CHAPTER 1

Modeling and Simulation

1.1 NUMERICAL APPROXIMATIONS

Numerical methods is an area of study in mathematics that discusses the solutions to various mathematical problems involving differential equations, curve fittings, integrals, eigenvalues, and root findings through approximations rather than exact solutions. This discussion is necessary because the exact solutions to these problems are difficult to obtain through the analytical approach. For example, it may be wise to evaluate

$$\int_{-1}^{5} x \sin x \, dx,$$

as the exact solution can be obtained through a well-known technique in calculus called *integration by parts*. However, it is not possible to apply the same method or any other analytical method to solve

$$\int_{0}^{2} \frac{e^{-x}}{3 \sin x + x^2} \, dx.$$

The given equation in the above integral is difficult to solve as it is not subject to the exact methods discussed in ordinary calculus. Therefore, a numerical method is needed to produce a reasonably good approximated solution. A good approximated solution, whose value may differ from the exact solution by some fractions, is definitely better than nothing.

Numerical methods are also needed in cases where the mathematical function for a given problem is not given. In many practical situations, the governing equations for a given problem cannot be determined. Instead, an engineer may have a set of n data (x_i, y_i) collected from the site to analyze in order to produce a working model. In this case, a numerical method is applied to fit a curve that corresponds to this set of data for modeling the scenario.

Solutions to numerical problems can be obtained on the computer in two ways: using a ready-made software program or programming using a primitive language. A ready-made software program is a commercial package that has been designed to solve specific problems without the hassle of going through programming. The software provides quick solutions to the problems with just a few commands. The solution to a problem, such as finding the inverse of a matrix, is obtained by typing just one or two lines of command. Matlab, Maple, and Mathematica are some of the most common examples of ready-made software programs that are tailored to solve numerical problems.

The easy approach using ready-made software programs has its drawback. The software behaves like a black box where the user does not need to know the details of the method for solving the problem. The underlying concepts in solving the problem are hidden in the software, and this approach does not really test the mathematical skill of the user. Very often, the user does not understand how the method works as all he or she gets is the generated solution. In addition, the user may face difficulty in trying to figure out why a particular solution fails because of problems such as singularity in the domain.

A ready-made software program also does not provide the flexibility of customizing the solution according to the user's requirement. The user may need special features to visualize the solution, but these features may not be supported in the software. In addition, a ready-made software program generates files that are relatively large in size. The large size is the result of the large number of program modules from its library that are stored in order to run the program.

The real challenge in solving a numerical problem is through the native language programming. It is through programming that a person will understand the whole method comprehensively. The developer will need to start from scratch and will need to understand all the fundamental concepts for solving a given problem before the solution to the problem can be developed. The whole process in providing the solution may take a long time, but a successful solution indicates the programmer fully understands the whole process.

It is also important for us to accept both approaches. Ready-made software programs are needed in cases where the program needs to be delivered fast. A ready-made software program can produce the desired solution within a short period of time if all required routines are available in the software, which is done by following the right commands and procedures in handling the software. In cases where some required actions are not supported in the software, it may be necessary to integrate the solution with a programming language such as C++. Conversely, if one starts from a programming language, it may be necessary call a ready-made software program to handle some difficult tasks. For example, C++ may be used to draw up the numerical

solution to the heat distribution problem. To see the solution in the form of surface graph, it may be wise to call a few routines from Matlab as the same feature in C++ will require a long time to develop.

1.2 C++ FOR NUMERICAL MODELING

C++ is a language that has its origin from C, developed in early 1980s. C++ retains all the procedural structure of C but adds the object-oriented features in order to meet the new requirements in computing. Both C and C++ are heavily structured high-level languages that produce small executable files. The two languages are also suitable for producing low-level routines that run the device drivers of many electronic components.

C++ is popular because of its general-purpose features to support a wide variety of applications, such as data processing, numerical, scientific, and engineering. C++ is available in all computing platforms, including Windows, UNIX, Macintosh, and operating systems for mainframe and minicomputers. C++ is a revolutionary language that has a very strong following from students, practitioners, researchers, and software developers all over the world. The language is taught in most universities and colleges in the world as a one- or two-semester subject to support numerical and general-purpose applications.

C++ is a language that strongly supports object-oriented programming. *Object-oriented programming* is a programming approach based on objects. An *object* is an instance of a class. A *class* is a set of entities that share the same parent. As it stood, C++ is one of the most popular object-oriented programming languages in the world. The main reason for its popularity is because it is a high-level language, but at the same time, it runs as powerful as the assembly language. But the real strength of C++ lies in its takeover from C to move to the era of object-oriented programming in the late 1980s. This conquest provides C++ with the powerful features of the procedural C and an added flavor for object-oriented programming.

The original product from Microsoft consists of the C compiler that runs under the Microsoft DOS (disk operating system), and it has been designed to compete against Turbo C, which was produced by the Borland Corp. In 1988, C++ was added to C and the compiler was renamed Microsoft C++. In early 1989, Microsoft launched the Microsoft Windows operating system, which includes the Windows API (application programming interface). This interface is based on 16 bits and it supports the procedural mode of programming using C.

Improvements were made over the following years that include the Windows Software Development Kit (SDK). This development takes advantage of the API for the graphical user interface (GUI) applications with the release of the Microsoft C compiler. As this language is procedural, the demands in the applications require an upgrade to the object-oriented language design approach, and this contributes to the release of the Microsoft C++ compiler. With the appearance of the 32-bit Windows API (or Win32 API) in the early 1990s, C++ was reshaped to tackle the extensive demands on Windows programming and this brings about the release of the Microsoft Foundation Classes (MFC) library. The library is based on C++, and it

has been tailored with the object-oriented methodology for supporting the application architecture and implementation.

The main reason why Visual C++.Net is needed in numerical methods is its powerful simulation and visualization tools. The Net platform refers to a huge collection of library functions and objects for creating full-featured applications on both the desktop and the enterprise Web. The classes and objects provide support for friendly user interface functions like multiple windows, menus, dialog boxes, message boxes, buttons, scroll bars, and labels. Besides, the platform also includes several tedious task-handling jobs like file management, error handling, and multiple threading. This platform also supports advanced frameworks and environments such as the Passport, Windows XP, and the Tablet PC. The strength of the Net platform is obvious in providing the Internet and Web enterprise solutions. Web services include information sharing, e-commerce, HTTP, XML, and SOAP. XML, or Extensible Markup Language, is a platform-independent approach for creating markup languages needed in a Web application.

A new approach in Visual C++.Net is the Managed Extension, which performs automatic garbage collection for optimizing the code. Garbage collection involves the removal of memory and resources unused any more in the application, which is often neglected by the programmer. The managed extension is a more structured way in programming, and it is now the default in Visual C++.Net. Central to the .Net platform is the Visual Studio integrated development environment (IDE). It is in this platform that applications are built from a choice of several powerful programming languages that include Visual Basic, Visual C++, Visual C#, and Visual J++.

In addition, IDE also provides the integration of these languages in tackling a particular problem under the .Net banner. Visual C++.Net is one of the high-performance compilers that makes up the .NET platform. This highly popular language has its root in C and was improved to include the object-oriented elements; now with the .Net extension, it is capable of creating solutions for the Web enterprise requirements. A relatively new language called Visual C# in the .Net family was developed by taking the best features from Visual Basic visual tools with the programming power of Visual C++.

In addition to its single-machine prowess, Visual C++.Net presents a powerful approach for building applications that interact with databases through ADO.NET. This product evolves from the earlier ActiveX Data Objects (ADO) technology, and it encompasses XML and other tools for accessing and manipulating databases for several large-scale applications. This feature makes possible Visual C++.Net as an ideal tool for several Web-based database applications.

1.3 MATHEMATICAL MODELING

Many problems arise in science and engineering that have their roots in mathematics. Problems of this nature are best described through mathematical models that provide the fundamental concepts needed in solving the problem. A successful mathematical modeling always leads to a successful implementation of the given project.

A *mathematical model* is an abstract model that uses mathematical language to describe the behavior of a system. It is an attempt to find the analytical solutions for enabling the prediction of the behavior of the system from a set of parameters, and their initial and boundary conditions in a given problem. A mathematical model is composed of variables and operators to represent a given problem. *Variables* are the abstractions of the quantities of interest in the described systems, whereas *operators* are the mechanisms that act on these variables. An operator can be expressed in the form of an algebraic operator, a function, a differential operator, and so on.

One good example of mathematical modeling is in the heat distribution problem in a two-dimensional plane, which is modeled as a Laplace function given by

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

In the above equation, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is an operator that acts on the heat quantity u, whose independent variables are x and y. The model describes the analytical solution of heat distribution in a given domain subject to certain initial and boundary conditions. We will discuss the numerical solution to this problem later in Chapter 11.

A mathematical model is often represented as variables in terms of objective functions and constraints. If the objective functions and constraints in the problem are represented entirely by linear equations, then the model is regarded as a *linear model*. If one or more of the objective functions or constraints are represented with a nonlinear equation, then the model is known as a *nonlinear model*. Most problems in science and engineering today are modeled as nonlinear.

A *deterministic* model is one where every set of variable states is uniquely determined by parameters in the model and by sets of previous states of these variables. For example, the path defined by a delivery truck for distributing petrol in a city is defined as a deterministic model in the form of a graph. In this case, the petrol stations can be modeled as the nodes of the graph, whereas the path is defined as the edges between the nodes. Conversely, in a *probabilistic* or *stochastic* model, randomness is present, and the variable states are not described by unique values. Very often, the model is represented in the form of probability distribution functions.

A mathematical model can be classified as static or dynamic. A *static* model does not account for the element of time, whereas a dynamic model does. Dynamic models typically are represented with difference equations or differential equations. A model is said to be *homogeneous* if it is in a consistent state throughout the entire system. If the state varies according to certain controlling mechanism, then the model is *heterogeneous*. If the model is homogeneous, then the parameters are *lumped*, or confined to a central depository. A heterogeneous, model has its parameters distributed. Distributed parameters are typically represented with ordinary or partial differential equations.

Mathematical modeling problems are also classified into black-box or white-box models, according to how much *a priori* information is available in the system. A *black-box* model is a system of which no *a priori* information is available. A *white-box*

model, also called glass box or clear box, is a system where all necessary information is available.

1.4 SIMULATION AND ITS VISUALIZATION

Mathematical modeling is often followed by a series of numerical simulations to support and verify its validity and correctness. *Simulation* is an imitation of some real thing, state of affairs, or process representing certain key characteristics or behaviors of a selected physical or abstract system. It is also the objective of a simulation to optimize the results by controlling the variables that make up the problem. By changing the variables during simulation, predictions may be made about the behavior of the system. Simulation is necessary to save time, cost, human capital, and other resources. Good results from a simulation contribute in some critical decision-making process.

Simulation can be implemented using three approaches: microscopic, macroscopic, and mesoscopic. In the *microscopic* simulation, the detail physical and performance characteristics, such as the properties of the elements that make up the problem, are considered. The simulation involves some tiny properties of the individuals or elements that make up the pieces. The results from a microscopic simulation are always reliable and accurate. However, this approach could be very costly and time consuming as data from the individuals or elements are not easy to obtain.

An easier approach is the *macroscopic simulation* that considers the deterministic factors of the whole population, rather than all the individuals or elements. In this approach, factors such as the governing mathematical equations and their macro data are considered. The steps in this approach may skip the detail components, and therefore, the results may not be as accurate as the one produced in the microscopic approach. However, this approach saves time and is not as costly as the microscopic approach.

A more realistic and practical approach is the *mesoscopic simulation* that combines the good parts of the microscopic and macroscopic approaches to produce a more versatile model. In this approach, some deterministic properties of the elements in the system are integrated with the detail information to produce a workable model.

A good simulation has several visualization features that accurately describe the elements in a system. Visualization is a form of graphical or textual presentation that easily describes the solution to a particular problem. An effective visualization includes components such as text, graphics, diagrams, images, animation, and sound in order to describe the system.

In most cases, numerical simulations are carried out effectively in a computer. A computer can accurately describe the functionality and behavior of the elements that make up the system. Today's computers are fast and have all the required resources to perform most college-level numerical simulations. Supercomputers also exist that are multiprocessor systems capable of processing numeric-intensive applications with a whopping gigaflop speed. Several parallel and distributed computer systems are also available in processing these numeric-intensive applications. Computers are also grouped into clusters to work cooperatively in grid computing networks that span across many countries in the globe.

1.5 NUMERICAL METHODS

Numerical methods is a branch of mathematics that consists of seven core areas, as follows:

- Nonlinear equations
- System of linear equations
- Interpolation and approximation
- Differentiation and integration
- Eigenvalues and eigenvectors
- Ordinary differential equations
- Partial differential equations

The basic problem in a nonlinear equation is in finding the zeros of a given function. The problem translates into finding the roots of the equation, or the points along the x-axis where the function crosses. The roots may exist as real numbers. In cases where the real roots do not exist, their corresponding imaginary roots may become a topic of study.

Linear equations are often encountered in various science and engineering problems. The problem arises frequently as the original problem reduces to linear equations at some stage in the solution. For example, mesh modeling using the finite-element method in a fluid dynamics results in a system of hundreds of linear equations. As the size of the matrix in this problem is large, the solution needs to be tackled using a fast numerical method.

Interpolation and approximation are curve fitting problems that contribute in things like designing the surface of an aircraft. The techniques are also applied in other problems, such as forecasting, pattern matching, and routing.

Differentiation and integration are fundamental topics that arise in many problems. Good approximations are needed to these two topics as their exact values may not be easy to obtain.

Problems involving ordinary differential equations arise in modeling and simulation. Exact solutions are difficult to obtain as the models are subject to variation because of the presence of constraints and nonlinear factors. Therefore, numerical methods are needed in their successful implementation.

Modeling and simulation involving partial differential equations commonly use numerical techniques as their fundamental elements. Numerical methods contribute to provide the desired solutions in most cases as the exact solutions are not practical for implementation on the computer.

1.6 NUMERICAL APPLICATIONS

Most problems in science and engineering are inherently nonlinear in nature. This nonlinearity is because the problems are dependent on variables and parameters that are nonlinear and are subject to many constraints. Many problems are also

dynamic and nondeterministic with no *a priori* information. Because of their nature, the solution to these problems will not be a straightforward task.

In most cases, the normal approach for solving nonlinear problems in science and engineering is to start from the fundamental concepts that are based on mathematics. A mathematical model that describes the system needs to be developed to represent the problem. Data from the problem are collected as an input. Numerical simulation based on the theoretical model is then performed on the data. During the simulation, the results obtained are periodically compared and matched with some real values in order to verify the correctness of the simulation.

We discuss some common modeling and simulation work that makes use of numerical methods.

Bacteria Population Growth

Population growth of bacteria in a geographical region over a period of time has been successfully modeled using a differential equation, given as

$$\frac{dx}{dt} = kx,$$

where $x(t)$ represents the size of the population at time t and k is a constant. In a broader scope, $x(t)$ in the above equation may represent the number of bacteria in a sample container.

The numerical solution to this model consists of solving an initial value problem involving the first-order ordinary differential equation. A suitable solution is provided in the form of the Runge–Kutta method of order 4, as will be discussed in Chapter 10. However, the solution is not solely provided by this method alone. Constraints such as temperature and the acidity of the fluid in the container may affect the validity of the mathematical model. The bacteria may grow faster when the temperature is high and when the acidity of the fluid is low. Therefore, a numerical simulation is needed to integrate the mathematical model with other variables and parameters. Several parameters can be included in the simulation in order to produce the correct model for this problem.

Computational Fluid Dynamics

Computational fluid dynamics (CFD) is an area of research that deals with the dynamic behavior and movement of fluid under certain physical conditions. Numerical methods and algorithms are used extensively to solve and analyze problems involving fluid flow. For example, the interaction between particles in fluids and gases are studied through simulation on the computer. Millions of calculations are performed in this simulation, as the original problem reduces to numerical problems such as matrix multiplications, matrix inverse, system of linear equations, and the computation of eigenvalues.

One area of study in computational fluid dynamics is the blood flow modeling in stenosed artery of the human body. The study contributes in predicting the occurrence of cardiovascular diseases such as heart attack and stroke. CFD simulations have also

been carried out in the aerospace and automotive industries for evaluating the air flow around moving aircrafts and cars.

The fundamental tool in CFD simulation is the Navier–Stokes equation, which describes a single-phase fluid flow. Also, a set of ordinary or partial differential equations with their initial and boundary conditions is given. The solution to these problems makes use of the finite-difference method or finite-element methods, which reduces the problem to several systems of linear equations.

Finite-Element Modeling

The finite-element method is a numerical technique for evaluating things like stresses and displacements in mechanical objects and systems. The finite-element method has been successfully applied for modeling problems involving heat transfer, fluid dynamics, electromagnetism, and solid state diffusion. At the National Aeronautics and Space Administration (NASA), the finite-element method has been applied in modeling the turbulence that occurs during the aircraft flight.

The finite-element method requires the domain in the problem to be divided into several elements in the form of line segments, triangles, rectangular meshes, volume, and so on. The method provides flexibility where the elements need not be of equal in terms of dimension and size. Solutions are obtained from these elements, and they are grouped to produce the overall solution.

The fundamentals of finite-element methods rest heavily on numerical methods. They include curve and surface fittings using interpolation or approximation techniques, systems of linear equation, and ordinary and partial differential equations.

Printed-Circuit Board Design

Massive simulations are carried out to produce optimal designs for printed-circuit boards (PCBs). A PCB forms the main circuitry of all electronic devices. A typical PCB can accommodate thousands or millions of microelectronic components such as pins, vias, transistors, processors, and memory chips. In addition, the PCB has massive wirings that connect these components.

As the space on a PCB is limited, the designer must optimize the placement of components and its routing (wiring) so that the board is capable of accommodating as many components as possible within the limited space area. We can imagine a PCB function like a city where the buildings and streets need to be designed properly so that the city will not be too congested with problems such as improper housing and traffic jams.

One technique commonly applied for routing in the PCB design is single-row routing. The problem is about designing non-crossing tracks between pairs of pins that are arranged in a single row in such a way that the tracks do not cross. Single-row routing has been known to be NP-complete with many interacting degrees of freedom. Figure 1.1 shows an optimal output from single-row routing involving 42 pins. The proven methods for solving this problem involve computer simulations using graph theory, simulated annealing, and genetic algorithm. A technique from the authors called ESSR in 2003 has been successful in producing optimal results for the problem.

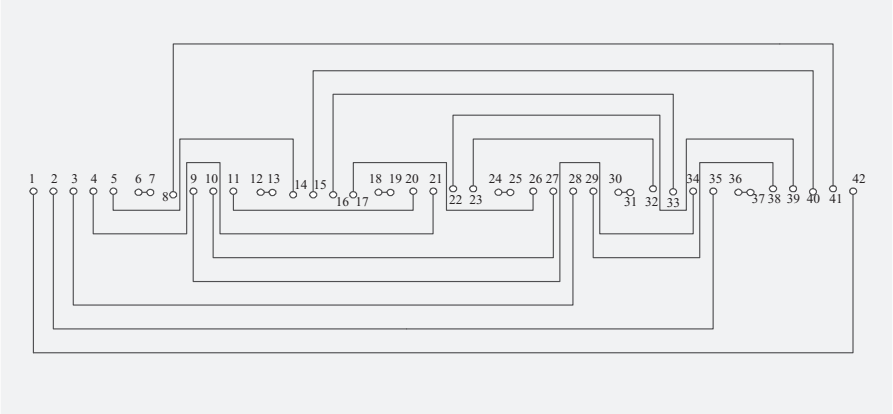

FIGURE 1.1. Optimal single-row routing involving 42 pins.

Wireless Sensor Networks

A sensor network is a deployment of massive numbers of small, inexpensive, self-powered devices called motes that can sense, compute, and communicate with other devices for the purpose of gathering local information to make global decisions about a physical environment. The object of the network is to detect certain items over a geographical region, such as the presence of harmful chemical, bacteria, electromagnetic field, and temperature distribution. In one scenario, thousands of tiny sensor motes are distributed from an aircraft over a region. Sensor network research originates from a DARPA project called Smartdust in late 1990s [2]. The research attracts interest from many disciplines because of several developments in the micro-electro mechanical system (MEMS) technology that produces many cheap and small sensor motes.

Locating sensor motes at their geographical location has become one major issue in forming the network, because many constraints need to be considered in the problem. Each node in a sensor network has a short lifetime based on a battery. To save energy, the node sleeps most of the time and becomes awake only occasionally. Therefore, the process of training the nodes in order to locate their correct location is a highly nonlinear problem with many constraints.

A coarse-grain model developed in Ref. 3 proposed a method called asynchronous training for locating the nodes in a sensor network. Figure 1.2 shows a model consisting of three networks for training the sensor nodes. Each network is indicated by concentric circles that originate from a center called *sink*. The sink is represented by a device called the *aggregating and forwarding node* (AFN), which has a powerful transmitter and receiver for reaching all nodes in its network. In this model, AFN is responsible for training the nodes, storing and retrieving data from the sensor nodes, and performing all the necessary computations on the data received to produce the desired results. The model makes use of the dynamic coordinate system based on coronas and wedges to locate the nodes through a series of transmission from the sink.

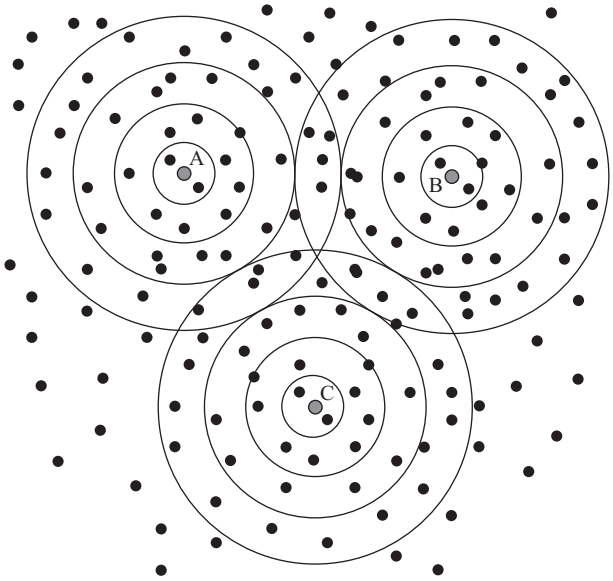

FIGURE 1.2. The asynchronous model for training the sensor nodes.

This theoretical model is not implemented yet in the real world, but it has a strong potential for applications. Modeling and simulation on the computer definitely helps in convincing the lawmakers to realize the benefit from this research.

Flood Control Modeling

The city of Kuala Lumpur, Malaysia, recently completed the construction of a tunnel that has dual purposes, as a road during the normal time and as a waterway tunnel whenever flood hits the city. The project is called SMART or the *Stormwater Management and Road Tunnel*. This project is the first of its kind in the world, and it contributes in controlling flash flood, which frequently disrupts communication in the city.

SMART is an underground tunnel about 10 km long that was constructed using tunnel-boring machines. The tunnel connects two rivers, one in the middle of the city and the other away from the city. During the normal days, the tunnel serves as two trunk roads going into and out of the city, and this contributes in reducing traffic jam in some parts of the city. During heavy rain, parts of the city get flooded as water overflows from the main river in the city. The tunnel is immediately closed to traffic, and it is converted into a waterway to divert the water from the flood area into the second river. This has the immediate effect of reducing the occurrence of flood in these areas. When the flood ends, the tunnel is cleaned, and it is open for traffic again.

A lot of work involving modeling and simulation has been carried out before the project gets started. As the project is costly, modeling and simulation using computer

help in providing all the necessary input before a decision on the viability of the project is made. The heavy costs involved are well justified here.

REFERENCES

1. S. Salleh, B. Sanugi, H. Jamaluddin, S. Olariu, and A. Y. Zomaya, Enhanced simulated annealing technique for the single-row routing problem, *Journal of Supercomputing*, 21(3), 2002, 285–302.
2. S. Olariu and Q. Xu, A simple self-organization protocol for massively deployed sensor networks, *Computer Communications*, 28, 2005, 1505–1516.
3. Q. Xu, R. Ishak, S. Olariu, and S. Salleh, On asynchronous training in sensor networks, *Journal of Mobile Multimedia*, 3(1), 2007.