

## CHAPTER 1

---

# INTRODUCTORY CONCEPTS AND CALCULUS REVIEW

---

It is best to start this book with a question: What do we mean by “Numerical Methods and Analysis”? What kind of mathematics is this book about?

Generally and broadly speaking, this book covers the mathematics and methodologies that underlie the techniques of *scientific computation*. More prosaically, consider the button on your calculator that computes the sine of the number in the display. Exactly how does the calculator know that correct value? When we speak of using the computer to solve a complicated mathematics or engineering problem, exactly what is involved in making that happen? Are computers “born” with the knowledge of how to solve complicated mathematical and engineering problems? No, they are not. Mostly they are *programmed* to do it, and the programs implement algorithms that are based on the kinds of things we will talk about in this text.

Textbooks and courses in this area generally follow one of two main themes: Those titled “Numerical methods” tend to emphasize the implementation of the algorithms, perhaps at the expense of the underlying mathematical theory that explains why the methods work; those titled “Numerical analysis” tend to emphasize this underlying mathematical theory, perhaps at the expense of some of the implementation issues. The best approach, of course, is to properly mix the study of the algorithms and their implementation (“methods”) with the study of the mathematical theory (“analysis”) that supports them. This is the goal of the present text.

Whenever someone speaks of using a computer to design an airplane or predict the weather, or otherwise solve a complex science or engineering problem, that person is talking about using numerical methods and analysis. The problems and areas of endeavor that

use these kinds of techniques are continually expanding. For example, computational mathematics—another name for the material that we consider here—is now commonly used in the study of financial markets and investment structures, an area of study that does not ordinarily come to mind when we think of “scientific” computation. Similarly, the increasingly frequent use of computer-generated animation in film production is based on a heavy dose of spline approximations, which we introduce in §4.8.

There are a number of different ways to break down the subject into component parts. We will discuss the derivation of the algorithms, we will discuss the implementation of the algorithms, and we will also analyze the algorithms, mathematically, in order to learn how best to use them and how best to implement them. In our study of each technique, we will usually be concerned with two issues that often are in competition with each other:

- **Accuracy:** Very few of our computations will yield the exact answer to the problem, so we will have to understand how much error is made, and how to control (or even diminish) that error.
- **Efficiency:** Does the algorithm take an inordinate amount of computer time? This might seem to be an odd question to concern ourselves with—after all, computers are fast, right?—but there are slow ways to do things and fast ways to do things. All else being equal (it rarely is), we prefer the fast ways.

We say these two issues compete with each other because, generally speaking, the steps that can be taken to make an algorithm more accurate usually make it more costly, that is, less efficient.

There is a third issue of importance, but it does not become as evident as the others (although it is still present) until Chapter 6:

- **Stability:** Does the method produce similar results for similar data? If we change the data by a small amount, do we get vastly different results? If so, we say the method is unstable, and unstable methods tend to produce unreliable results. It is entirely possible to have an accurate method that is efficiently implemented and yet is horribly unstable; see §6.4.4 for an example of this.

## 1.1 BASIC TOOLS OF CALCULUS

### 1.1.1 Taylor’s Theorem

Computational mathematics does not require a large amount of background, but it does require a good knowledge of that background. The most important single result in numerical computations, from all of the calculus, is Taylor’s Theorem,<sup>1</sup> which we now state:

**Theorem 1.1 (Taylor’s Theorem with Remainder)** *Let  $f(x)$  have  $n+1$  continuous derivatives on  $[a, b]$  for some  $n \geq 0$ , and let  $x, x_0 \in [a, b]$ . Then,*

$$f(x) = p_n(x) + R_n(x)$$

<sup>1</sup>Brook Taylor (1685 – 1731) was educated at St. John’s College of Cambridge University, entering in 1701 and graduating in 1709. He published what we know as Taylor’s Theorem in 1715, although it appears that he did not appreciate its larger importance and he certainly did not bother with a formal proof. He was elected a member of the prestigious Royal Society of London in 1712.

Taylor acknowledged that his work was based on that of Newton and Kepler and others, but he did not acknowledge that the same result had been discovered by Johann Bernoulli and published in 1694. (But then Taylor discovered integration by parts first, although Bernoulli claimed the credit.)

for

$$p_n(x) = \sum_{k=0}^n \frac{(x-x_0)^k}{k!} f^{(k)}(x_0) \quad (1.1)$$

and

$$R_n(x) = \frac{1}{n!} \int_{x_0}^x (x-t)^n f^{(n+1)}(t) dt. \quad (1.2)$$

Moreover, there exists a point  $\xi_x$  between  $x$  and  $x_0$  such that

$$R_n(x) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi_x). \quad (1.3)$$

The point  $x_0$  is usually chosen at the discretion of the user, and is often taken to be 0. Note that the two forms of the remainder are equivalent: the “pointwise” form (1.3) can be derived from the “integral” form (1.2); see Problem 23.

Taylor’s Theorem is important because it allows us to represent, exactly, fairly general functions in terms of polynomials with a *known, specified, boundable* error. This allows us to replace, in a computational setting, these same general functions with something that is much simpler—a polynomial—and yet at the same time we are able to bound the error that is made. No other tool will be as important to us in this course as Taylor’s Theorem, so it is worth spending some time on it here at the outset.

The usual calculus treatment of Taylor’s Theorem should leave the student familiar with three particular expansions (for all three of these we have used  $x_0 = 0$ ):

$$\begin{aligned} e^x &= 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{n!}x^n + \frac{1}{(n+1)!}x^{n+1}e^{\xi_x} \\ &= \sum_{k=0}^n \frac{1}{k!}x^k + R_n(x), \\ \sin x &= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \dots + \frac{(-1)^n}{(2n+1)!}x^{2n+1} - \frac{(-1)^{n+1}}{(2n+3)!}x^{2n+3} \cos \xi_x \\ &= \sum_{k=0}^n \frac{(-1)^k}{(2k+1)!}x^{2k+1} + R_n(x), \\ \cos x &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \dots + \frac{(-1)^n}{(2n)!}x^{2n} - \frac{(-1)^{n+1}}{(2n+2)!}x^{2n+2} \cos \xi_x \\ &= \sum_{k=0}^n \frac{(-1)^k}{(2k)!}x^{2k} + R_n(x). \end{aligned}$$

(Strictly speaking, the index on the last two remainders should be  $2n+1$  and  $2n$ , because those are the exponents in the last terms of the expansion, but it is commonplace to present it as we did here.) In fact, Taylor’s Theorem provides us with our first and simplest example of an approximation and an error estimate. Consider the problem of approximating the exponential function on the interval  $[-1, 1]$ . Taylor’s Theorem tells us that we can represent  $e^x$  using a polynomial with a (known) remainder:

$$e^x = \underbrace{1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{n!}x^n}_{p_n(x), \text{ polynomial}} + \underbrace{\frac{1}{(n+1)!}x^{n+1}e^{\xi_x}}_{R_n(x), \text{ remainder}}$$

where  $c_x$  is an unknown point between  $x$  and 0. Since we want to consider the most general case, where  $x$  can be any point in  $[-1, 1]$ , we have to consider that  $c_x$  can be any point in  $[-1, 1]$ , as well. For simplicity, let's denote the polynomial by  $p_n(x)$ , and the remainder by  $R_n(x)$ , so that the above becomes

$$e^x = p_n(x) + R_n(x).$$

Suppose we want this approximation to be accurate to within  $10^{-6}$  in absolute error, i.e., we want

$$|e^x - p_n(x)| \leq 10^{-6}$$

for *all*  $x$  in the interval  $[-1, 1]$ . Note that if we can make  $|R_n(x)| \leq 10^{-6}$  for all  $x \in [-1, 1]$ , then we will have

$$|e^x - p_n(x)| = |R_n(x)| \leq 10^{-6}$$

so that the error in the approximation will be less than  $10^{-6}$ . The best way to proceed is to create a *simple* upper bound for  $|R_n(x)|$ , and then use that to determine the number of terms necessary to make this upper bound less than  $10^{-6}$ .

Thus we proceed as follows:

$$\begin{aligned} |R_n(x)| &= \frac{|x^{n+1} e^{c_x}|}{(n+1)!}, \\ &= \frac{|x|^{n+1} e^{c_x}}{(n+1)!}, \quad \text{because } e^z > 0 \text{ for all } z, \\ &\leq \frac{e^{c_x}}{(n+1)!}, \quad \text{because } |x| \leq 1 \text{ for all } x \in [-1, 1], \\ &\leq \frac{e}{(n+1)!}, \quad \text{because } e^{c_x} \leq e \text{ for all } x \in [-1, 1]. \end{aligned}$$

Thus, if we find  $n$  such that

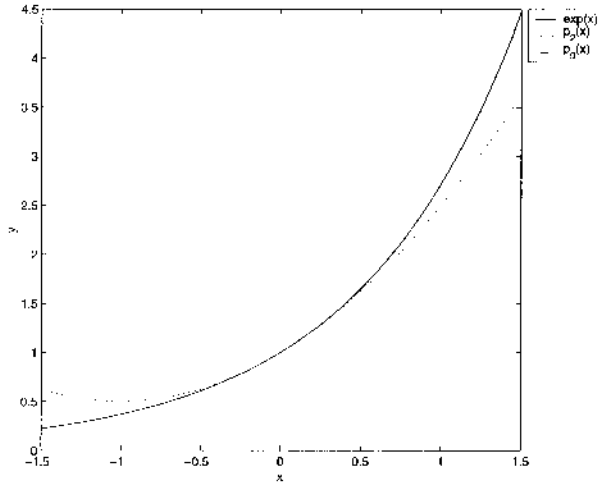
$$\frac{1}{(n+1)!} e \leq 10^{-6}$$

then we will have

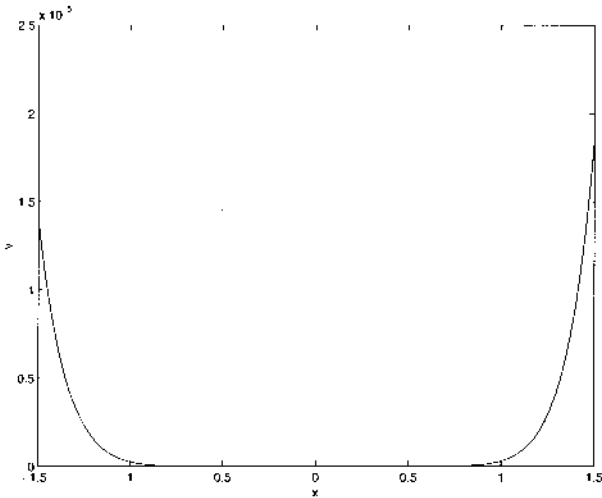
$$|e^x - p_n(x)| = |R_n(x)| \leq \frac{1}{(n+1)!} e \leq 10^{-6}$$

and we will *know* that the error is less than the desired tolerance, for *all* values of  $x$  of interest to us, i.e., all  $x \in [-1, 1]$ . A little bit of exercise with a calculator shows us that we need to use  $n = 9$  in order to get the desired accuracy. Figure 1.1 shows a plot of the exponential function  $e^x$ , the approximation  $p_9(x)$ , as well as the less accurate  $p_2(x)$ ; since it is impossible to distinguish by eye between the plots for  $e^x$  and  $p_9(x)$ , we have also provided Figure 1.2, which is a plot of the error  $e^x - p_9(x)$ ; note that we begin to lose accuracy as we get away from the interval  $[-1, 1]$ . This is not surprising. Since the Taylor polynomial is constructed to match  $f$  and its first  $n$  derivatives at  $x = x_0$ , it ought to be the case that  $p_n$  is a good approximation to  $f$  only when  $x$  is near  $x_0$ .

Here we are in the first section of the text, and we have already constructed our first approximation and proved our first theoretical result. Theoretical result? Where? Why, the error estimate, of course. The material in the previous several lines is a proof of the following result:



**Figure 1.1** Taylor approximation:  $e^x$ ,  $p_9(x) \approx e^x$  and  $p_2(x) \approx e^x$ . Note that  $e^x$  and  $p_9(x)$  are indistinguishable on this plot.



**Figure 1.2** Error in Taylor approximation:  $e^x - p_9(x)$ .

**Proposition 1.1** Let  $p_9(x)$  be the Taylor polynomial of degree 9 for the exponential function. Then, for all  $x \in [-1, 1]$ , the error in the approximation of  $p_9(x)$  to  $e^x$  is less than  $10^{-6}$ , i.e.,

$$|e^x - p_9(x)| \leq 10^{-6}$$

for all  $x \in [-1, 1]$ .

While this result is not of major significance to our work—the exponential function is approximated more efficiently by different means—it does illustrate one of the most important aspects of the subject. The result tells us, ahead of time, that we can approximate the exponential function to within  $10^{-6}$  accuracy using a specific polynomial, and this accuracy holds for all  $x$  in a specified interval. That is the kind of thing we will be doing throughout the text—constructing approximations to difficult computations that are accurate in some sense, and we *know* how accurate they are.

■ **EXAMPLE 1.1**

Let  $f(x) = \sqrt{x+1}$ ; then the second order Taylor polynomial (computed about  $x_0 = 0$ ) is computed as follows:

$$\begin{aligned} f(x_0) &= f(0) = 1; \\ f'(x) &= \frac{1}{2}(x+1)^{-1/2} \Rightarrow f'(x_0) = \frac{1}{2}; \\ f''(x) &= -\frac{1}{2} \times \frac{1}{2}(x+1)^{-3/2} \Rightarrow f''(x_0) = -\frac{1}{4}; \\ p_2(x) &= f(x_0) + (x-x_0)f'(x_0) + \frac{1}{2}(x-x_0)^2 f''(x_0) = 1 + \frac{1}{2}x - \frac{1}{8}x^2. \end{aligned}$$

The error in using  $p_2$  to approximate  $\sqrt{x+1}$  is given by  $R_2(x) = \frac{1}{3!}(x-x_0)^3 f'''(\xi_x)$ , where  $\xi_x$  is between  $x$  and  $x_0$ . We can simplify the error as follows:

$$\begin{aligned} |R_2(x)| &= \left| \frac{1}{3!}(x-x_0)^3 f'''(\xi_x) \right|, \\ &= \frac{1}{6}|x|^3 \left| \frac{3}{2} \times \frac{1}{2} \times \frac{1}{2}(\xi_x+1)^{-5/2} \right|, \\ &= \frac{1}{16}|x|^3 |\xi_x+1|^{-5/2}. \end{aligned}$$

If we want to consider this for all  $x \in [0, 1]$ , then, we observe that  $x \in [0, 1]$  and  $\xi_x$  between  $x$  and 0 imply that  $\xi_x \in [0, 1]$ , therefore

$$|\xi_x+1|^{-5/2} \leq |0+1|^{-5/2} = 1,$$

so that the upper bound on the error becomes  $|R_2(x)| \leq 1/16 = 0.0625$ , for all  $x \in [0, 1]$ . If we are only interested in  $x \in [0, \frac{1}{2}]$ , then the error is much smaller:

$$\begin{aligned} |R_2(x)| &= \frac{1}{16}|x|^3 |\xi_x+1|^{-5/2}, \\ &\leq \frac{1}{16}(1/2)^3, \\ &= \frac{1}{128}, \\ &= 0.0078125. \end{aligned}$$

■ EXAMPLE 1.2

Consider the problem of finding a polynomial approximation to the function  $f(x) = \sin \pi x$  that is accurate to within  $10^{-4}$  for all  $x \in [-\frac{1}{2}, \frac{1}{2}]$ , using  $x_0 = 0$ . A direct computation with Taylor's Theorem gives us that (note that the indexing here is a little different, because we want to take advantage of the fact that the Taylor polynomial here has only odd terms)

$$p_n(x) = \pi x - \frac{1}{6}\pi^3 x^3 + \frac{1}{120}\pi^5 x^5 + \cdots + (-1)^n \frac{1}{(2n+1)!} \pi^{2n+1} x^{2n+1},$$

$$R_n(x) = (-1)^{n+1} \frac{1}{(2n+3)!} \pi^{2n+3} x^{2n+3} \cos \xi_x.$$

Thus, the error in the approximation is

$$|f(x) - p_n(x)| = |R_n(x)| = \frac{(\pi x)^{2n+3}}{(2n+3)!} |\cos \xi_x|,$$

where  $\xi_x$  is between  $x$  and  $0$ ; this can be bounded for all  $x \in [-\frac{1}{2}, \frac{1}{2}]$  in the following fashion:

$$|R_n(x)| = \frac{|\pi x|^{2n+3}}{(2n+3)!} |\cos \xi_x| \leq \frac{(\pi/2)^{2n+3}}{(2n+3)!},$$

and some experimentation with a calculator shows that

$$|R_4(x)| \leq 0.3599 \times 10^{-5}, \quad |R_3(x)| \leq 0.1604 \times 10^{-3},$$

from which we conclude that  $n = 4$  will achieve the desired accuracy, thus we want

$$p_4(x) = \pi x - \frac{1}{6}\pi^3 x^3 + \frac{1}{120}\pi^5 x^5 - \frac{1}{5040}\pi^7 x^7 + \frac{1}{362880}\pi^9 x^9$$

as our approximation. Figure 1.3 shows the error between this polynomial and  $f(x)$  over the interval  $[-\frac{1}{2}, \frac{1}{2}]$ ; note that the error is much better than the predicted error, especially in the middle of the interval.

While the usual approach is to construct Taylor expansions by directly computing the necessary derivatives of the function  $f$ , sometimes a more subtle approach can be used.

Consider the problem of computing a Taylor series for the arctangent function,  $f(x) = \arctan x$ , about the point  $x_0 = 0$ . It won't take many terms before we get tired of trying to find a general expression for  $f^{(n)}$ . What do we do?

Recall, from calculus, that

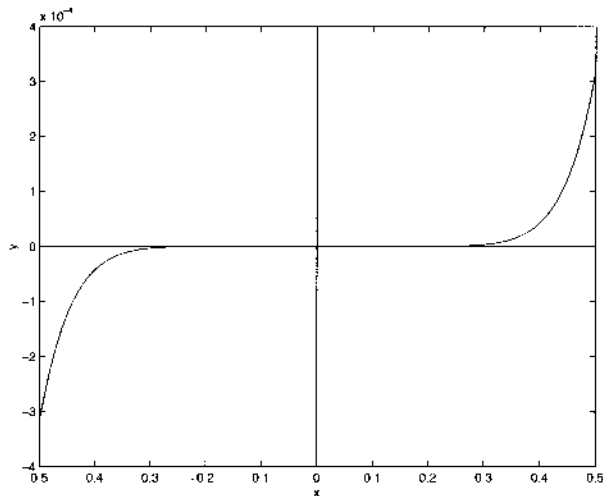
$$\arctan x = \int_0^x \frac{dt}{1+t^2},$$

so we can get the Taylor series for the arctangent from the Taylor series for  $(1+t^2)^{-1}$  by a simple integration. Now recall the summation formula for the geometric series:

$$\sum_{k=0}^n r^k = \frac{1-r^{n+1}}{1-r}.$$

If we substitute  $r = -t^2$  into this, and re-arrange terms a bit, we get

$$\frac{1}{1+t^2} = \sum_{k=0}^n (-1)^k t^{2k} + \frac{(-t^2)^{n+1}}{1-t^2}.$$



**Figure 1.3** Error in Taylor approximation to  $f(x) = \sin \pi x$  over  $[-\frac{1}{2}, \frac{1}{2}]$ .

Thus,

$$\begin{aligned} \arctan x &= \int_0^x \frac{dt}{1+t^2} \\ &= \int_0^x \left( \sum_{k=0}^n (-1)^k t^{2k} + \frac{(-t^2)^{n+1}}{1+t^2} \right) dt \\ &= \sum_{k=0}^n (-1)^k (2k+1)^{-1} x^{2k+1} + (-1)^{n+1} \int_0^x \frac{t^{2n+2}}{1+t^2} dt. \end{aligned}$$

This is known as Gregory's series<sup>2</sup> for the arctangent and was the basis for one of the early methods for computing  $\pi$ . (See Problems 14–16.)

### EXAMPLE 1.3

Let's use the Gregory series to determine the error in a ninth degree Taylor approximation to the arctangent function. Since  $2n+1=9$  implies that  $n=4$ , we have

$$R_9(x) = - \int_0^x \frac{t^{10}}{1+t^2} dt,$$

so that (assuming  $x \geq 0$ )

$$|R_9(x)| = \int_0^x \frac{t^{10}}{1+t^2} dt,$$

<sup>2</sup>James Gregory (1638–1675) was born in Aberdeen, Scotland, where he first went to university. He later (1664–1668) studied in Italy before returning to Britain. He published a work on optics and several works devoted to finding the areas under curves, including the circle. He knew about what we call Taylor series more than 40 years before Taylor published the theorem, and he might have developed the calculus before Newton did if he had been of a less geometric and more analytical frame of mind.

and we can bound the error as follows:

$$|R_9(x)| = \int_0^x \frac{t^{10}}{1+t^2} dt \leq \int_0^x t^{10} dt = \frac{1}{11} x^{11}.$$

So, for all  $x \in [-\frac{1}{2}, \frac{1}{2}]$  the remainder is bounded above by

$$|R_9(x)| \leq \frac{1}{11} (1/2)^{11} = 4.44 \times 10^{-5}.$$

Finally, we close this section with an illustration of a special kind of Taylor expansion that will be used again and again.

Consider the problem of expanding  $f(x+h)$  in a Taylor series, about the point  $x_0 = x$ . Here  $h$  is generally considered to be a small parameter. Direct application of Taylor's Theorem gives us the following:

$$\begin{aligned} f(x+h) &= f(x) + (|x+h-x|)f'(x) + \frac{1}{2}(|x+h-x|)^2 f''(x) \\ &+ \cdots + \frac{1}{n!}(|x+h-x|)^n f^{(n)}(x) \\ &+ \frac{1}{(n+1)!}(|x+h-x|)^{n+1} f^{(n+1)}(\xi) \\ &- f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots + \frac{1}{n!}h^n f^{(n)}(x) \\ &+ \frac{1}{(n+1)!}h^{n+1} f^{(n+1)}(\xi). \end{aligned}$$

This kind of expansion will be useful time and again in our studies.

### 1.1.2 Mean Value and Extreme Value Theorems

We need to spend some time reviewing other results from calculus that have an impact on numerical methods and analysis. All of these theorems are included in most calculus texts but are usually not emphasized as much as we will need here.

**Theorem 1.2 (Mean Value Theorem)** *Let  $f$  be a given function, continuous on  $[a, b]$  and differentiable on  $(a, b)$ . Then there exists a point  $\xi \in [a, b]$  such that*

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}. \quad (1.4)$$

In the context of calculus, the importance of this result might seem obscure, at best. However, from the point of view of numerical methods and analysis, the Mean Value Theorem (MVT) is probably second in importance only to Taylor's Theorem. Why? Well, consider a slightly reworked form of (1.4):

$$f(x_1) - f(x_2) = f'(\xi)(x_1 - x_2).$$

Thus, the MVT allows us to replace differences of function values with differences of argument values, if we scale by the derivative of the function. For example, we can use the MVT to tell us that

$$|\cos x_1 - \cos x_2| \leq |x_1 - x_2|,$$

because the derivative of the cosine is the sine, which is bounded by one in absolute value. Note also that the MVT is simply a special case of Taylor's Theorem, for  $n = 0$ .

Similar to the MVT is the Intermediate Value Theorem:

**Theorem 1.3 (Intermediate Value Theorem)** *Let  $f \in C([a, b])$  be given, and assume that  $W$  is a value between  $f(a)$  and  $f(b)$ , that is, either  $f(a) \leq W \leq f(b)$ , or  $f(b) \leq W \leq f(a)$ . Then there exists a point  $c \in [a, b]$  such that  $f(c) = W$ .*

This seems to be a very abstract result; it says that a certain point exists, but doesn't give us much information about its numeric value. (Besides which, we might ask, why do we even care that  $c$  exists?) But it is this very theorem that is the basis for our first method for finding the roots of functions, an algorithm called the bisection method. (See §3.1.) Moreover, this is the theorem that tells us that a continuous function is one whose graph can be drawn without lifting the pen off the paper, because it says that between any two function values, we have a point on the curve for all possible argument values. Sometimes very abstract results have very concrete consequences.

A related result is the Extreme Value Theorem, which is the basis for the max/min problems that are a staple of most beginning calculus courses.

**Theorem 1.4 (Extreme Value Theorem)** *Let  $f \in C([a, b])$  be given; then there exists a point  $m \in [a, b]$  such that  $f(m) \leq f(x)$  for all  $x \in [a, b]$ , and a point  $M \in [a, b]$  such that  $f(M) \geq f(x)$  for all  $x \in [a, b]$ . Moreover,  $f$  achieves its maximum and minimum values on  $[a, b]$  either at the endpoints  $a$  or  $b$ , or at a critical point.*

(The student should recall that a critical point is a point where the first derivative is either undefined or equal to zero. The theorem thus says that we have one of  $M = a$ ,  $M = b$ ,  $f'(M)$  doesn't exist, or  $f'(M) = 0$ , and similarly for  $m$ .)

There are other "mean value theorems," and we need to look at two in particular, as they will come up in our early error analysis.

**Theorem 1.5 (Integral Mean Value Theorem)** *Let  $f$  and  $g$  both be in  $C([a, b])$ , and assume further that  $g$  does not change sign on  $[a, b]$ . Then there exists a point  $\xi \in [a, b]$  such that*

$$\int_a^b g(t)f(t)dt = f(\xi) \int_a^b g(t)dt. \quad (1.5)$$

**Proof:** Since this result is not commonly covered in the calculus sequence, we will go ahead and prove it.

We first assume, without any loss of generality, that  $g(t) \geq 0$ ; the argument changes in an obvious way if  $g$  is negative (see Problem 26). Let  $f_M$  be the maximum value of the function on the interval,  $f_M = \max_{x \in [a, b]} f(x)$ , so that  $g(t)f(t) \leq g(t)f_M$  for all  $t \in (a, b)$ ; then

$$\int_a^b g(t)f(t)dt \leq \int_a^b g(t)f_M dt = f_M \int_a^b g(t)dt.$$

Similarly, we have

$$\int_a^b g(t)f(t)dt \geq \int_a^b g(t)f_m dt = f_m \int_a^b g(t)dt$$

where  $f_m = \min_{x \in [a, b]} f(x)$  is the minimum value of  $f$  on the interval. Since  $g$  does not change sign on the interval of integration, the only way that we can have

$$\int_a^b g(x) dx = 0$$

is if  $g$  is identically zero on the interval, in which case the theorem is trivially true, since both sides of (1.5) would be zero. So we can assume that

$$\int_a^b g(x) dx \neq 0.$$

Now define

$$W = \frac{\int_a^b g(t)f(t) dt}{\int_a^b g(t) dt}$$

so that we have

$$f_m \leq W \leq f_M.$$

By the Extreme Value Theorem, there is a point  $M \in [a, b]$  such that  $f(M) = f_M$ , and similarly there is a point  $m \in [a, b]$  such that  $f(m) = f_m$ . Therefore

$$f(m) \leq W \leq f(M)$$

and we can apply the Intermediate Value Theorem to establish that there is a point  $\xi$  in the interval defined by  $m$  and  $M$  such that  $f(\xi) = W$ ; but this implies (1.5) (why?) and we are done. •

This result will be useful for simplifying some error estimates for numerical integration rules in Chapter 5. A related result is the Discrete Average Value Theorem:

**Theorem 1.6 (Discrete Average Value Theorem)** *Let  $f \in C([a, b])$  and consider the sum*

$$S = \sum_{k=1}^n a_k f(x_k)$$

where each point  $x_k \in [a, b]$ , and the coefficients satisfy

$$a_k \geq 0, \quad \sum_{k=1}^n a_k = 1.$$

Then there exists a point  $\eta \in [a, b]$  such that  $f(\eta) = S$ , i.e.,

$$f(\eta) = \sum_{k=1}^n a_k f(x_k).$$

**Proof:** The proof is similar in spirit and technique to the previous one. We quickly have that

$$S = \sum_{k=1}^n a_k f(x_k) \leq f_M \sum_{k=1}^n a_k = f_M$$

and similarly  $S \geq f_m$ , where  $f_M$  and  $f_m$  are defined as in the previous proof. Now define  $W = S$  and proceed as before to get that there is a point  $\eta \in [a, b]$  such that  $f(\eta) = S$ . •

All three of the mean value theorems are useful to us in that they allow us to simplify certain expressions that will occur in the process of deriving error estimates for our approximate computations.

### Exercises:

1. What is the third-order Taylor polynomial for  $f(x) = \sqrt{x+1}$ , about  $x_0 = 0$ ?
2. What is the sixth-order Taylor polynomial for  $f(x) = \sqrt{1+x^2}$ , using  $x_0 = 0$ ?
3. Show that the third-order Taylor polynomial for  $f(x) = (x+1)^{-1}$ , about  $x_0 = 0$ , is

$$p_3(x) = 1 - x + x^2 - x^3.$$

4. Given that

$$R(x) = \frac{|x|^6}{6!} e^\xi$$

for  $x \in [-1, 1]$ , where  $\xi$  is between  $x$  and 0, find an upper bound for  $|R|$ , valid for all  $x \in [-1, 1]$ , that is independent of  $x$  and  $\xi$ .

5. Repeat the above, but this time require that the upper bound be valid only for all  $x \in [-\frac{1}{2}, \frac{1}{2}]$ .

6. Given that

$$R(x) = \frac{|x|^4}{4!} \left( \frac{-1}{1+\xi} \right)$$

for  $x \in [-\frac{1}{2}, \frac{1}{2}]$ , where  $\xi$  is between  $x$  and 0, find an upper bound for  $|R|$ , valid for all  $x \in [-\frac{1}{2}, \frac{1}{2}]$ , that is independent of  $x$  and  $\xi$ .

7. What is the fourth-order Taylor polynomial for  $f(x) = 1/(x+1)$ , about  $x_0 = 0$ ?
8. What is the fourth-order Taylor polynomial for  $f(x) = 1/x$ , about  $x_0 = 1$ ?
9. Use a Taylor polynomial to find an approximate value for  $\sqrt{e}$  that is accurate to within  $10^{-3}$ .
10. Find the Taylor polynomial of third-order for  $\sin x$ , using:
  - (a)  $x_0 = \pi/6$ ;
  - (b)  $x_0 = \pi/4$ ;
  - (c)  $x_0 = \pi/2$ .
11. For each function below construct the third-order Taylor polynomial approximation, using  $x_0 = 0$ , and then estimate the error by computing an upper bound on the remainder, over the given interval.
  - (a)  $f(x) = e^{-x}$ ,  $x \in [0, 1]$ ;
  - (b)  $f(x) = \ln(1+x)$ ,  $x \in [-1, 1]$ ;
  - (c)  $f(x) = \sin x$ ,  $x \in [0, \pi]$ ;
  - (d)  $f(x) = \ln(1+x)$ ,  $x \in [-1/2, 1/2]$ ;
  - (e)  $f(x) = 1/(x+1)$ ,  $x \in [-1/2, 1/2]$ .

12. Construct a Taylor polynomial approximation that is accurate to within  $10^{-3}$ , over the indicated interval, for each of the following functions, using  $x_0 = 0$ .

- (a)  $f(x) = \sin x$ ,  $x \in [0, \pi]$ ;  
 (b)  $f(x) = e^{-x}$ ,  $x \in [0, 1]$ ;  
 (c)  $f(x) = \ln(1+x)$ ,  $x \in [-1/2, 1/2]$ ;  
 (d)  $f(x) = 1/(x+1)$ ,  $x \in [-1/2, 1/2]$ ;  
 (e)  $f(x) = \ln(1+x)$ ,  $x \in [-1, 1]$ .

13. Repeat the above, this time with a desired accuracy of  $10^{-6}$ .

14. Since

$$\frac{\pi}{4} = \arctan 1,$$

we can estimate  $\pi$  by estimating  $\arctan 1$ . How many terms are needed in the Gregory series for the arctangent to approximate  $\pi$  to 100 decimal places? 1,000? *Hint:* Use the error term in the Gregory series to predict when the error gets sufficiently small.

15. Elementary trigonometry can be used to show that

$$\arctan(1/239) = 4 \arctan(1/5) - \arctan(1).$$

This formula was developed in 1706 by the English astronomer John Machin. Use this to develop a more efficient algorithm for computing  $\pi$ . How many terms are needed to get 100 digits of accuracy with this form? How many terms are needed to get 1,000 digits? *Historical note:* Until 1961 this was the basis for the most commonly used method for computing  $\pi$  to high accuracy.

16. In 1896 a variation on Machin's formula was found:

$$\arctan(1/239) = \arctan(1) - 6 \arctan(1/8) - 2 \arctan(1/57),$$

and this began to be used in 1961 to compute  $\pi$  to high accuracy. How many terms are needed when using this expansion to get 100 digits of  $\pi$ ? 1,000 digits?

17. What is the Taylor polynomial of order 4 for  $f(x) = x^4 + 1$ , using  $x_0 = 0$ ? Simplify as much as possible.
18. What is the Taylor polynomial of order 3 for  $f(x) = x^4 + 1$ , using  $x_0 = 0$ ?
19. What is the Taylor polynomial of order 2 for  $f(x) = x^3 + x$ , using  $x_0 = 1$ ?
20. What is the Taylor polynomial of order 3 for  $f(x) = x^3 + x$ , using  $x_0 = 1$ ? Simplify as much as possible.
21. Let  $p(x)$  be an arbitrary polynomial of degree less than or equal to  $n$ . What is its Taylor polynomial of degree  $n$ , about an arbitrary  $x_0$ ?
22. The Fresnel integrals are defined as

$$C(x) = \int_0^x \cos(\pi t^2/2) dt$$

and

$$S(x) = \int_0^x \sin(\pi t^2/2) dt.$$

Use Taylor expansions to find approximations to  $C(x)$  and  $S(x)$  that are  $10^{-4}$  accurate for all  $x$  with  $|x| \leq \frac{1}{2}$ . *Hint:* Substitute  $x = \pi t^2/2$  into the Taylor expansions for the cosine and sine.

23. Use the Integral Mean Value Theorem to show that the “pointwise” form (1.3) of the Taylor remainder (usually called the *Lagrange* form) follows from the “integral” form (1.2) (usually called the *Cauchy* form).
24. For each function in Problem 11, use the Mean Value Theorem to find a value  $M$  such that

$$|f(x_1) - f(x_2)| \leq M|x_1 - x_2|$$

is valid for all  $x_1, x_2$  in the interval used in Problem 11.

25. A function is called *monotone* on an interval if its derivative is strictly positive or strictly negative on the interval. Suppose  $f$  is continuous and monotone on the interval  $[a, b]$ , and  $f(a)f(b) < 0$ ; prove that there is exactly one value  $\alpha \in [a, b]$  such that  $f(\alpha) = 0$ .
26. Finish the proof of the Integral Mean Value Theorem (Theorem 1.5) by writing up the argument in the case that  $g$  is negative.
27. Prove Theorem 1.6, providing all details.
28. Let  $c_k > 0$  be given,  $1 \leq k \leq n$ , and let  $x_k \in [a, b]$ ,  $1 \leq k \leq n$ . Then, use the Discrete Average Value Theorem to prove that, for any function  $f \in C([a, b])$ ,

$$\frac{\sum_{k=1}^n c_k f(x_k)}{\sum_{k=1}^n c_k} = f(\xi)$$

for some  $\xi \in [a, b]$ .

29. Discuss, in your own words, whether or not the following statement is true: “The Taylor polynomial of degree  $n$  is the best polynomial approximation of degree  $n$  to the given function near the point  $x_0$ .”



## 1.2 ERROR, APPROXIMATE EQUALITY, AND ASYMPTOTIC ORDER NOTATION

We have already talked about the “error” made in a simple Taylor series approximation. Perhaps it is time we got a little more precise.

### 1.2.1 Error

If  $A$  is a quantity we want to compute and  $A_h$  is an approximation to that quantity, then the error is the difference between the two:

$$\text{error} = A - A_h;$$

the absolute error is simply the absolute value of the error:

$$\text{absolute error} = |A - A_h|; \tag{1.6}$$

and the relative error normalizes by the absolute value of the exact value:

$$\text{relative error} = \frac{|A - A_h|}{|A|}, \tag{1.7}$$

where we assume that  $A \neq 0$ .

Why do we need two different measures of error? Consider the problem of approximating the number

$$x = e^{-16} = 0.1125351747 \times 10^{-6}.$$

Because  $x$  is so small, the absolute error in  $y = 0$  as an approximation to  $x$  is also small. In fact,  $|x - y| < 1.2 \times 10^{-7}$ , which is decent accuracy in many settings. However, this “approximation” is clearly not a good one.

On the other hand, consider the problem of approximating

$$z = e^{16} = 0.8886110521 \times 10^7.$$

Because  $z$  is so large, the absolute error in almost any approximation will be large, even though almost all of the digits are matched. For example, if we take  $w = 0.8886110517 \times 10^7$ , then we have  $|z - w| = 4 \times 10^{-3}$ , hardly very small, even though  $y$  matches  $z$  to 9 decimal digits.

The point is that relative error gives a measure of the number of correct digits in the approximation. Thus

$$\left| \frac{x - y}{x} \right| \approx 1,$$

which tells us that not many digits are matched in that example, whereas

$$\left| \frac{z - w}{z} \right| = \frac{4 \times 10^{-3}}{0.8886110521 \times 10^7} = 0.4501 \times 10^{-9},$$

which shows that about nine digits are correct. Generally speaking, using a relative error protects us from misjudging the accuracy of an approximation because of scale extremes (very large or very small numbers). As a practical matter, however, we sometimes are not able to obtain an error estimate in the relative sense.

In the definitions (1.6) and (1.7), we have used the subscript  $h$  to suggest that, in general, the approximation depends (in part, at least) on a parameter. For the most part, our computations will indeed be constructed this way, usually with either a real parameter  $h$  which tends toward zero, or with an integer parameter  $n$  which tends toward infinity. So we might want to think in terms of one of the two cases

$$\lim_{h \rightarrow 0} A_h = A$$

or

$$\lim_{n \rightarrow \infty} A_n = A.$$

In actual applied problems there are, of course, lots of sources of error: simple mistakes, measurement errors, modeling errors, etc. We are concerned here only with the computational errors caused by the need to construct computable approximations. The common terminology is *truncation error* or *approximation error* or *mathematical error*.

### 1.2.2 Notation: Approximate Equality

If two quantities are approximately equal to each other, we will use the notation “ $\approx$ ” to denote this relationship, as in

$$A \approx B.$$

This is an admittedly vague notion. Is  $0.99 \approx 1$ ? Probably so. Is  $0.8 \approx 1$ ? Maybe not. We will almost always use the  $\approx$  symbol in the sense of one of the two contexts outlined previously, of a parameterized set of approximations converging to a limit. Note that the definition of limit means that

$$\lim_{h \rightarrow 0} A_h = A \quad \Rightarrow \quad A_h \approx A$$

for all  $h$  “sufficiently small” (and similarly for the case of  $A_n \rightarrow A$  as  $n \rightarrow \infty$ , for  $n$  “sufficiently large”). For example, one way to write the definition of the derivative of a function  $y = f(x)$  is as follows:

$$\lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} = f'(x).$$

We therefore conclude that, for  $h$  small enough,

$$\frac{f(x+h) - f(x)}{h} \approx f'(x).$$

Moreover, approximate equality does satisfy the transitive, symmetric, and reflexive properties of what abstract algebra calls an “equivalence relation”:

$$\begin{aligned} A \approx B, \quad B \approx C &\Rightarrow A \approx C, \\ A \approx B &\Rightarrow B \approx A, \\ A &\approx A. \end{aligned}$$

Consequently, we can manipulate approximate equalities much like ordinary equalities (i.e., equations). We can solve them, integrate both sides, etc.

Despite its vagueness, approximate equality is a useful notion to have around in a course devoted to approximations.

### 1.2.3 Notation: Asymptotic Order

Another notation of use is the so-called “Big O” notation, more formally known as “asymptotic order” notation. Suppose we have a value  $y$  and a family of values  $\{y_h\}$ , each of which approximates this value,

$$y \approx y_h$$

for small values of  $h$ . If we can find a constant  $C > 0$ , independent of  $h$ , such that

$$|y - y_h| \leq C\beta(h) \tag{1.8}$$

for all  $h$  sufficiently small, then we say that

$$y - y_h = \mathcal{O}(\beta(h)), \text{ as } h \rightarrow 0.$$

meaning that  $y - y_h$  is “on the order of”  $\beta(h)$ . Here  $\beta(h)$  is a function of the parameter  $h$ , and we assume that

$$\lim_{h \rightarrow 0} \beta(h) = 0.$$

The utility of this notation is that it allows us to concentrate on the important issue in the approximation—the way that the error  $y - y_h$  depends on the parameter  $h$ , which is determined by  $\beta(h)$ —while ignoring the unimportant details like the precise size of the constant,  $C$ . The usage is similar if we have a sequence  $x_n$  which approximates a given value  $x$  for large values of  $n$ . If

$$|x - x_n| \leq C\beta(n) \tag{1.9}$$

for all  $n$  sufficiently large, then we say that

$$x - x_n = \mathcal{O}(\beta(n)), \text{ as } n \rightarrow \infty.$$

The formal definitions are as follows.

**Definition 1.1 (Asymptotic Order Notation)** For a given value  $y$ , let  $\{y_h\}$  be a set of values parameterized by  $h$ , which we assume is small, such that  $y_h \approx y$  for small  $h$ . If there exists a positive function  $\beta(h)$ ,  $\beta(h) \rightarrow 0$  as  $h \rightarrow 0$ , and a constant  $C > 0$ , such that for all  $h$  sufficiently small,

$$|y - y_h| \leq C\beta(h),$$

then we say that

$$y - y_h = \mathcal{O}(\beta(h)).$$

Similarly, if  $\{y_n\}$  is a set of values parameterized by  $n$ , which we assume is large, such that  $y_n \approx y$  for large  $n$ , and if there exists a positive function  $\beta(n)$ ,  $\beta(n) \rightarrow 0$  as  $n \rightarrow \infty$ , and a constant  $C > 0$ , such that for all  $n$  sufficiently large,

$$|y - y_n| \leq C\beta(n),$$

then we say that

$$y - y_n = \mathcal{O}(\beta(n)).$$

■ **EXAMPLE 1.4**

Let

$$A = \int_0^\infty e^{-2x} dx,$$

$$A_n = \int_0^n e^{-2x} dx.$$

Simple calculus shows that  $A = \frac{1}{2}$  and  $A_n = \frac{1}{2} - \frac{1}{2}e^{-2n}$ , so that we have  $A - A_n = \mathcal{O}(e^{-2n})$ . Here  $\beta(n) = e^{-2n}$ .

### EXAMPLE 1.5

Another example—and many such, in fact—can be generated from Taylor's Theorem. We have that

$$\cos x = 1 - \frac{1}{2}x^2 \cos \xi_x$$

where  $\xi_x$  is between  $x$  and 0. Since  $\cos \xi_x$  is bounded by 1 in absolute value, we easily have that

$$|\cos h - 1| \leq \frac{1}{2}h^2$$

so we can write  $\cos h - 1 = \mathcal{O}(h^2)$ . Similarly, we can write that  $e^h = 1 + h + \mathcal{O}(h^2)$ . In both these cases we have  $\beta(h) = h^2$ .

The following theorem shows how the asymptotic order relationship can be manipulated.

**Theorem 1.7** Let  $y = y_h + \mathcal{O}(\beta(h))$  and  $z = z_h + \mathcal{O}(\gamma(h))$ , with  $b\beta(h) > \gamma(h)$  for all  $h$  near zero. Then

$$\begin{aligned} y + z &= y_h + z_h + \mathcal{O}(\beta(h) + \gamma(h)), \\ y - z &= y_h - z_h + \mathcal{O}(\beta(h)), \\ Ay &= Ay_h + \mathcal{O}(\beta(h)). \end{aligned}$$

In the last equation,  $A$  is an arbitrary constant, independent of  $h$ .

**Proof:** We simply show that each relationship above satisfies (1.8) for some constant  $C$ . For example,

$$\begin{aligned} |(y + z) - (y_h + z_h)| &\leq |y - y_h| + |z - z_h|, \\ &\leq C_1\beta(h) + C_2\gamma(h), \\ &\leq C(\beta(h) + \gamma(h)), \end{aligned}$$

where  $C = \max(C_1, C_2)$ . Thus  $y + z = y_h + z_h + \mathcal{O}(\beta(h) + \gamma(h))$ . Moreover, since  $b\beta(h) > \gamma(h)$ , we also have that

$$\begin{aligned} |(y + z) - (y_h + z_h)| &\leq C(\beta(h) + \gamma(h)), \\ &\leq C(\beta(h) + b\beta(h)), \\ &\leq C(1 + b)\beta(h). \end{aligned}$$

Also,

$$\begin{aligned} |Ay - Ay_h| &= A|y - y_h|, \\ &\leq C_1A\beta(h), \\ &= C\beta(h), \end{aligned}$$

so that  $Ay = Ay_h + \mathcal{O}(\beta(h))$ . •

A similar result holds for  $u_n \approx u$  with  $u = u_n + \mathcal{O}(\beta(n))$ , and  $v_n \approx v$  with  $v = v_n + \mathcal{O}(\gamma(n))$ , etc.

**EXAMPLE 1.6**

We close this section with a simple example that illustrates the utility of the  $\mathcal{O}$  notation. Consider the combination of function values

$$D = -f(x + 2h) + 4f(x + h) - 3f(x)$$

where  $f$  is assumed to be continuous and smooth, and  $h$  is a (small) parameter. We can use Taylor's Theorem, together with the definition of the  $\mathcal{O}$  notation, to write

$$f(x + h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \mathcal{O}(h^3)$$

and

$$f(x + 2h) = f(x) + 2hf'(x) + 2h^2 f''(x) + \mathcal{O}(h^3).$$

Therefore,

$$\begin{aligned} D &= -f(x + 2h) + 4f(x + h) - 3f(x) \\ &= -(f(x) + 2hf'(x) + 2h^2 f''(x) + \mathcal{O}(h^3)) \\ &\quad + 4(f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \mathcal{O}(h^3)) - 3f(x) \\ &= (-1 + 4 - 3)f(x) + (-2h + 4h)f'(x) + (-2h^2 + 2h^2)f''(x) + \mathcal{O}(h^3) \\ &= 2hf'(x) + \mathcal{O}(h^3). \end{aligned}$$

If we then solve this for  $f'(x)$  we get

$$f'(x) = \frac{-f(x + 2h) + 4f(x + h) - 3f(x)}{2h} + \mathcal{O}(h^2),$$

where we have used the fact that  $\mathcal{O}(h^3)/h = \mathcal{O}(h^2)$  (see Problem 10); thus we can use the expression on the right as an approximation to the derivative, and the remainder will be bounded by a constant times  $h^2$ . See §2.2 and §4.5 for more on approximations to the derivative. This particular approximation is derived again, by alternate means, in §4.5.

Note that we have rather quickly obtained an approximation to the first derivative, along with some notion of the error—it behaves proportionally to  $h^2$ —by using the  $\mathcal{O}$  notation.

## Exercises:

1. Use Taylor's Theorem to show that

$$\sqrt{1+x} = 1 + \frac{1}{2}x + \mathcal{O}(x^2)$$

for  $x$  sufficiently small.

2. Use Taylor's Theorem to show that  $e^x = 1 + x + \mathcal{O}(x^2)$  for  $x$  sufficiently small.
3. Use Taylor's Theorem to show that  $\frac{1-\cos x}{x} = \frac{1}{2}x + \mathcal{O}(x^3)$  for  $x$  sufficiently small.
4. Show that

$$\sin x = x + \mathcal{O}(x^3).$$

5. Use Taylor's Theorem to show that

$$(1+x)^{-1} = 1 - x + x^2 + \mathcal{O}(x^3)$$

for  $x$  sufficiently small.

6. Recall the summation formula

$$1 + r + r^2 + r^3 + \cdots + r^n = \sum_{k=0}^n r^k = \frac{1 - r^{n+1}}{1 - r}.$$

Use this to prove that

$$\sum_{k=0}^n r^k = \frac{1}{1-r} + \mathcal{O}(r^{n+1}).$$

*Hint:* What is the *definition* of the  $\mathcal{O}$  notation?

7. Use the above result to show that 9 terms are all that is needed to compute

$$S = \sum_{k=0}^{\infty} e^{-k}$$

to within  $10^{-4}$  absolute accuracy.

8. Recall the summation formula

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}.$$

Use this to show that

$$\sum_{k=1}^n k = \frac{1}{2}n^2 + \mathcal{O}(n).$$

9. State and prove the version of Theorem 1.7 which deals with relationships of the form  $x - x_n + \mathcal{O}(\beta(n))$ .
10. Use the definition of  $\mathcal{O}$  to show that if  $y = y_n + \mathcal{O}(h^p)$ , then  $hy = hy_n + \mathcal{O}(h^{p-1})$ .
11. Show that if  $a_n = \mathcal{O}(n^p)$  and  $b_n = \mathcal{O}(n^q)$ , then  $a_n b_n = \mathcal{O}(n^{p+q})$ .
12. Suppose that  $y = y_h + \mathcal{O}(\beta(h))$  and  $z = z_h + \mathcal{O}(\beta(h))$ , for  $h$  sufficiently small. Does it follow that  $y - z = y_h - z_h$  (for  $h$  sufficiently small)?
13. Show that

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2)$$

for all  $h$  sufficiently small. *Hint:* Expand  $f(x \pm h)$  out to the fourth-order terms.

14. Explain, in your own words, why it is necessary that the constant
- $C$
- in (1.8) be independent of
- $h$
- .

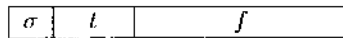
### 1.3 A PRIMER ON COMPUTER ARITHMETIC

We need to spend some time reviewing how the computer actually does arithmetic. The reason for this is simple: Computer arithmetic is generally inexact, and while the errors that are made are very small, they can accumulate under some circumstances and actually dominate the calculation. Thus we need to understand computer arithmetic well enough to anticipate and deal with this phenomenon.

Most computer languages use what is called *floating point arithmetic*. While the details will differ from machine to machine, the basic idea is the same. Every number is represented using a (fixed, finite) number of binary digits, usually called *bits*. A typical implementation would represent the number in the form:

$$x = \sigma \times f \times \beta^{t-p}.$$

Here  $\sigma$  is the sign of the number ( $\pm 1$ ), denoted by a single bit;  $f$  is the mantissa or fraction;  $\beta$  is the base of the internal number system, usually binary ( $\beta = 2$ ) or hexadecimal ( $\beta = 16$ ), although other systems have sometimes been used;  $t$  is the (shifted) exponent, i.e., the value that is actually stored; and  $p$  is the shift required to recover the actual exponent. (Shifting in the exponent is done to avoid the need for a sign bit in the exponent itself.) The number would be stored by storing only the values of  $\sigma$ ,  $f$ , and  $t$ . The standard way to represent the computer word containing a floating point number is as follows:



To keep things simple, we will use a base 2 representation for our floating point examples here.

The total number of bits devoted to the number would be fixed by the computer architecture, and the fraction and exponent would each be allowed a certain number of bits. For example, a common situation would allow 32 bits for the entire word,<sup>3</sup> assigned as follows: 24 bits for the fraction, 7 bits for the exponent, and a single bit for the sign.

Note that this imposes limits on the numbers that can be represented. For example, a 7 bit exponent means that

$$0 \leq t \leq 127.$$

In order to allow for an equal range of positive and negative exponents, the shift should be taken to be  $p = 63$  so that

$$-63 \leq t - p \leq 64.$$

Attempts to create larger exponents result in what is called an *overflow*. Attempts to create smaller exponents result in *underflow*<sup>4</sup>. The fraction is also limited in size by the number of bits available:

$$0 \leq f \leq \sum_{k=1}^{24} 2^{-k} = 1 - 2^{-24}.$$

<sup>3</sup>A "word" is the largest unit of computer storage. Usually a word consists of two or more *bytes* that themselves consist of a certain number of bits, typically 8.

<sup>4</sup>Most modern computers adhere to the so-called IEEE standard for arithmetic, which uses a kind of "extended floating point" number system. In addition to ordinary numbers, the IEEE standard allows for results Inf (infinity) and NaN (not a number), and includes rules for manipulating with ordinary floating point numbers and these special values. For example,  $x/\text{Inf}$  yields 0 as a result, while  $x/0$  yields plus or minus Inf, depending on the sign of  $x$ . Most manipulations with NaN return NaN as their result. In older arithmetic schemes, if an overflow or divide by zero occurred, program execution usually terminated.

In practice, most architectures assume that the fraction is *normalized* to be between  $\beta^{-1}$  and 1; any leading zeroes would be dropped and the exponent adjusted accordingly.<sup>5</sup> Thus we actually have

$$\frac{1}{2} \leq f \leq \sum_{k=1}^{24} 2^{-k} = 1 - 2^{-24}.$$

The errors in computer arithmetic come about because the floating point numbers are allowed only a fixed number of bits, and not every number can be exactly represented in a fixed number of bits. The common name for the error that is caused by this finite representation is *rounding error*.

Let's consider the simple addition of two numbers

$$x = 0.1, \quad y = 0.00003$$

assuming that they are represented in the scheme outlined above. The exact answer, of course, is

$$z = x + y = 0.10003.$$

We note first that neither of these numbers can be exactly represented in our scheme. The best we can do is<sup>6</sup>

$$\begin{aligned} \tilde{x} &= 0.00011001\ 10011001\ 10011001\ 100_2 \\ &= 0.09999999910395\dots \end{aligned}$$

and

$$\begin{aligned} \tilde{y} &= 0.00000000\ 00000001\ 11110111\ 01010001\ 0000010_2 \\ &= 0.0000299999992421\dots \end{aligned}$$

Thus we have (admittedly, small) errors being made even before the addition occurs.

In our floating point scheme these two numbers would be stored as

$$\begin{aligned} \hat{x} &= .11001100\ 11001100\ 11001100_2 \times 2^{60-63}, \\ \hat{y} &= .11110111\ 10101000\ 10000010_2 \times 2^{48-63}. \end{aligned}$$

Because the two numbers are of somewhat different sizes, a normalization is required in order to get equal exponents. One way to do this—the precise details would depend on the particular computer architecture—would give us

$$\begin{aligned} \tilde{\tilde{x}} &= 1100\ 11001100\ 11001100\ 11000000\ 00000000_2 \times 2^{-39} \\ \tilde{\tilde{y}} &= 0000\ 00000000\ 11110111\ 10101000\ 10000010_2 \times 2^{-39} \end{aligned}$$

and we can now add the mantissas to get the sum, which is first written as

$$\tilde{\tilde{w}} = 1100\ 11001101\ 11001000\ 01101000\ 10000010_2 \times 2^{-39}.$$

Note that the fraction here is too long to be stored with only 24 bits. What is done with the extra bits? Depending on the machine, they are either thrown away, regardless of their

<sup>5</sup>Some architectures take advantage of this assumption to avoid actually storing that leading bit—all the basic arithmetic algorithms are written to assume an extra leading 1—and thus they are able to get 25 bits of information into 24 bits of storage space.

<sup>6</sup>Note that we have used the subscript "2" to indicate that the number should be interpreted as a base-2 fraction.

size (chopping), or the result would be rounded up or down, depending on size (rounding). Rounding is more accurate, of course, but chopping is faster. If we round, then we have

$$\begin{aligned}\tilde{z} &= 11001100\ 11011100\ 10000111_2 \times 2^{-27} \\ &= .11001100\ 11011100\ 10000111_2 \times 2^{60-63} \\ &= 0.100029997527599\dots\end{aligned}$$

and the error is

$$|z - \tilde{z}| = 0.24724\dots \times 10^{-8}.$$

Let's next assume that our machine uses chopping, in which case we end up with

$$\begin{aligned}\tilde{z} &\dots 11001100\ 11011100\ 10000110_2 \times 2^{-27} \\ &\dots .11001100\ 11011100\ 10000110_2 \times 2^{60-63} \\ &\dots 0.1000299900770\dots\end{aligned}$$

and a final error of

$$|z - \tilde{z}| = 0.992298\dots \times 10^{-8}.$$

Note that the chopping error is indeed larger than the error when we used rounding. Similar errors would of course occur with the other arithmetic operations.

The difference here—whether we chop or round—is indeed very small, and we might be tempted to ignore it as being too small to worry about. In fact, this is usually the case. But it is possible for the effects of different rounding errors to combine in such a way as to dominate and ruin a calculation. We can illustrate this point with simple decimal arithmetic so long as we insist on using only a small number of digits.

Consider an 8 digit approximation to  $a = e^{(1/100)^2} = 0.99990001$  and a similar approximation to  $b = e^{-(1/1000)^2} = 0.99999900$ . By construction, both of these numbers are accurate to 8 decimal digits. What about their difference  $c = a - b = -0.00009899$ ? How many accurate digits do we have here? The answer is: only *four*. Because we were subtracting two nearly equal numbers, we lost a great deal of accuracy. This phenomenon is called *subtractive cancellation*. If we had started with more accurate approximations, then the difference would contain more accurate digits; try this by looking at the 16 digit values

$$\begin{aligned}a &= 0.9999000049998333, \\ b &= 0.9999990000005000, \\ c &= -0.0000989950006667.\end{aligned}$$

The result  $c$  is now accurate to 12 digits.

To see how subtractive cancellation can destroy almost all the accuracy in a calculation, consider the quantity

$$D = (f(x_1) - f(x_2)) - (f(x_2) - f(x_3))$$

where  $f(x) = e^{-x^2}$  and  $x_1 = 999/10,000$ ,  $x_2 = 1/10$ ,  $x_3 = 1,001/10,000$ . Then the calculation, as organized above and done in 8 digit arithmetic, yields  $D = -0.1 \times 10^{-7}$ . But when we do it in 16 digit arithmetic, we get  $D = -0.194049768 \times 10^{-7}$ . The 8 digit calculation had *no* accurate digits. Subtractive cancellation is therefore something to avoid as much as possible, and to be aware of when it is unavoidable.

Sometimes the problem with rounding error can be eliminated by increasing the precision of the computation. Traditionally, floating-point arithmetic systems used a single word for each number ("single precision") by default, and a second word could be used ("double precision") by properly specifying the type of data format to be used. Most languages now use double word arithmetic by default. Sometimes the entire extra word is used to extend the length of the fraction; sometimes the length of the exponent is also extended as well. *If changing the precision of a calculation dramatically changes the results, then it is almost certain that the computation is being seriously affected by rounding errors.*

Another example of subtractive cancellation occurs with the evaluation of the function

$$f(x) = \frac{e^x - 1}{x}$$

for values of  $x$  near zero. L'Hôpital's Rule<sup>7</sup> can be easily used to show that

$$\lim_{x \rightarrow 0} f(x) = 1$$

and Taylor's Theorem can be used to show that

$$f(x) - 1 \approx \frac{1}{2}x + \mathcal{O}(x^2)$$

for small  $x$ , but the evaluation of  $f$  for  $x$  near 0 will exhibit subtractive cancellation that is amplified by the division by  $x$  (since  $x$  is small precisely when the subtractive cancellation is worst). Table 1.1 shows the results of computing  $f$  using single precision and double precision floating point arithmetic. Note that the error in the single precision results increases dramatically for  $x \leq 2^{-12} \approx 1/4096$ , which is not that small a number. A second threshold of inaccuracy is reached at around  $x = 2^{-24}$ . Note that the use of double precision arithmetic defers the onset and lessens the severity of the error, but does not eliminate it entirely, as the last few rows show. (Even though the limiting value of  $f$  is 1, the error in the computed value for  $x = 2^{-30}$  is still non-zero;  $f(2^{-30}) - 1 = 0.1657 \times 10^{-9}$ . While this error would be acceptable in single precision arithmetic, it is not acceptable for double precision arithmetic.) How do we fix this?

One approach would be to use Taylor's Theorem:

$$f(x) = \frac{(1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots + \frac{1}{n!}x^n + \frac{1}{(n+1)!}x^{n+1}e^{c_x}) - 1}{x} \\ = 1 + \frac{1}{2}x + \frac{1}{6}x^2 + \dots + \frac{1}{n!}x^{n-1} + \frac{1}{(n+1)!}x^n e^{c_x}$$

where  $c_x$  is between  $x$  and 0; the value of  $n$  would depend on our required accuracy. We would thus define  $f$ , for computational purposes, as

$$f(x) = \begin{cases} 1 + \frac{1}{2}x + \frac{1}{6}x^2 + \dots + \frac{1}{n!}x^{n-1}, & |x| \text{ close to } 0, \\ x^{-1}(e^x - 1), & \text{otherwise.} \end{cases}$$

If we wanted more accuracy we would use more terms in the Taylor expansion.

<sup>7</sup>Guillaume François Antoine Marquis de L'Hôpital (1661–1704) was not trained in mathematics, but took it up after resigning from the military due to poor eyesight. He studied with Johann Bernoulli in the 1690's, and in 1692 published what is generally regarded to be the first calculus textbook, *Analyse des infiniment petits pour l'intelligence des lignes courbes*. What we know as "L'Hôpital's Rule" first appears in this book, and is probably actually due to Bernoulli.

**Table 1.1** Illustration of subtractive cancellation, using  $f(x) = \frac{e^x - 1}{x}$ .

$k$	$x_k - 2^{-k}$	Single Precision	Double Precision
1	0.50000000000000E+00	0.12974424362183E+01	0.12974425414003D+01
2	0.25000000000000E+00	0.11361017227173E+01	0.11361016667510D+01
3	0.12500000000000E+00	0.10651874542236E+01	0.10651876245346D+01
4	0.62500000000000E-01	0.10319118499756E+01	0.10319113426858D+01
5	0.31250000000000E-01	0.10157890319824E+01	0.10157890399713D+01
6	0.15625000000000E-01	0.10078506469727E+01	0.10078533495479D+01
7	0.78125000000000E-02	0.10039215087891E+01	0.10039164424253D+01
8	0.39062500000000E-02	0.10019531250000E+01	0.10019556706170D+01
9	0.19531250000000E-02	0.10009765625000E+01	0.10009771985934D+01
10	0.97656250000000E-03	0.10004882812500E+01	0.10004884402344D+01
11	0.48828125000000E-03	0.10002441406250E+01	0.10002441803663D+01
12	0.24414062500000E-03	0.10000000000000E+01	0.10001220802469D+01
13	0.12207031250000E-03	0.10000000000000E+01	0.10000610376392D+01
14	0.61035156250000E-04	0.10000000000000E+01	0.10000305182002D+01
15	0.30517578125000E-04	0.10000000000000E+01	0.10000152589419D+01
16	0.15258789062500E-04	0.10000000000000E+01	0.10000076294382D+01
17	0.76293945312500E-05	0.10000000000000E+01	0.10000038146973D+01
18	0.38146972656250E-05	0.10000000000000E+01	0.10000019073486D+01
19	0.19073486328125E-05	0.10000000000000E+01	0.10000009536743D+01
20	0.95367431640625E-06	0.10000000000000E+01	0.10000004768372D+01
21	0.47683715820312E-06	0.10000000000000E+01	0.10000002384186D+01
22	0.23841857910156E-06	0.10000000000000E+01	0.10000001192093D+01
23	0.11920928955078E-06	0.10000000000000E+01	0.10000000596046D+01
24	0.59604644775391E-07	0.00000000000000E+00	0.10000000298023D+01
25	0.29802322387695E-07	0.00000000000000E+00	0.10000000149012D+01
26	0.14901161193848E-07	0.00000000000000E+00	0.10000000000000D+01
27	0.7450580596238E-08	0.00000000000000E+00	0.10000000000000D+01
28	0.37252902984619E-08	0.00000000000000E+00	0.10000000000000D+01
29	0.18626451492310E-08	0.00000000000000E+00	0.10000000000000D+01
30	0.93132257461548E-09	0.00000000000000E+00	0.10000000000000D+01

We close this section with a definition. In a floating point computer system, there will exist many nonzero numbers  $x$  such that

$$1 + x = 1$$

within computer precision. For instance, in the 32 bit implementation outlined at the beginning of the section, it is clear that this will hold for any  $x < 2^{-24}$ , since  $1 + 2^{-24}$  in binary will require 25 bits of storage, and we only have 24 to work with. We define the *machine rounding unit*, or *machine epsilon*,  $u$ , to be the largest such number:

**Definition 1.2 (Machine epsilon)** *The machine epsilon (alternately, the machine rounding unit),  $u$ , is the largest floating point number  $x$  such that  $x + 1$  cannot be distinguished from 1 on the computer:*

$$u = \max\{x \mid 1 + x = 1, \text{ in computer arithmetic}\}.$$

It is possible to compute the machine epsilon based on knowledge of the floating point number system on the computer in question.

### ■ EXAMPLE 1.7

Suppose we want to compute the machine epsilon for a simple three digit decimal computer which rounds (i.e., a machine that does decimal arithmetic, keeps only three digits, and rounds its results). Thus we can write any number  $x$  that is stored on the computer as  $x = d_1d_2d_3 \times 10^t$ , where  $t$  is the exponent and  $d_1d_2d_3$  represent the decimal digits of the fraction. Now, based on the definition of machine epsilon, we know that  $x_1 = 1.00 \times 10^{-2}$  is too large, because

$$1 + x_1 = 1.00 + 0.01 = 1.01 \neq 1.00.$$

On the other hand, we know that  $x_2 = 1.00 \times 10^{-3}$  is certainly small enough (perhaps too small) because we have

$$1 + x_2 = 1.00 + 0.001 = 1.001 \Rightarrow 1.00 = 1.00.$$

Thus, the computer cannot distinguish between 1 and  $1 + x_2$ . But  $u$  is the *largest* such number, so we have to look a bit further. It's not a matter of using the right formula, it is really a matter of experimentation and trial and error. We have

$$1 + 0.002 = 1.002 \rightarrow 1.00 = 1,$$

so we know that  $x_3 = 2.00 \times 10^{-3}$  is small enough; the same argument would apply to  $x_4 = 3.00 \times 10^{-3}$  and  $x_5 = 4.00 \times 10^{-3}$ . But

$$1 + 0.005 = 1.005 \rightarrow 1.01 / 1$$

(because the machine rounds its computations), thus  $x_6 = 5.00 \times 10^{-3}$  is too large. But it is just barely too large, as any smaller number would have resulted in the sum being rounded down to 1.00. Thus we want the next smaller number *within the floating point number system*, i.e., we want  $u = 4.99 \times 10^{-3}$ . This gives us

$$1.00 + 0.00499 = 1.00499 \rightarrow 1.00 = 1,$$

and it is clear that any larger number that we can represent in our floating point system would result in the sum being rounded up to  $1.01 / 1$ . The student ought to be able to show, now, that if the machine chops instead of rounds, that  $u = 9.99 \times 10^{-3}$ .

There are a number of interesting consequences of the existence of the machine epsilon. For example, consider the numbers

$$a = 1, b = u, c = u.$$

If we want to add up these numbers we quickly learn that the way we organize the calculation matters. For example,

$$(a + b) + c = (1 + u) + u = 1 + u = 1$$

whereas

$$a + (b + c) = 1 + 2u / 1.$$

From a technical point of view, this means that the associative law of arithmetic (for addition) does not hold in floating point arithmetic systems. In other words, *the order in which we do operations sometimes matters*.

It is possible to estimate the machine epsilon by constructing a loop which adds increasingly small numbers to 1, and only terminates this when the result cannot be distinguished from 1.

The basic result on computer arithmetic is the following.

**Theorem 1.8 (Computer Arithmetic Error)** *Let  $*$  denote any of the basic binary operations (addition, subtraction, multiplication, or division), and let  $fl(x)$  denote the floating point value of  $x$ . Then there exists a constant  $C > 0$  such that, for all  $x$  and  $y$ ,*

$$|x * y - fl(x * y)| \leq C u |x * y|.$$

Thus the computer value of  $x * y$  is relatively accurate to within  $\mathcal{O}(u)$ .

The point of this section is that we should now be aware that computer arithmetic is not the 100% reliable thing we might have thought it was. However, the errors that crop up do tend to appear only when extremely large or small numbers are involved (exceptions do occur, so we have to be careful), and are themselves very small. Moreover, rounding errors tend to cancel themselves out over the long run. Rounding error and related effects are things we have to watch out for and be aware of, but they are *usually* dominated by the mathematical error that is made in constructing the approximations. Exceptions to this rule of thumb do exist, however, as we shall see in §2.2 and §4.11.1.

## Exercises:

- In each problem below,  $A$  is the exact value, and  $A_h$  is an approximation to  $A$ . Find the absolute error and the relative error.
  - $A = \pi, A_h = 22/7$ ;
  - $A = e, A_h = 2.71828$ ;
  - $A = \frac{1}{6}, A_h = 0.1666$ ;
  - $A = \frac{1}{6}, A_h = 0.1667$ .
- Perform the indicated computations in each of three ways: (i) Exactly; (ii) Using 3 digit decimal arithmetic, with chopping; (iii) Using 3 digit decimal arithmetic, with rounding. For both approximations, compute the absolute error and the relative error.

- (a)  $\frac{1}{6} + \frac{1}{10}$ ;  
 (b)  $\frac{1}{7} + \left(\frac{1}{10} + \frac{1}{9}\right)$ ;  
 (c)  $\left(\frac{1}{7} + \frac{1}{10}\right) + \frac{1}{9}$ ;  
 (d)  $\frac{1}{6} \times \frac{1}{10}$ .

3. For each function below explain why a naive construction will be susceptible to significant rounding error (for  $x$  near certain values), and explain how to avoid this error.

(a)  $f(x) = (1-x)^{-1}(\ln x - \sin \pi x)$ ;

(b)  $f(x) = x^{-1}(1 - \cos x)$ ;

(c)  $f(x) = (\sqrt{x+9} - 3)x^{-1}$ ;

(d)  $f(x) = (e^{1+x} - e^{1-x})(2x)^{-1}$ ;

(e)  $f(x) = (\cos(\pi + x) - \cos \pi)x^{-1}$ .

4. For  $f(x) = (e^x - 1)/x$ , how many terms in a Taylor expansion are needed to get single precision accuracy (7 decimal digits) for all  $x \in [0, \frac{1}{2}]$ ? How many terms are needed for double precision accuracy (14 decimal digits) over this same range?

5. Using single precision arithmetic only, carry out each of the following computations, using first the form on the left side of the equals sign, then using the form on the right side, and compare the two results. Comment on what you get in light of the material in §1.3.

(a)  $(x + \epsilon)^3 - 1 = x^3 + 3x^2\epsilon + 3x\epsilon^2 + \epsilon^3 - 1$ ,  $x = 1.0$ ,  $\epsilon = 0.000001$ .

(b)  $-b + \sqrt{b^2 - 2c} = 2c(-b + \sqrt{b^2 - 2c})^{-1}$ ,  $b = 1,000$ ,  $c = \pi$ .

6. Consider the sum

$$S = \sum_{k=0}^m e^{-14(1-e^{-0.05k})}$$

where  $m = 2 \times 10^5$ . Again using only single precision, compute this two ways: First, by summing in the order indicated in the formula; second, by summing *backwards*, that is, starting with the  $k = 200,000$  term and ending with the  $k = 0$  term. Compare your results and comment upon them.

7. Using the computer of your choice, find three values  $a$ ,  $b$ , and  $c$ , such that

$$(a + b) + c \neq a + (b + c).$$

Repeat using your pocket calculator.

8. Assume we are using 3-digit decimal arithmetic. For  $\epsilon = 0.0001$ ,  $a_1 = 5$ , compute

$$a_2 = a_0 + \left(\frac{1}{\epsilon}\right) a_1$$

for  $a_0$  equal to each of 1, 2, and 3. Comment.

9. Let  $\epsilon \leq u$ . Explain, in your own words, why the computation

$$a_2 = a_0 + \left(\frac{1}{c}\right) a_1$$

is potentially rife with rounding error. (Assume that  $a_0$  and  $a_1$  are of comparable size.) *Hint:* See Problem 8.

10. Using the computer and language of your choice, write a program to estimate the machine epsilon.
11. We can compute  $e^{-x}$  using Taylor polynomials in two ways, either using

$$e^{-x} \approx 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \dots$$

or using

$$e^{-x} \approx \frac{1}{1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots}$$

Discuss, in your own words, which approach is more accurate. In particular, which one is more (or less) susceptible to rounding error?

12. What is the machine epsilon for a computer that uses binary arithmetic, 24 bits for the fraction, and rounds? What if it chops?
13. What is the machine epsilon for a computer that uses *octol* (base 8) arithmetic, assuming it retains 8 octol digits in the fraction?



## 1.4 A WORD ON COMPUTER LANGUAGES AND SOFTWARE

In the early 1970's the standard computer language for scientific computation was FORTRAN, with Algol being perhaps the second choice. BASIC, in many incarnations, was also a possibility. By the late 1980's, Pascal had entered the fray and FORTRAN was considered passé in some quarters, especially since various easy-to-use integrated environment packages for Pascal were being marketed to personal computer users. By the 1990's, Pascal was fading away, but we now had C, or C++, or even Java. And FORTRAN, despite the predictions of many and the desires of a few, was still with us. In addition, the increased power of personal computers meant that software packages such as MATLAB or MathCAD or Maple or Mathematica might be used to do scientific computation. (MATLAB, in particular, has had a tremendous influence on scientific computation.)

In short, if you don't like the present state of affairs with regard to computing languages, wait around a little while—it will change.

Given the inevitable change in computing languages, this text tries to a great extent to avoid the entire issue by being as "language neutral" as possible. Most examples will be given in a generic pseudo-code. The idea is to make the text as independent as possible of anyone's prejudices about languages, and also to pay attention to the notion that the student should be comfortable in as many different languages as possible—even though most scientific programming *today* might be done in MATLAB, it is still the case that there

is a lot of computer code (called *legacy code*) that is still being used and that was written in FORTRAN or Pascal or Algol or C.

Finally, the more involved algorithms will be presented in more of an “outline” style, rather than a line-by-line of code style.

There exist a number of sources for good mathematical software. Traditionally, two of the best sources were the IMSL and NAG libraries, collections of FORTRAN routines for a wide variety of computational tasks. More specialized packages have also been developed, notably QUADPACK (numerical integration), LINPACK (linear algebra), EISPACK (eigenvalue methods), LAPACK (an updated package that combines and actually replaces LINPACK and EISPACK), and others. A repository of public domain mathematics software is maintained at NETLIB.<sup>8</sup> Recently, of course, there has been the development of commercial packages like MATLAB, MathCAD, Maple, Mathematica, etc.

Despite the wide availability of general purpose mathematical software, it is still important for students to learn how to write and (most important) debug their own codes. For this reason many of the exercises in this text involve computer programming.

The text assumes that students are familiar with the use of elementary packages (such as Maple or Mathematica) for producing simple plots of functions. Whenever the exercises call for graphs or plots to be produced, it is assumed that such modern technology will be used. Students should also feel free (with their instructor’s permission, of course) to use Maple or Mathematica to simplify some of the more involved manipulations in the exercises.

## 1.5 SIMPLE APPROXIMATIONS

We have already used Taylor series to construct a simple approximation to the exponential function. Here we will do a similar but slightly more involved example to reinforce the basic ideas and also to illustrate the usage of the asymptotic order notation introduced above.

The *error function* occurs often in probability theory and other areas of applied mathematics (the solution of heat conduction problems, for example). It is defined by an integral:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

It is not possible to evaluate this integral by means of the fundamental theorem of calculus; there is no elementary anti-derivative for  $e^{-t^2}$ . (In Chapter 5 we will derive techniques that can be applied to directly approximate the integral.) Here we will use Taylor’s Theorem to approximate the integrand as a polynomial, and exactly integrate that polynomial.

*This is a fundamental idea in numerical methods: When confronted with a computation which cannot be done exactly, we often replace the relevant function with something simpler which approximates it, and carry out the computation exactly on the simple approximation.*

We might be tempted to compute the Taylor approximation to  $e^{-t^2}$  by appealing directly to the formula (1.1), but this will lead to a lot of unnecessary work. We know that

$$\begin{aligned} e^x &= 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{k!}x^k + \frac{x^{k+1}}{(k+1)!}e^c \\ &= p_k(x) + R_k(x) \end{aligned}$$

<sup>8</sup>On the World Wide Web at <http://www.netlib.org>.

where

$$p_k(x) = \sum_{i=0}^k \frac{x^i}{i!}$$

and

$$R_k(x) = \frac{x^{k+1}}{(k+1)!} e^c.$$

Therefore we can get the expansion and error for  $e^{-t^2}$  by a simple substitution:

$$e^{-t^2} = p_k(-t^2) + R_k(-t^2).$$

Thus we have

$$\begin{aligned} \operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x p_k(-t^2) dt + \frac{2}{\sqrt{\pi}} \int_0^x R_k(-t^2) dt \\ &= \frac{2}{\sqrt{\pi}} \left( \sum_{i=0}^k \frac{(-1)^i x^{2i+1}}{(2i+1)i!} \right) + \frac{2}{\sqrt{\pi}} \int_0^x R_k(-t^2) dt. \end{aligned}$$

For simplicity define the polynomial that approximates the error function by  $q_k$ :

$$\begin{aligned} q_k(x) &= \frac{2}{\sqrt{\pi}} \left( \sum_{i=0}^k \frac{(-1)^i x^{2i+1}}{(2i+1)i!} \right) \\ &= \frac{2}{\sqrt{\pi}} \left( x - \frac{1}{3}x^3 + \frac{1}{10}x^5 - \frac{1}{42}x^7 + \dots + \frac{(-1)^k x^{2k+1}}{(2k+1)k!} \right) \end{aligned}$$

so that we have

$$\operatorname{erf}(x) - q_k(x) = \frac{2}{\sqrt{\pi}} \int_0^x R_k(-t^2) dt.$$

We want to simplify this error and produce an error bound, so we will also set

$$E_k(x) = \frac{2}{\sqrt{\pi}} \int_0^x R_k(-t^2) dt.$$

Thus,

$$\operatorname{erf}(x) - q_k(x) = E_k(x). \quad (1.10)$$

To simplify and bound the error, we note that

$$\begin{aligned} E_k(x) &= \frac{2}{\sqrt{\pi}} \int_0^x R_k(-t^2) dt, \\ &= \frac{2}{\sqrt{\pi}} \int_0^x \frac{(-t^2)^{k+1}}{(k+1)!} e^c dt, \\ &= \frac{2(-1)^{k+1}}{(k+1)! \sqrt{\pi}} \int_0^x t^{2k+2} e^c dt, \end{aligned}$$

where  $c$  depends on  $t$ , and  $-t^2 \leq c \leq 0$ . Now, since both functions in the integrand above are positive, we can apply the Integral Mean Value Theorem to get

$$\int_0^x t^{2k+2} e^c dt = e^\xi \int_0^x t^{2k+2} dt = e^\xi \frac{x^{2k+3}}{2k+3}$$

for some  $\xi$  between 0 and  $-x^2$ . Thus

$$E_k(x) = \frac{2(-1)^{k+1}x^{2k+3}}{(2k+3)(k+1)!\sqrt{\pi}}e^\xi. \quad (1.11)$$

This error statement, although applicable only to a very elementary problem, nonetheless contains all of the common features that will appear in the error estimates for more realistic algorithms applied to more complicated problems. Thus it is worthwhile discussing it before we proceed to simplify it further.

Let us write the error in a more structured form:

$$E_k(x) = \left(\frac{2}{\sqrt{\pi}}\right) \left(\frac{(-1)^{k+1}x^{2k+3}}{(2k+3)(k+1)!}\right) e^\xi - C\delta_k(x)M$$

where

$$C = \frac{2}{\sqrt{\pi}}, \quad \delta_k(x) = \frac{(-1)^{k+1}x^{2k+3}}{(2k+3)(k+1)!}, \quad M = e^\xi.$$

This divides the error into three distinct parts:

1. A raw numeric constant,  $C$ ;
2. An expression depending on the computational parameters ( $k, x$ ) of the problem; this is  $\delta_k(x)$ ;
3. A factor that depends on the function or its derivatives, evaluated at an indeterminate, unknown point; this is  $M$ .

We are most interested in the second part of the error. The raw numeric constant is usually of less interest, since we cannot do anything to make it smaller. The function-dependent part of the error is of some concern to us in more complicated approximations; here we will simply note that we are not able to actually compute that value in most cases, so we resort to upper bounds.

It is the parameter-dependent part of the error that determines convergence and accuracy and how fast we achieve either of them, and that is, at least partly, under our control. In fact, we might use the asymptotic order notation to write (1.10) as

$$\operatorname{erf}(x) = q_k(x) + \mathcal{O}(\delta_k(x)).$$

Alternately, we could use the approximate equality notation to write

$$\operatorname{erf}(x) \approx q_k(x)$$

where it is understood (perhaps only implicitly) that this approximation is valid only for  $k$  large or for  $x$  small. We might want to simplify the error a little more by removing one of the variables from  $\delta_k(x)$ . For example, if we were only interested in values of  $x$  between 0 and 2, then we could easily establish that

$$|\delta_k(x)| \leq \frac{2^{2k+3}}{(2k+3)(k+1)!}$$

for all  $x \in [0, 2]$ . Moreover, the  $2k+3$  factor in the denominator is not going to affect the error bound nearly as much as the factorial, so we can further simplify to get

$$|\delta_k(x)| \leq \frac{8}{5} \frac{2^{2k}}{(k+1)!} = \frac{8}{5} \frac{4^k}{(k+1)!}$$

for all  $k \geq 1$ . Thus we can write

$$\operatorname{erf}(x) - q_k(x) = \mathcal{O}\left(\frac{1^k}{(k+1)!}\right) \quad 0 \leq x \leq 2.$$

This extra simplification has indeed increased our estimate of the error, but only slightly, and not in a manner that ignores the most important factors in the convergence of the approximation—the factorial and the power. The benefit we get from this slightly increased error estimate is the ability to quickly and easily gauge the accuracy of an approximation using a specified number of terms. For example, we have that the error in a 20 term approximation is on the order of  $2.2 \times 10^{-8}$ , whereas a 10 term approximation is accurate only to within about 0.026.

## Exercises:

1. Consider the error (1.11) in approximating the error function. If we restrict ourselves to  $k \leq 3$ , then over what range of values of  $x$  is the approximation accurate to within  $10^{-3}$ ?
2. If we are interested only in  $x \in [0, \frac{1}{2}]$ , then how many terms in the error function approximation do we need to take to get an accuracy of  $10^{-4}$ ?
3. Repeat the above for  $x \in [0, 1]$ .
4. Assume that  $x \in [0, 1]$  and write the error in the approximation to the error function using the asymptotic order notation.
5. Construct a Taylor approximation for

$$f(x) = \int_0^x t^p e^{-t^2} dt, \quad 0 \leq p < 1, \quad x \in \left[0, \frac{1}{2}\right]$$

that is accurate to within  $10^{-3}$  for all values of  $p$  in the indicated range.

6. Repeat the above for

$$f(x) = \int_0^x e^{-t^2} dt, \quad x \in \left[-\frac{1}{2}, \frac{1}{2}\right].$$

7. For

$$f(x) = \int_0^x t^{-1} \sin t dt, \quad x \in [-\pi/4, \pi/4],$$

construct a Taylor approximation that is accurate to within  $10^{-4}$  over the indicated interval.

8. Does it make a difference in the previous problem if we restrict  $p$  to  $p \in [0, \frac{1}{2}]$ ?
9. What is the error in a Taylor polynomial of degree 4 for  $f(x) = \sqrt{x}$  using  $x_0 = 9/16$ , for all  $x \in [1/4, 1]$ ?
10. What is the error in the Taylor polynomial of degree 5 for  $f(x) = 1/x$ , using  $x_0 = 3/4$ , for  $x \in [\frac{1}{2}, 1]$ ?
11. How many terms must be taken in the above to get an error of less than  $10^{-2}$ ?  $10^{-4}$ ?

## 1.6 APPLICATION: APPROXIMATING THE NATURAL LOGARITHM

In this section we will put together many of the basic ideas from previous sections to construct a reasonable approximation to the natural logarithm function. The primary tool will be Taylor's Theorem, and our goal will be to produce an approximation to the logarithm that is accurate to within  $\epsilon = 10^{-16}$ .

We first observe that if we can assume that  $\tau = \ln 2$  is known to arbitrary precision, then we really only need to construct a logarithm approximation that is valid over a short interval. This follows because of the way that the computer stores numbers. Since any  $z$  is stored as  $z = f \cdot 2^\beta$  for  $\frac{1}{2} \leq f \leq 1$  and some  $\beta$ , we have

$$\ln z = \ln f + \beta \ln 2.$$

Since  $f \in [\frac{1}{2}, 1]$ , we will get the best results by choosing the center of the expansion to be  $x_0 = \frac{3}{4}$ . This is a little unusual, but not unprecedented. The Taylor expansion for the logarithm is then (you ought to verify this)

$$\begin{aligned} \ln x &= \ln x_0 + \frac{x - x_0}{x_0} - \frac{1}{2} \left( \frac{x - x_0}{x_0} \right)^2 + \dots + (-1)^n \frac{1}{n} \left( \frac{x - x_0}{x_0} \right)^n \\ &+ (-1)^n \int_{x_0}^x (x - t)^{n-1} t^{-n} dt. \end{aligned} \quad (1.12)$$

Since the remainder here is a little more complicated than in the usual case, let's look at it carefully. We have

$$R_n(x) = (-1)^n \int_{x_0}^x (x - t)^n t^{-n-1} dt$$

so that

$$|R_n(x)| = \left| \int_{x_0}^x (x - t)^n t^{-n-1} dt \right|.$$

We now take upper bounds using the elementary fact that

$$\left| \int_a^b f(x) dx \right| \leq |b - a| \max_{x \in [a, b]} |f(x)|.$$

Thus,

$$|R_n(x)| \leq |x - x_0| \max_t \left| \frac{(x - t)^n}{t^{n+1}} \right|.$$

Now, we know that  $x_0 = \frac{3}{4}$ , that  $x \in [\frac{1}{2}, 1]$  and that  $t$  is between  $x$  and  $x_0$ . It follows, then, that

$$|x - x_0| \leq \frac{1}{4}$$

and that

$$\left| \frac{(x - t)^n}{t^{n+1}} \right| = |t|^{-1} \left| \frac{x}{t} - 1 \right|^n \leq 2 \left| \frac{x - t}{t} \right|^n.$$

At this point we have to proceed carefully. We want to bound the function

$$g(t) = \frac{x - t}{t}$$

in absolute value. What do we know about  $t$ ? Since  $t$  is between  $x$  and  $x_0$ , we have either  $x_0 \leq t \leq x$  or  $x \leq t \leq x_0$ . We have  $g'(t) = -x/t^2$  so there are no critical points (since  $x \neq 0$ ). Therefore, the Extreme Value Theorem tells us that the maximum and minimum of  $g$  occurs at the end points, that is for  $t = x$  and for  $t = x_0$ . Since  $g(x) = 0$ , we clearly have  $|g(t)| \leq |g(x_0)|$ . Thus, substituting into the previous inequality, we have

$$\left| \frac{(x-t)^n}{t^{n+1}} \right| \leq 2 \left| \frac{x-x_0}{x_0} \right|^n \leq 2 \left| \frac{\frac{1}{4}}{\frac{3}{4}} \right|^n.$$

Therefore,

$$|R_n(x)| \leq \frac{1}{2} \left( \frac{1}{3} \right)^n,$$

from which we conclude that  $n = 33$  is sufficient to guarantee that  $|R_n(x)| < 10^{-16}$  for all  $x \in [\frac{1}{3}, 1]$ . It thus requires a 33 degree polynomial to approximate the logarithm in this fashion, along with an accurate representation for  $\ln 2$  and  $\ln \frac{3}{4}$ .

Problem 5 asks you to implement this as a subprogram and check it against the intrinsic natural logarithm function on your computer.

Can this be improved? Yes, it is possible to construct an equally accurate logarithm approximation that uses fewer computations. Problem 6 asks you to look into this by using a clever combination of logarithm expansions that results in faster convergence.

## Exercises:

- Write each of the following in the form  $x = f \times 2^\beta$  for some  $f \in [\frac{1}{2}, 1]$ .
  - $x = 25$ ;
  - $x = 13$ ;
  - $x = \frac{1}{3}$ ;
  - $x = \frac{1}{10}$ .
- For each value in the previous problem, compute the logarithm approximation using the degree 4 Taylor polynomial from (1.12). What is the error compared to the logarithm on your calculator?
- Repeat the above for the degree 6 Taylor approximation.
- Repeat the above for the degree 10 Taylor approximation.
- Implement (as a computer program) the logarithm approximation constructed in this section. Compare it to the intrinsic logarithm function over the interval  $[\frac{1}{2}, 1]$ . What is the maximum observed error?
- Let's consider how we might improve on our logarithm approximation from this section.
  - Compute the Taylor expansions, with remainder, for  $\ln(1+x)$  and  $\ln(1-x)$  (use the integral form of the remainder).
  - Combine the two to get the Taylor expansion for

$$f(x) = \ln \left( \frac{1-x}{1+x} \right).$$

What is the remainder in this expansion?

- (c) Given  $z \in [\frac{1}{2}, 1]$  show how to compute  $x$  such that  $z = (1-x)(1+x)^{-1}$ . What interval contains  $x$ ?
- (d) Use this to construct an approximation to  $\ln z$  that is accurate to within  $10^{-16}$ .
7. Use the logarithm expansion from the previous problem, but limited to the degree 4 case, to compute approximations to the logarithm of each value in the first problem of this section.
8. Repeat the above, using the degree 10 approximation.
9. Implement (as a computer program) the logarithm approximation constructed in Problem 6. Compare it to the intrinsic logarithm function over the interval  $[\frac{1}{2}, 1]$ . What is the maximum observed error?
10. Try to use the ideas from this section to construct an approximation to the reciprocal function,  $f(x) = x^{-1}$ , that is accurate to within  $10^{-16}$  over the interval  $[\frac{1}{2}, 1]$ .



## Literature Review

There are a lot of textbooks in numerical analysis and numerical methods. Some, like [8], [9], and [15], are considered classics. A list, by no means exhaustive, of numerical analysis or numerical methods texts is given below.

All of these books give decent treatments of the basic topics. Some are more mathematical than the others; some are designed for less well-prepared students. The books [2, 4, 18] are relatively current texts intended for a graduate student audience; [3, 5, 6, 7, 10, 13, 16, 17] are current texts intended for an undergraduate audience. The presentation in this text has been heavily influenced by [4] and the earlier editions of [6].

An interesting and light-hearted collection of projects for a numerical methods course can be found in [12].

There exist more specialized books that treat only the root-finding problem, or numerical integration, and so on. These will be discussed in the appropriate chapters.

## REFERENCES

---

1. Acton, Forman S.; *Numerical Methods That Work*, Mathematical Association of America, Washington, 1990 (originally published by Harper & Row, 1970).
2. Allen, Myron B., and Isaacson, Eli L.; *Numerical Analysis for Applied Science*, John Wiley & Sons, Inc., New York, 1998.
3. Asaithambi, A.S.; *Numerical Analysis, Theory and Practice*, Saunders College Publishing, Fort Worth, 1995.
4. Atkinson, Kendall; *An Introduction to Numerical Analysis*, John Wiley & Sons, Inc., New York, 1989 (2<sup>nd</sup> edition).
5. Atkinson, Kendall; *Elementary Numerical Analysis*, John Wiley & Sons, Inc., New York, 1993 (2<sup>nd</sup> edition).
6. Burden, Richard, and Faires, J. Douglas; *Numerical Analysis*, Brooks/Cole, Pacific Grove, CA, 1997 (6<sup>th</sup> Edition).
7. Cheney, Ward, and Kincaid, David; *Numerical Mathematics and Computing*, Brooks/Cole Publishing, Pacific Grove, CA, 1994 (3<sup>rd</sup> edition).
8. Conte, S.D., and de Boor, Carl; *Elementary Numerical Analysis*, McGraw-Hill, New York, 1980 (3<sup>rd</sup> edition).
9. Dahlquist, Germund, and Bjorck, Ake; *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
10. Faires, J. Douglas and Burden, Richard; *Numerical Methods*, Brooks/Cole, Pacific Grove, CA, 1998 (2<sup>nd</sup> edition).
11. Forsythe, George, Malcolm, Michael, and Moler, Cleve; *Computer Methods for Numerical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
12. Grandine, Thomas; *The Numerical Methods Programming Projects Book*, Oxford Science Publications, Oxford, 1970.

13. Gregory, John, and Redmond, Don; *Introduction to Numerical Analysis*, Jones and Bartlett Publishers, Boston, 1994.
14. Henrici, Peter; *Essentials of Numerical Analysis*, John Wiley & Sons, Inc., New York, 1982.
15. Isaacson, Eugene, and Keller, Herbert B.; *Analysis of Numerical Methods*, Dover Publications, New York, 1994; originally published by John Wiley & Sons, Inc., New York, 1966.
16. Kahaner, David, Moler, Cleve, and Nash, Stephen; *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
17. Maron, Melvin J., and Lopez, Robert J.; *Numerical Analysis, a Practical Approach*, Wadsworth, Belmont, CA, 1991 (3<sup>rd</sup> edition).
18. Stoer, J., and Bulirsch, R.; *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.
19. Young, David M., and Gregory, Robert Todd; *A Survey of Numerical Mathematics in Two Volumes*, Dover, New York, 1988 (originally published by Addison-Wesley in 1972 and 1973).