1.1 THE FIVE MAIN IDEAS

Numerical methods use numbers to simulate mathematical processes, which in turn usually simulate real-world situations. This implies that there is a purpose behind the computing. To cite the motto of the book, The Purpose of Computing Is Insight, Not Numbers. This motto is often thought to mean that the numbers from a computing machine should be read and used, but there is much more to the motto. The choice of the particular formula, or algorithm, influences not only the computing but also how we are to understand the results when they are obtained. The way the computing progresses, the number of iterations it requires, or the spacing used by a formula, often sheds light on the problem. Finally, the same computation can be viewed as coming from different models, and these different views often shed further light on the problem. Thus computing is, or at least should be, intimately bound up with both the source of the problem and the use that is going to be made of the answers—it is not a step to be taken in isolation from reality.

Much of the knowledge necessary to meet this goal comes from the field of application and therefore lies outside a general treatment of numerical methods. About all that can be done is to supply a rich assortment of methods and to
comment on their relevance in general situations. This art of connecting the specific problem with the computing is important, but it is best taught in connection with a field of application.

The second main idea is a consequence of the first. If the purpose of computing is insight, not numbers, as the motto states, then it is necessary to study families and to relate one family to another when possible, and to avoid isolated formulas and isolated algorithms. In this way a sensible choice can be made among the alternate ways of doing the problem, and once the computation is done, alternate ways of viewing the results can be developed. Thus, hopefully, the insight can arise. For these reasons we tend to concentrate on systematic methods for finding formulas and avoid the isolated, cute result. It is somewhat more difficult to systematize algorithms, but a unifying principle has been found.

This is perhaps the place to discuss some of the differences between numerical methods and numerical analysis (as judged by the corresponding textbooks). Numerical analysis seems to be the study in depth of a few, somewhat arbitrarily selected, topics and is carried out in a formal mathematical way devoid of relevance to the real world. Numerical methods, on the other hand, try to meet the need for methods to cope with the potentially infinite variety of problems that can arise in practice. The methods given are generally chosen for their wide applicability in creating formulas and algorithms as well as for the particular result being found at that point.

The third major idea is roundoff error. This effect arises from the finite nature of the computing machine which can only deal with finitely represented numbers. But the machine is used to simulate the mathematician's number system which uses infinitely long representations. In the machine the fraction \(\frac{1}{3}\) becomes the terminated decimal 0.333...3 with the obvious roundoff effect. At first this approximation does not seem to be very severe since usually a minimum of eight decimal places are carried at every step, but the incredible number of arithmetic operations that can occur in a problem lasting only a few seconds is the reason that roundoff plays an important role. The greatest loss of significance in the numbers occurs when two numbers of about the same size are subtracted so that most of the leading digits cancel out, and unless care is taken in advance, this can happen almost any place in a long computation.

Most books on computing stress the estimation of roundoff, especially the bounding of roundoff, but we shall concentrate on the avoidance of roundoff. It seems better to avoid roundoff than to estimate what did not have to occur if common sense and a few simple rules had been followed before the problem was put on the machine.

The fourth main idea is again connected with the finite nature of the machine, namely that many of the processes of mathematics, such as differentiation and in-
tegration, imply the use of a limit which is an infinite process. The machine has finite speed and can only do a finite number of operations in a finite length of time. This effect gives rise to the truncation error of a process.

We shall generally first give an exact expression for the truncation error and deduce from it various bounds. A moment's thought should reveal that if we had an exact expression, then it would be practically useless because to know the exact error is to know the exact answer. However, the exact-error expression is very useful in studying families of formulas, and it provides a starting point for a variety of error bounds.

The fifth main idea is feedback, which means, as its name implies, that numbers produced at one stage are fed back into the computer to be processed again and again; the program has a loop which uses the output of one cycle as the input for the next cycle. This feedback situation is very common in computing, as it is a very powerful tool for solving many problems.

Feedback leads immediately to the associated idea of stability of the feedback loop—will a small error grow or decay through the successive iterations? The answer may be given loosely in two equivalent ways: first, if the feedback of the error is too strong and is in the direction to eliminate the error (technically, negative feedback), then the system will break into an oscillation that grows with time; second and equivalently, if the feedback is delayed too long, the same thing will happen.

A simple example that illustrates feedback instability is the common home shower. Typically the shower begins with the water being too cold, and the user turns up the hot water to get the temperature he wants. If the adjustment is too strong (he turns the knob too far), he will soon find that the shower is too hot, whereupon he rapidly turns back to cold and soon finds it is too cold. If the reactions are too strong, or alternately the total system (pipes, valve, and human) is too slow, there will result a "hunting" that grows more and more violent as time goes on. Another familiar example is the beginning automobile driver who over-reacts while steering and swings from side to side of the street. This same kind of behavior can happen for the same reasons in feedback computing situations, and therefore the stability of a feedback system needs to be studied before it is put on the computer.

1.2 SECOND-LEVEL IDEAS

Below the main ideas in Sec. 1.1 are about 50 second-level ideas which are involved in both theoretical and practical work. Some of these are now discussed.

At the foundation of all numerical computing are the actual numbers
themselves. The floating-point number system used in most scientific and engineering computations is significantly different from the mathematician's usual number system. The floating-point numbers are not equally spaced, and the numbers do not occur with equal frequency. For example, it is well known that a table of physical constants will have about 60 percent of the numbers with a leading digit of 1, 2, or 3, and the other digits—4, 5, 6, 7, 8, and 9—comprise only 40 percent.

Although this number system lies at the foundation of most of computing, it is rarely investigated with any care. People tend to start computing, and only after having frequent trouble do they begin to look at the system that is causing it. Immediately above the number system is the apparently simple matter of evaluating functions accurately. Again people tend to think that they know how to do it, and it takes a lot of painful experience to teach them to examine the processes they use before putting them on a computer.

These two mundane, pedestrian topics need to be examined with the care they deserve before going on to more advanced matters; otherwise they will continually intrude in later developments.

Perhaps the simplest problem in computing is that of finding the zeros of a function. In the evaluation of a function near a zero there is almost exact cancellation of the positive and negative parts, and the two topics we just discussed, roundoff of the numbers and function evaluation, are basic, since if we do not compute the function accurately, there can be little meaning to the zeros we find. Because of the discrete structure of the computer's number system it is very unlikely that there will be a number \( x \) which will make the function \( y = f(x) \) exactly zero. Instead, we generally find a small interval in which the function changes sign. The size of the interval we can use is related to the size of the argument \( x \), since for large \( x \) the number system has a coarse spacing and for \( x \) small (in size) it has a fine spacing. This is one of the reasons that the idea of the relative error

\[
\text{Relative error} = \left| \frac{\text{true} - \text{calculated}}{\text{true}} \right|
\]

plays such a leading role in scientific and engineering computations. Classical mathematics uses the absolute error

\[
\text{Absolute error} = |\text{true} - \text{calculated}|
\]

most of the time, and it requires a positive effort to unlearn the habits acquired in the conventional mathematics courses. The relative error has trouble near places where the true value is approximately zero, and in such cases it is customary to use as the denominator

\[
\max\{|x|, |f(x)|\}
\]

where \( f(x) \) is the function computed at \( x \).
The problem of finding the complex zeros of an analytic function occurs so often in practice that it cannot be ignored in a course on numerical methods, though it is almost never mentioned in numerical analysis. A simple method resembling one used to find the real zeros is very effective in practice.

In the special case of finding all the zeros of a polynomial the fact that the number of zeros (as well as other special characteristics) is known in advance makes the problem easier than for the general analytic function. One of the best methods for finding them is an adaptation of the usual Newton’s method for finding real zeros, and this discussion is used to extend, as well as to analyse further, Newton’s method. It is only in situations in which a careful analysis can be made that Newton’s method is useful in practice; otherwise its well-known defects outweigh its virtues.

What makes the problem of finding the zeros of a polynomial especially important, besides its frequency, is the use made of the zeros found. The method is a good example of the difference between the mathematical approach and the engineering approach. The first merely tries to find some numbers which make the function close to zero, while the second recognizes that a pair of “close” zeros will give rise to severe roundoff troubles when used at a later stage. In isolation the problem of finding the zeros is not a realistic problem since the zeros are to be used, not merely admired in a vacuum. Thus what is wanted in most practice is the finding of the multiple zeros as multiple zeros, not as close, separate ones. Similarly, zeros which are purely imaginary are to be preferred to ones with a small real part and a large imaginary part, provided the difference can reasonably be attributed to uncertainties in the underlying model.

Another standard algorithmic problem both in mathematics and in the use of computation to solve problems is the solution of simultaneous linear equations. Unfortunately much of what is commonly taught is usually not relevant to the problem as it occurs in practice; nor is any completely satisfactory method of solution known at present. Because the solution of simultaneous linear equations is so often a standard library package supplied by the computing center and because the corresponding description is so often misleading, it is necessary to discuss the limitations (and often the plain foolishness) of the method used by the package. Thus it is necessary to examine carefully the obvious flaws and limitations, rather than pretending they do not exist.

The various algorithms for finding zeros, solving simultaneous linear equations, and inverting matrices are the classic algorithms of numerical analysis. Each is usually developed as a special trick, with no effort to show any underlying principles. The idea of an invariant algorithm provides one common idea linking, or excluding, various methods. An invariant algorithm is one that in a very real sense attacks the problem rather than the particular representation supplied to the
computer. The idea of an invariant algorithm is actually fairly simple and obvious once understood. In many kinds of problems there are one or more classes of transformations that will transform one representation of the equations into another of the same form. For example, given a polynomial

\[ P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = 0 \]

the transformation of multiplying the equation by any nonzero constant does not really change the problem. Similarly, when \( a_0 \neq 0 \), replacing \( x \) by \( 1/x \) while also multiplying the equation by \( x^n \) merely reverses coefficients. These transformations form a group (provided we recognize the finite limitations of computing), and it is natural to ask for algorithms that are invariant with respect to this group, where invariant means that if the problem is transformed to some equivalent form, then the algorithm uses, at all stages, the equivalent numbers (within roundoff, of course). In a sense the invariance is like dimensional analysis—the scaling of the problem should scale the algorithm in exactly the same way. It is more than dimensional analysis since, as in the example of the polynomial, some of the transformations to be used in the problem may involve more than simple scaling.

It is surprising how many common algorithms do not satisfy this criterion. The principle does more than merely reject some methods; it also, like dimensional analysis, points the way to proper ones by indicating possible forms that might be tried.

1.3 THE FINITE DIFFERENCE CALCULUS

After examining the simpler algorithms, it is necessary to develop more general tools if we are to go further. The finite difference calculus provides both the notation and the framework of ideas for many computations. The finite difference calculus is analogous to the usual infinitesimal calculus. There are the difference calculus, the summation calculus, and difference equations. Each has slight variations from the corresponding infinitesimal calculus because instead of going to the limit, the finite calculus stops at a fixed step size. This reveals why the finite calculus is relevant to many applications of computing: in a sense it undoes the limiting process of the usual calculus. It should be evident that if a limit process cannot be undone, then there is a very real question as to the soundness of the original derivation, because it is usually based on constructing a believable finite approximation and then going to the limit.

The finite difference calculus provides a tool for estimating the roundoff effects that appear in a table of numbers regardless of how the table was computed. This tool is of broad and useful application because instead of carefully studying
each particular computation, we can apply this general method without regard to
the details of the computation. Of course, such a general method is not as power-
ful as special methods hand-tailored to the problem, but for much of computation
it saves both trouble and time.

The summation calculus provides a natural tool for approaching the very
common (and often neglected) problem of the summation of infinite series, which is
the simplest of the limiting processes (since the index \( n \) of the number of terms
taken runs through the integers only).

The solution of finite difference equations is analogous to the solution of
differential equations, especially the very common case of linear difference equa-
tions with constant coefficients, which is a valuable tool for the study of feedback
loops and their stability. Thus finite difference equations have both a practical
and a theoretical value in computing.

### 1.4 ON FINDING FORMULAS

Once past the easier algorithms and tools for doing simple things in computing,
it is natural to attack one of the central problems of numerical methods, namely,
the approximation of infinite operations (operators) by finite methods. Inter-
polation is the simplest case. In interpolation we are given some samples of the
function, say, \( y(-1) \), \( y(0) \), and \( y(1) \), and we are asked to guess at the missing
values—to read between the lines of a table. While it is true that because of the
finite nature of the number system used there are only a finite number of values to
be found, nevertheless this number is so high that it might as well be infinite. Thus
interpolation is an infinite operator to be approximated.

There is no sense to the question of interpolation unless some additional as-
sumptions are made. The classical assumption is that given \( n + 1 \) samples of the
function, these samples determine a unique polynomial of degree \( n \), and this poly-
nomial is to be used to give the interpolated values. With the above data consist-
ing of three points, the quadratic through these points is

\[
P(x) = \frac{x(x - 1)}{2} y(-1) + (1 - x^2) y(0) + \frac{x(x + 1)}{2} y(1)
\]

We are to use this polynomial \( P(x) \) as if it were the function. This method is
known as the exact matching of the function to the data.

The error of this interpolation can be expressed as the \((n + 1)st\) derivative
(of the original function) evaluated at some generally unknown point \( \theta \) in the
interval. Unfortunately in practice it is rare to have any idea of the size of this
derivative.
For samples of the function we may use not only function values $y(x)$ but also values of the derivatives $y'(x)$, $y''(x)$, etc., at various points. For example, the cubic exactly matching the data $y(0)$, $y(1)$, $y'(0)$, and $y'(1)$ is

$$P(x) = (1 - 3x^2 + 2x^3)y(0) + (3x^2 - 2x^3)y(1) + (x - 2x^2 + x^3)y'(0) + (x^3 - x^2)y'(1)$$

It is important to use analytically found derivatives when possible. Then we can usually get a higher order of approximation at little extra cost since generally once the function values are computed, the derivatives are relatively easy to compute. No new radicals, logs, exponentials, etc., arise, and these are the time-consuming parts of most function evaluation. Of course a sine goes into a cosine when differentiated, but this is about the only new term needed for the higher derivatives. Even the higher transcendental functions, like the Bessel functions, satisfy a second-order linear differential equation, and once both the function and the first derivative are found, the higher derivatives can be computed from the differential equation and its derivatives (which are easy to compute). Thus we shall emphasize the use of derivatives as well as function values for our samples.

Although a wide variety of function and derivative values may be used to determine the interpolating polynomial, there are some sets, rather naturally occurring, for which $n + 1$ data samples do not determine an $n$th-degree polynomial. Perhaps the best example is the data $y(-1)$, $y(0)$, $y(1)$, $y''(-1)$, $y''(0)$, and $y''(1)$ which do not determine a fifth-degree polynomial—the positions and accelerations at three equally spaced points do not determine a quintic in general.

The classic method for finding formulas for other infinite operators, such as integration and differentiation, is to use the interpolating polynomial as if it were the function and then to apply the infinite operator to the polynomial. For example, if we wish to find the integral of a function from $-1$ to $+1$, given the values $y(-1)$, $y(0)$, and $y(1)$, we find the interpolating quadratic as above and integrate it to get the classical Simpson's formula:

$$\int_{-1}^{1} y(x) \, dx = \frac{1}{3} y(-1) + \frac{1}{3} y(0) + \frac{1}{3} y(1)$$

This process is called analytic substitution; in place of the function we could not handle we take some samples, exactly match a polynomial to the data, and finally analytically operate on this polynomial. This is the classical method for finding formulas. It is a two-step method: find the interpolating function and then apply the operator to this function.

There is another direct method that is almost equivalent to the analytic-substitution method. In this method we make the formula true for a sequence of
functions $y(x) = 1, x, x^2, x^3, \ldots, x^m$. For example, to derive Simpson's formula by this method we assume the form

$$\int_{-1}^{1} y(x) \, dx = a_{-1} y(-1) + a_0 y(0) + a_1 y(1)$$

and substitute the sequence of functions $1, x, x^2$. The three resulting equations determine the three unknown coefficients $a_i$, and the resulting formula is exactly the same (in this case). The two methods differ in the case where there is no interpolating polynomial; it may be that there is a formula even if there is no interpolating polynomial. For example, we have the formula

$$\int_{-1}^{1} y(x) \, dx = \frac{1}{2} [5y(-1) + 32y(0) + 5y(1)] - \frac{1}{3} y''(-1) - 32y''(0) + y''(1)$$

which is exact for sixth-degree polynomials when, as we have noted above, there is in general no interpolating polynomial of fifth degree.

It would seem as if the two methods were equivalent, for if there were an interpolating polynomial, then the formula would surely be true for the corresponding powers of $x$; and conversely, if it were true for the individual powers, then it would be true for any linear combination, namely a polynomial. The difference lies in the words if there is an interpolating polynomial, then .... It can happen that the two-stage process fails on the first step, but the one-step direct method will work.

There are two main advantages of the direct method. First the derivations are much easier, and second the direct method provides a basis for extensive generalizations. The importance of this method is hard to overestimate. It means that we can find a very wide range of formulas, all within a common framework of ideas and methods, and that we will therefore be able to compare one formula to another and then decide which one to use. Perhaps more important, it means that we can decide the kind of formula we want and then with this single method and its generalizations find almost any formula we want—we can fit the formula to the problem rather than fit the problem to the formula. Thus we can try to achieve the insight that is the main goal of the book, as stated in our motto, The Purpose of Computing Is Insight, Not Numbers.

With a general method for finding polynomial approximation formulas it is necessary to have a corresponding method for finding the error of the formula. The general method of finding the error is somewhat difficult to understand the first time. It is based on the use of a Taylor series with the integral remainder, and by substituting this into the formula for which we want the error (and manipulating the results a bit) we get the desired error formula. Once we have the exact-error term, it can be transformed in various ways to get suitable practical-error estimates.
The method for finding the truncation error term for polynomial approximation unfortunately gives it in the form of a derivative, much as the interpolation method did. This, as noted before, is unfortunate because the high-order derivatives are seldom available.

This brings up a central dilemma. Should one use a high-order formula (error term has a high-order derivative) or use the repetition of a low-order formula—the composite formula? The answer is simple in principle. It depends on the size of the high-order derivative as compared to the other lower-order derivative. This is, of course, almost no answer at all, because we seldom can decide which is better, and the basis of the choice depends on the location in the complex plane of the singularities of the function being integrated—something we seldom know.

1.5 CLASSICAL NUMERICAL ANALYSIS

Much of classical numerical analysis, as we have indicated, is based on polynomial approximation for the infinite operations of differentiation, integration, and interpolation. The polynomial approximation is also used in the numerical integration of ordinary differential equations. The most widely used methods for this are the predictor-corrector methods. A polynomial is fitted to some of the data at past points and is used to extrapolate to the next point—to predict. The predicted value is used in the differential equation to get the predicted slope. This slope along with past data is used to find another polynomial which produces the corrected value, and the corrected value of the slope is found. If the predicted and corrected values are sufficiently close, then the step is accepted as accurate enough, and if not, the step size of integration may be halved (doubled if the two values are too close).

There are so many possible predictor-corrector methods of the same order of accuracy that it is necessary to have a general theory to compare the various formulas within a common framework. Otherwise chaos and prejudice would reign.

So far we have discussed the exact-matching interpolating polynomial, and this is the more usual method. There are other methods, more or less classical, for the selection of the approximating polynomial. One method is the minimum sum of squares of the residuals between the formula and the data given. Another more modern method picks the polynomial with the minimum maximum error—the minimax, or Chebyshev, approximation. Still other criteria could be used if desired, though the labor of finding the polynomial may be fairly high in some cases.
1.6 MODERN NUMERICAL METHODS—
FOURIER APPROXIMATION

The difficulty with polynomial approximation in practice is that it is in the nature of polynomials to "wiggle" and to go to infinity for large absolute values of the argument \( x \). Physically occurring functions tend to wiggle much less than polynomials and to remain bounded for large values of the argument. Thus polynomials are a poor basis for approximation, even though they are easy to compute and to think about. The fact that the Weierstrass approximation theorem states that any continuous function can be uniformly approximated in a closed interval by a polynomial is irrelevant for two reasons. First, the degree of the Weierstrass polynomial is generally very high for even a low degree of approximation; second, we are not finding the polynomial the way the theorem states it can be found. Indeed, it is "well known"\(^1\) that for the simple function \( y(x) = 1/(1 + x^2) \) in the interval \( |x| \leq 3.63 \ldots \) the sequence of polynomials that exactly matches the function at a set of equally spaced points does not approach the function uniformly as the number of points increases indefinitely—the function and the polynomial differ by arbitrarily large amounts, and the sequence of approximating polynomials fails to converge.

Since polynomials are rather poor functions to use for approximating many functions that occur in practice, it is natural to look for other sets of functions. Among the many sets that are known to be complete (meaning that they can approximate any continuous function in a closed interval) the functions \( \sin nx \) and \( \cos nx \) \((n = 0, 1, \ldots)\) have been the most studied and are the most useful. Approximation in terms of them is usually called Fourier approximation because J. B. J. Fourier (1768–1830) used them extensively in his work.

In the simplest case of approximating a function in an interval we are given a periodic function of period, say \( 2\pi \), and are asked to approximate the function \( y(x) \) by a form

\[
y(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx)
\]

It is easy to show that since the functions are orthogonal, that is,

\[
\int_{0}^{2\pi} \cos kx \cos mx \, dx = \begin{cases} 0 & k \neq m \\ \pi & k = m \neq 0 \\ 2\pi & k = m = 0 \end{cases}
\]

\[
\int_{0}^{2\pi} \sin kx \cos mx \, dx = 0
\]

\[
\int_{0}^{2\pi} \sin kx \sin mx \, dx = \begin{cases} 0 & k \neq m \\ \pi & k = m \neq 0 \end{cases}
\]

\(^1\) Meaning that it can be found in the literature.
the coefficients in the expansion are given by

\[ a_k = \frac{1}{\pi} \int_{0}^{2\pi} y(x) \cos kx \, dx \quad b_k = \frac{1}{\pi} \int_{0}^{2\pi} y(x) \sin kx \, dx \]

The Fourier functions have a number of interesting properties beyond merely remaining bounded for all values. The error of an approximation that uses only a finite number of terms can be expressed in terms of the function rather than, as in the polynomial case, some high-order derivative. Furthermore, the rate of convergence, that is, how fast the finite series approaches the function as we take more and more terms, can be estimated easily from the discontinuities of the function and its derivatives.

Perhaps most important is the simple fact that the effect of taking equally spaced samples of the continuous function can be easily understood. The higher frequencies (speeds of rotation) appear as if they were lower frequencies. This effect, called aliasing, because one frequency goes under the name of another, is a familiar phenomenon to the watchers of TV and movie westerns. As the stage coach starts up, the wheels start going faster and faster, but then they gradually slow down, stop, go backwards, slow down, stop, go forward, etc. This effect is due solely to the sampling the picture makes of the real scene. Figures 1.6.1 and 1.6.2 should make the effect clear. Once we know the sampling rate, we know exactly what frequencies will go into what frequencies. The highest frequency that is correct is called the Nyquist, or folding, frequency. In the polynomial situation we have no such simple understanding of the effect of sampling.

It might appear that to calculate all the coefficients of a Fourier expansion would involve a great deal of computing, but the recently discovered fast Fourier transform (FFT) method requires about \( N \log N \) operations to fit \( N \) data points. This discovery has greatly increased the importance of Fourier approximation.
Most of our functions are not periodic, and so the Fourier series is of limited importance. For the general nonperiodic function there is a corresponding Fourier integral

\[ f(t) = \int_{-\infty}^{\infty} F(\sigma)e^{2\pi i \sigma t} \, d\sigma \]

\[ F(\sigma) = \int_{-\infty}^{\infty} f(t)e^{-2\pi i \sigma t} \, dt \]

The two functions \( f(t) \) and the transform \( F(\sigma) \) have the same information: one describes the function in the space of the variable \( t \), while the other describes it in the space of frequencies \( \sigma \). These two equivalent views are part of the reason that the Fourier integral gives such a useful approach to many problems.

When we resort to using a finite number of sample points, we again face aliasing, exactly the same effect as before. The folding frequency now provides a more natural barrier and leads to the concept of a "band-limited function," meaning that the frequencies in the function are confined to a band.

The idea of a band-limited function is mirrored very closely in real problems. A hi-fi system handles all the frequencies in a band and cuts off above and below rather sharply; the better the hi-fi system, the wider the band. The idea of a band of frequencies applies to most information transmission systems, servomechanisms, and feedback control situations. This means, among other things, that from the physics of the problem we can estimate the spacing we will need for our samples, and vice versa, that from the spacing we can estimate the frequency content of the solution.
With the aid of the Fourier integral, which closely parallels the Fourier series with its nice mathematical properties, we can show the effect of taking a finite slice of a function from a potentially infinitely long function. For example, the light from a pulsar or Cepheid variable star shines for many years; yet we observe it for a night or two and from that limited record try to estimate what the star is doing. It is often important to know the effect of the length of the observation on what we can hope to learn about the star. Similarly in other control problems, the length of the observation affects what we can see, and the Fourier integral enables us to understand this limitation.

Using this new tool of Fourier approximation, we can then look back at the polynomial approximation methods and see them in a new way. Once this new way becomes familiar, we can understand much more clearly what we were doing before. Exactly the same computations can be viewed in a new, more revealing light. This is especially true for the communications and control problems that tend to dominate our technological age.

Once this new way of looking at computing becomes familiar, it is natural to begin designing formulas to meet new criteria. In place of the discrete set of integers that were the exponents of the variable $x$ in polynomial approximation, we now have a continuous band of frequencies to use and examine. We can pick the formula that minimizes some property of this continuous error curve (in the frequency space to be sure). One popular method is to make the error curve have a minimax (Chebyshev) property.

This new approach is relevant to many design problems. For example, in designing a simulator for humans to use while training for airplane or space travel, we are interested in building the simulator so that it "feels right" to the human who is being trained. This to a first approximation means that the Fourier transform (more generally, the Laplace transform) of the simulator should be close to the transform of the real thing. It is less important that the simulator and real vehicle go exactly the same place than it is to "feel right." Thus we are no longer to judge the method of integrating a system of differential equations that describe the simulator by how well the solutions agree, but rather by how well the transforms agree—which is not the same thing! This new method of design is known as the method of zeros and poles, and unfortunately it involves classical network theory.

The role of digital communications systems is steadily increasing; more and more we are sampling the continuous analog signal that occurs naturally in the real world and converting it to a sequence of discrete digits. Thus we face not only sampling but quantization effects of the digitizing of the analog signal. This effect is like that of roundoff in many ways, but it is usually much more severe, and because it occurs at the beginning, it again limits us to what we can
hope to see of the original underlying physical phenomena. Without a good understanding of these limitations we are not likely to understand what the numbers coming out of the computer mean and do not mean.

This leads gradually to the design of digital filters, which filter digital signals much as the old analog filters were used in processing analog signals, with radio and television being a couple of familiar examples. The differences from the continuous signals are significant in the digital case, and it is the digital case that digital computers must use.

1.7 OTHER CLASSES OF FUNCTIONS USED IN APPROXIMATIONS

After polynomial and Fourier approximations, the exponential set of functions is most used. The three sets, polynomials, Fourier, and exponentials (together with combinations of the three), are invariant under a translation of the origin. This is an important property because in many situations there is no natural origin, and without this property the choice of the origin would affect the answer. For these three classes the answer is independent of the origin, though the particular set of coefficients used in the approximation will differ as the origin is chosen differently.

For the exponential functions there is the Laplace transform corresponding to the Fourier transform, but it has much more difficult properties from the computing point of view.

1.8 MISCELLANEOUS

When a problem has a singularity, as many practical problems do (often because of the mathematical idealization), then the structure of the singularity indicates the class of approximating functions to use, and the position gives the natural origin. The methods used in this case are similar to those of the three usual cases.

Sometimes the problem has a natural set of functions to use, and again the methods we have developed can be applied, though the details may get a bit messy in many cases.

Optimization occurs frequently in practice, and the person practicing numerical methods needs to know something about this rapidly growing field. Most simulations imply an optimization in the background—the simulation is being done to optimize some aspect of the situation.

A central idea of mathematics is linear independence. In computing it is
natural that this idea, which involves a yes no situation, becomes more vague. Clearly in computing there will be some degree of linear independence, and various forms for representing the same information will have varying degrees of linear independence. The idea is still in its infancy and needs a great deal more development, but it is clearly a central idea in computing.

One of the more difficult problems requiring an algorithm is finding the eigenvalues and eigenvectors of a matrix. Unfortunately, there is a great lack of understanding of what the problem actually is and of what the answers are to be used for, and there are no widely accepted methods for the general case. For the particular case of a symmetric (also for a Hermitian) matrix reasonably effective methods are known.

1.9 REFERENCES

The problem of supplying further references is a vexing one. The literature is rapidly changing when compared to the lifetime of a book, and as a result most references would soon be out of date and misleading. Furthermore, in a text like this where most chapters can, and some have, been expanded into whole books, there is little point in giving a lot of isolated references which will probably be ignored by most readers. We shall assume that a few standard textbooks are available and usually refer the reader to them for further information. The occasional reference to the literature is to amplify a point that is not in standard textbooks.