CHAPTER 6

Numerical Error

6.0 Introduction

In the problems of interest to us, the solution is either a vector or a function. How can we measure the error of approximations to such solutions? One approach is to measure the error in each component of the vector or in each value of the function. Then the error is itself a vector or a function. However, such error vectors and functions provide a wealth of useless information, and may actually obscure critical evaluation. Thus, for most purposes, we prefer a single number measuring the overall size of the error. Such a measure is called a vector or functional *norm*.

After discussing error norms, this chapter describes the two basic categories of numerical errors: round-off errors and discretization errors. *Round-off errors* are any errors caused by the use of finite-precision real numbers, rather than the true infinite-precision real numbers. *Discretization errors* are any errors caused by using a finite-dimensional vector rather than the true infinite-dimensional vector, or a finite sequence rather than the true infinite sequence, or a finite series rather than the true infinite series, or a finite sequence or series such as a truncated Taylor series rather than the true function.

6.1 Norms and Inner Products

This section concerns vector and functional norms and inner products. Consider a scalar x. The absolute value $|x|$ measures the size of x. Recall that the absolute value has the following properties:

$$|x| \geq 0 \quad \text{and} \quad |x| = 0 \text{ if and only if } x = 0,$$
$$|\alpha x| = |\alpha||x| \quad \text{for any scalar } \alpha,$$
$$|x + y| \leq |x| + |y| \quad \text{for any } x \text{ and } y.$$

The last property is sometimes called the *triangle inequality*.

Consider a vector $\mathbf{x}$. The *norm* $\|\mathbf{x}\|$ measures the size of $\mathbf{x}$. More specifically, $\|\mathbf{x}\|$ is any scalar function of $\mathbf{x}$ such that

$$\|\mathbf{x}\| \geq 0 \quad \text{and} \quad \|\mathbf{x}\| = 0 \text{ if and only if } \mathbf{x} = \mathbf{0},$$
$$\|\alpha\mathbf{x}\| = |\alpha|\,\|\mathbf{x}\| \quad \text{for any scalar } \alpha,$$
$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad \text{for any } \mathbf{x} \text{ and } \mathbf{y}.$$

Vector norms are the natural extensions of absolute value to vectors. However, whereas there is only one absolute value, there are infinitely many vector norms. The most popular vector norms are the *p-norms* or *l_p-norms* defined as follows:

$$\|\mathbf{x}\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_N|^p\right)^{1/p}, \tag{6.1}$$

where $(x_1, \dots, x_N)$ are the components of the vector $\mathbf{x}$ in any orthonormal basis, and p is any integer. For example, the 1-*norm* is defined as

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_N|. \tag{6.2}$$

For another example, the 2-*norm* or *Euclidean norm* is defined as

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}. \tag{6.3}$$

As a final example, the ∞-*norm* or *maximum norm* is defined as

$$\|\mathbf{x}\|_\infty = \max(|x_1|, |x_2|, \dots, |x_N|). \tag{6.4}$$

Consider any function f. The *norm* $\|f\|$ measures the size of f. More specifically, $\|f\|$ is any scalar function of f such that

$$\|f\| \geq 0 \quad \text{and} \quad \|f\| = 0 \text{ if and only if } f = 0,$$
$$\|\alpha f\| = |\alpha| \|f\| \quad \text{for any scalar } \alpha,$$
$$\|f + g\| \leq \|f\| + \|g\| \quad \text{for any } f \text{ and } g.$$

Functional norms are the natural extension of absolute value to functions. The most popular functional norms are the *p-norms* or L_p*-norms* defined as follows:

$$\|f\|_p = \left[\int_a^b |f(x)|^p \, dx\right]^{1/p}, \tag{6.5}$$

where $[a, b]$ is the domain of f and p is any integer. This definition obviously assumes that $|f(x)|^p$ is integrable over $[a, b]$. For example, the 1-*norm* is defined as

$$\|f\|_1 = \int_a^b |f(x)| \, dx, \tag{6.6}$$

assuming that $|f(x)|$ is integrable over $[a, b]$. For another example, the 2-*norm* is defined as

$$\|f\|_2 = \sqrt{\int_a^b f(x)^2 \, dx}, \tag{6.7}$$

assuming that $f(x)^2$ is integrable over $[a, b]$. As a final example, the ∞-*norm* or *maximum norm* is defined as

$$\|f\|_\infty = \operatorname*{ess\,sup}_{a \leq x \leq b} |f(x)|. \tag{6.8}$$

The term "ess sup" or "essential supremum" may not be familiar to many readers. The *supremum* or *sup* is the least upper bound. Although a function must attain its maximum it need not attain its supremum. Thus, for example, a horizontal asymptote is a supremum but not a maximum. For another example, if a function peaks at a jump discontinuity, such as a peak in the pressure function at a shock, then the peak is a supremum but not a maximum. An *essential supremum* or *ess sup* is like a supremum except that it ignores any strange or uncharacteristic points. For example, suppose a function suddenly leaps up at a single point and then leaps back down – this affects the maximum and supremum but not the essential supremum. Similarly, the *infemum* or *inf* is the greatest lower bound, and the *essential infemum* or *ess inf* is like an infemum except that it ignores any strange or uncharacteristic points. Having read this explanation, everyone except the mathematicians in the audience should now mentally replace "ess sup" by "max" in Equation (6.8).

Example 6.1 Suppose $f(x)$ is approximated by $g(x)$ with an error $e(x)$ as follows:

$$e(x) = f(x) - g(x) = 1 - x^2.$$

Measure the error in the 1-norm, 2-norm, and ∞-norm on the domain $[-2, 2]$.

Solution Assuming that the reader can perform the required integrations and maximizations, the answers are $\|e\|_1 = 4$, $\|e\|_2 = 2.477$, and

$$\|e\|_\infty = |e(2)| = |e(-2)| = 3.$$

Norms measure the size of vectors or functions. Norms also measure the distance between vectors or between functions. In particular, $\|\mathbf{x} - \mathbf{y}\|$ measures the *absolute difference* between vectors $\mathbf{x}$ and $\mathbf{y}$, while $\|\mathbf{x} - \mathbf{y}\|/\|\mathbf{x}\|$ or $\|\mathbf{x} - \mathbf{y}\|/\|\mathbf{y}\|$ measure the *relative* or *percentage difference* between vectors $\mathbf{x}$ and $\mathbf{y}$. Similar definitions apply to functional norms. Relative differences provide the most meaningful results. For example, if $\mathbf{x} = 100{,}000$ and $\mathbf{y} = 90{,}000$ the absolute difference is 10,000, which sounds large, but the relative difference is only 10%, which sounds smaller, and properly so.

Different norms are good for different things. For example, in one application, it might be easy to prove that there is a certain upper bound on the 1-norm but hard to prove that there is an upper bound on the 2-norm. For another example, it might be important to minimize the "worst case" ∞-norm error in one application, whereas it might be important to minimize the average error as measured in the 1-norm or 2-norm in another application. Fortunately, a number of inequalities exist that demonstrate that something large in one norm is also reasonably large in all other norms. For example,

$$\|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1 \le \sqrt{N}\|\mathbf{x}\|_2,$$

where N is the vector dimension. In other words, relative judgements such as "large" and "small" are not usually terribly sensitive to the choice of norm.

This completes the discussion of vector and functional norms. The section ends with a brief discussion of vector and functional inner products. An ordinary product of real numbers has the following properties:

$$\begin{aligned} &xx \ge 0 \quad \text{and} \quad xx = 0 \text{ if and only if } x = 0,\\ &xy = yx \quad \text{for any } x \text{ and } y,\\ &x(y + z) = xy + xz \quad \text{for any } x, y \text{ and } z. \end{aligned}$$

A *vector inner product* or *scalar product* maps two vectors to a scalar such that

$$\begin{aligned} &\mathbf{x} \cdot \mathbf{x} \ge 0 \quad \text{and} \quad \mathbf{x} \cdot \mathbf{x} = 0 \text{ if and only if } \mathbf{x} = 0,\\ &\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} \quad \text{for any } \mathbf{x} \text{ and } \mathbf{y},\\ &\mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z} \quad \text{for any } \mathbf{x}, \mathbf{y}, \text{ and } \mathbf{z},\\ &\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y} \quad \text{for any } \mathbf{x}, \mathbf{y}, \text{ and scalar } \alpha. \end{aligned}$$

The most popular vector inner product is the Euclidean inner product

$$\mathbf{x}\cdot\mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_N y_N, \tag{6.9}$$

where $(x_1, \ldots, x_N)$ and $(y_1, \ldots, y_N)$ are the components of **x** and **y** in any orthonormal basis. In standard notation, the dot is often reserved for the Euclidean inner product; general inner products are denoted by $\langle \mathbf{x}, \mathbf{y}\rangle$ or $(\mathbf{x}, \mathbf{y})$. Every vector inner product is naturally associated with a norm as follows:

$$\mathbf{x}\cdot\mathbf{x} = \|\mathbf{x}\|^2. \tag{6.10}$$

For example, the Euclidean inner product is naturally associated with the 2-norm. There are many other possible vector products besides the inner product, including outer products and cross products. These will not be discussed here.

A *functional inner product* maps two functions to a scalar such that

$$\begin{aligned}
&f\cdot f \geq 0 \quad \text{and} \quad f\cdot f = 0 \text{ if and only if } f = 0,\\
&f\cdot g = g\cdot f \quad \text{for any } f \text{ and } g,\\
&f\cdot(g+h) = f\cdot g + f\cdot h \quad \text{for any } f, g, \text{ and } h,\\
&\alpha(f+g) = \alpha f + \alpha g \quad \text{for any } f, g, \text{ and scalar } \alpha.
\end{aligned}$$

The most popular functional inner products have the form

$$f\cdot g = \int_a^b w(x) f(x) g(x)\, dx, \tag{6.11}$$

where $w(x)$ is a nonnegative *weighting function.* Functional inner products of all varieties are more commonly denoted by $\langle f, g\rangle$ or (f, g). Every functional inner product is naturally associated with a norm as follows:

$$f\cdot f = \|f\|^2. \tag{6.12}$$

For example, if $w(x) = 1$, then the inner product defined by Equation (6.12) is naturally associated with the functional 2-norm.

6.2 Round-Off Error

This section very briefly discusses round-off error. A *binary digit* or *bit* is a digit in a base-two number system; it is either zero or one. A *byte* is eight bits. Computers do not deal in individual bits or bytes. Instead, computers deal in *words*, typically 32 bits or four-bytes long. A 32-bit word can represent roughly $2^{32} \approx 4.29 \times 10^9$ different objects. Those objects can be real numbers, integers, characters, and so forth. For example, consider integers. On a typical computer, the 2^{32} integers are more or less evenly divided between positive integers and negative integers. In other words, the integers run between $-2^{32}/2$ and $2^{32}/2$ give or take one. As another example, consider real numbers. Instead of the true infinite continuum of real numbers, there must be a maximum computer number, generally around 10^{39}, and any larger numbers cause *overflow*. There must also be a minimum computer number, generally around 10^{-39}, and any smaller numbers cause *underflow*. Finally, there must be a finite spacing between computer numbers. The spacing varies with the size of the numbers such that the *relative* spacing remains constant. Then the relative spacing between computer numbers, also called the *machine epsilon* ϵ, bounds the *relative round-off error* as follows:

$$\textit{relative round-off error} = \frac{|\textit{true number} - \textit{computer number}|}{|\textit{true number}|} \leq \epsilon.$$

Machine epsilon is typically about $\epsilon = 10^{-8}$; hence computer numbers have roughly eight decimal places of accuracy in base 10. As an alternative to single word representations, computers can also use two, four, or more words to represent objects such as real numbers; for example, double precision increases the maximum real number to something like 10^{309}, decreases the minimum real number to something like 10^{-309}, and decreases the machine epsilon, allowing roughly sixteen decimal places of accuracy in base 10.

Round-off error may seem insignificant, since the relative round-off error is always less than or equal to the machine epsilon ϵ. However, although round-off error starts small, there are at least three computational mechanisms that may magnify round-off errors. First, in a large sum, suppose that there is a wide range of numbers and that small numbers are repeatedly added to large numbers. For example, suppose that 1 and ϵ are added repeatedly. After each addition, the sum is still 1 since $1 + \epsilon = 1$ according to the computer, by the definition of ϵ. After $1/\epsilon$ such additions, the true sum should be 2 whereas the computer yields 1; thus the relative round-off error is 100%! The best solution to this problem is to always add numbers from smallest to largest, although this may require an expensive sorting procedure.

Another mechanism that magnifies round-off error is called *subtractive cancellation*. Suppose two nearly equal numbers are subtracted. For example, $44.55 - 44.54 = 0.01$. Although the original numbers have four decimal places of accuracy in base 10, their difference has only one decimal place of accuracy! Another example of subtractive cancellation follows.

Example 6.2 If $x = 1984$, find $\sqrt{x+1} - \sqrt{x}$ truncating all numbers to four decimal places in base 10. Suggest a way to avoid subtractive cancellation.

Solution If $x = 1984$ then

$$\sqrt{x+1} - \sqrt{x} = 44.55 - 44.54 = 0.01.$$

Alternatively,

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

Then

$$\sqrt{1985} - \sqrt{1984} = \frac{1}{\sqrt{1985} + \sqrt{1984}} = \frac{1}{44.55 + 44.54} = 0.01122.$$

This second result has four decimal places of accuracy rather than one.

A third mechanism that magnifies round-off error is solution sensitivity. In a sensitive problem, small changes in the problem lead to large changes in the solution. Round-off errors inevitably introduce small changes, which create larges changes in the solution. For example, a matrix A is called *ill-conditioned* if, for any vector $\mathbf{b}$, small changes in A or $\mathbf{b}$ cause large changes in $\mathbf{x}$, where $\mathbf{x}$ is the solution to $A\mathbf{x} = \mathbf{b}$. Matrices are often ill-conditioned when there is a large range of sizes in the elements of the matrix, or when one row of the matrix is nearly equal to some linear combination of other rows. See Golub and van Loan (1989) for a more detailed description of matrix ill-conditioning. Another example of sensitivity follows.

Example 6.3 Consider the following iterative equation:

$$u_{n+1} = 4\lambda u_n(1 - u_n),$$

where $0 \leq \lambda \leq 1$ is any constant and u_0 is any starting value. This simple quadratic equation is called the *logistic map*. Show that the solution to the logistic map is extremely sensitive to small errors.

Solution Notice that if $0 \leq u_0 \leq 1$ then $0 \leq u_n \leq 1$ for all n. However, if $u_0 > 1$ or $u_0 < 0$ then u_n quickly diverges to $\pm\infty$ as n increases. In this example, we shall choose $0 \leq u_0 \leq 1$ and $\lambda = 1$. In particular, let us compare the results using the three starting values $u_0 = 0.4 + 10^{-8}$, $u_0 = 0.4$, and $u_0 = 0.4 - 10^{-8}$. Using double-precision arithmetic, the three solutions are nearly equal for small n. However, the three solutions differ significantly for $n = 20$ and the three solutions are entirely unrelated for $n = 30$. This is a classic example of *chaos*. See Tabor (1989) for a more complete description of chaos. Chaos is but one interesting behavior studied under the heading of *dynamical systems*. Traditional computational gasdynamics does not allow particularly complicated dynamical behaviors or, in particular, chaos; however, computational gasdynamics with strong source terms, such as chemically reacting flows, allows all sorts of devious dynamical behaviors, which may pose extreme challenges for numerical models. For more information, see Yee, Sweby, and Griffiths (1991).

Sometimes extreme sensitivity is due to a mistake in the original problem statement. For example, the problem statement may not specify enough information or it may specify contradictory information; in other words, the problem may be *ill-posed*. In other cases, the problem statement is fine, and the extreme sensitivity is the result of a faulty numerical approximation. In yet other cases, the problem is simply inherently sensitive, and it is unreasonable to expect to approximate its details. For example, turbulent gas flows are constantly changing in a chaotic like fashion, and thus no approximation can be expected to capture the exact pattern of a given turbulent flow. This sort of sensitivity is sometimes called physical *instability*; see Chapter 14 for more discussion along these lines.

6.3 Discretization Error

The last section concerned round-off errors, that is, errors in the computer representation of numbers. This section concerns discretization error, that is, errors in the computer representation of functions and functional operators. As every reader will certainly recall, a *function* maps a domain to a range. A function is *continuously defined* if the domain is continuous; for example, the domain might equal the real numbers between a and b. A function is *discretely defined* if the domain is discrete; for example, a *sequence* is a discretely defined function whose domain is a set of consecutive integers, called the *index*. The term *vector* either refers to any sequence, especially when the index starts from one or zero, or to a sequence with special coordinate transformation properties, as discussed in Chapter 7.

The distinction between continuously defined and discretely defined functions is not as clear-cut as it first appears – in fact, continuously defined functions can often be naturally interchanged with discretely defined functions. For example, the discretely defined function $(a_0, a_1, a_2, a_3, \ldots)$ can be transformed to a continuously defined function $f(x)$ as follows:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots.$$

Vice versa, a continuously defined function $f(x)$ can be transformed to a discretely defined function $(a_0, a_1, a_2, a_3, \ldots)$ using Taylor series, where

$$a_n = \frac{1}{n!}\frac{d^n f}{dx^n}(0).$$

How do computers represent functions? Computers can only perform the four basic arithmetic operations: addition, subtraction, multiplication, and division. Then, for example, a computer can easily represent $f(x) = ax^2 + bx + c$ since this function involves only basic arithmetic operations. In contrast $\ln x = \int_1^x \frac{dy}{y}$ involves integration in addition to basic arithmetic operations. However, consider the following series:

$$\ln x = \frac{2}{1}\frac{x-1}{x+1} + \frac{2}{3}\left(\frac{x-1}{x+1}\right)^2 + \frac{2}{5}\left(\frac{x-1}{x+1}\right)^5 + \cdots.$$

Then the continuously defined function $\ln x$ can be recovered from the discretely defined sequence $(2, 2/3, 2/5, \ldots)$ using only the four basic arithmetic operations.

Whereas many functions can be perfectly represented by *infinite* sequences and basic arithmetic operations, computers can only store *finite* sequences. Thus the true sequence representing the true function must be artificially truncated to a finite number of terms. To summarize: *Computers represent functions by finite sequences of numbers.* Many problems involve not only continuously defined functions but also operations on continuously defined functions such as integration and differentiation. If continuously defined functions are replaced by discretely defined functions, then all operators on continuously defined functions must also be replaced by operators on discretely defined functions. *Discretization* is the process of replacing all continuously defined functions and functional operators by discretely defined functions and functional operators. *Discretization error* is, naturally enough, any error caused by discretization. *Truncation* is the process of replacing infinite sequences or series by finite sequences or series, and *truncation error* is any error caused by truncation. Truncation is a type of discretization, since it is part of the process of putting functions and functional operators in a form suitable for computation; then truncation error is a type of discretization error.

Unfortunately, just as a finite number of bits can represent only a limited number of real numbers, a finite number of numbers $(a_0, a_1, \ldots, a_N)$ can represent only a limited number of functions. In other words, computers can rarely represent real numbers exactly, and real numbers are represented by the closest computer number; similarly, computers can rarely represent functions exactly, and functions are represented by the closest computer function.

To continue the analogy, round-off error is any error that results from replacing a real number by a computer number; similarly, discretization error is any error that results from replacing a function by a computer function (i.e., the error that results from replacing a function by a finite sequence of numbers). Unlike the computer representation of numbers, the computer representation of functions is completely under the user's control. The following examples illustrate some of the principles involved.

Example 6.4 If the required functions are always linear or nearly linear, a linear representation $f(x) \approx ax + b$ is the best choice. Linear representations require only two real numbers: a and b. However, unless the exact function is truly linear, a line can accurately represent the function only over a limited domain. Outside of the domain of accuracy, the approximating line strays progressively further and further from the true function; this occurs, for example, with tangent and secant lines. To approximate a nonlinear function over a wide region generally requires a representation with more than two numbers, such as a quadratic or cubic representation. Even within the domain of accuracy, for general functions, the accuracy of linear approximations is relatively low. Thus linear representations place severe limitations on both the accuracy and the domain of accuracy for general functions.

Example 6.5 If the required functions contain periodic oscillations, a trigonometric series may be the best choice. For example, $f(x) \approx a_0 + a_1 \cos x + b_1 \sin x$. This is a compact representation – it requires only three real numbers a_0, a_1, and b_1 – and captures the expected periodic oscillations. When using trigonometric series representations we assume that the computer has adequate representations for $\cos x$ and $\sin x$. Indeed, most programming languages have efficient built-in libraries of trigonometric functions.

Example 6.6 Suppose that, as part of some computer method, functions are frequently differentiated. This is extremely easy to do in a Taylor series form representation. In particular, suppose that

$$f(x) \approx a_0 + a_1(x-b) + a_2(x-b)^2 + \cdots + a_N(x-b)^N.$$

Then

$$\frac{df}{dx}(x) \approx a_1 + 2a_2(x-b) + \cdots + Na_N(x-b)^{N-1}.$$

Differentiation is more difficult with most other functional forms.

Suppose that a function $f(x)$ is replaced by an approximate function $g_N(x)$, where $g_N(x)$ is defined by a finite sequence of N numbers $(a_0, a_1, \ldots, a_N)$. The discretization error $e_N(x)$ is defined as follows:

$$e_N(x) = f(x) - g_N(x), \tag{6.13}$$

How does $e_N(x)$ depend on N? If $|e_N(x)| \to 0$ as $N \to \infty$ for all x, then the approximation function *pointwise converges* to the true function. If $\|e_N\|_2 \to 0$ as $N \to \infty$, then the approximation function *converges in the mean* to the true function; and if $\|e_N\|_\infty \to 0$ as

$N \to \infty$, then the approximation function *uniformly converges* to the true function. How does $e_N(x)$ depend on x? Suppose that $|e_N(x)| \le M|x^R|$ for all x, where M is some constant; then the approximation has *order of approximation* or *order of accuracy* R. In this case, a common notation is $e_N(x) = O(x^R)$.

Example 6.7 By Taylor series, a fourth-order accurate approximation to e^x is

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + O(x^4).$$

Also, a fifth-order accurate approximation to $\sin x$ is

$$\sin x = x - \frac{x^3}{3!} + O(x^5).$$

Then

$$\begin{aligned} e^x + \sin x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + O(x^4) + x - \frac{x^3}{3!} + O(x^5) \\ &= 1 + 2x + \frac{x^2}{2!} + O(x^4). \end{aligned}$$

Multiply the last two approximations to obtain the following approximation to $e^x \sin x$:

$$\begin{aligned} e^x \sin x &= \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + O(x^4)\right)\left(x - \frac{x^3}{3!} + O(x^5)\right) \\ &= x - \frac{x^3}{3!} + O(x^5) + x^2 - \frac{x^4}{3!} + O(x^6) + \frac{x^3}{2!} - \frac{x^5}{2!3!} + O(x^7) \\ &\quad + \frac{x^4}{3!} - \frac{x^6}{3!3!} + O(x^8) + O(x^5) + O(x^7) + O(x^9) \\ &= x + x^2 + \left(\frac{1}{2!} - \frac{1}{3!}\right)x^3 + \left(\frac{1}{3!} - \frac{1}{3!}\right)x^4 + O(x^5) \\ &= x + x^2 + \frac{x^3}{3} + O(x^5). \end{aligned}$$

Rather than using a single function to represent all of $f(x)$, suppose that the domain of $f(x)$ is decomposed into cells and that a different function is used to represent $f(x)$ in each cell. This is illustrated by the following example:

Example 6.8 Consider a linear approximation

$$f(x) = ax + b + O(x^2).$$

This could be a secant line or a tangent line centered on $x = 0$. Notice that the error is large if $|x|$ is large. So, instead of a single line, suppose that the domain is decomposed into cells

$[x_0, x_1], [x_1, x_2], \ldots, [x_{N-1}, x_N]$ and that a different line is used to represent $f(x)$ in each cell. For example,

$$f(x) \approx \begin{cases} a_1(x - x_1) + b_1 + O[(x - x_1)^2] & x_0 \le x < x_1, \\ a_2(x - x_2) + b_2 + O[(x - x_2)^2] & x_1 \le x < x_2, \\ \vdots & \vdots \\ a_N(x - x_N) + b_N + O[(x - x_N)^2] & x_{N-1} \le x \le x_N, \end{cases}$$

where each line could be a local secant or tangent line approximation. Now suppose that $x_{i+1} = x_i + \Delta x$. Then

$$O(x - x_i) = O(\Delta x) \quad \text{for} \quad x_i \le x \le x_{i+1}.$$

Then

$$f(x) \approx \begin{cases} a_1(x - x_1) + b_1 + O(\Delta x^2) & x_0 \le x < x_1, \\ a_2(x - x_2) + b_2 + O(\Delta x^2) & x_1 \le x < x_2, \\ \vdots & \vdots \\ a_N(x - x_N) + b_N + O(\Delta x^2) & x_{N-1} \le x \le x_N. \end{cases}$$

Now the error depends mostly on the single constant parameter $\Delta x = (x_N - x_0)/N$ and much less on the variable x.

This last example illustrates *piecewise-polynomial* approximations. A piecewise-polynomial approximation *pointwise converges* if $e(x) \to 0$ as $\Delta x \to 0$ for all x. Similarly, a piecewise-polynomial approximation *converges in the mean* if $\|e\|_2 \to 0$ as $\Delta x \to 0$. Finally, a piecewise-polynomial approximation *converges uniformly* if $\|e\|_\infty \to 0$ as $\Delta x \to 0$. Rather than just requiring that the error go to zero as $\Delta x \to 0$, suppose we require that the error go to zero at some minimum rate; this leads us to new definitions for order of accuracy. In particular, a piecewise-polynomial approximation is *pointwise Rth-order accurate* if $|e(x)| \le M|\Delta x^R|$ for all x; *Rth-order accurate in the mean* if $\|e\|_2 \le M|\Delta x^R|$; and *uniformly Rth-order accurate* if $\|e\|_\infty \le M|\Delta x^R|$ for some constant M. Notice that all three definitions of order of accuracy imply convergence as $\Delta x \to 0$.

Unfortunately, the order of accuracy may vary from cell to cell. For example, referring to Example 6.8, if $f(x)$ varies linearly or nearly linearly in a certain cell, then the pointwise order of accuracy of the piecewise-linear approximation in that cell may be anything between three and infinity. In contrast, if $f(x)$ is discontinuous in a certain cell, then the pointwise order of accuracy in that cell may be very small, typically between one and zero. When a piecewise-polynomial approximation is called "Rth-order accurate," this usually means "the approximation has pointwise Rth-order accuracy except in certain cells." By tradition, some exceptions are explicitly noted while others are simply understood. Specifically, local losses in the order of accuracy caused by jump discontinuities in the function or its derivatives are usually understood and are not usually noted explicitly. However, any local losses in the order of accuracy near extrema are almost always explicitly noted. For example, many approximations in computational gasdynamics have second-order accuracy in most places but only first-order accuracy near smooth extrema, due to a phenomenon called clipping;

see Chapter 16 and Part V. In conclusion, as commonly used, "Rth-order accuracy," "order-of-accuracy R," and $O(\Delta x^R)$ are a bit vague – these measures depend on the choice of norm and may vary from place to place and from function to function.

Note that an approximation can have a high order of accuracy and yet have very poor absolute or relative accuracy. Thus the error for a given Δx may be quite large, regardless of the fact that the error would decrease rapidly if Δx were decreased. *Do not make the common mistake of confusing high accuracy with high order of accuracy.*

Remember that the order of accuracy refers to discretization error and does not include the effects of round-off error. Thus, in practice, the error of a numerical approximation based on piecewise-polynomials will decrease at the expected rate as Δx decreases, but only to a certain point, usually when Δx nears machine epsilon. Relative round-off error increases as Δx decreases so that eventually increases in relative round-off errors will completely offset decreases in the relative discretization error, and the overall relative error will stop decreasing unless the machine precision is increased.

It is important to be able to distinguish between discretization errors and round-off errors because their causes and cures are completely different. The classic technique is to increase the machine precision. Under ordinary circumstances, changing from single to double precision should only affect the results starting in the seventh or eighth decimal place. However, if the computation magnifies round-off error, the results may change even in the first or second decimal place. In this case, the machine precision should be doubled again and again, if possible, until the low-order decimal places stop changing. Then any error in the stable low-order decimal places must be discretization error.

References

Golub, G. H., and van Loan, C. F. 1989. *Matrix Computations*, 2nd ed., Baltimore, MD: Johns Hopkins University Press, Chapter 2.

Tabor, M. 1989. *Chaos and Integrability in Nonlinear Dynamics: An Introduction*, New York: Wiley.

Yee, H. C., Sweby, P. K., and Griffiths, D. F. 1991. "Dynamical Approach Study of Spurious Steady-State Numerical Solutions of Nonlinear Differential Equations. I. The Dynamics of Time Discretization and Its Implications for Algorithm Development in Computational Fluid Dynamics," *Journal of Computational Physics*, 97:249–310.

Problems

6.1 (a) Consider the following inner product:

$$f \cdot g = \int_{-1}^{1} f(x)g(x)\,dx.$$

Find $\|g\|$ in the natural norm and $f \cdot g$ where $f(x) = 1 - x + \frac{1}{3}x^2 + 5x^3$ and $g(x) = x - 3x^2$.

(b) Consider the following inner product:

$$f \cdot g = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx.$$

Find $\|g\|$ in the natural norm and $f \cdot g$ where $f(x) = \sin^{-1} x$ and $g(x) = x$. Please feel free to avail yourself of a table of integrals.

6.2 Suppose that $f(x)$ is approximated by $g(x)$ with a pointwise-error $e(x) = \sin x$.

(a) Measure the error in the 1-norm, 2-norm, and ∞-norm on the domain $[-\pi/3, \pi/4]$.
(b) Suppose $\|f\|_1 = 0.1$ and $\|g\|_1 = 0.523$. What are the relative errors in the 1-norm? Is the error large or small, as measured in the 1-norm?

6.3 In each of the following cases, underflow will occur on most machines using single-precision arithmetic. In each case, suppose that any quantities that underflow are replaced by zero. What are the resulting absolute and relative round-off errors? Based on the relative round-off error, in which cases is it reasonable to replace the quantities that underflow by zero?
(a) $x + y$ where $x = 10^{-50}$ and $y = 1$.
(b) $x + y$ where $x = y = 10^{-25}$.
(c) xy where $x = 10^{-100}$ and $y = 10^{-99}$.
(d) y/x where $x = 10^{-100}$ and $y = 10^{-99}$.

6.4 Consider the roots of a quadratic $ax^2 + bx + c = 0$. The quadratic equation yields the following two solutions:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

For one of the roots, show that this expression may magnify round-off error when b^2 is much larger than $4ac$. What causes this error magnification? Which root is subject to round-off error magnification and which root is not? For the root subject to round-off error magnification, suggest an equivalent expression that avoids the round-off error.

6.5 Add 0.333, 2.31, 50.2, and 263 to find the exact value for the sum. Now add the numbers from largest to smallest, truncating all intermediate results to three decimal places in base 10. What is the relative round-off error of the sum? Now add the numbers from smallest to largest, again truncating all intermediate results to three decimal places in base 10. What is the relative round-off error of the sum?

6.6 In each case, suggest an equivalent expression that avoids round-off error magnification. Feel free to use trigonometric and logarithmic identities.
(a) $1 - \cos x$ when $x \approx 0$.
(b) $\ln(x + 1) - \ln x$ for large x.

6.7 Consider the following Taylor series approximations:

$$\cos x = 1 - \frac{1}{2}x^2 + O(x^4),$$
$$\sin x = x - \frac{1}{6}x^3 + O(x^5).$$

Using these expressions, find approximations for $\sec x = 1/\cos x$ and $\tan x = \sin x/\cos x$; what is the order of accuracy of these approximations? Use the fact that

$$\frac{1}{1+x} = 1 + x + x^2 + x^3 + x^4 + x^5 + \cdots.$$

6.8 Consider cells $[-\pi/16, \pi/16]$, $[\pi/16, 3\pi/16]$, $[3\pi/16, 5\pi/16]$, $[5\pi/16, 7\pi/16]$, $[7\pi/16, 9\pi/16]$, $[9\pi/16, 11\pi/16]$, $[11\pi/16, 13\pi/16]$, $[13\pi/16, 15\pi/16]$, $[15\pi/16, 17\pi/16]$.
(a) Consider a piecewise-constant approximation to $\sin x$ based on the value of $\sin x$ at the cell center. For each cell, what is the order of accuracy? Use Taylor series. Be especially careful in the last cell. Does the order of accuracy depend on the choice of norm?
(b) Consider a piecewise-linear approximation to $\sin x$ based on a Taylor series approximation to $\sin x$ about the cell center. For each cell, what is the order of accuracy? Be especially careful in the first cell. Does the order of accuracy depend on the choice of norm?