should be left to the expert attention of specialists.

In addition to setting the wheels in motion along a path that led to the computers of today, Leibniz was also the co-founder of differential calculus. The creation of a dynamic model is deeply rooted in mathematical analysis. We will sacrifice theoretical rigor in order to simplify the mathematics in a way that is intelligible to the working engineer and computer professional. Wherever possible we will explain mathematical operations by means of computer programs that approximate their behavior.

Thanks go to the many reviewers whose suggestions helped to bring this book into its present form. I hope that it will give readers the benefit of experience that was gained by working with many colleagues on diverse projects. Thanks to Philip de Vaal, whose enthusiastic support strengthened me to persevere in the face of setbacks, and to Becker van Niekerk, whose industry constantly invigorated me to complete this project. This is the second time that Aileen Storry has helped me bring a book to see the light of day. Thanks to the professionalism of the Wiley team, led by Kimberly, Mustaq and Patricia, that managed the publication process.

I must thank my dearest wife for her constant, unstinting support. Not only is her spelling better than mine, but she has a way of provoking me to write more lucidly for you, the reader. May it thus give you food for thought, and better equip you for the future!

1 Introduction

1.1 Engineering uses of computer models

Computer models are used throughout every phase of industrial research, design, and development. Simulation studies can play a large part in the concept phase of an engineering project since computer models can usually be created in less time and cost less than hardware prototypes. The very creation of such models forces everyone to delve deeper into the underlying physics of the plant. This can give valuable insight into plant operation that can aid the plant designers. If such simulators are developed before the plant is built, they can be used to evaluate the design before expensive decisions are made, and thereby help to avoid costly mistakes. In the later stages of a development program, we can greatly reduce the time and cost of commissioning and qualification by using simulator studies to reduce the scale of hardware testing. The model can also be used to determine safety limits and emergency procedures by simulating tests that would be hazardous in the real plant. There is also a growing market for training simulators to ensure the competence of the operators who will run the plant. We can foresee that this will tie up with the creation of virtual reality and the computer game industry.

Computer modeling of dynamic processes has a long association with control engineering, where there is often an interest in the speed with which the system settles to a required operating condition. The designers generally make use of feedback control loops, where they have to allow for the time response of the plant to control inputs. Computer models were used to study simple servomechanisms, and this led to their extensive use in very large and complex feedback control problems. They have been widely employed in the aerospace industry for the design of flight control systems. The process control industry has applied computer modeling to such large-scale studies as optimizing a complete chemical process. The growing field of robotics will also make use of dynamic models.

"The purpose of computing is insight, not numbers."

R.W. Hamming (1962)

Computer models also find application in many diverse scientific fields, including areas that are known as the softer sciences. As we use machinery to perform manual labor, we can use computers to mechanize difficult repetitive mathematical processes and thereby free our minds to consider questions of a more intellectual nature.

1.1.1 Mission statement

This book is written for you, the professional worker in the field of computer modeling, who wishes to widen your horizon and needs to consider new approaches, applications, and advanced techniques. When we use the word "model" we are thinking of a program that simulates the dynamic behavior of a physical phenomenon. Perhaps you have not already entered the field, but wish to apply your training in applied mathematics and experience with computers to explore the behavior of an object that evolves in this way. This book proceeds from basic methods of programming and sets up mathematical models of fairly common objects that can be described by reasonably simple equations.

A series of case studies covers a myriad of different topics in order to provide a vista of the challenges that fall within this discipline. These topics have been fitted into a framework that progresses from introductory material to subjects of increasing complexity. You will meet scores of examples that have been carefully chosen to take you step by step along a logical learning curve that leads to greater enlightenment.

It is shown how the computer has progressed from being a mere tool to convert mathematical equations into numbers. We can interact with it as with virtual reality where its graphical output makes the equations become alive.

There is another mission to be accomplished. If there is to be synergy between yourself and the computer, the simulation process has to stand firmly on the following legs:

Techniques of computer modeling,
Mathematical analysis, and
Imagination, Inspiration, and Creative Thinking

1.2 The subject matter

The book provides a long series of case studies that are based on personal experience. The emphasis is on the simulation of moving objects or the propagation of energy in a physical medium. Computer models are presented to the reader either as program listings or as block diagrams. Most of them are programmed on the MATLAB® or Scilab® platforms:

MATLAB is a registered trademark of The MathWorks, Inc.
Scilab is registered under the GPLv2 license (previously CeCILL) as circulated by CEA, CNRS, and Inria.
The computer code fragments use statements that are easily understood by Python and C users.

Chapter 2 gives an introduction to computer programming, using the instruction sets from MATLAB and Scilab. This firstly shows how to perform various repetitive operations, and then goes on to consider the use of a digital computer to simulate differential equations. These techniques will serve as a springboard for the creation of even the most advanced computer models.

Chapter 3 serves as an object lesson to show how creative thinking is a tool in the model building process. It explains how scientific theories are actually conceptual models that describe physical phenomena and shows how the evolution of science depended on a series of inspired guesses. It describes the development of three different mathematical frameworks that are used in later chapters to illustrate the creation of various computer models.

Chapter 4 shows how differential equations can be implemented as computer models, which can then be exploited to produce graphs and numbers. It also shows how mathematical tools based on calculus can be used to find analytical solutions that satisfy the equations. It thereby illustrates how problems can be studied by two independent methods to increase our confidence in the results.

Chapter 5 considers the creation of differential equations that describe the motion of point masses and their implementation as computer models. It also shows how analytical solutions can be used to perform cross-checks, to verify the computed results.

Chapter 6 then considers the motion of a rigid body and the creation of a flight simulator in the aircraft industry.

Chapter 7 analyses the flow of heat through a solid body by exploiting mathematical methods that are based on the concept of scalar and vector fields. These are used to show how the evolution of a temperature field can be described by a partial differential equation. It is demonstrated how a computer model can be created by approximating the body as a series of finite elements, where each individual element can be described by an ordinary differential equation.

Chapter 8 analyses wave propagation of an energy field through a continuous medium. It again demonstrates how a computer model can be created by approximating the medium as a series of finite elements. Methods are used that are similar to those described in Chapter 7. The analysis of the physical phenomena discussed in these two chapters is much more complex than the study of mechanical motion. The case studies use a combination of computer modeling, mathematical analysis, and physical analogies as a way to gain a better understanding of the physical processes that are considered.

Chapter 9 widens our horizon by exploring the realm of uncertainty and statistical analysis. The examples that appear in this chapter consider various topics, which range from imperfect theoretical knowledge to measurement noise, as well as business models and digital images.

Chapter 10 shows how simulation engineers can work as members of a design team within an engineering project. At the beginning of the project the computer model can be used as an electronic prototype that is created more quickly and cheaply than hardware. It also provides the facility for checking the sensitivity of the product to variations in parameters within the design tolerances. There must be a configuration management process to keep track of changes to the model during the course of the project. The need for formal procedures is emphasized in order to create trustworthy models. This includes cross-checks on simulation results by means of mathematical analysis to see that the computer model has been implemented correctly.

When engineers use computer models to assist them with making critical design decisions they must remember that the solutions obtained with the computer are greatly dependent on the equations that were created to represent the physical hardware. Approximations in these equations result in approximate solutions. The equations used in this book may be sufficient for an introduction to the topic, and indeed many have been used for preliminary investigations of the problems that will be discussed. However, before readers attempt serious investigations of this type, it is recommended that they satisfy themselves about the form of the equations that they choose to use.

The moment of truth arrives when the hardware is tested. After this, the computer model can still serve as the means whereby we can quickly and efficiently predict the behavior of the hardware under different operating conditions.

1.3 Mathematical material

The advance of science led to the creation of new mathematical methods that describe the new theoretical concepts. Thus, Newton's laws of motion led to the creation of vector analysis, while Maxwell's equations required the concept of a vector field. Such mathematical analysis forms an integral part of the modeling process, both in the definition of the models and in checking the results. It was thus preferable to include selected mathematical material in separate appendices that can be studied separately or consulted when referred to in the chapters.

Appendix A describes frequency response techniques that are used in engineering, such as the design of feedback controllers. It shows how MATLAB and Scilab can determine linear state equations and transfer functions that approximate a given computer model.

Appendix B describes the use of vectors to create kinetic models of point masses and rigid bodies.

Appendix C describes the application of vector fields to solve diffusion and wave equations.

Appendix D discusses the mathematical modeling and analysis of random events.

1.4 Some remarks

Mathematical modeling began when Galileo and Newton married the sciences of astronomy and mathematics. Many professions have since used calculus to analyze systems of every description. As the calculations became more complex, scientists began to create computer models. This began with the creation of analog computers and the development of sophisticated simulation techniques. Some of the books that were written on the subject are listed below and are still a valuable source of information to anyone who wishes to create computer models. Once the model has been mathematically defined, it is now possible to use a digital simulation platform that greatly simplifies its programming on the computer. We can then experiment with the model and obtain answers with astonishing rapidity. This creates a risk that we sometimes forget to take some time off to consider where we are going. The IBM Corporation used to hand out posters that continue to give good advice "THINK." The scientists of yesteryear can still serve as a role model in this regard.

This book aims to show how creative thinking, mathematical analysis, and computer models can be used together to achieve a synergy that may not otherwise be apparent to the reader. It also takes true wisdom to decide how realistic a model must be and what is enough to satisfy our needs.

Bibliography

Abbasov, I.B. (2019). *Computer Modeling in the Aerospace Industry*. New York: Wiley.

Hamming, R.W. (1962). *Numerical Methods for Scientists and Engineers*. New York: McGraw-Hill.

Karplus, W.J. (1958). *Analog Simulation*. New York: McGraw Hill.

Korn, G.A. and Korn, T.M. (1956). *Electronic Analog Computers*. New York: McGraw Hill.

Paynter, H.M. (1960). *A Palimpsest on the Electronic Analog Art, Dedham*. MA: Philbrick Researches.

Raczynski, S. (2014). *Modeling and Simulation*. New York: Wiley.

Rogers, A.E. and Connoly, T.W. (1960). *Analog Computation in Engineering Design*. New York: McGraw Hill.

Rubin, O. (2016). *Control Engineering in Development Projects, Dedham*. MA: Artech House.

Shearer J.L., Murphy, A.T. and Richardson, H.H., *Introduction to System Dynamics*, Reading, MA: Addison-Wesley, 1967

Soroka, W.W. (1954). *Analog Methods in Computation and Simulation*. New York: McGraw Hill.

Tomovic, R. and Karplus, W.J. (1962). *High Speed Analog Computers*. New York: Wiley.

2

From Computer Hardware to Software

This chapter spends less than 5% of its time on the computer and over 95% of its time on the software.

"For the machine is not a thinking being, but simply an automaton which acts according to the laws imposed upon it."
Ada, Countess of Lovelace, 1843
(*Source:* In the Public Domain, Rights Holder Augusta Ada King)

We will code computer models in a programming language that bears no resemblance to the code that controls the computing circuits.

2.1 Introduction

The mission statement in Chapter 1 described a learning curve that progresses from basic programming to the most challenging techniques of computer modeling. This chapter begins by giving a short introduction to the digital computer. It will then go on to describe a programming language that will typically be used in later chapters. It will also show some fundamental techniques whereby this language can be used to perform relatively complex tasks. Later chapters will consider models of continuous motion by means of differential equations; thus we require methods whereby a digital computer, which is a discrete device, can emulate continuous behavior. Section 2.7 presents techniques whereby this can be achieved.

2.2 Computing machines

Our word "calculate" comes from the Roman *calculus* (a pebble). A primitive abacus used pebbles that were laid in furrows of sand. Over many centuries machines were developed to speed up such calculations. Leonardo da Vinci (1452–1519) left sketches of the mechanisms that are used in a mechanical adding machine. Pascal (1623–1662) invented a calculator that was used by his tax-collector father. Then Babage (1791–1871) attempted to create a machine that had the capability of the modern computer (Swade, 1991).

> One evening I was sitting in the rooms of the Analytical Society at Cambridge..., with a table of logarithms lying open before me. Another member...called out. 'Well, Babage, what are you dreaming about?' to which I replied, 'I am thinking that all these tables might be calculated by machinery."
>
> Charles Babage

Babage's first machine, the "Difference Engine," had an ability to repeat calculations that had never been achieved before. He abandoned this to embark on a much more ambitious project, the "Analytical Engine." This machine embodied many of the features of modern electronic computers. It was programmable using punched cards, so complex actions could be achieved by means of a group of more elementary instructions. It had a "store" where numbers and intermediate results were held and a separate "mill" where the arithmetic processing was performed. The machine could implement computing loops and was capable of performing conditional branching:

A form of : IF ... THEN ... ELSE

The machine would have been the size of a small locomotive – 15 feet high, 6 feet across, and, in one version, 20 feet long. Had it been built, "calculating by steam" would have been a prophetic wish come true.

In 1944 Aiken built an electromechanical version of Babage's Analytical Engine (Trask, 1971). Bernstein (1963) describes the operation of this machine: "One could go in and listen to the gentle clicking of its relays, which sounded like a room full of ladies knitting." This was followed in 1945 by the ENIAC, an electronic digital computer that contained 18 000 thermionic valves that consumed 150 kW of electricity (Trask, 1971). The invention of transistors and integrated circuits then allowed the creation of low-cost, high-performance microprocessors.

2.2.1 The software interface

Digital computers are able to perform complex operations by executing many simple actions in discrete steps. The first computers were programmed by writing each step separately in binary code. Such a set of instructions, known as a program, can then be loaded into the computer memory bank. The computer then performs the particular operation by reading the instructions from memory and executing them step-by-step. Special programs were then developed and installed in the computer that allow us to write our instructions in a simpler language. Such an installed program then translates our statements into binary instructions that are executed by the computer. Today there can be several layers of software that act as an interface between the user and the actual computer hardware.

With the advent of personal computers (PCs) in the 1980s the stage was set for the development of the digital simulation platforms of today. The first PCs were slow and had little memory, but John Little anticipated that they would eventually be capable of effective technical computing and initiated the development of the product that is known as MATLAB® (Moler, 2006). This can execute program files to perform many scientific and engineering tasks. Numerical results can then be displayed or plotted in high-resolution graphics.

2.3 Computer programming

Scientific computations can be done through underlying software that allows us to write our instructions in a language that resembles the equations that are familiar to mathematicians. A typical instruction to perform an arithmetic calculation takes the form of a statement that has the following format:

variable = expression;

Mathematicians write equations in this way to define variables as algebraic functions of other variables. The above instruction will cause the computer to perform a calculation that is defined by the *expression* on the right-hand side of the equal (=) sign, and save the result as the *variable* that is defined on the left-hand side. This is saved as a binary number in its memory. It is necessary to identify the location (address) of this *variable* in the memory bank, so that it can be retrieved for later use. The underlying software platform that supports the scientific language provides a user interface where numbers are displayed as decimals, while the address of the *variable* is displayed as a name that is written in alphanumeric characters.

Several scientific languages have been developed, each of which has a slightly different algebraic syntax.

The program listings given in this book are extracted from programs that were created and run on either the Scilab or the MATLAB software platform. Programs that are saved in a file with the extension *.sce* can be run by Scilab, while files with the extension *.m* can be run by MATLAB. The individual instructions in these programs can also be typed and run in the command windows of the respective platforms. The next sections (2.3.1–2.3.6) give a short introduction to the basic syntax that is used. Readers who write programs using these platforms can consult their built-in Help facilities for further information. User Guides are also available, which give more basic training.

2.3.1 Algebraic expressions

MATLAB and Scilab perform arithmetical calculations on numerical objects. The instruction to perform a particular calculation is created by typing a statement of the form that was shown in the previous section. For example, we could define the radius of a circle by the following statement:

```
r = 1.2;
```

This would instruct the computer to save the numerical value of the radius in its memory bank. If we now type the letter `r` in the command window we would see the following:

```
r =
   1.2
```

This quantity is now available for further use. If we are using Scilab we could calculate the area within this circle by the following statement:

```
A = %pi * (r^2);
```

Scilab has a predefined variable `%pi` that is equal to the ratio of a circle's circumference to its diameter, while MATLAB uses the symbol `pi` to identify this number. The brackets define the order of execution. The above statement will thus cause the square of the radius to be calculated before it is multiplied by `%pi`.

Now suppose that we have defined the length of a cylinder by the symbol `L`. We could then calculate its volume by the following statement:

```
V = (%pi*(r^2)) * L;
```

Other arithmetic operations are similarly coded. Their default order of execution is as follows:

^	exponentiation
+	addition
-	subtraction
*	multiplication
/	division

Scilab and MATLAB allow us to create variables "on the fly" as they are defined by statements within a program. Their numerical values can then be changed by subsequent statements. If a statement refers to a calculation that involves a variable `y` that has not been previously created, the program will end and display an error message:

```
Undefined variable: y
```

MATLAB and Scilab are designed to perform arithmetical calculations on numerical vectors and matrices, where scalars are regarded to be 1-by-1 matrices. Suppose that we have created the following scalars:

```
a11 = 1;
a12 = 2;
a21 = 3;
a22 = 4;
```

We could then create the following row vectors:

```
R1 = [a11, a12];
R2 = [a21, a22];
```

If we now type the name `R1` in the command window we would see the following:

```
R1 =
   1.          2.
```

These row vectors can then be combined to form a matrix:

```
M = [R1; R2];
```

If we now type the name `M` in the command window, we would see the following:

```
M =
   1.          2.
   3.          4.
```

We can also address the individual elements in a matrix, in order to use their values in further calculations. For example, if we type the `M(1,2)` in the command window we would see the following:

```
ans =
   2.
```

Similarly, if we type the `M(2,1)` in the command window we would see the following:

```
ans =
   3.
```

Alternatively, we could have created the following column vectors:

```
C1 = [a11; a21];
C2 = [a12; a22];
```

If we now type the name `C1` in the command window we would see the following:

```
C1 =
   1.
   3.
```

These column vectors can then be combined to form the same matrix:

```
M = [C1, C2];
```

We can also address the row and column vectors in a matrix. For example, if we type `M(2,:)` in the command window we would see the following:

```
ans =
   3.          4.
```

while if we type `M(:,2)` in the command window we would see the following:

```
ans =
   2.
   4.
```

A matrix `N` can be added to or subtracted from a second matrix (`M`) provided that they have the same dimensions. Consider the following statement:

```
A = M + N;
```

The individual elements of the matrices are added together, so that A(i,j) = M(i,j) + N(i,j).

Two matrices `M` and `N` can be multiplied using the following statement, provided that the number of columns of `M` equals the number of rows of `N`:

```
A = M * N;
```

For example, if M is a row vector [m1, m2] and N is a column vector [n1; n2], the above statement produces the scalar product:

$$A = (m1\ n1) + (m2\ n2)$$

If M is a square matrix the expression `M^2` is equivalent to `M*M` while `M^3` is equivalent to `M*M*M`.

Similarly, `M^0.5*M^0.5` is equal to `M` and `M^1.5` is equal to `M*M^0.5`.

Two matrices can be multiplied element-by-element provided that they have the same dimensions. The following statement gives such a matrix that has the same dimensions as `M` and `N`:

```
A = M .* N;
```

The expression `M./N` gives element-by-element division, while the expression `M.^2` is equivalent to `M.*M`.

The Help facilities and User Guides give better information on such operations; for example, how to divide one matrix by another.

2.3.2 Math functions

MATLAB and Scilab have built-in functions, such as:

`sqrt`	square root
`sin`	sine
`cos`	cosine
`tan`	tangent
`asin`	arcsine
`atan`	arctangent
`atan2`	four quadrant arctangent
`exp`	exponential to base e
`log`	natural logarithm
`log10`	log to base 10
`factorial(4) = 1*2*3*4`	
`M'`	matrix transpose (row vector to column vector and vice versa)

If `M` is a matrix, the expression `sqrt(M)` will apply the square root operation element-by-element. Other functions operate in the same way. We can also nest functions, as shown by the following example:

```
s = sqrt(exp(A))
```

This brings us to complex numbers. If we type the expression `sqrt(-1)` in the Scilab command window we would see the following symbol for the unitary imaginary number:

```
ans =
  i
```

Scilab has predefined this quantity, so the statement `c = 3 + (4.5*%i)` creates a complex variable.

If we then type the symbol `c` in the command window, we would see the following:

```
c =
   3. + 4.5i
```

We can similarly create a complex matrix such as `M + (N*%i)`.

Unlike `sqrt(M)`, the expression `M^0.5` can operate on a real matrix to give a complex matrix.

The Help facilities and User Guides give better information on available functions.

2.3.3 Computation loops

We are now coming to those features that distinguish a computer from a calculator.

Consider the calculation of the following polynomial to produce a variable `y` as a function of a given variable (`x`):

```
y = a0 + a(1)*x + a(2)*x^2 + ... + a(99)*x^99
```

where the coefficients are defined as the elements of a vector:

```
[a(1), a(2), ... ,a(99)]
```

We can use a calculator as follows, to calculate the individual terms and add them successively to a running total:

```
y = a0
y = y + a(1)*x
y = y + a(2)*x^2
 ///////////
y = y + a(99)*x^99
```

On a computer, we can avoid the laborious process of separately programming each individual step by creating a computing loop. MATLAB and Scilab have statements that allow us to do this as follows:

```
y = a0;
K = 99;
for k = 1:K
     y = y + a(k)*x^k;
end;
```

All the operations that appear between the `for` statement and the `end` keyword will be executed 99 times. In this example there is a single operation, which calculates a term `a(k)*x^k` and adds it to the running total (`y`). The computing loop goes around from `for` to `end` and back to `for`. The `for` loop uses a counter, the variable `k`, to keep track of the number of times that the operation has been iterated. This is a true variable that can be used within the calculations. In this example it has been used to address the elements `a(k)` of the vector, and also to determine the power to which the variable `x` must be raised. The