CHAPTER 2

Programming and Software

In Chap. 1, we used a net force to develop a mathematical model to predict the fall velocity of a parachutist. This model took the form of a differential equation,

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

We also learned that a solution to this equation could be obtained by a simple numerical approach called Euler's method,

$$v_{i+1} = v_i + \frac{dv_i}{dt}\Delta t$$

Given an initial condition, this equation can be implemented repeatedly to compute the velocity as a function of time. However, to obtain good accuracy, many small steps must be taken. This would be extremely laborious and time-consuming to implement by hand. However, with the aid of the computer, such calculations can be performed easily.

So our next task is to figure out how to do this. The present chapter will introduce you to how the computer is used as a tool to obtain such solutions.

2.1 PACKAGES AND PROGRAMMING

Today, there are two types of software users. On one hand, there are those who take what they are given. That is, they limit themselves to the capabilities found in the software's standard mode of operation. For example, it is a straightforward proposition to solve a system of linear equations or to generate of plot of x-y values with either Excel or MATLAB software. Because this usually involves a minimum of effort, most users tend to adopt this "vanilla" mode of operation. In addition, since the designers of these packages anticipate most typical user needs, many meaningful problems can be solved in this way.

But what happens when problems arise that are beyond the standard capability of the tool? Unfortunately, throwing up your hands and saying, "Sorry boss, no can do!" is not acceptable in most engineering circles. In such cases, you have two alternatives.

First, you can look for a different package and see if it is capable of solving the problem. That is one of the reasons we have chosen to cover both Excel and MATLAB in this book. As you will see, neither one is all encompassing and each has different strengths.

By being conversant with both, you will greatly increase the range of problems you can address.

Second, you can grow and become a "power user" by learning to write Excel VBA[1] macros or MATLAB M-files. And what are these? They are nothing more than computer programs that allow you to extend the capabilities of these tools. Because engineers should never be content to be tool limited, they will do whatever is necessary to solve their problems. A powerful way to do this is to learn to write programs in the Excel and MATLAB environments. Furthermore, the programming skills required for macros and M-files are the same as those needed to effectively develop programs in languages like Fortran 90 or C.

The major goal of the present chapter is to show you how this can be done. However, we do assume that you have been exposed to the rudiments of computer programming. Therefore, our emphasis here is on facets of programming that directly affect its use in engineering problem solving.

2.1.1 Computer Programs

Computer programs are merely a set of instructions that direct the computer to perform a certain task. Since many individuals write programs for a broad range of applications, most high-level computer languages, like Fortran 90 and C, have rich capabilities. Although

CHAPTER 3

Approximations and Round-Off Errors

Because so many of the methods in this book are straightforward in description and application, it would be very tempting at this point for us to proceed directly to the main body of the text and teach you how to use these techniques. However, understanding the concept of error is so important to the effective use of numerical methods that we have chosen to devote the next two chapters to this topic.

The importance of error was introduced in our discussion of the falling parachutist in Chap. 1. Recall that we determined the velocity of a falling parachutist by both analytical and numerical methods. Although the numerical technique yielded estimates that were close to the exact analytical solution, there was a discrepancy, or *error*, because the numerical method involved an approximation. Actually, we were fortunate in that case because the availability of an analytical solution allowed us to compute the error exactly. For many applied engineering problems, we cannot obtain analytical solutions. Therefore, we cannot compute exactly the errors associated with our numerical methods. In these cases, we must settle for approximations or estimates of the errors.

Such errors are characteristic of most of the techniques described in this book. This statement might at first seem contrary to what one normally conceives of as sound engineering. Students and practicing engineers constantly strive to limit errors in their work. When taking examinations or doing homework problems, you are penalized, not rewarded, for your errors. In professional practice, errors can be costly and sometimes catastrophic. If a structure or device fails, lives can be lost.

Although perfection is a laudable goal, it is rarely, if ever, attained. For example, despite the fact that the model developed from Newton's second law is an excellent approximation, it would never in practice exactly predict the parachutist's fall. A variety of factors such as winds and slight variations in air resistance would result in deviations from the prediction. If these deviations are systematically high or low, then we might need to develop a new model. However, if they are randomly distributed and tightly grouped around the prediction, then the deviations might be considered negligible and the model deemed adequate. Numerical approximations also introduce similar discrepancies into the analysis. Again, the question is: How much the next error is present in our calculations and is it tolerable?

This chapter and Chap. 4 cover basic topics related to the identification, quantification, and minimization of these errors. In this chapter, general information concerned with the quantification of error is reviewed in the first sections. This is followed by a section on one

of the two major forms of numerical error: round-off error. *Round-off error* is due to the fact that computers can represent only quantities with a finite number of digits. Then Chap. 4 deals with the other major form: truncation error. *Truncation error* is the discrepancy introduced by the fact that numerical methods may employ approximations to represent exact mathematical operations and quantities. Finally, we briefly discuss errors not directly connected with the numerical methods themselves. These include blunders, formulation or model errors, and data uncertainty.

3.1 SIGNIFICANT FIGURES

This book deals extensively with approximations connected with the manipulation of numbers. Consequently, before discussing the errors associated with numerical methods, it is useful to review basic concepts related to approximate representation of the numbers themselves.

Whenever we employ a number in a computation, we must have assurance that it can be used with confidence. For example, Fig. 3.1 depicts a speedometer and odometer from an automobile. Visual inspection of the speedometer indicates that the car is traveling between 48 and 49 km/h. Because the indicator is higher than the midpoint between the

CHAPTER 1

Mathematical Modeling and Engineering Problem Solving

Knowledge and understanding are prerequisites for the effective implementation of any tool. No matter how impressive your tool chest, you will be hard-pressed to repair a car if you do not understand how it works.

This is particularly true when using computers to solve engineering problems. Although they have great potential utility, computers are practically useless without a fundamental understanding of how engineering systems work.

This understanding is initially gained by empirical means—that is, by observation and experiment. However, while such empirically derived information is essential, it is only half the story. Over years and years of observation and experiment, engineers and scientists have noticed that certain aspects of their empirical studies occur repeatedly. Such general behavior can then be expressed as fundamental laws that essentially embody the cumulative wisdom of past experience. Thus, most engineering problem solving employs the two-pronged approach of empiricism and theoretical analysis (Fig. 1.1).

It must be stressed that the two prongs are closely coupled. As new measurements are taken, the generalizations may be modified or new ones developed. Similarly, the generalizations can have a strong influence on the experiments and observations. In particular, generalizations can serve as organizing principles that can be employed to synthesize observations and experimental results into a coherent and comprehensive framework from which conclusions can be drawn. From an engineering problem-solving perspective, such a framework is most useful when it is expressed in the form of a mathematical model.

The primary objective of this chapter is to introduce you to mathematical modeling and its role in engineering problem solving. We will also illustrate how numerical methods figure in the process.

1.1 A SIMPLE MATHEMATICAL MODEL

A mathematical model can be broadly defined as a formulation or equation that expresses the essential features of a physical system or process in mathematical terms. In a very general sense, it can be represented as a functional relationship of the form

$$\begin{matrix}\text{Dependent}\\ \text{variable}\end{matrix} = f\left(\begin{matrix}\text{independent}\\ \text{variables}\end{matrix}, \text{ parameters, } \begin{matrix}\text{forcing}\\ \text{functions}\end{matrix}\right) \tag{1.1}$$