

Chapter 1

Computational Methods

Modern electronic computers owe their origin, to a large extent, to the needs in science and engineering. In the 50 years or so since their appearance, computers have out-performed their original goals of solving numerical problems and keeping tracks of information. They are now an essential tool in almost every aspect of the daily routines of engineers and scientists, from data collection to writing technical reports. The high speed of computation available to us these days opens up not only new ways of carrying out traditional tasks but also new areas of endeavor that have implications going well beyond what we can realize at the moment. Our concern here is limited to a small, albeit important, corner of the role of modern computers in science and engineering, namely some of the general techniques to solve common problems encountered in physics and engineering.

1-1 Numerical calculations and beyond

When we use a computer to solve a problem in science, the general assumption is that it is done numerically. Indeed, the proper name of most computers in the market is “digital computer,” reminding us of the fact that numbers are being manipulated. However, in addition to mathematical operations, such as addition and multiplication, the central processor of a computer is also capable of logical operations, that is, making decisions depending on whether a particular condition is true or false. Furthermore, a binary digit, or “bit,” of the computer memory may be regarded as a logical unit, representing the value “true” if it is on ($= 1$) and “false” if it is off ($= 0$). In this way, a computer can be programmed equally well to carry out logical decisions or, more generally, symbolic manipulations.

At the same time, the computer screen is made of lines of horizontal dots or “pixels.” On a monochrome screen, each dot can be turned on or off. For a color monitor, there is the further capability of displaying different colors at each dot. As a result, very effective graphical images can be displayed. The same is also true for paper output. For this reason, in addition to numerical calculation and symbolic manipulation, computers are used extensively for graphics.

In this volume, we shall be mainly concerned with numerical calculations. Before we get totally immersed in the topic, it is useful to remind ourselves that both symbolic manipulation and graphic presentation are also important in scientific applications of computers. We shall give a brief description of both topics before getting on to numerical calculations.

Computer algebra Among the various possibilities of using computers for “artificial intelligence” applications, symbolic manipulation, more commonly referred to as computer algebra, is an important tool in scientific endeavors. However, we shall not go into the subject in this volume. The main reason for this choice is that, most computer algebra are carried out using one of the available “packages,” such as Maple (Symbolic Computation Group, University of Waterloo), Mathematica (Wolfram Research, Inc), and Reduce (Rand Corporation). Although the Lisp language, for example, is designed for the purpose of symbolic manipulation, most of computer algebra applications these days are carried without explicitly going to the level of actually programming a computer using one of the general purpose programming languages.

In most computer applications, it is desirable for us to give the instructions in a language that is as close as possible to the working language of the subject we are involved with. In the case of algebraic calculations, it is natural for us to want to work in terms of algebraic equations. For example, if we wish to solve the set of equations:

$$\begin{aligned} ax + by &= c \\ dx + ey &= f \end{aligned} \tag{1-1}$$

it would be nice if all we need to say to the computer is something like

```
SOLVE
  ax + by = c
  dx + ey = f
FOR x AND y.
```

However, to solve Eq. (1-1) requires some knowledge of linear algebra that is beyond the basic mathematical and logical operations a computer is designed for.

As we shall see later in §5-1, the solution may be expressed in terms of ratios of determinants. In general, we cannot expect a computer or any general purpose programming language to possess such advanced knowledge of algebra. On the other hand, for commonly encountered applications, one can think of developing a set of codes that specialize in these problems. In this way, we can always call up the codes whenever we want to solve, for example, a set of linear equations as in the example above. In fact, we can put together a number of such codes that carry out related functions, such as evaluating determinants, inverting matrices, finding the roots of linear equations, and write a “driver” to manage them. Such a collection is often loosely referred to as a “package.” The development of computer algebra, to a large

extent, has followed this route. As a result, algebraic calculations are usually carried on computers in terms of one of the existing packages. In fact, some of these symbolic manipulation packages are so versatile that they can be regarded as programming languages for a large class of calculations.

Because of this tendency, discussions on algebraic calculations are often based on one of the popular packages. Since most textbooks in physics and engineering have already done a good job in presenting the algebraic aspects of various topics, the main work we need to do is to cast the problems in terms of the language of one of the packages. We shall not do this here. Instead, we shall give an example of algebraic calculation later in §1-4, just to provide some introduction to those who have not been exposed to the wonders of computer algebra.

The development of numerical methods for physics and engineering is somewhat different. Although there are many excellent “packages” available to carry out specific calculations, such as eigenvalue problems and matrices,[2] they tend to exist in the form of subprogram libraries. In this case, the user often has to write a “calling” program to transform the problem in hand into one that can take advantage of the library to perform some of the calculations. Similar to other tools, a basic knowledge of the numerical methods used in these subprogram is essential in this case.

Computer graphics An important market for computers these days is in graphics. This is, in part, due to the success in computer animation and computer aided design. Such applications clearly belong to totally different treatments from what we intend here. However, there is a small area of computer graphics that is important to numerical work, namely graphical representation of results.

In numerical calculations, the results often appear as a table of numbers and, for a complicated problem, such a table can be an extensive one. A good way to gain an overall feeling for a large set of numbers is to view them in the form of a plot. Before computers, plots are usually done on a sheet of graph paper. For simplicity, let us consider the problem of making a linear plot for some function y of a single independent variable x . The graph paper we shall use in this case is nothing but a sheet of paper with $(N_x + 1)$ evenly spaced horizontal lines and $(N_y + 1)$ evenly spaced vertical lines. The plotting area of our graph paper may therefore be regarded as made up of $N_x \times N_y$ squares. We can select one of the horizontal lines as our x -axis and a vertical line as the y -axis. The scales of our two axes are set by the ranges of values we wish to display. The actual plotting is carried out by putting on the graph paper a symbol for the value of y corresponding to each one of the values of x we are interested in. For a continuous function, the plot of y versus x is a continuous curve. However, for our purpose here, such a continuous curve may be regarded as a collection of closely spaced points, one for each possible values of the independent variable x .

We can follow basically the same steps to plot the graph on a computer screen or a sheet of output. The reason is that both types of device are made of a number of dots or pixels, very similar to the squares on our graph paper. For example, many

monitors are said to have a resolution of 1024×746 . For our present purpose, it may be regarded as a graph paper with 1024 horizontal lines and 746 vertical lines. The only difference is that, for a variety of technical reasons which we do not need to go into here, the spacings between the horizontal lines and vertical lines are not necessarily equal. In fact, the aspect ratio, i.e., the ratio of the horizontal and vertical size of each element, on a computer screen is usually less than one. As a result, each one of the basic elements on our "graph paper" is, often, a rectangle instead of a square as on a normal sheet of graph paper. While this difference causes some nuisance in displaying a graph, it does not impose any fundamental problem and we shall ignore it here. For output on paper, it is not difficult to obtain resolutions of 300 to 600 dots per inch. This means that we can easily have the equivalent of 300^2 to 600^2 elements on each square inch of a sheet of output.

The computer screen (often paper output as well) has the advantage of color. As a result, we have an additional dimension to express our "graphs" that is not easily available on graph papers. Furthermore, it is possible to generate many frames of a "graph" in a short time and, as a result, one can have a "movie" of our calculated results to show, for example, the time development of a process.

Although the basic principles of computer graphics are simple, the actual applications require some preparations. Most computers come with machine-level instructions to access and to manipulate the pixels. However, it will be extremely tedious to plot a graph using such low-level instructions. In fact, what we prefer is that, for example, once two arrays, say `X_ARRAY` and `Y_ARRAY`, are generated, we can issue an instruction like

```
PLOT Y_ARRAY versus X_ARRAY.
```

The computer will then go and find the best axes and scales to represent y as a function of x and display the results on the screen. Another instruction,

```
OUTPUT PLOT
```

will produce a printed version of the plot on a sheet of paper. For more complicated plots, such as histograms, log plots, contours, and surfaces, we can think of, at least in principle, developing similar conversation-like instructions.

Needless to say, we are not close to this ideal level of graphics programming on computers. Partly because of the fact that the standardization of computer graphics hardware and software is still in the development stage, there are only, at the time of writing, the beginnings of common high-level graphics programming languages and "interfaces" that are portable between different types of computers. As a result, we must once again resort to "packages." Similar to computer algebra, there are many fairly extensive plotting packages available, both public domain and commercial, and most plottings are done using one of these packages.