1

Introduction

1.1 Introduction

This text is concerned with *modeling and simulation of systems*, both natural and engineered. We treat both analytical and computational methods for a range of applications to electrical, mechanical, biological, financial, social, and other systems.

The notion of what constitutes a system is very broad and application dependent. Generally, by a system, one means *a combination of interrelated components or parts that works in synergy to collectively perform a desired function*. In business, for example, a system can mean a set of processes or procedures working together, such as a payroll system or inventory system. In physiology, the circulatory system is composed of the heart, blood, arteries, veins, and capillaries and delivers oxygen and nutrients to cells and removes waste products, such as carbon dioxide. In computer science, an operating system is the software that manages hardware and software resources and provides services to application programs. In biology, an ecosystem refers to a community of interacting species and their physical environment.

The distinction between component and system is application dependent. For example, in the semiconductor industry a microprocessor is a highly complex system, whereas in the automotive industry a microprocessor is a component in systems that control engine emissions, braking, cruise control, and other functions.

1.1.1 Systems Engineering

Systems engineering is an interdisciplinary field of engineering that focuses on how to design and manage systems over their life cycles [3, 15, 17]. A key problem in systems engineering is how to deal with the increasing complexity of modern systems. Modeling and simulation are important tools to help design and

Introduction to Modeling and Simulation: A Systems Approach, First Edition. Mark W. Spong.

Companion Website: www.wiley.com/go/spong/engineeringmodeling

analyze modern complex systems. Good models allow meaningful simulation for testing, design validation, hardware-in-the-loop testing and iterative methods. Simulations are less expensive than prototype testing and can be more rapidly developed and modified.

In this text we take the viewpoint that modeling is the more important part of modeling and simulation. Simulations based on poor models are of relatively little use. Thus, we focus on the modeling aspects of systems and use simulations to illustrate their performance and gain insight into their structure.

1.1.2 The Input/Output Viewpoint

At an abstract level, we can view a system as an object or process Σ that transforms *inputs* u to *outputs* y, which we write as $y = \Sigma(u)$; see Figure 1.1. Within this input/output system paradigm various assumptions on the nature of the system must be made to derive concrete models that can be used for analysis, simulation, and prediction. How to do this is the principle subject of this text.

Figure 1.1 A general characterization of a system as an object or process that transforms inputs u to outputs y.

1.1.3 Some Examples

Examples of systems as objects that transform inputs to output are found in numerous fields of engineering and nature. Some representative examples include the following:

(1) **An electric circuit.** In the circuit shown in Figure 1.2 we can take the voltage V_1 as input and the voltage V_2 as output. The system then transforms the input to the output according to Kirchhoff's laws.

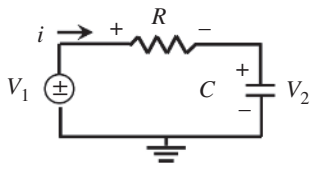

Figure 1.2 An electric circuit with input voltage V_1 and output voltage V_2. The output voltage is determined by Kirchhoff's Laws.

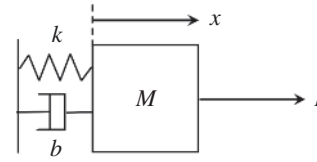

Figure 1.3 A mass–spring–damper system with input force F and output position x. The motion of the mass as a function of time is governed by Newton's laws.

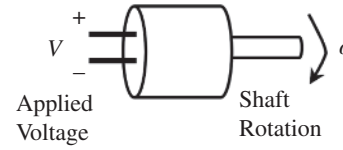

Figure 1.4 A DC motor with input voltage V and output speed ω. Both electrical and mechanical models govern the input/output behavior of this system.

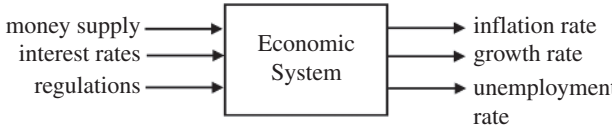

Figure 1.5 An economic system with multiple inputs and outputs. The choice of what variables to choose as inputs and what variables to choose as outputs is also an important part of the modeling process.

(2) **A mechanical system.** The system in Figure 1.3 shows an interconnected mass, spring, and damper. If an input force F acts on the mass, then the output position x of the mass is governed by Newton's laws of motion.

(3) **A DC motor.** In a DC motor, as shown in Figure 1.4, we can take the applied voltage V as input and the shaft angular velocity ω as output. The system then transforms electrical energy into mechanical energy to perform useful work.

(4) **An economic system.** A system may have multiple inputs and/or multiple outputs. For example, in an economic system, variables such as money supply, interest rates, and regulations can be taken as inputs and variables such as inflation rate, growth rate, unemployment rate can be taken as outputs (Figure 1.5).

(5) **A manufacturing plant.** A factory (Figure 1.6) can be thought of as a system that transforms raw materials (the inputs) to finished products (the outputs). The manufacturing process is composed of many individual processes that occur as discrete events, such as metal forming, welding, painting, final assembly, and other processes.

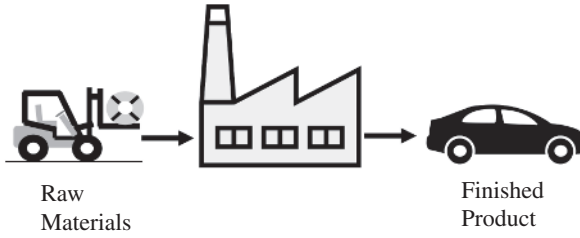

Figure 1.6 A manufacturing plant transforms input raw materials into output finished products. The system can be modeled as the interconnection of many subsystems, such as metal forming, welding, painting, assembly, and so on.

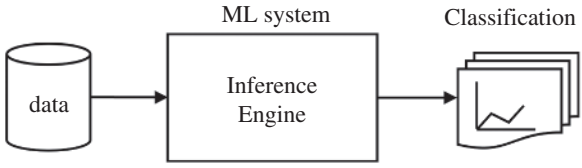

Figure 1.7 A machine learning (ML) system takes input data and produces output decisions. The generation of the input data and the algorithms used for the inference engine are key features of a learning system.

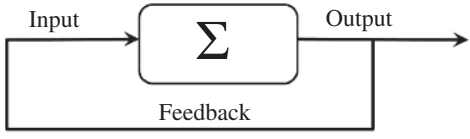

Figure 1.8 A feedback system is one where the output affects future inputs.

(6) **Machine learning.** A machine learning system can be thought of as a computer program that takes data as input and produces estimates (classifications, decisions, suggestions) as outputs (Figure 1.7).
(7) **Feedback systems.** Feedback is a fundamental property in both natural and engineered systems. By a *feedback system* we mean "a system where the input is influenced by the output."

We refer to the system in Figure 1.8 as a *closed-loop system* and the system in Figure 1.1 as an *open-loop system*. Feedback can be positive, which tends to have a destabilizing effect, as in a nuclear chain reaction, or negative, which tends to have a stabilizing effect, for example, when we sweat in hot weather to reduce our body temperature.

Feedback control is indispensable in modern engineering systems, such as automotive systems, aircraft, robotics, chemical and oil refineries, manufacturing,

energy and power systems, and a host of other applications. In biological systems, feedback is present in regulating many processes, from body temperature to cell metabolism, gene expression, hormone production, as well as balance and locomotion. In weather and climate dynamics, feedback is an important mechanism affecting global temperatures, weather patterns, ocean currents, and so on.

1.2 Model Classification

A model is a mathematical representation and is both an abstraction and an approximation of physical reality. Developing models of systems takes several forms.

Models from first principles. These are physics-based models, typically differential or difference equations derived from Newton's laws, Maxwell's equations, Kirchhoff's laws, and so on.

Black box models. These are models based on measured data to identify an input/output structure. Such data-driven methods include neural networks, genetic algorithms, frequency-response methods, machine learning, or optimization methods.

Gray box models. These are a combination of the above two using a-priori assumptions on the model together with measured data for parameter identification, systems identification, regression, or other methods.

System models may be classified in several ways, which can be useful to determine the best tools for their analysis. Some of the most useful classifications are listed below.

1.2.1 Static and Dynamic Systems

In a static system, there is an algebraic or functional relationship between input and output, as in a resistor where $V(t) = RI(t)$. Static models will be used, for example, when we discuss optimization, regression, and graph theory. In a dynamic system the output depends on the input together with the current state of the system, which typically evolves according to a differential or difference equation.

1.2.2 Linear and Nonlinear Systems

All systems possess nonlinearities. Despite this fact, linear systems are an important class of systems for two primary reasons. First, nonlinear systems are approximately linear near their equilibrium states. Second, linear systems are

well understood and their behaviors are easily characterized. Nonlinear systems, on the other hand, are difficult to analyze and can often only be studied using numerical methods. At the same time, nonlinear systems have extremely rich behavior not possessed by linear systems.

The key to linearity is the principle of superposition, with which the output response to an arbitrary input can be understood in relation to outputs generated by simpler inputs. Specifically, we have the following definition.

Definition 1.1 ***(Linearity)*** A system Σ is linear if and only if it satisfies the principle of superposition: If $y_1 = \Sigma(u_1)$ and $y_2 = \Sigma(u_2)$, then the output y corresponding to the input $u = a_1u_1 + a_2u_2$, where a_1 and a_2 are constant, is given by $y = a_1y_1 + a_2y_2$. In other words,

$$\Sigma(a_1u_1 + a_2u_2) = a_1\Sigma(u_1) + a_2\Sigma(u_2).$$

The principle of superposition says that if y_1 is the output corresponding to the input u_1 and y_2 is the output corresponding to the input u_2, then a linear combination of the inputs u_1 and u_2 produces the same linear combination of the outputs y_1 and y_2. Using superposition, one can gain an understanding of systems by analyzing their constituent components.

This approach is related to the principle of *reductionism*, which refers to a scientific approach in which a system is decomposed into subsystems, each of which is more amenable to analysis. The increasing complexity of modern engineering systems limits the usefulness of the reductionist approach. Complexity leads to *emergent behavior*, which means roughly that the whole is greater than the sum of the parts. Thus, we need to develop methods that are directly applicable to nonlinear, interconnected systems. We will discuss emergent behavior in complex networks in the final chapter of this text.

1.2.3 Distributed-Parameter Systems

By a distributed-parameter system, we mean one that is described by a partial differential equation. Examples of distributed-parameter systems are vibrating strings and membranes, the temperature distribution along a bar, and quantum mechanical systems.

1.2.4 Hybrid and Discrete-Event Systems

Hybrid systems are systems that combine both continuous-time and discrete-time components. A simple example of a hybrid system is a bouncing ball, whose

motion is governed by continuous dynamics together with a discrete *reset map* when the ball contacts the ground.

Discrete-event systems are logic-based rather than time-based. For example, in a manufacturing system an assembly operation may take place in discrete stages as parts are transferred between machines.

1.2.5 Deterministic and Stochastic Systems

In a deterministic system the output is uniquely determined from the model and is repeatable. In a stochastic system, the output is determined according to a probability, and multiple repeated trials will generally produce different outputs.

1.2.6 Large-Scale Systems

By a *large-scale system* one typically means a system with numerous components and with many variables and parameters needed to model them. These systems are also complex and difficult to model accurately. A few example of large-scale engineered systems are given below in Figures 1.9 to 1.15.

Figure 1.9 Warehouse robots for order retrieval are an example of factory automation, where the robots act with little or no human supervision.

Figure 1.10 Oil and chemical refineries are large-scale, complex systems that are made more efficient through modeling, simulation, and systems engineering principles for process control and optimization.

Figure 1.11 The electric power grid consists of an interconnected network of power generation stations, transmission lines, substations, and power distribution stations. China is currently the largest producer of electric energy followed by the United States.

Figure 1.12 A wind farm is a collection of wind turbines connected to the power grid. Each turbine is a complicated engineered system. However, in order to optimize energy production, it is not sufficient to study the individual turbines. Modeling the interactions among the turbines is also important.

Figure 1.13 Supply chains and logistics systems are responsible for moving manufactured goods on a global scale. These systems are interconnected networks of ships, airplanes, trucks, and railroads, as well as pipeline networks, loading docks, warehouses and a host of other systems that must function efficiently and economically.

Figure 1.14 Systems of autonomous vehicles are already changing the trucking industry, the shipping industry, and personal transportation. As in the wind-farm example, one must model both the individual vehicles and the interconnections among multiple vehicles.

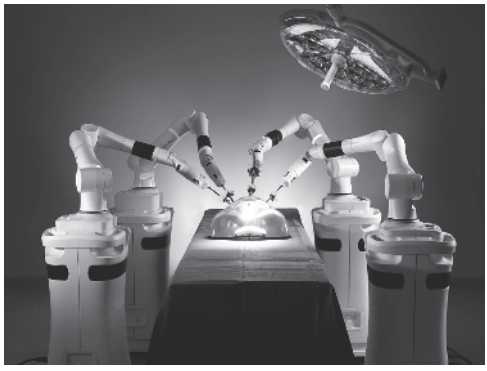

Figure 1.15 Surgical robots are examples of human/machine/computer interfaces and require new methods for modeling human dynamics in addition to robot dynamics.

1.3 Simulation Languages

There are several general purpose and special purpose simulation software programs and packages available, for example:

- **Matlab/Simulink** is a programming language and numerical computing environment. Matlab also allows symbolic computation and graphical representation of data.
- **SysML** is a graphical modeling language for analyzing complex systems. SysML is useful for modeling system requirements, system interconnections, and other features.
- **AnyLogic** is a simulation tool designed primarily for business applications like supply chains, logistics, markets and other applications.
- **FlexSim** is a simulation package designed for modeling and simulation of discrete-event systems.
- **Mathematica** is a general-purpose programming language for symbolic and numerical computation.
- **Modelica** is an object-oriented modeling language, as opposed to a programming language, designed to model systems containing electrical, mechanical, hydraulic, thermal, and other components.

- **ANSYS** is a finite-element modeling package for numerical analysis of mechanical systems.
- **SolidWorks** is a computer-aided design and computer-aided engineering package that runs primarily on Microsoft Windows.
- **Arena** is a discrete-event simulation and automation package for business and production processes.

A discussion of specialized programs for topics such as project management is outside the scope of this text. Instead, we will use Matlab/Simulink as a general purpose programming language to illustrate simulation methods for system models.

1.4 Outline of the Text

Each chapter of the text is intended to give an overview of a specific topic in systems modeling and simulation. After studying the chapters in this text the reader should be prepared for more in-depth treatment of each topic.

Chapter 2 begins with a treatment of second-order systems. Second-order systems form an important class of systems because many physical laws, like Newton's second law, are stated in terms of second-order differential equations. We characterize the behavior of linear second-order systems in terms of the eigenvalues and eigenvectors of the system coefficient matrices. We also discuss the existence of periodic solutions for nonlinear systems and present the Poincaré–Bendixson theorem, which gives conditions for the existence of periodic solutions of nonlinear systems.

Chapter 3 treats the time response of general continuous-time systems. We state sufficient conditions that guarantee a unique solution for nonlinear differential equations. We also discuss linear approximations of nonlinear systems and we introduce the matrix exponential, with which we can characterize the solutions of nth-order linear systems. We also discuss the Hartman–Grobman theorem which, in effect, says that near a hyperbolic equilibrium point, the response of a nonlinear system and its linear approximation have the same qualitative behavior. We also discuss singularly perturbed systems, which are used to model systems with very large or very small parameters that can pose particular challenges for simulation methods.

In Chapter 4 we introduce compartmental modeling and discuss examples and applications, including models of population dynamics, the spread of epidemics, and predator–prey models. The basis for these models is the so-called logistic equation, which we treat in some detail.

Chapter 5 deals with stability theory for linear and nonlinear systems. For systems operating in equilibrium configurations, it is important to know how the system will respond to perturbations and disturbances that move the system

state away from equilibrium. We discuss two main results, namely stability in the sense of Lyapunov and LaSalle's invariance principle.

In Chapter 6 we discuss discrete-time systems. We present results that mimic the results for continuous-time systems, including the existence and uniqueness of solutions and stability. We discuss the discrete-time logistic equation, which shows important differences between the continuous-time and discrete-time equations, and we present cobweb diagrams as a convenient way to visualize solutions of one-dimensional discrete-time maps.

In Chapter 7 we discuss numerical methods to simulate system models. We discuss numerical differentiation, numerical integration and the most common methods for numerical solution of ordinary differential equations using Matlab and Simulink.

Chapter 8 is an introduction to optimization theory. We will discuss general necessary and sufficient conditions for solutions of nonlinear programming problems. We discuss gradient search methods, including the method of steepest descent. We then treat Newton's method for optimization and root finding. For optimization subject to constraints we introduce Lagrange multipliers and we present the so-called Karush–Kuhn–Tucker conditions for optimality. We also discuss the importance of convexity in optimization.

Chapter 9 treats system identification. We discuss linear and nonlinear regression, and the method of recursive least squares. We also present a brief introduction to logistic regression and perceptron neural networks for binary classification.

Chapter 10 treats stochastic systems. In particular, we focus on Markov chain models and Monte Carlo methods for simulation and optimization.

In Chapter 11 we discuss feedback systems. We give examples of feedback systems and present an introduction to feedback control.

In Chapter 12 we discuss partial differential equation (PDE) models of systems. We discuss the three most important examples of PDE models, namely, the wave equation, heat equation, and Laplace's equation.

Chapter 13 treats complex networks. Many of the most important systems in the modern world are modeled as complex networks, such as the internet, World Wide Web, airline networks, social networks, and others. We give an introduction to the most relevant concepts in graph theory applied to both deterministic and random networks.

Problems

1.1 For each of the systems below, what would you choose as appropriate inputs and outputs to model the system?

1. Roof-top solar cells
2. A home thermostat

3. An insurance company
4. The respiratory system

Linear Algebra Background

1.2 Consider the matrix

$$A = \begin{bmatrix} -5 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}.$$

1. Compute the determinant of A.
2. Compute the principal minor determinants of A.
3. Compute the trace of A.
4. Compute the inverse of A.
5. Compute $A + A^T$.
6. Compute AA^T.

1.3 Compute the range and null space of the matrix

$$A = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -2 & -2 \end{bmatrix}.$$

1.4 Consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}.$$

1. Compute the eigenvalues λ_1 and λ_2 of A.
2. Compute unit eigenvectors v_1 and v_2 corresponding λ_1 and λ_2, respectively.
3. Define the matrix $T = [v_1\ v_2]$ and compute the matrix $T^{-1}AT$.

1.5 Consider the vectors $x = [1, -1]^T$ and $y = [2, 1]^T$.

1. Compute the inner product $x^T y$.
2. Computer the norms $||x||$ and $||y||$.
3. Verify that the triangle inequality and Cauchy–Schwarz inequality hold for these vectors.

Real Analysis Background

1.6 Consider the function $f(x_1, x_2) = 2x_1^2 + x_1x_2 + 4x_2^2$. Compute the gradient and Hessian of f. Show that the Hessian is a positive-definite matrix.

1.7 Compute the Jacobian of the vector field

$$f(x_1, x_2) = \begin{bmatrix} 3x_1^3 + 2x_2 \\ x_1 x_2^2 \end{bmatrix}.$$

1.8 Verify the Taylor series expansions of $\sin(x)$ and $\cos(x)$ given in Appendix B by direct calculation.

Probability Background

1.9 Suppose that you have 100 tickets in a hat numbered 1 to 100. What is the probability of drawing a ticket number divisible by 2 or 5?

1.10 Suppose that you have an exam with 10 multiple-choice questions. Each question has four choices, only one of which is correct. What is the probability of getting five or more correct answers by choosing answers randomly?

1.11 Here is a list of ten students' grades on an exam:

98, 93, 89, 77, 68, 83, 94, 88, 65, 77.

What are the mean, median, and standard deviation of the set of grades?

Matlab

1.12 Write a Matlab script to input and multiply two matrices. Your script should check that the matrices have compatible dimensions and print an error message if the dimensions are not compatible.

1.13 Given the two matrices,

$$A = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 3 & 4 \\ 0 & 3 & -1 \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

verify that the Matlab commands `inv(A)`, $A\backslash I$, and I/A return the same matrix, A^{-1}. Check that `A*inv(A)=I`.

1.14 Write a Matlab script to input an $n \times n$ matrix and compute its determinant and its eigenvalues using the Matlab functions `det` and `eig`.

1.15 Write a Matlab script to input an $n \times n$ matrix A and compute the square of each entry of A. Note that this is different from computing A^2.

1.16 Write a Matlab script to input an anonymous function $f(x)$ of a single variable x and plot the graph of $f(x)$ on an interval $[-a, a]$.

1.17 Review the documentation on Matlab symbolic variables and functions, `sym` and `syms`. Compute the symbolic derivative and integrals of the following functions:

1. $\sin(3x)$ and $\cos(3x)$
2. $x^3 - 3x^2 + 4$
3. e^{-2t}

1.18 Write a Matlab script to input two polynomials, multiply them together, and find the roots of the resulting polynomial.

1.19 A particle of mass $m = 1$ is dropped in a gravitational field $g = 9.8$ m/s^2, starting at a height $h = 100$ meters above the ground. Plot its position and velocity for 3 seconds.

1.20 Repeat the above problem assuming now that the particle is initially thrown upward with a velocity of 10 m/s.