CHAPTER 19

Numerics in General

Numeric analysis or briefly numerics has a distinct flavor that is different from basic calculus, from solving ODEs algebraically, or from other (nonnumeric) areas. Whereas in calculus and in ODEs there were very few choices on how to solve the problem and your answer was an algebraic answer, in numerics you have many more choices and your answers are given as tables of values (numbers) or graphs. You have to make judicous choices as to what numeric method or algorithm you want to use, how accurate you need your result to be, with what value (starting value) do you want to begin your computation, and others. This chapter is designed to provide a good transition from the algebraic type of mathematics to the numeric type of mathematics.

We begin with the general concepts such as floating point, roundoff errors, and general numeric errors and their propagation. This is followed in Sec. 19.2 by the important topic of solving equations of the type $f(x) = 0$ by various numeric methods, including the famous Newton method. Section 19.3 introduces interpolation methods. These are methods that construct new (unknown) function values from known function values. The knowledge gained in Sec. 19.3 is applied to spline interpolation (Sec. 19.4) and is useful for understanding numeric integration and differentiation covered in the last section.

Numerics provides an invaluable extension to the knowledge base of the problem-solving engineer. Many problems have no solution formula (think of a complicated integral or a polynomial of high degree or the interpolation of values obtained by measurements). In other cases a complicated solution formula may exist but may be practically useless. It is for these kinds of problems that a numerical method may generate a good answer. Thus, it is very important that the applied mathematician, engineer, physicist, or scientist becomes familiar with the essentials of numerics and its ideas, such as estimation of errors, order of convergence, numerical methods expressed in algorithms, and is also informed about the important numeric methods.

Prerequisite: Elementary calculus.
References and Answers to Problems: App. 1 Part E, App. 2.

19.1 Introduction

As an engineer or physicist you may deal with problems in elasticity and need to solve an equation such as $x \cosh x = 1$ or a more difficult problem of finding the roots of a higher order polynomial. Or you encounter an integral such as

$$\int_0^1 \exp(-x^2)\, dx$$

[see App. 3, formula (35)] that you cannot solve by elementary calculus. Such problems, which are difficult or impossible to solve algebraically, arise frequently in applications. They call for **numeric methods**, that is, systematic methods that are suitable for solving, numerically, the problems on computers or calculators. Such solutions result in tables of numbers, graphical representation (figures), or both. Typical numeric methods are iterative in nature and, for a well-choosen problem and a good starting value, will frequently converge to a desired answer. The evolution from a given problem that you observed in an experimental lab or in an industrial setting (in engineering, physics, biology, chemistry, economics, etc.) to an approximation suitable for numerics to a final answer usually requires the following steps.

1. **Modeling.** We set up a mathematical model of our problem, such as an integral, a system of equations, or a differential equation.
2. **Choosing a numeric method** and parameters (e.g., step size), perhaps with a preliminary error estimation.
3. **Programming.** We use the algorithm to write a corresponding program in a CAS, such as Maple, Mathematica, Matlab, or Mathcad, or, say, in Java, C or C^{++}, or FORTRAN, selecting suitable routines from a software system as needed.
4. **Doing the computation.**
5. **Interpreting the results** in physical or other terms, also deciding to rerun if further results are needed.

Steps 1 and 2 are related. A slight change of the model may often admit of a more efficient method. To choose methods, we must first get to know them. Chapters 19–21 contain efficient algorithms for the most important classes of problems occurring frequently in practice.

In Step 3 the program consists of the given data and a sequence of instructions to be executed by the computer in a certain order for producing the answer in numeric or graphic form.

To create a good understanding of the nature of numeric work, we continue in this section with some simple general remarks.

Floating-Point Form of Numbers

We know that in decimal notation, every real number is represented by a finite or an infinite sequence of decimal digits. Now most computers have two ways of representing numbers, called *fixed point* and *floating point*. In a **fixed-point** system all numbers are given with a fixed number of decimals after the decimal point; for example, numbers given with 3 decimals are 62.358, 0.014, 1.000. In a text we would write, say, 3 decimals as 3D. Fixed-point representations are impractical in most scientific computations because of their limited range (explain!) and will not concern us.

In a **floating-point** system we write, for instance,

$$0.6247 \cdot 10^{3}, \qquad 0.1735 \cdot 10^{-13}, \qquad -0.2000 \cdot 10^{-1}$$

or sometimes also

$$6.247 \cdot 10^{2}, \qquad 1.735 \cdot 10^{-14}, \qquad -2.000 \cdot 10^{-2}.$$

We see that in this system the number of significant digits is kept fixed, whereas the decimal point is "floating." Here, a **significant digit** of a number c is any given digit of c, except possibly for zeros to the left of the first nonzero digit; these zeros serve only to fix the position of the decimal point. (Thus any other zero is a significant digit of c.) For instance,

$$13600, \quad 1.3600, \quad 0.0013600$$

all have 5 significant digits. In a text we indicate, say, 5 significant digits, by 5S.

The use of exponents permits us to represent very large and very small numbers. Indeed, theoretically any nonzero number a can be written as

$$(1) \qquad a = \pm m \cdot 10^n, \qquad 0.1 \leqq |m| < 1, \qquad n \text{ integer.}$$

On modern computers, which use binary (base 2) numbers, m is limited to k binary digits (e.g., $k = 8$) and n is limited (see below), giving representations (for finitely many numbers only!)

$$(2) \qquad \bar{a} = \pm \bar{m} \cdot 2^n, \qquad \bar{m} = 0.d_1 d_2 \cdots d_k, \qquad d_1 > 0.$$

These numbers $\bar{a}$ are called *k-digit binary* **machine numbers**. Their fractional part m (or $\bar{m}$) is called the *mantissa*. This is not identical with "mantissa" as used for logarithms. n is called the *exponent* of $\bar{a}$.

It is important to realize that there are only finitely many machine numbers and that they become less and less "dense" with increasing a. For instance, there are as many numbers between 2 and 4 as there are between 1024 and 2048. Why?

The smallest positive machine number *eps* with $1 + \text{eps} > 1$ is called the *machine accuracy*. It is important to realize that there are no numbers in the intervals $[1, 1 + \text{eps}]$, $[2, 2 + 2 \cdot \text{eps}], \cdots, [1024, 1024 + 1024 \cdot \text{eps}], \cdots$. This means that, if the mathematical answer to a computation would be $1024 + 1024 \cdot \text{eps}/2$, the computer result will be *either* 1024 or $1024 \cdot \text{eps}$ so it is impossible to achieve greater accuracy.

Underflow and Overflow. The range of exponents that a typical computer can handle is very large. The IEEE (Institute of Electrical and Electronic Engineers) floating-point standard for **single precision** is from 2^{-126} to 2^{128} (1.175×10^{-38} to 3.403×10^{38}) and for **double precision** it is from 2^{-1022} to 2^{1024} (2.225×10^{-308} to 1.798×10^{308}).

As a minor technicality, to avoid storing a minus in the exponent, the ranges are shifted from $[-126, 128]$ by adding 126 (for double precision 1022). Note that shifted exponents of 255 and 1047 are used for some special cases such as representing infinity.

If, in a computation a number outside that range occurs, this is called **underflow** when the number is smaller and **overflow** when it is larger. In the case of underflow, the result is usually set to zero and computation continues. Overflow might cause the computer to halt. Standard codes (by IMSL, NAG, etc.) are written to avoid overflow. Error messages on overflow may then indicate programming errors (incorrect input data, etc.). From here on, we will be discussing the decimal results that we obtain from our computations.

Roundoff

An error is caused by **chopping** (= discarding all digits from some decimal on) or **rounding**. This error is called **roundoff error**, regardless of whether we chop or round. The rule for rounding off a number to k decimals is as follows. (The rule for rounding off to k significant digits is the same, with "decimal" replaced by "significant digit.")

Roundoff Rule. To round a number x to k decimals, and $5 \cdot 10^{-(k+1)}$ to x and chop the digits after the $(k + 1)$st digit.

EXAMPLE 1 Roundoff Rule

Round the number 1.23454621 to **(a)** 2 decimals, **(b)** 3 decimals, **(c)** 4 decimals, **(d)** 5 decimals, and **(e)** 6 decimals.

Solution. **(a)** For 2 decimals we add $5 \cdot 10^{-(k+1)} = 5 \cdot 10^{-3} = 0.005$ to the given number, that is, $1.2345621 + 0.005 = 1.23\,954621$. Then we chop off the digits "954621" after the space or equivalently $1.23954621 - 0.00954621 = 1.23$.
(b) $1.23454621 + 0.0005 = 1.235\,04621$, so that for 3 decimals we get 1.234.
(c) 1.23459621 after chopping give us 1.2345 (4 decimals).
(d) 1.23455121 yields 1.23455 (5 decimals).
(e) 1.23454671 yields 1.234546 (6 decimals).
Can you round the number to 7 decimals? ■

Chopping is not recommended because the corresponding error can be larger than that in rounding. (Nevertheless, some computers use it because it is simpler and faster. On the other hand, some computers and calculators improve accuracy of results by doing intermediate calculations using one or more extra digits, called *guarding digits.*)

Error in Rounding. Let $\bar{a} = \mathrm{fl}(a)$ in (2) be the floating-point computer approximation of a in (1) obtained by rounding, where fl suggests **floating**. Then the roundoff rule gives (by dropping exponents) $|m - \bar{m}| \leqq \frac{1}{2} \cdot 10^{-k}$. Since $|m| \geqq 0.1$, this implies (when $a \neq 0$)

$$\left|\frac{a - \bar{a}}{a}\right| \approx \left|\frac{m - \bar{m}}{m}\right| \leqq \frac{1}{2} \cdot 10^{1-k}. \tag{3}$$

The right side $u = \frac{1}{2} \cdot 10^{1-k}$ is called the **rounding unit**. If we write $\bar{a} = a(1 + \delta)$, we have by algebra $(\bar{a} - a)/a = \delta$, hence $|\delta| \leqq u$ by (3). *This shows that the rounding unit u is an error bound in rounding.*

Rounding errors may ruin a computation completely, even a small computation. In general, these errors become the more dangerous the more arithmetic operations (perhaps several millions!) we have to perform. It is therefore important to analyze computational programs for expected rounding errors and to find an arrangement of the computations such that the effect of rounding errors is as small as possible.

As mentioned, the arithmetic in a computer is not exact and causes further errors; however, these will not be relevant to our discussion.

Accuracy in Tables. Although available software has rendered various tables of function values superfluous, some tables (of higher functions, of coefficients of integration formulas, etc.) will still remain in occasional use. If a table shows k significant digits, it is conventionally assumed that any value $\tilde{a}$ in the table deviates from the exact value a by at most $\pm\frac{1}{2}$ unit of the kth digit.

Loss of Significant Digits

This means that a result of a calculation has fewer correct digits than the numbers from which it was obtained. This happens if we subtract two numbers of about the same size, for example, $0.1439 - 0.1426$ ("subtractive cancellation"). It may occur in simple problems, but it can be avoided in most cases by simple changes of the algorithm—if one is aware of it! Let us illustrate this with the following basic problem.

EXAMPLE 2 Quadratic Equation. Loss of Significant Digits

Find the roots of the equation

$$x^2 + 40x + 2 = 0,$$

using 4 significant digits (abbreviated 4S) in the computation.

Solution. A formula for the roots x_1, x_2 of a quadratic equation $ax^2 + bx + c = 0$ is

$$x_1 = \frac{1}{2a}\left(-b + \sqrt{b^2 - 4ac}\right), \qquad x_2 = \frac{1}{2a}\left(-b - \sqrt{b^2 - 4ac}\right). \tag{4}$$

Furthermore, since $x_1 x_2 = c/a$, another formula for those roots

$$x_1 = \frac{c}{ax_2}, \qquad x_2 \text{ as in (4).} \tag{5}$$

We see that this avoids cancellation in x_1 for positive b.

If $b < 0$, calculate x_1 from (4) and then $x_2 = c/(ax_1)$.

For $x^2 + 40x + 2 = 0$ we obtain from (4) $x = -20 \pm \sqrt{398} = -20 \pm 19.95$, hence $x_2 = -20.00 - 19.95$, involving no difficulty, and $x_1 = -20.00 + 19.95 = -0.05$, a poor value involving loss of digits by subtractive cancellation.

In contrast, (5) gives $x_1 = 2.000/(-39.95) = -0.05006$, the absolute value of the error being less than one unit of the last digit, as a computation with more digits shows. The 10S-value is -0.05006265674. ■

Errors of Numeric Results

Final results of computations of unknown quantities generally are **approximations**; that is, they are not exact but involve errors. Such an error may result from a combination of the following effects. **Roundoff errors** result from rounding, as discussed above. **Experimental errors** are errors of given data (probably arising from measurements). **Truncating errors** result from truncating (prematurely breaking off), for instance, if we replace a Taylor series with the sum of its first few terms. These errors depend on the computational method used and must be dealt with individually for each method. [“Truncating” is sometimes used as a term for chopping off (see before), a terminology that is not recommended.]

Formulas for Errors. If $\tilde{a}$ is an approximate value of a quantity whose exact value is a, we call the difference

$$\epsilon = a - \tilde{a} \tag{6}$$

the **error** of $\tilde{a}$. Hence

$$a = \tilde{a} + \epsilon, \qquad \text{True value} = \text{Approximation} + \text{Error}. \tag{6*}$$

For instance, if $\tilde{a} = 10.5$ is an approximation of $a = 10.2$, its error is $\epsilon = -0.3$. The error of an approximation $\tilde{a} = 1.60$ of $a = 1.82$ is $\epsilon = 0.22$.

CAUTION! In the literature $|a - \tilde{a}|$ (“absolute error”) or $\tilde{a} - a$ are sometimes also used as definitions of error.

The **relative error** ϵ_r of $\tilde{a}$ is defined by

$$\epsilon_r = \frac{\epsilon}{a} = \frac{a - \tilde{a}}{a} = \frac{\text{Error}}{\text{True value}} \qquad (a \neq 0). \tag{7}$$

This looks useless because a is unknown. But if $|\epsilon|$ is much less than $|\tilde{a}|$, then we can use $\tilde{a}$ instead of a and get

$$\epsilon_r \approx \frac{\epsilon}{\tilde{a}}. \tag{7'}$$

This still looks problematic because ϵ is unknown—if it were known, we could get $a = \tilde{a} + \epsilon$ from (6) and we would be done. But what one often can obtain in practice is an **error bound** for $\tilde{a}$, that is, a number β such that

$$|\epsilon| \leqq \beta, \qquad \text{hence} \qquad |a - \tilde{a}| \leqq \beta.$$

This tells us how far away from our computed $\tilde{a}$ the unknown a can at most lie. Similarly, for the relative error, an error bound is a number β_r such that

$$|\epsilon_r| \leqq \beta_r, \qquad \text{hence} \qquad \left|\frac{a - \tilde{a}}{a}\right| \leqq \beta_r.$$

Error Propagation

This is an important matter. It refers to how errors at the beginning and in later steps (roundoff, for example) propagate into the computation and affect accuracy, sometimes very drastically. We state here what happens to error bounds. Namely, bounds for the *error* add under addition and subtraction, whereas bounds for the *relative error* add under multiplication and division. You do well to keep this in mind.

THEOREM 1

Error Propagation

(a) *In addition and subtraction, a bound for the* ***error*** *of the results is given by the sum of the error bounds for the terms.*

(b) *In multiplication and division, an error bound for the* ***relative error*** *of the results is given (approximately) by the sum of the bounds for the relative errors of the given numbers.*

PROOF **(a)** We use the notations $x = \tilde{x} + \epsilon_x$, $y = \tilde{y} + \epsilon_y$, $|\epsilon_x| \leqq \beta_x$, $|\epsilon_y| \leqq \beta_y$. Then for the error ϵ of the *difference* we obtain

$$\begin{aligned} |\epsilon| &= |x - y - (\tilde{x} - \tilde{y})| \\ &= |x - \tilde{x} - (y - \tilde{y})| \\ &= |\epsilon_x - \epsilon_y| \leqq |\epsilon_x| + |\epsilon_y| \leqq \beta_x + \beta_y. \end{aligned}$$

The proof for the *sum* is similar and is left to the student.

(b) For the relative error ϵ_r of $\tilde{x}\tilde{y}$ we get from the relative errors ϵ_{rx} and ϵ_{ry} of $\tilde{x}, \tilde{y}$ and bounds β_{rx}, β_{ry}

$$\begin{aligned} |\epsilon_r| &= \left|\frac{xy - \tilde{x}\tilde{y}}{xy}\right| = \left|\frac{xy - (x - \epsilon_x)(y - \epsilon_y)}{xy}\right| = \left|\frac{\epsilon_x y + \epsilon_y x - \epsilon_x \epsilon_y}{xy}\right| \\ &\approx \left|\frac{\epsilon_x y + \epsilon_y x}{xy}\right| \leqq \left|\frac{\epsilon_x}{x}\right| + \left|\frac{\epsilon_y}{y}\right| = |\epsilon_{rx}| + |\epsilon_{ry}| \leqq \beta_{rx} + \beta_{ry}. \end{aligned}$$

This proof shows what “approximately” means: we neglected $\epsilon_x \epsilon_y$ as small in absolute value compared to $|\epsilon_x|$ and $|\epsilon_y|$. The proof for the quotient is similar but slightly more tricky (see Prob. 13). ■

Basic Error Principle

Every numeric method should be accompanied by an error estimate. If such a formula is lacking, is extremely complicated, or is impractical because it involves information (for instance, on derivatives) that is not available, the following may help.

Error Estimation by Comparison. *Do a calculation twice with different accuracy. Regard the difference $\tilde{a}_2 - \tilde{a}_1$ of the results $\tilde{a}_1, \tilde{a}_2$ as a (perhaps crude) estimate of the error ϵ_1 of the inferior result $\tilde{a}_1$.* Indeed, $\tilde{a}_1 + \epsilon_1 = \tilde{a}_2 + \epsilon_2$ by formula (4*). This implies $\tilde{a}_2 - \tilde{a}_1 = \epsilon_1 - \epsilon_2 \approx \epsilon_1$ because $\tilde{a}_2$ is generally more accurate than $\tilde{a}_1$, so that $|\epsilon_2|$ is small compared to $|\epsilon_1|$.

Algorithm. Stability

Numeric methods can be formulated as algorithms. An **algorithm** is a step-by-step procedure that states a numeric method in a form (a "**pseudocode**") understandable to humans. (See Table 19.1 to see what an algorithm looks like.) The algorithm is then used to write a program in a programming language that the computer can understand so that it can execute the numeric method. Important algorithms follow in the next sections. For routine tasks your CAS or some other software system may contain programs that you can use or include as parts of larger programs of your own.

Stability. To be useful, an algorithm should be **stable**; that is, small changes in the initial data should cause only small changes in the final results. However, if small changes in the initial data can produce large changes in the final results, we call the algorithm **unstable**.

This "*numeric instability*," which in most cases can be avoided by choosing a better algorithm, must be distinguished from "*mathematical instability*" of a problem, which is called "*ill-conditioning*," a concept we discuss in the next section.

Some algorithms are stable only for certain initial data, so that one must be careful in such a case.

PROBLEM SET 19.1

1. **Floating point.** Write 84.175, -528.685, 0.000924138, and -362005 in floating-point form, rounded to 5S (5 significant digits).
2. Write -76.437125, 60100, and -0.00001 in floating-point form, rounded to 4S.
3. **Small differences of large numbers** may be particularly strongly affected by rounding errors. Illustrate this by computing $0.81534/(35 \cdot 724 - 35.596)$ as given with 5S, then rounding stepwise to 4S, 3S, and 2S, where "stepwise" means round the rounded numbers, not the given ones.
4. **Order of terms**, in adding with a fixed number of digits, will generally affect the sum. Give an example. Find empirically a rule for the best order.
5. **Rounding and adding.** Let $a_1, \cdots, a_n$ be numbers with a_j correctly rounded to S_j digits. In calculating the sum $a_1 + \cdots + a_n$, retaining $S = \min S_j$ significant digits, is it essential that we first add and then round the result or that we first round each number to S significant digits and then add?
6. **Nested form.** Evaluate

$$f(x) = x^3 - 7.5x^2 + 11.2x + 2.8$$
$$= ((x - 7.5)x + 11.2)x + 2.8$$

at $x = 3.94$ using 3S arithmetic and rounding, in both of the given forms. The latter, called the *nested form*, is usually preferable since it minimizes the number of operations and thus the effect of rounding.

7. **Quadratic equation.** Solve $x^2 - 30x + 1 = 0$ by (4) and by (5), using 6S in the computation. Compare and comment.

8. Solve $x^2 - 40x + 2 = 0$, using 4S-computation.

9. Do the computations in Prob. 7 with 4S and 2S.

10. **Instability.** For small $|a|$ the equation $(x - k)^2 = a$ has nearly a double root. Why do these roots show instability?

11. **Theorems on errors.** Prove Theorem 1(a) for addition.

12. **Overflow and underflow** can sometimes be avoided by simple changes in a formula. Explain this in terms of $\sqrt{x^2 + y^2} = x\sqrt{1 + (y/x)^2}$ with $x^2 \geqq y^2$ and x so large that x^2 would cause overflow. Invent examples of your own.

13. **Division.** Prove Theorem 1(b) for division.

14. **Loss of digits. Square root.** Compute $\sqrt{x^2 + 4} - 2$ with 6S arithmetic for $x = 0.001$ **(a)** as given and **(b)** from $x^2/(\sqrt{x^2 + 4} + 2)$ (derive!).

15. **Logarithm.** Compute $\ln a - \ln b$ with 6S arithmetic for $a = 4.00000$ and $b = 3.99900$ **(a)** as given and **(b)** from $\ln(a/b)$.

16. **Cosine.** Compute $1 - \cos x$ with 6S arithmetic for $x = 0.02$ **(a)** as given and **(b)** by $2 \sin^2 \frac{1}{2}x$ (derive!).

17. Discuss the numeric use of (12) in App. A3.1 for $\cos v - \cos u$ when $u \approx v$.

18. **Quotient near 0/0. (a)** Compute $(1 - \cos x)/\sin x$ with 6S arithmetic for $x = 0.005$. **(b)** Looking at Prob. 16, find a much better formula.

19. **Exponential function.** Calculate $1/e = 0.367879$ (6S) from the partial sums of 5–10 terms of the Maclaurin series **(a)** of e^{-x} with $x = 1$, **(b)** of e^x with $x = 1$ and then taking the reciprocal. Which is more accurate?

20. Compute e^{-10} with 6S arithmetic in two ways (as in Prob. 19).

21. **Binary conversion.** Show that

$$\begin{aligned} 23 &= 20 \cdot 10^1 + 3 \cdot 10^0 = 16 + 4 + 2 + 1 \\ &= 2^4 + 2^2 + 2^1 + 2^0 = (1 \quad 0 \quad 1 \quad 1 \quad 1.)_2 \end{aligned}$$

can be obtained by the division algorithm

$$\begin{array}{lll} 2\,\underline{|23} & \text{Remainder} & 1 = c_0 \\ 2\,\underline{|11} & & 1 = c_1 \\ 2\,\underline{|5\ } & & 1 = c_2 \\ 2\,\underline{|2\ } & & 0 = c_3 \\ \ \ 0 & & 1 = c_4 \end{array}$$

22. Convert $(0.59375)_{10}$ to $(0.10011)_2$ by successive multiplication by 2 and dropping (removing) the integer parts, which give the binary digits $c_1, c_2, \cdots$:

$$\begin{array}{rl} & 0\ .59375 \cdot 2 \\ c_1 = & \boxed{1}\ .1875 \cdot 2 \\ c_2 = & \boxed{0}\ .375 \cdot 2 \\ c_3 = & \boxed{0}\ .75 \cdot 2 \\ c_4 = & \boxed{1}\ .5 \cdot 2 \\ c_5 = & \boxed{1}\ .0 \end{array}$$

23. Show that 0.1 is not a binary machine number.

24. Prove that any binary machine number has a finite decimal representation. Is the converse true?

25. **CAS EXPERIMENT. Approximations.** Obtain $x = 0.1 = \frac{3}{2}\sum_{m=1}^{\infty} 2^{-4m}$ from Prob. 23. Which machine number (partial sum) S_n will first have the value 0.1 to 30 decimal digits?

26. **CAS EXPERIMENT. Integration from Calculus.** Integrating by parts, show that $I_n = \int_0^1 e^x x^n\, dx = e - nI_{n-1}$, $I_0 = e - 1$. **(a)** Compute I_n, $n = 0, \cdots$, using 4S arithmetic, obtaining $I_8 = -3.906$. Why is this nonsense? Why is the error so large?

    (b) Experiment in (a) with the number of digits $k > 4$. As you increase k, will the first negative value $n = N$ occur earlier or later? Find an empirical formula for $N = N(k)$.

27. **Backward Recursion. In Prob. 26.** Using $e^x < e$ $(0 < x < 1)$, conclude that $|I_n| \leqq e/(n + 1) \to 0$ as $n \to \infty$. Solve the iteration formula for $I_{n-1} = (e - I_n)/n$, start from $I_{15} \approx 0$ and compute 4S values of $I_{14}, I_{13}, \cdots, I_1$.

28. **Harmonic series.** $1 + \frac{1}{2} + \frac{1}{3} + \cdots$ diverges. Is the same true for the corresponding series of computer numbers?

29. **Approximations of $\pi = 3.14159265358979 \cdots$** are $22/7$ and $355/113$. Determine the corresponding errors and relative errors to 3 significant digits.

30. **Compute π by Machin's approximation** $16 \arctan(\frac{1}{5}) - 4 \arctan(\frac{1}{239})$ to 10S (which are correct). [In 1986, D. H. Bailey (NASA Ames Research Center, Moffett Field, CA 94035) computed almost 30 million decimals of π on a CRAY-2 in less than 30 hrs. The race for more and more decimals is continuing. See the Internet under pi.]

19.2 Solution of Equations by Iteration

For each of the remaining sections of this chapter, we select basic kinds of problems and discuss numeric methods on how to solve them. The reader will learn about a variety of important problems and become familiar with ways of thinking in numerical analysis.

Perhaps the easiest conceptual problem is to find solutions of a single equation

$$f(x) = 0, \tag{1}$$

where f is a given function. A **solution** of (1) is a number $x = s$ such that $f(s) = 0$. Here, s suggests "solution," but we shall also use other letters.

It is interesting to note that the task of solving (1) is a question made for numeric algorithms, as in general there are no direct formulas, except in a few simple cases.

Examples of single equations are $x^3 + x = 1$, $\sin x = 0.5x$, $\tan x = x$, $\cosh x = \sec x$, $\cosh x \cos x = -1$, which can all be written in the form of (1). The first of the five equations is an **algebraic equation** because the corresponding f is a polynomial. In this case the solutions are called **roots** of the equation and the solution process is called *finding roots*. The other equations are **transcendental equations** because they involve transcendental functions.

There are a very large number of applications in engineering, where we have to solve a single equation (1). You have seen such applications when solving characteristic equations in Chaps. 2, 4, and 8; partial fractions in Chap. 6; residue integration in Chap. 16, finding eigenvalues in Chap. 12, and finding zeros of Bessel functions, also in Chap. 12. Moreover, methods of finding roots are very important in areas outside of classical engineering. For example, in finance, the problem of determining how much a bond is worth amounts to solving an algebraic equation.

To solve (1) when there is no formula for the exact solution available, we can use an approximation method, such as an **iteration method**. This is a method in which we start from an initial guess x_0 (which may be poor) and compute step by step (in general better and better) approximations $x_1, x_2, \cdots$ of an unknown solution of (1). We discuss three such methods that are of particular practical importance and mention two others in the problem set.

It is very important that the reader understand these methods and their underlying ideas. The reader will then be able to select judiciously the appropriate software from among different software packages that employ variations of such methods and not just treat the software programs as "black boxes."

In general, iteration methods are easy to program because the computational operations are the same in each step—just the data change from step to step—and, more importantly, if in a concrete case a method converges, it is stable in general (see Sec. 19.1).

Fixed-Point Iteration for Solving Equations $f(x) = 0$

Note: Our present use of the word "fixed point" has absolutely nothing to do with that in the last section.

By some *algebraic steps* we transform (1) into the form

$$x = g(x). \tag{2}$$

Then we choose an x_0 and compute $x_1 = g(x_0)$, $x_2 = g(x_1)$, and in general

$$x_{n+1} = g(x_n) \qquad (n = 0, 1, \cdots). \tag{3}$$

A solution of (2) is called a **fixed point** of g, motivating the name of the method. This is a solution of (1), since from $x = g(x)$ we can return to the original form $f(x) = 0$. From (1) we may get several different forms of (2). The behavior of corresponding iterative sequences $x_0, x_1, \cdots$ may differ, in particular, with respect to their speed of convergence. Indeed, some of them may not converge at all. Let us illustrate these facts with a simple example.

EXAMPLE 1 An Iteration Process (Fixed-Point Iteration)

Set up an iteration process for the equation $f(x) = x^2 - 3x + 1 = 0$. Since we know the solutions

$$x = 1.5 \pm \sqrt{1.25}, \qquad \text{thus} \qquad 2.618034 \qquad \text{and} \qquad 0.381966,$$

we can watch the behavior of the error as the iteration proceeds.

Solution. The equation may be written

$$x = g_1(x) = \tfrac{1}{3}(x^2 + 1), \qquad \text{thus} \qquad x_{n+1} = \tfrac{1}{3}(x_n^2 + 1). \tag{4a}$$

If we choose $x_0 = 1$, we obtain the sequence (Fig. 426a; computed with 6S and then rounded)

$$x_0 = 1.000, \quad x_1 = 0.667, \quad x_2 = 0.481, \quad x_3 = 0.411, \quad x_4 = 0.390, \cdots$$

which seems to approach the smaller solution. If we choose $x_0 = 2$, the situation is similar. If we choose $x_0 = 3$, we obtain the sequence (Fig. 426a, upper part)

$$x_0 = 3.000, \quad x_1 = 3.333, \quad x_2 = 4.037, \quad x_3 = 5.766, \quad x_4 = 11.415, \cdots$$

which diverges.

Our equation may also be written (divide by x)

$$x = g_2(x) = 3 - \frac{1}{x}, \qquad \text{thus} \qquad x_{n+1} = 3 - \frac{1}{x_n}, \tag{4b}$$

and if we choose $x_0 = 1$, we obtain the sequence (Fig. 426b)

$$x_0 = 1.000, \quad x_1 = 2.000, \quad x_2 = 2.500, \quad x_3 = 2.600, \quad x_4 = 2.615, \cdots$$

which seems to approach the larger solution. Similarly, if we choose $x_0 = 3$, we obtain the sequence (Fig. 426b)

$$x_0 = 3.000, \quad x_1 = 2.667, \quad x_2 = 2.625, \quad x_3 = 2.619, \quad x_4 = 2.618, \cdots.$$

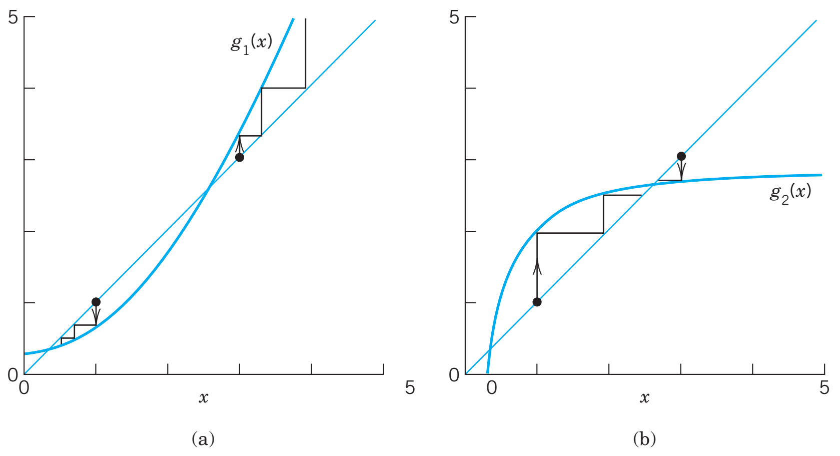

Fig. 426. Example 1, iterations (4a) and (4b)

Our figures show the following. In the lower part of Fig. 426a the slope of $g_1(x)$ is less than the slope of $y = x$, which is 1, thus $|g_1'(x)| < 1$, and we seem to have convergence. In the upper part, $g_1(x)$ is steeper ($g_1'(x) > 1$) and we have divergence. In Fig. 426b the slope of $g_2(x)$ is less near the intersection point ($x = 2.618$, fixed point of g_2, solution of $f(x) = 0$), and both sequences seem to converge. From all this we conclude that convergence seems to depend on the fact that, in a neighborhood of a solution, the curve of $g(x)$ is less steep than the straight line $y = x$, and we shall now see that this condition $|g'(x)| < 1$ (= slope of $y = x$) is sufficient for convergence. ■

An iteration process defined by (3) is called **convergent** for an x_0 if the corresponding sequence $x_0, x_1, \cdots$ is convergent.

A sufficient condition for convergence is given in the following theorem, which has various practical applications.

THEOREM 1

Convergence of Fixed-Point Iteration

Let $x = s$ be a solution of $x = g(x)$ and suppose that g has a continuous derivative in some interval J containing s. Then, if $|g'(x)| \leqq K < 1$ in J, the iteration process defined by (3) *converges for any x_0 in J. The limit of the sequence $\{x_n\}$ is s.*

PROOF By the mean value theorem of differential calculus there is a t between x and s such that

$$g(x) - g(s) = g'(t)(x - s) \qquad (x \text{ in } J).$$

Since $g(s) = s$ and $x_1 = g(x_0), x_2 = g(x_1), \cdots$, we obtain from this and the condition on $|g'(x)|$ in the theorem

$$|x_n - s| = |g(x_{n-1}) - g(s)| = |g'(t)||x_{n-1} - s| \leqq K|x_{n-1} - s|.$$

Applying this inequality n times, for $n, n - 1, \cdots, 1$ gives

$$|x_n - s| \leqq K|x_{n-1} - s| \leqq K^2|x_{n-2} - s| \leqq \cdots \leqq K^n|x_0 - s|.$$

Since $K < 1$, we have $K^n \to 0$; hence $|x_n - s| \to 0$ as $n \to \infty$. ■

We mention that a function g satisfying the condition in Theorem 1 is called a **contraction** because $|g(x) - g(v)| \leqq K|x - v|$, where $K < 1$. Furthermore, K gives information on the speed of convergence. For instance, if $K = 0.5$, then the accuracy increases by at least 2 digits in only 7 steps because $0.5^7 < 0.01$.

EXAMPLE 2 **An Iteration Process. Illustration of Theorem 1**

Find a solution of $f(x) = x^3 + x - 1 = 0$ by iteration.

Solution. A sketch shows that a solution lies near $x = 1$. **(a)** We may write the equation as $(x^2 + 1)x = 1$ or

$$x = g_1(x) = \frac{1}{1 + x^2}, \qquad \text{so that} \qquad x_{n+1} = \frac{1}{1 + x_n^2}. \qquad \text{Also} \qquad |g_1'(x)| = \frac{2|x|}{(1 + x^2)^2} < 1$$

for any x because $4x^2/(1 + x^2)^4 = 4x^2/(1 + 4x^2 + \cdots) < 1$, so that by Theorem 1 we have convergence for any x_0. Choosing $x_0 = 1$, we obtain (Fig. 427)

$$x_1 = 0.500, \quad x_2 = 0.800, \quad x_3 = 0.610, \quad x_4 = 0.729, \quad x_5 = 0.653, \quad x_6 = 0.701, \cdots.$$

The solution exact to 6D is $s = 0.682328$.

(b) The given equation may also be written

$$x = g_2(x) = 1 - x^3. \qquad \text{Then} \qquad |g_2'(x)| = 3x^2$$

and this is greater than 1 near the solution, so that we cannot apply Theorem 1 and assert convergence. Try $x_0 = 1, x_0 = 0.5, x_0 = 2$ and see what happens.

The example shows that the transformation of a given $f(x) = 0$ into the form $x = g(x)$ with g satisfying $|g'(x) \leqq K < 1$ may need some experimentation. ■

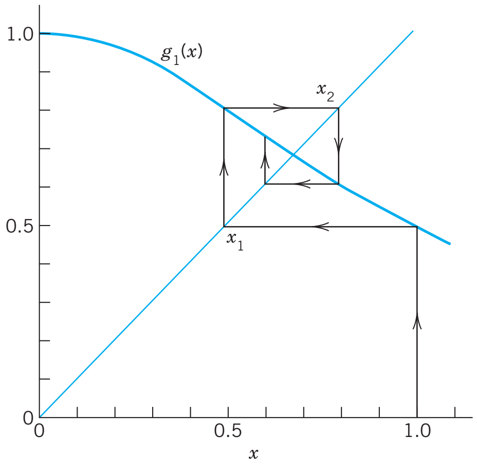

Fig. 427. Iteration in Example 2

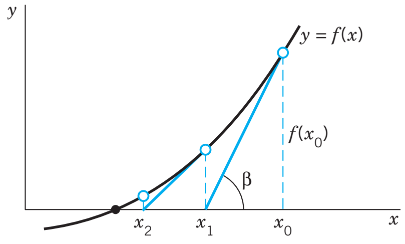

Fig. 428. Newton's method

Newton's Method for Solving Equations $f(x) = 0$

Newton's method, also known as **Newton–Raphson's method**,[1] is another iteration method for solving equations $f(x) = 0$, where f is assumed to have a continuous derivative f'. The method is commonly used because of its simplicity and great speed.

The underlying idea is that we approximate the graph of f by suitable tangents. Using an approximate value x_0 obtained from the graph of f, we let x_1 be the point of intersection of the x-axis and the tangent to the curve of f at x_0 (see Fig. 428). Then

$$\tan \beta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1}, \qquad \text{hence} \qquad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

In the second step we compute $x_2 = x_1 - f(x_1)/f'(x_1)$, in the third step x_3 from x_2 again by the same formula, and so on. We thus have the algorithm shown in Table 19.1. Formula (5) in this algorithm can also be obtained if we algebraically solve Taylor's formula

$$(5^*) \qquad f(x_{n+1}) \approx f(x_n) + (x_{n+1} - x_n)f'(x_n) = 0.$$

[1]JOSEPH RAPHSON (1648–1715), English mathematician who published a method similar to Newton's method. For historical details, see Ref. [GenRef2], p. 203, listed in App. 1.

Table 19.1 Newton's Method for Solving Equations $f(x) = 0$

```
ALGORITHM NEWTON ($f, f', x_0, \epsilon, N$)

This algorithm computes a solution of $f(x) = 0$ given an initial approximation $x_0$ (starting
value of the iteration). Here the function $f(x)$ is continuous and has a continuous
derivative $f'(x)$.

    INPUT:  $f$, $f'$, initial approximation $x_0$, tolerance $\epsilon > 0$, maximum number of
            iterations $N$.

    OUTPUT: Approximate solution $x_n$ ($n \leqq N$) or message of failure.

    For $n = 0, 1, 2, \cdots, N - 1$ do:
1           Compute $f'(x_n)$.
2           If $f'(x_n) = 0$ then OUTPUT "Failure." Stop.
                [Procedure completed unsuccessfully]
3           Else compute
            (5)        $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.
4           If $|x_{n+1} - x_n| \leqq \epsilon |x_{n+1}|$ then OUTPUT $x_{n+1}$. Stop.
                [Procedure completed successfully]
    End
5   OUTPUT "Failure". Stop.
                [Procedure completed unsuccessfully after N iterations]

End NEWTON
```

If it happens that $f'(x_n) = 0$ for some n (see line 2 of the algorithm), then try another starting value x_0. Line 3 is the heart of Newton's method.

The inequality in line 4 is a **termination criterion**. If the sequence of the x_n converges and the criterion holds, we have reached the desired accuracy and stop. Note that this is just a form of the relative error test. It ensures that the result has the desired number of significant digits. If $|x_{n+1}| = 0$, the condition is satisfied if and only if $x_{n+1} = x_n = 0$, otherwise $|x_{n+1} - x_n|$ must be sufficiently small. The factor $|x_{n+1}|$ is needed in the case of zeros of very small (or very large) absolute value because of the high density (or of the scarcity) of machine numbers for those x.

WARNING! The criterion by itself does not imply convergence. *Example.* The harmonic series diverges, although its partial sums $x_n = \Sigma_{k=1}^{n} 1/k$ satisfy the criterion because $\lim (x_{n+1} - x_n) = \lim (1/(n + 1)) = 0$.

Line 5 gives another termination criterion and is needed because Newton's method may diverge or, due to a poor choice of x_0, may not reach the desired accuracy by a reasonable number of iterations. Then we may try another x_0. If $f(x) = 0$ has more than one solution, different choices of x_0 may produce different solutions. Also, an iterative sequence may sometimes converge to a solution different from the expected one.

EXAMPLE 3 Square Root

Set up a Newton iteration for computing the square root x of a given positive number c and apply it to $c = 2$.

Solution. We have $x = \sqrt{c}$, hence $f(x) = x^2 - c = 0, f'(x) = 2x$, and (5) takes the form

$$x_{n+1} = x_n - \frac{x_n^2 - c}{2x_n} = \frac{1}{2}\left(x_n + \frac{c}{x_n}\right).$$

For $c = 2$, choosing $x_0 = 1$, we obtain

$$x_1 = 1.500000, \qquad x_2 = 1.416667, \qquad x_3 = 1.414216, \qquad x_4 = 1.414214, \cdots.$$

x_4 is exact to 6D. ■

EXAMPLE 4 Iteration for a Transcendental Equation

Find the positive solution of $2 \sin x = x$.

Solution. Setting $f(x) = x - 2 \sin x$, we have $f'(x) = 1 - 2 \cos x$, and (5) gives

$$x_{n+1} = x_n - \frac{x_n - 2 \sin x_n}{1 - 2 \cos x_n} = \frac{2(\sin x_n - x_n \cos x_n)}{1 - 2 \cos x_n} = \frac{N_n}{D_n}.$$

n	x_n	N_n	D_n	x_{n+1}
0	2.00000	3.48318	1.83229	1.90100
1	1.90100	3.12470	1.64847	1.89552
2	1.89552	3.10500	1.63809	1.89550
3	1.89550	3.10493	1.63806	1.89549

From the graph of f we conclude that the solution is near $x_0 = 2$. We compute:
$x_4 = 1.89549$ is exact to 5D since the solution to 6D is 1.895494. ■

EXAMPLE 5 Newton's Method Applied to an Algebraic Equation

Apply Newton's method to the equation $f(x) = x^3 + x - 1 = 0$.

Solution. From (5) we have

$$x_{n+1} = x_n - \frac{x_n^3 + x_n - 1}{3x_n^2 + 1} = \frac{2x_n^3 + 1}{3x_n^2 + 1}.$$

Starting from $x_0 = 1$, we obtain

$$x_1 = 0.750000, \quad x_2 = 0.686047, \quad x_3 = 0.682340, \quad x_4 = 0.682328, \cdots$$

where x_4 has the error $-1 \cdot 10^{-6}$. A comparison with Example 2 shows that the present convergence is much more rapid. This may motivate the concept of the *order of an iteration process,* to be discussed next. ■

Order of an Iteration Method. Speed of Convergence

The quality of an iteration method may be characterized by the speed of convergence, as follows.

Let $x_{n+1} = g(x_n)$ define an iteration method, and let x_n approximate a solution s of $x = g(x)$. Then $x_n = s - \epsilon_n$, where ϵ_n is the error of x_n. Suppose that g is differentiable a number of times, so that the Taylor formula gives

$$\begin{aligned} x_{n+1} = g(x_n) &= g(s) + g'(s)(x_n - s) + \tfrac{1}{2} g''(s)(x_n - s)^2 + \cdots \\ &= g(s) - g'(s)\epsilon_n + \tfrac{1}{2} g''(s)\epsilon_n^2 + \cdots . \end{aligned} \tag{6}$$

The exponent of ϵ_n in the first nonvanishing term after $g(s)$ is called the **order** of the iteration process defined by g. The order measures the speed of convergence.

To see this, subtract $g(s) = s$ on both sides of (6). Then on the left you get $x_{n+1} - s = -\epsilon_{n+1}$, where ϵ_{n+1} is the error of x_{n+1}. And on the right the remaining expression equals approximately its first nonzero term because $|\epsilon_n|$ is small in the case of convergence. Thus

$$\begin{aligned} &\text{(a)} \quad \epsilon_{n+1} \approx +g'(s)\epsilon_n && \text{in the case of first order,} \\ &\text{(b)} \quad \epsilon_{n+1} \approx -\tfrac{1}{2} g''(s)\epsilon_n^2 && \text{in the case of second order,} \qquad \text{etc.} \end{aligned} \tag{7}$$

Thus if $\epsilon_n = 10^{-k}$ in some step, then for second order, $\epsilon_{n+1} = c \cdot (10^{-k})^2 = c \cdot 10^{-2k}$, so that the number of significant digits is about doubled in each step.

Convergence of Newton's Method

In Newton's method, $g(x) = x - f(x)/f'(x)$. By differentiation,

$$\begin{aligned} g'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} \\ &= \frac{f(x)f''(x)}{f'(x)^2} . \end{aligned} \tag{8}$$

Since $f(s) = 0$, this shows that also $g'(s) = 0$. Hence Newton's method is at least of second order. If we differentiate again and set $x = s$, we find that

$$g''(s) = \frac{f''(s)}{f'(s)} \tag{8*}$$

which will not be zero in general. This proves

THEOREM 2

Second-Order Convergence of Newton's Method

If $f(x)$ is three times differentiable and f' and f'' are not zero at a solution s of $f(x) = 0$, then for x_0 sufficiently close to s, Newton's method is of second order.

Comments. For Newton's method, (7b) becomes, by (8*),

$$\epsilon_{n+1} \approx -\frac{f''(s)}{2f'(s)}\epsilon_n^2. \tag{9}$$

For the rapid convergence of the method indicated in Theorem 2 it is important that s be a *simple* zero of $f(x)$ (thus $f'(s) \neq 0$) and that x_0 be close to s, because in Taylor's formula we took only the linear term [see (5*)], assuming the quadratic term to be negligibly small. (With a bad x_0 the method may even diverge!)

EXAMPLE 6 Prior Error Estimate of the Number of Newton Iteration Steps

Use $x_0 = 2$ and $x_1 = 1.901$ in Example 4 for estimating how many iteration steps we need to produce the solution to 5D-accuracy. This is an **a priori estimate** or **prior estimate** because we can compute it after only one iteration, prior to further iterations.

Solution. We have $f(x) = x - 2\sin x = 0$. Differentiation gives

$$\frac{f''(s)}{2f'(s)} \approx \frac{f''(x_1)}{2f'(x_1)} = \frac{2\sin x_1}{2(1 - 2\cos x_1)} \approx 0.57.$$

Hence (9) gives

$$|\epsilon_{n+1}| \approx 0.57\epsilon_n^2 \approx 0.57(0.57\epsilon_{n-1}^2)^2 = 0.57^3\epsilon_{n-1}^4 \approx \cdots \approx 0.57^M\epsilon_0^{M+1} \leqq 5 \cdot 10^{-6}$$

where $M = 2^n + 2^{n-1} + \cdots + 2 + 1 = 2^{n+1} - 1$. We show below that $\epsilon_0 \approx -0.11$. Consequently, our condition becomes

$$0.57^M 0.11^{M+1} \leqq 5 \cdot 10^{-6}.$$

Hence $n = 2$ is the smallest possible n, according to this crude estimate, in good agreement with Example 4.

$\epsilon_0 \approx -0.11$ is obtained from $\epsilon_1 - \epsilon_0 = (\epsilon_1 - s) - (\epsilon_0 - s) = -x_1 + x_0 \approx 0.10$, hence $\epsilon_1 = \epsilon_0 + 0.10 \approx -0.57\epsilon_0^2$ or $0.57\epsilon_0^2 + \epsilon_0 + 0.10 \approx 0$, which gives $\epsilon_0 \approx -0.11$. ■

Difficulties in Newton's Method. Difficulties may arise if $|f'(x)|$ is very small near a solution s of $f(x) = 0$. For instance, let s be a zero of $f(x)$ of second or higher order. Then Newton's method converges only linearly, as is shown by an application of l'Hopital's rule to (8). Geometrically, small $|f'(x)|$ means that the tangent of $f(x)$ near s almost coincides with the x-axis (so that double precision may be needed to get $f(x)$ and $f'(x)$ accurately enough). Then for values $x = \tilde{s}$ far away from s we can still have small function values

$$R(\tilde{s}) = f(\tilde{s}).$$

In this case we call the equation $f(x) = 0$ **ill-conditioned**. $R(\tilde{s})$ is called the **residual** of $f(x) = 0$ at $\tilde{s}$. Thus a small residual guarantees a small error of $\tilde{s}$ only if the equation is ***not*** ill-conditioned.

EXAMPLE 7 An Ill-Conditioned Equation

$f(x) = x^5 + 10^{-4}x = 0$ is ill-conditioned, $x = 0$ is a solution. $f'(0) = 10^{-4}$ is small. At $\tilde{s} = 0.1$ the residual $f(0.1) = 2 \cdot 10^{-5}$ is small, but the error -0.1 is larger in absolute value by a factor 5000. Invent a more drastic example of your own. ■

Secant Method for Solving $f(x) = 0$

Newton's method is very powerful but has the disadvantage that the derivative f' may sometimes be a far more difficult expression than f itself and its evaluation therefore computationally expensive. This situation suggests the idea of replacing the derivative with the difference quotient

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} .$$

Then instead of (5) we have the formula of the popular secant method

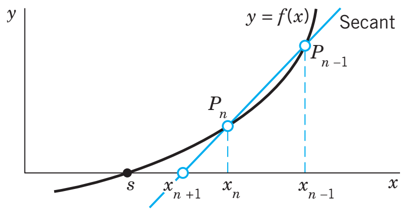

Fig. 429. Secant method

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} . \tag{10}$$

Geometrically, we intersect the x-axis at x_{n+1} with the secant of $f(x)$ passing through P_{n-1} and P_n in Fig. 429. We need two starting values x_0 and x_1. Evaluation of derivatives is now avoided. It can be shown that convergence is **superlinear** (that is, more rapid than linear, $|\epsilon_{n+1}| \approx \text{const} \cdot |\epsilon_n|^{1.62}$; see [E5] in App. 1), almost quadratic like Newton's method. The algorithm is similar to that of Newton's method, as the student may show.

CAUTION! It is ***not*** good to write (10) as

$$x_{n+1} = \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})} ,$$

because this may lead to loss of significant digits if x_n and x_{n-1} are about equal. (Can you see this from the formula?)

EXAMPLE 8 Secant Method

Find the positive solution of $f(x) = x - 2 \sin x = 0$ by the secant method, starting from $x_0 = 2, x_1 = 1.9$.

Solution. Here, (10) is

$$x_{n+1} = x_n - \frac{(x_n - 2 \sin x_n)(x_n - x_{n-1})}{x_n - x_{n-1} + 2(\sin x_{n-1} - \sin x_n)} = x_n - \frac{N_n}{D_n} .$$

Numeric values are:

n	x_{n-1}	x_n	N_n	D_n	$x_{n+1} - x_n$
1	2.000000	1.900000	−0.000740	−0.174005	−0.004253
2	1.900000	1.895747	−0.000002	−0.006986	−0.000252
3	1.895747	1.895494	0		0

$x_3 = 1.895494$ is exact to 6D. See Example 4. ■

Summary of Methods. The methods for computing solutions s of $f(x) = 0$ with given continuous (or differentiable) $f(x)$ start with an initial approximation x_0 of s and generate a sequence $x_1, x_2, \cdots$ by **iteration**. **Fixed-point methods** solve $f(x) = 0$ written as $x = g(x)$, so that s is a *fixed point* of g, that is, $s = g(s)$. For $g(x) = x - f(x)/f'(x)$ this is **Newton's method**, which, for good x_0 and simple zeros, converges quadratically (and for multiple zeros linearly). From Newton's method the **secant method** follows by replacing $f'(x)$ by a difference quotient. The ***bisection method*** and the ***method of false position*** in Problem Set 19.2 always converge, but often slowly.

PROBLEM SET 19.2

1–13 FIXED-POINT ITERATION

Solve by fixed-point iteration and answer related questions where indicated. Show details.

1. **Monotone sequence.** Why is the sequence in Example 1 monotone? Why not in Example 2?
2. Do the iterations (b) in Example 2. Sketch a figure similar to Fig. 427. Explain what happens.
3. $f = x - 0.5 \cos x = 0, \quad x_0 = 1$. Sketch a figure.
4. $f = x - \operatorname{cosec} x$ the zero near $x = 1$.
5. Sketch $f(x) = x^3 - 5.00x^2 + 1.01x + 1.88$, showing roots near ± 1 and 5. Write $x = g(x) = (5.00x^2 - 1.01x + 1.88)/x^2$. Find a root by starting from $x_0 = 5, 4, 1, -1$. Explain the (perhaps unexpected) results.
6. Find a form $x = g(x)$ of $f(x) = 0$ in Prob. 5 that yields convergence to the root near $x = 1$.
7. Find the smallest positive solution of $\sin x = e^{-x}$.
8. Solve $x^4 - x - 0.12 = 0$ by starting from $x_0 = 1$.
9. Find the negative solution of $x^4 - x - 0.12 = 0$.
10. **Elasticity.** Solve $x \cosh x = 1$. (Similar equations appear in vibrations of beams; see Problem Set 12.3.)
11. **Drumhead. Bessel functions.** A partial sum of the Maclaurin series of $J_0(x)$ (Sec. 5.5) is $f(x) = 1 - \frac{1}{4}x^2 + \frac{1}{64}x^4 - \frac{1}{2304}x^6$. Conclude from a sketch that $f(x) = 0$ near $x = 2$. Write $f(x) = 0$ as $x = g(x)$ (by dividing $f(x)$ by $\frac{1}{4}x$ and taking the resulting x-term to the other side). Find the zero. (See Sec. 12.10 for the importance of these zeros.)
12. **CAS EXPERIMENT. Convergence.** Let $f(x) = x^3 + 2x^2 - 3x - 4 = 0$. Write this as $x = g(x)$, for g choosing (1) $(x^3 - f)^{1/3}$, (2) $(x^2 - \frac{1}{2}f)^{1/2}$, (3) $x + \frac{1}{3}f$, (4) $(x^3 - f)/x^2$, (5) $(2x^2 - f)/(2x)$, and (6) $x - f/f'$ and in each case $x_0 = 1.5$. Find out about convergence and divergence and the number of steps to reach 6S-values of a root.
13. **Existence of fixed point.** Prove that if g is continuous in a closed interval I and its range lies in I, then the equation $x = g(x)$ has at least one solution in I. Illustrate that it may have more than one solution in I.

14–23 NEWTON'S METHOD

Apply Newton's method (6S-accuracy). First sketch the function(s) to see what is going on.

14. **Cube root.** Design a Newton iteration. Compute $\sqrt[3]{7}, x_0 = 2$.
15. $f = 2x - \cos x, \quad x_0 = 1$. Compare with Prob. 3.
16. What happens in Prob. 15 for any other x_0?
17. **Dependence on x_0.** Solve Prob. 5 by Newton's method with $x_0 = 5, 4, 1, -3$. Explain the result.
18. **Legendre polynomials.** Find the largest root of the Legendre polynomial $P_5(x)$ given by $P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x)$ (Sec. 5.3) (to be needed in ***Gauss integration*** in Sec. 19.5) **(a)** by Newton's method, **(b)** from a quadratic equation.
19. **Associated Legendre functions.** Find the smallest positive zero of $P_4^2 = (1 - x^2)P_4'' = \frac{15}{2}(-7x^4 + 8x^2 - 1)$ (Sec. 5.3) **(a)** by Newton's method, **(b)** exactly, by solving a quadratic equation.
20. $x + \ln x = 2, \quad x_0 = 2$
21. $f = x^3 - 5x + 3 = 0, \quad x_0 = 2, \quad 0, -2$
22. **Heating, cooling.** At what time x (4S-accuracy only) will the processes governed by $f_1(x) = 100(1 - e^{-0.2x})$ and $f_2(x) = 40e^{-0.01x}$ reach the same temperature? Also find the latter.
23. **Vibrating beam.** Find the solution of $\cos x \cosh x = 1$ near $x = \frac{3}{2}\pi$. (This determines a frequency of a vibrating beam; see Problem Set 12.3.)
24. **Method of False Position (Regula falsi).** Figure 430 shows the idea. We assume that f is continuous. We compute the x-intercept c_0 of the line through $(a_0, f(a_0))$, $(b_0, f(b_0))$. If $f(c_0) = 0$, we are done. If $f(a_0)f(c_0) < 0$ (as in Fig. 430), we set $a_1 = a_0$, $b_1 = c_0$ and repeat to get c_1, etc. If $f(a_0)f(c_0) > 0$, then $f(c_0)f(b_0) < 0$ and we set $a_1 = c_0$, $b_1 = b_0$, etc.

    (a) Algorithm. Show that

    $$c_0 = \frac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}$$

    and write an algorithm for the method.

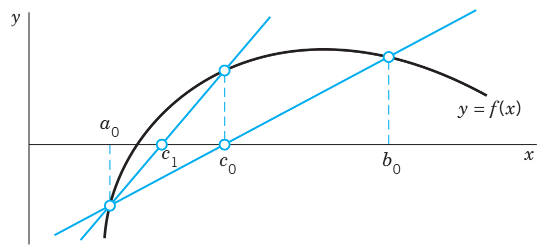

Fig. 430. Method of false position

(b) Solve $x^4 = 2$, $\cos x = \sqrt{x}$, and $x + \ln x = 2$, with $a = 1$, $b = 2$.

25. TEAM PROJECT. Bisection Method. This simple but slowly convergent method for finding a solution of $f(x) = 0$ with continuous f is based on the **intermediate value theorem**, which states that if a continuous function f has opposite signs at some $x = a$ and $x = b (> a)$, that is, either $f(a) < 0, f(b) > 0$ or $f(a) > 0, f(b) < 0$, then f must be 0 somewhere on $[a, b]$. The solution is found by repeated bisection of the interval and in each iteration picking that half which also satisfies that sign condition.

(a) Algorithm. Write an algorithm for the method.

(b) Comparison. Solve $x = \cos x$ by Newton's method and by bisection. Compare.

(c) Solve $e^{-x} = \ln x$ and $e^x + x^4 + x = 2$ by bisection.

26–29 SECANT METHOD

Solve, using x_0 and x_1 as indicated:

26. $e^{-x} - \tan x = 0, \quad x_0 = 1, \quad x_1 = 0.7$

27. Prob. 21, $x_0 = 1.0, \quad x_1 = 2.0$

28. $x = \cos x, \quad x_0 = 0.5, \quad x_1 = 1$

29. $\sin x = \cot x, \quad x_0 = 1, \quad x_1 = 0.5$

30. WRITING PROJECT. Solution of Equations. Compare the methods in this section and problem set, discussing advantages and disadvantages in terms of examples of your own. No proofs, just motivations and ideas.

19.3 Interpolation

We are given the values of a function $f(x)$ at different points $x_0, x_1, \cdots, x_n$. We want to find approximate values of the function $f(x)$ for "new" x's that lie between these points for which the function values are given. This process is called **interpolation**. The student should pay close attention to this section as interpolation forms the underlying foundation for both Secs. 19.4 and 19.5. Indeed, interpolation allows us to develop formulas for numeric integration and differentiation as shown in Sec. 19.5.

Continuing our discussion, we write these given values of a function f in the form

$$f_0 = f(x_0), \qquad f_1 = f(x_1), \qquad \cdots, \qquad f_n = f(x_n)$$

or as ordered pairs

$$(x_0, f_0), \qquad (x_1, f_1), \qquad \cdots, \quad (x_n, f_n).$$

Where do these given function values come from? They may come from a "mathematical" function, such as a logarithm or a Bessel function. More frequently, they may be measured or automatically recorded values of an "empirical" function, such as air resistance of a car or an airplane at different speeds. Other examples of functions that are "empirical" are the yield of a chemical process at different temperatures or the size of the U.S. population as it appears from censuses taken at 10-year intervals.

A standard idea in interpolation now is to find a polynomial $p_n(x)$ of degree n (or less) that assumes the given values; thus

$$\text{(1)} \qquad p_n(x_0) = f_0, \qquad p_n(x_1) = f_1, \qquad \cdots, \qquad p_n(x_n) = f_n.$$

We call this p_n an **interpolation polynomial** and $x_0, \cdots, x_n$ the **nodes**. And if $f(x)$ is a mathematical function, we call p_n an **approximation** of f (or a **polynomial approximation**, because there are other kinds of approximations, as we shall see later). We use p_n to get (approximate) values of f for x's between x_0 and x_n ("**interpolation**") or sometimes outside this interval $x_0 \leqq x \leqq x_n$ ("**extrapolation**").

Motivation. Polynomials are convenient to work with because we can readily differentiate and integrate them, again obtaining polynomials. Moreover, they approximate *continuous* functions with any desired accuracy. That is, for any continuous $f(x)$ on an interval J: $a \leqq x \leqq b$ and error bound $\beta > 0$, there is a polynomial $p_n(x)$ (of sufficiently high degree n) such that

$$|f(x) - p_n(x)| < \beta \quad \text{for all } x \text{ on } J.$$

This is the famous **Weierstrass approximation theorem** (for a proof see Ref. [GenRef7], App. 1).

Existence and Uniqueness. Note that the interpolation polynomial p_n satisfying (1) for given data exists and we shall give formulas for it below. Furthermore, p_n is unique: Indeed, if another polynomial q_n also satisfies $q_n(x_0) = f_0, \cdots, q_n(x_n) = f_n$, then $p_n(x) - q_n(x) = 0$ at $x_0, \cdots, x_n$, but a polynomial $p_n - q_n$ of degree n (or less) with $n + 1$ roots must be identically zero, as we know from algebra; thus $p_n(x) = q_n(x)$ for all x, which means uniqueness. ■

How Do We Find p_n? We shall explain several standard methods that give us p_n. By the uniqueness proof above, we know that, for given data, the different methods *must* give us the same polynomial. However, the polynomials may be expressed in different forms suitable for different purposes.

Lagrange Interpolation

Given $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$ with arbitrarily spaced x_j, Lagrange had the idea of multiplying each f_j by a polynomial that is 1 at x_j and 0 at the other n nodes and then taking the sum of these $n + 1$ polynomials. Clearly, this gives the unique interpolation polynomial of degree n or less. Beginning with the simplest case, let us see how this works.

Linear interpolation is interpolation by the straight line through (x_0, f_0), (x_1, f_1); see Fig. 431. Thus the linear Lagrange polynomial p_1 is a sum $p_1 = L_0 f_0 + L_1 f_1$ with L_0 the linear polynomial that is 1 at x_0 and 0 at x_1; similarly, L_1 is 0 at x_0 and 1 at x_1. Obviously,

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \qquad L_1(x) = \frac{x - x_0}{x_1 - x_0}.$$

This gives the linear Lagrange polynomial

$$p_1(x) = L_0(x) f_0 + L_1(x) f_1 = \frac{x - x_1}{x_0 - x_1} \cdot f_0 + \frac{x - x_0}{x_1 - x_0} \cdot f_1. \tag{2}$$

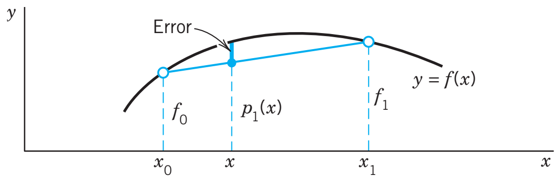

Fig. 431. Linear Interpolation

EXAMPLE 1 Linear Lagrange Interpolation

Compute a 4D-value of ln 9.2 from ln 9.0 = 2.1972, ln 9.5 = 2.2513 by linear Lagrange interpolation and determine the error, using ln 9.2 = 2.2192 (4D).

Solution. $x_0 = 9.0, x_1 = 9.5, f_0 = \ln 9.0, f_1 = \ln 9.5$. Ln (2) we need

$$L_0(x) = \frac{x - 9.5}{-0.5} = -2.0(x - 9.5), \qquad L_0(9.2) = -2.0(-0.3) = 0.6$$

$$L_1(x) = \frac{x - 9.0}{0.5} = 2.0(x - 9.0), \qquad L_1(9.2) = 2 \cdot 0.2 = 0.4$$

(see Fig. 432) and obtain the answer

$$\ln 9.2 \approx p_1(9.2) = L_0(9.2) f_0 + L_1(9.2) f_1 = 0.6 \cdot 2.1972 + 0.4 \cdot 2.2513 = 2.2188.$$

The error is $\epsilon = a - \tilde{a} = 2.2192 - 2.2188 = 0.0004$. Hence linear interpolation is not sufficient here to get 4D accuracy; it would suffice for 3D accuracy. ■

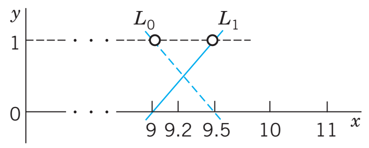

Fig. 432. L_0 and L_1 in Example 1

Quadratic interpolation is interpolation of given (x_0, f_0), (x_1, f_1), (x_2, f_2) by a second-degree polynomial $p_2(x)$, which by Lagrange's idea is

$$p_2(x) = L_0(x)f_0 + L_1(x)f_1 + L_2(x)f_2 \tag{3a}$$

with $L_0(x_0) = 1$, $L_1(x_1) = 1$, $L_2(x_2) = 1$, and $L_0(x_1) = L_0(x_2) = 0$, etc. We claim that

$$\begin{aligned} L_0(x) &= \frac{l_0(x)}{l_0(x_0)} = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \\ L_1(x) &= \frac{l_1(x)}{l_1(x_1)} = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \\ L_2(x) &= \frac{l_2(x)}{l_2(x_2)} = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}. \end{aligned} \tag{3b}$$

How did we get this? Well, the numerator makes $L_k(x_j) = 0$ if $j \neq k$. And the denominator makes $L_k(x_k) = 1$ because it equals the numerator at $x = x_k$.

EXAMPLE 2 Quadratic Lagrange Interpolation

Compute ln 9.2 by (3) from the data in Example 1 and the additional third value ln 11.0 = 2.3979.

Solution. In (3),

$$L_0(x) = \frac{(x - 9.5)(x - 11.0)}{(9.0 - 9.5)(9.0 - 11.0)} = x^2 - 20.5x + 104.5, \qquad L_0(9.2) = 0.5400,$$

$$L_1(x) = \frac{(x - 9.0)(x - 11.0)}{(9.5 - 9.0)(9.5 - 11.0)} = -\frac{1}{0.75}(x^2 - 20x + 99), \qquad L_1(9.2) = 0.4800,$$

$$L_2(x) = \frac{(x - 9.0)(x - 9.5)}{(11.0 - 9.0)(11.0 - 9.5)} = \frac{1}{3}(x^2 - 18.5x + 85.5), \qquad L_2(9.2) = -0.0200,$$

(see Fig. 433), so that (3a) gives, exact to 4D,

$$\ln 9.2 \approx p_2(9.2) = 0.5400 \cdot 2.1972 + 0.4800 \cdot 2.2513 - 0.0200 \cdot 2.3979 = 2.2192.$$ ■

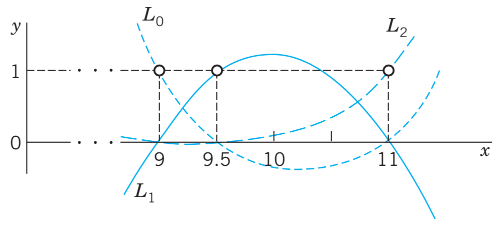

Fig. 433. L_0, L_1, L_2 in Example 2

General Lagrange Interpolation Polynomial. For general n we obtain

(4a)
$$f(x) \approx p_n(x) = \sum_{k=0}^{n} L_k(x) f_k = \sum_{k=0}^{n} \frac{l_k(x)}{l_k(x_k)} f_k$$

where $L_k(x_k) = 1$ and L_k is 0 at the other nodes, and the L_k are independent of the function f to be interpolated. We get (4a) if we take

(4b)
$$\begin{aligned} l_0(x) &= (x - x_1)(x - x_2) \cdots (x - x_n), \\ l_k(x) &= (x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n), \qquad 0 < k < n, \\ l_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned}$$

We can easily see that $p_n(x_k) = f_k$. Indeed, inspection of (4b) shows that $l_k(x_j) = 0$ if $j \neq k$, so that for $x = x_k$, the sum in (4a) reduces to the single term $(l_k(x_k)/l_k(x_k)) f_k = f_k$.

Error Estimate. If f is itself a polynomial of degree n (or less), it must coincide with p_n because the $n + 1$ data $(x_0, f_0), \cdots, (x_n, f_n)$ determine a polynomial uniquely, so the error is zero. Now the special f has its $(n + 1)$st derivative identically zero. This makes it plausible that for a *general* f its $(n + 1)$st derivative $f^{(n+1)}$ should measure the error

$$\epsilon_n(x) = f(x) - p_n(x).$$

It can be shown that this is true if $f^{(n+1)}$ exists and is continuous. Then, with a suitable t between x_0 and x_n (or between x_0, x_n, and x if we extrapolate),

(5)
$$\epsilon_n(x) = f(x) - p_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{(n+1)}(t)}{(n + 1)!}.$$

Thus $|\epsilon_n(x)|$ is 0 at the nodes and small near them, because of continuity. The product $(x - x_0) \cdots (x - x_n)$ is large for x away from the nodes. This makes extrapolation risky. And interpolation at an x will be best if we choose nodes on both sides of that x. Also, we get error bounds by taking the smallest and the largest value of $f^{(n+1)}(t)$ in (5) on the interval $x_0 \leqq t \leqq x_n$ (or on the interval also containing x if we *extra*polate).

Most importantly, since p_n is unique, as we have shown, we have

THEOREM 1

Error of Interpolation

Formula (5) *gives the error for* ***any*** *polynomial interpolation method if* $f(x)$ *has a continuous* $(n + 1)$*st derivative.*

Practical error estimate. If the derivative in (5) is difficult or impossible to obtain, apply the Error Principle (Sec. 19.1), that is, take another node and the Lagrange polynomial $p_{n+1}(x)$ and regard $p_{n+1}(x) - p_n(x)$ as a (crude) error estimate for $p_n(x)$.

EXAMPLE 3 **Error Estimate (5) of Linear Interpolation. Damage by Roundoff. Error Principle**

Estimate the error in Example 1 first by (5) directly and then by the Error Principle (Sec. 19.1).

Solution. **(A)** ***Estimation by*** **(5).** We have $n = 1$, $f(t) = \ln t$, $f'(t) = 1/t$, $f''(t) = -1/t^2$. Hence

$$\epsilon_1(x) = (x - 9.0)(x - 9.5)\frac{(-1)}{2t^2}, \qquad \text{thus} \qquad \epsilon_1(9.2) = \frac{0.03}{t^2}.$$

$t = 0.9$ gives the maximum $0.03/9^2 = 0.00037$ and $t = 9.5$ gives the minimum $0.03/9.5^2 = 0.00033$, so that we get $0.00033 \leqq \epsilon_1(9.2) \leqq 0.00037$, or better, 0.00038 because $0.3/81 = 0.003703 \cdots$.

But the error 0.0004 in Example 1 disagrees, and we can learn something! Repetition of the computation there with 5D instead of 4D gives

$$\ln 9.2 \approx p_1(9.2) = 0.6 \cdot 2.19722 + 0.4 \cdot 2.25129 = 2.21885$$

with an actual error $\epsilon = 2.21920 - 2.21885 = 0.00035$, which lies nicely near the middle between our two error bounds.

This shows that the discrepancy (0.0004 vs. 0.00035) was caused by rounding, which is not taken into account in (5).

(B) ***Estimation by the Error Principle.*** We calculate $p_1(9.2) = 2.21885$ as before and then $p_2(9.2)$ as in Example 2 but with 5D, obtaining

$$p_2(9.2) = 0.54 \cdot 2.19722 + 0.48 \cdot 2.25129 - 0.02 \cdot 2.39790 = 2.21916.$$

The difference $p_2(9.2) - p_1(9.2) = 0.00031$ is the approximate error of $p_1(9.2)$ that we wanted to obtain; this is an approximation of the actual error 0.00035 given above. ■

Newton's Divided Difference Interpolation

For given data $(x_0, f_0), \cdots, (x_n, f_n)$ the interpolation polynomial $p_n(x)$ satisfying (1) is unique, as we have shown. But for different purposes we may use $p_n(x)$ in different forms. **Lagrange's form** just discussed is useful for deriving formulas in numeric differentiation (approximation formulas for derivatives) and integration (Sec. 19.5).

Practically more important are Newton's forms of $p_n(x)$, which we shall also use for solving ODEs (in Sec. 21.2). They involve fewer arithmetic operations than Lagrange's form. Moreover, it often happens that we have to increase the degree n to reach a required accuracy. Then in Newton's forms we can use all the previous work and just add another term, a possibility without counterpart for Lagrange's form. This also simplifies the application of the Error Principle (used in Example 3 for Lagrange). The details of these ideas are as follows.

Let $p_{n-1}(x)$ be the $(n - 1)$st Newton polynomial (whose form we shall determine); thus $p_{n-1}(x_0) = f_0, p_{n-1}(x_1) = f_1, \cdots, p_{n-1}(x_{n-1}) = f_{n-1}$. Furthermore, let us write the nth Newton polynomial as

$$p_n(x) = p_{n-1}(x) + g_n(x); \tag{6}$$

hence

$$g_n(x) = p_n(x) - p_{n-1}(x). \tag{6'}$$

Here $g_n(x)$ is to be determined so that $p_n(x_0) = f_0, p_n(x_1) = f_1, \cdots, p_n(x_n) = f_n$.

Since p_n and p_{n-1} agree at $x_0, \cdots, x_{n-1}$, we see that g_n is zero there. Also, g_n will generally be a polynomial of nth degree because so is p_n, whereas p_{n-1} can be of degree $n - 1$ at most. Hence g_n must be of the form

$$g_n(x) = a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}). \tag{6''}$$

We determine the constant a_n. For this we set $x = x_n$ and solve (6″) algebraically for a_n. Replacing $g_n(x_n)$ according to (6′) and using $p_n(x_n) = f_n$, we see that this gives

$$a_n = \frac{f_n - p_{n-1}(x_n)}{(x_n - x_0)(x_n - x_1) \cdots (x_n - x_{n-1})}. \tag{7}$$

We write a_k instead of a_n and show that a_k equals the ***k*th divided difference**, recursively denoted and defined as follows:

$$a_1 = f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0}$$

$$a_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

and in general

$$a_k = f[x_0, \cdots, x_k] = \frac{f[x_1, \cdots, x_k] - f[x_0, \cdots, x_{k-1}]}{x_k - x_0}. \tag{8}$$

If $n = 1$, then $p_{n-1}(x_n) = p_0(x_1) = f_0$ because $p_0(x)$ is constant and equal to f_0, the value of $f(x)$ at x_0. Hence (7) gives

$$a_1 = \frac{f_1 - p_0(x_1)}{x_1 - x_0} = \frac{f_1 - f_0}{x_1 - x_0} = f[x_0, x_1],$$

and (6) and (6″) give the Newton interpolation polynomial of the first degree

$$p_1(x) = f_0 + (x - x_0)f[x_0, x_1].$$

If $n = 2$, then this p_1 and (7) give

$$a_2 = \frac{f_2 - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)} = \frac{f_2 - f_0 - (x_2 - x_0)f[x_0, x_1]}{(x_2 - x_0)(x_2 - x_1)} = f[x_0, x_1, x_2]$$

where the last equality follows by straightforward calculation and comparison with the definition of the right side. (Verify it; be patient.) From (6) and (6″) we thus obtain the second Newton polynomial

$$p_2(x) = f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2].$$

For $n = k$, formula (6) gives

$$p_k(x) = p_{k-1}(x) + (x - x_0)(x - x_1) \cdots (x - x_{k-1}) f[x_0, \cdots, x_k]. \tag{9}$$

With $p_0(x) = f_0$ by repeated application with $k = 1, \cdots, n$ this finally gives **Newton's divided difference interpolation formula**

$$\begin{aligned} f(x) \approx f_0 &+ (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &+ \cdots + (x - x_0)(x - x_1) \cdots (x - x_{n-1}) f[x_0, \cdots, x_n]. \end{aligned} \tag{10}$$

An algorithm is shown in Table 19.2. The first do-loop computes the divided differences and the second the desired value $p_n(\hat{x})$.

Example 4 shows how to arrange differences near the values from which they are obtained; the latter always stand a half-line above and a half-line below in the preceding column. Such an arrangement is called a (divided) **difference table**.

Table 19.2 Newton's Divided Difference Interpolation

```
ALGORITHM INTERPOL ($x_0, \cdots, x_n$; $f_0, \cdots, f_n$; $\hat{x}$)

This algorithm computes an approximation $p_n(\hat{x})$ of $f(\hat{x})$ at $\hat{x}$.

    INPUT: Data $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$; $\hat{x}$
    OUTPUT: Approximation $p_n(\hat{x})$ of $f(\hat{x})$
    Set $f[x_j] = f_j \quad (j = 0, \cdots, n)$.
    For $m = 1, \cdots, n - 1)$ do:
        For $j = 0, \cdots, n - m$ do:
            $f[x_j, \cdots, x_{j+m}] = \dfrac{f[x_{j+1}, \cdots, x_{j+m}] - f[x_j, \cdots, x_{j+m-1}]}{x_{j+m} - x_j}$
        End
    End
    Set $p_0(x) = f_0$.
    For $k = 1, \cdots, n$ do:
        $p_k(\hat{x}) = p_{k-1}(\hat{x}) + (\hat{x} - x_0) \cdots (\hat{x} - x_{k-1}) f[x_0, \cdots, x_k]$
    End
    OUTPUT $p_n(\hat{x})$
End INTERPOL
```

EXAMPLE 4 Newton's Divided Difference Interpolation Formula

Compute $f(9.2)$ from the values shown in the first two columns of the following table.

x_j	$f_j = f(x_j)$	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	$f[x_j, \cdots, x_{j+3}]$
8.0	2.079442			
		0.117783		
9.0	2.197225		−0.006433	
		0.108134		0.000411
9.5	2.251292		−0.005200	
		0.097735		
11.0	2.397895			

Solution. We compute the divided differences as shown. Sample computation:

$$(0.097735 - 0.108134)/(11 - 9) = -0.005200.$$

The values we need in (10) are circled. We have

$$f(x) \approx p_3(x) = 2.079442 + 0.117783(x - 8.0) - 0.006433(x - 8.0)(x - 9.0) \\ + 0.000411(x - 8.0)(x - 9.0)(x - 9.5).$$

At $x = 9.2$,

$$f(9.2) \approx 2.079442 + 0.141340 - 0.001544 - 0.000030 = 2.219208.$$

The value exact to 6D is $f(9.2) = \ln 9.2 = 2.219203$. Note that we can nicely see how the accuracy increases from term to term:

$$p_1(9.2) = 2.220782, \qquad p_2(9.2) = 2.219238, \qquad p_3(9.2) = 2.219208.$$ ■

Equal Spacing: Newton's Forward Difference Formula

Newton's formula (10) is valid for *arbitrarily spaced* nodes as they may occur in practice in experiments or observations. However, in many applications the x_j's are *regularly spaced*—for instance, in measurements taken at regular intervals of time. Then, denoting the distance by h, we can write

(11)
$$x_0, \quad x_1 = x_0 + h, \quad x_2 = x_0 + 2h, \quad \cdots, \quad x_n = x_0 + nh.$$

We show how (8) and (10) now simplify considerably!

To get started, let us define the *first forward difference* of f at x_j by

$$\Delta f_j = f_{j+1} - f_j,$$

the *second forward difference* of f at x_j by

$$\Delta^2 f_j = \Delta f_{j+1} - \Delta f_j,$$

and, continuing in this way, the ***k*th forward difference** of f at x_j by

(12)
$$\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j \qquad (k = 1, 2, \cdots).$$

Examples and an explanation of the name "forward" follow on the next page. What is the point of this? We show that if we have regular spacing (11), then

(13)
$$f[x_0, \cdots, x_k] = \frac{1}{k!h^k} \Delta^k f_0.$$

PROOF We prove (13) by induction. It is true for $k = 1$ because $x_1 = x_0 + h$, so that

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0} = \frac{1}{h}(f_1 - f_0) = \frac{1}{1!h} \Delta f_0.$$

Assuming (13) to be true for all forward differences of order k, we show that (13) holds for $k + 1$. We use (8) with $k + 1$ instead of k; then we use $(k + 1)h = x_{k+1} - x_0$, resulting from (11), and finally (12) with $j = 0$, that is, $\Delta^{k+1} f_0 = \Delta^k f_1 - \Delta^k f_0$. This gives

$$\begin{aligned} f[x_0, \cdots, x_{k+1}] &= \frac{f[x_1, \cdots, x_{k+1}] - f[x_0, \cdots, x_k]}{(k + 1)h} \\ &= \frac{1}{(k+1)h} \left[\frac{1}{k!h^k} \Delta^k f_1 - \frac{1}{k!h^k} \Delta^k f_0 \right] \\ &= \frac{1}{(k+1)!h^{k+1}} \Delta^{k+1} f_0 \end{aligned}$$

which is (13) with $k + 1$ instead of k. Formula (13) is proved. ■

In (10) we finally set $x = x_0 + rh$. Then $x - x_0 = rh$, $x - x_1 = (r - 1)h$ since $x_1 - x_0 = h$, and so on. With this and (13), formula (10) becomes **Newton's** (or *Gregory*[2]*–Newton's*) **forward difference interpolation formula**

(14)
$$\begin{aligned} f(x) \approx p_n(x) &= \sum_{s=0}^{n} \binom{r}{s} \Delta^s f_0 \qquad (x = x_0 + rh, \quad r = (x - x_0)/h) \\ &= f_0 + r\Delta f_0 + \frac{r(r-1)}{2!} \Delta^2 f_0 + \cdots + \frac{r(r-1) \cdots (r-n+1)}{n!} \Delta^n f_0 \end{aligned}$$

where the **binomial coefficients** in the first line are defined by

(15)
$$\binom{r}{0} = 1, \quad \binom{r}{s} = \frac{r(r-1)(r-2) \cdots (r-s+1)}{s!} \qquad (s > 0, \text{ integer})$$

and $s! = 1 \cdot 2 \cdots s$.

Error. From (5) we get, with $x - x_0 = rh$, $x - x_1 = (r - 1)h$, etc.,

(16)
$$\epsilon_n(x) = f(x) - p_n(x) = \frac{h^{n+1}}{(n+1)!} r(r-1) \cdots (r-n) f^{(n+1)}(t)$$

with t as characterized in (5).

[2]JAMES GREGORY (1638–1675), Scots mathematician, professor at St. Andrews and Edinburgh. Δ in (14) and ∇^2 (on p. 818) have nothing to do with the Laplacian.

Formula (16) is an exact formula for the error, but it involves the unknown t. In Example 5 (below) we show how to use (16) for obtaining an error estimate and an interval in which the true value of $f(x)$ must lie.

Comments on Accuracy. (A) The order of magnitude of the error $\epsilon_n(x)$ is about equal to that of the next difference not used in $p_n(x)$.

(B) One should choose $x_0, \cdots, x_n$ such that the x at which one interpolates is as well centered between $x_0, \cdots, x_n$ as possible.

The reason for (A) is that in (16),

$$f^{n+1}(t) \approx \frac{\Delta^{n+1} f(t)}{h^{n+1}}, \qquad \frac{|r(r-1)\cdots(r-n)|}{1\cdot 2 \cdots (n+1)} \leqq 1 \qquad \text{if} \qquad |r| \leqq 1$$

(and actually for any r as long as we do not *extrapolate*). The reason for (B) is that $|r(r-1)\cdots(r-n)|$ becomes smallest for that choice.

EXAMPLE 5 Newton's Forward Difference Formula. Error Estimation

Compute cosh 0.56 from (14) and the four values in the following table and estimate the error.

j	x_j	$f_j = \cosh x_j$	Δf_j	$\Delta^2 f_j$	$\Delta^3 f_j$
0	0.5	$\boxed{1.127626}$			
			$\boxed{0.057839}$		
1	0.6	1.185465		$\boxed{0.011865}$	
			0.069704		$\boxed{0.000697}$
2	0.7	1.255169		0.012562	
			0.082266		
3	0.8	1.337435			

Solution. We compute the forward differences as shown in the table. The values we need are circled. In (14) we have $r = (0.56 - 0.50)/0.1 = 0.6$, so that (14) gives

$$\begin{aligned}\cosh 0.56 &\approx 1.127626 + 0.6 \cdot 0.057839 + \frac{0.6(-0.4)}{2} \cdot 0.011865 + \frac{0.6(-0.4)(-1.4)}{6} \cdot 0.000697 \\ &= 1.127626 + 0.034703 - 0.001424 + 0.000039 \\ &= 1.160944.\end{aligned}$$

Error estimate. From (16), since the fourth derivative is $\cosh^{(4)} t = \cosh t$,

$$\begin{aligned}\epsilon_3(0.56) &= \frac{0.1^4}{4!} \cdot 0.6(-0.4)(-1.4)(-2.4) \cosh t \\ &= A \cosh t,\end{aligned}$$

where $A = -0.00000336$ and $0.5 \leqq t \leqq 0.8$. We do not know t, but we get an inequality by taking the largest and smallest $\cosh t$ in that interval:

$$A \cosh 0.8 \leqq \epsilon_3(0.62) \leqq A \cosh 0.5.$$

Since

$$f(x) = p_3(x) + \epsilon_3(x),$$

this gives

$$p_3(0.56) + A \cosh 0.8 \leqq \cosh 0.56 \leqq p_3(0.56) + A \cosh 0.5.$$

Numeric values are

$$1.160939 \leqq \cosh 0.56 \leqq 1.160941.$$

The exact 6D-value is $\cosh 0.56 = 1.160941$. It lies within these bounds. Such bounds are not always so tight. Also, we did not consider roundoff errors, which will depend on the number of operations. ■

This example also explains the name "*forward* difference formula": we see that the differences in the formula slope forward in the difference table.

Equal Spacing: Newton's Backward Difference Formula

Instead of forward-sloping differences we may also employ backward-sloping differences. The difference table remains the same as before (same numbers, in the same positions), except for a very harmless change of the running subscript j (which we explain in Example 6, below). Nevertheless, purely for reasons of convenience it is standard to introduce a second name and notation for differences as follows. We define the *first backward difference* of f at x_j by

$$\nabla f_j = f_j - f_{j-1},$$

the *second backward difference* of f at x_j by

$$\nabla^2 f_j = \nabla f_j - \nabla f_{j-1},$$

and, continuing in this way, the ***k*th backward difference** of f at x_j by

(17)
$$\nabla^k f_j = \nabla^{k-1} f_j - \nabla^{k-1} f_{j-1} \qquad (k = 1, 2, \cdots).$$

A formula similar to (14) but involving backward differences is **Newton's** (or *Gregory–Newton's*) **backward difference interpolation formula**

(18)
$$\begin{aligned} f(x) \approx p_n(x) &= \sum_{s=0}^{n} \binom{r+s-1}{s} \nabla^s f_0 \qquad (x = x_0 + rh,\ r = (x - x_0)/h) \\ &= f_0 + r\nabla f_0 + \frac{r(r+1)}{2!} \nabla^2 f_0 + \cdots + \frac{r(r+1)\cdots(r+n-1)}{n!} \nabla^n f_0. \end{aligned}$$

EXAMPLE 6 Newton's Forward and Backward Interpolations

Compute a 7D-value of the Bessel function $J_0(x)$ for $x = 1.72$ from the four values in the following table, using (a) Newton's forward formula (14), (b) Newton's backward formula (18).

j_{for}	j_{back}	x_j	$J_0(x_j)$	1st Diff.	2nd Diff.	3rd Diff.
0	−3	1.7	0.3979849			
				−0.0579985		
1	−2	1.8	0.3399864		−0.0001693	
				−0.0581678		0.0004093
2	−1	1.9	0.2818186		0.0002400	
				−0.0579278		
3	0	2.0	0.2238908			

Solution. The computation of the differences is the same in both cases. Only their notation differs.

(a) Forward. In (14) we have $r = (1.72 - 1.70)/0.1 = 0.2$, and j goes from 0 to 3 (see first column). In each column we need the first given number, and (14) thus gives

$$\begin{aligned} J_0(1.72) &\approx 0.3979849 + 0.2(-0.0579985) + \frac{0.2(-0.8)}{2}(-0.0001693) + \frac{0.2(-0.8)(-1.8)}{6} \cdot 0.0004093 \\ &= 0.3979849 - 0.0115997 + 0.0000135 + 0.0000196 = 0.3864183, \end{aligned}$$

which is exact to 6D, the exact 7D-value being 0.3864185.

(b) Backward. For (18) we use j shown in the second column, and in each column the last number. Since $r = (1.72 - 2.00)/0.1 = -2.8$, we thus get from (18)

$$\begin{aligned} J_0(1.72) &\approx 0.2238908 - 2.8(-0.0579278) + \frac{-2.8(-1.8)}{2} \cdot 0.0002400 + \frac{-2.8(-1.8)(-0.8)}{6} \cdot 0.0004093 \\ &= 0.2238908 + 0.1621978 + 0.0006048 - 0.0002750 \\ &= 0.3864184. \end{aligned}$$

There is a third notation for differences, called the **central difference notation**. It is used in numerics for ODEs and certain interpolation formulas. See Ref. [E5] listed in App. 1.

PROBLEM SET 19.3

1. **Linear interpolation.** Calculate $p_1(x)$ in Example 1 and from it ln 9.3.
2. **Error estimate.** Estimate the error in Prob. 1 by (5).
3. **Quadratic interpolation. Gamma function.** Calculate the Lagrange polynomial $p_2(x)$ for the values $\Gamma(1.00) = 1.0000$, $\Gamma(1.02) = 0.9888$, $\Gamma(1.04) = 0.9784$ of the gamma function [(24) in App. A3.1] and from it approximations of $\Gamma(1.01)$ and $\Gamma(1.03)$.
4. **Error estimate for quadratic interpolation.** Estimate the error for $p_2(9.2)$ in Example 2 from (5).
5. **Linear and quadratic interpolation.** Find $e^{-0.25}$ and $e^{-0.75}$ by linear interpolation of e^{-x} with $x_0 = 0$, $x_1 = 0.5$ and $x_0 = 0.5$, $x_1 = 1$, respectively. Then find $p_2(x)$ by quadratic interpolation of e^{-x} with $x_0 = 0$, $x_1 = 0.5$, $x_2 = 1$ and from it $e^{-0.25}$ and $e^{-0.75}$. Compare the errors. Use 4S-values of e^{-x}.
6. **Interpolation and extrapolation.** Calculate $p_2(x)$ in Example 2. Compute from it approximations of ln 9.4, ln 10, ln 10.5, ln 11.5, and ln 12. Compute the errors by using exact 5S-values and comment.
7. **Interpolation and extrapolation.** Find the quadratic polynomial that agrees with $\sin x$ at $x = 0, \pi/4, \pi/2$ and use it for the interpolation and extrapolation of $\sin x$ at $x = -\pi/8, \pi/8, 3\pi/8, 5\pi/8$. Compute the errors.
8. **Extrapolation.** Does a sketch of the product of the $(x - x_j)$ in (5) for the data in Example 2 indicate that extrapolation is likely to involve larger errors than interpolation does?
9. **Error function** (35) in App. A3.1. Calculate the Lagrange polynomial $p_2(x)$ for the 5S-values $f(0.25) = 0.27633$, $f(0.5) = 0.52050$, $f(1.0) = 0.84270$ and from $p_2(x)$ an approximation of $f(0.75)$ $(= 0.71116)$.

10. Error bound. Derive an error bound in Prob. 9 from (5).

11. Cubic Lagrange interpolation. Bessel function J_0. Calculate and graph L_0, L_1, L_2, L_3 with $x_0 = 0$, $x_1 = 1, x_2 = 2, x_3 = 3$ on common axes. Find $p_3(x)$ for the data $(0, 1)$, $(1, 0.765198)$, $(2, 0.223891)$, $(3, -0.260052)$ [values of the Bessel function $J_0(x)$]. Find p_3 for $x = 0.5, 1.5, 2.5$ and compare with the 6S-exact values $0.938470, 0.511828, -0.048384$.

12. Newton's forward formula (14). Sine integral. Using (14), find $f(1.25)$ by linear, quadratic, and cubic interpolation of the data (values of (40) in App. A31); 6S-value $\text{Si}(1.25) = 1.14645)$ $f(1.0) = 0.94608$, $f(1.5) = 1.32468$, $f(2.0) = 1.60541$, $f(2.5) = 1.77852$, and compute the errors. For the linear interpolation use $f(1.0)$ and $f(1.5)$, for the quadratic $f(1.0)$, $f(1.5)$, $f(2.0)$, etc.

13 Lower degree. Find the degree of the interpolation polynomial for the data $(-4, 50)$, $(-2, 18)$, $(0, 2)$, $(2, 2)$, $(4, 18)$, using a difference table. Find the polynomial.

14. Newton's forward formula (14). Gamma function. Set up (14) for the data in Prob. 3 and compute $\Gamma(1.01)$, $\Gamma(1.03)$, $\Gamma(1.05)$.

15. Divided differences. Obtain p_2 in Example 2 from (10).

16. Divided differences. Error function. Compute $p_2(0.75)$ from the data in Prob. 9 and Newton's divided difference formula (10).

17. Backward difference formula (18). Use $p_2(x)$ in (18) and the values of erf x, $x = 0.2, 0.4, 0.6$ in Table A4 of App. 5, compute erf 0.3 and the error. (4S-exact erf $0.3 = 0.3286$).

18. In Example 5 of the text, write down the difference table as needed for (18), then write (18) with general x and then with $x = 0.56$ to verify the answer in Example 5.

19. CAS EXPERIMENT. Adding Terms in Newton Formulas. Write a program for the forward formula (14). Experiment on the increase of accuracy by successively adding terms. As data use values of some function of your choice for which your CAS gives the values needed in determining errors.

20. TEAM PROJECT. Interpolation and Extrapolation. **(a) Lagrange practical error estimate** (after Theorem 1). Apply this to $p_1(9.2)$ and $p_2(9.2)$ for the data $x_0 = 9.0, x_1 = 9.5, x_2 = 11.0, f_0 = \ln x_0, f_1 = \ln x_1$, $f_2 = \ln x_2$ (6S-values).

(b) Extrapolation. Given $(x_j, f(x_j)) = (0.2, 0.9980)$, $(0.4, 0.9686)$, $(0.6, 0.8443)$, $(0.8, 0.5358)$, $(1.0, 0)$. Find $f(0.7)$ from the quadratic interpolation polynomials based on (α) 0.6, 0.8, 1.0, (β) 0.4, 0.6, 0.8, (γ) 0.2, 0.4, 0.6. Compare the errors and comment. [Exact $f(x) = \cos\left(\frac{1}{2}\pi x^2\right)$, $f(0.7) = 0.7181$ (4S).]

(c) Graph the product of factors $(x - x_j)$ in the error formula (5) for $n = 2, \cdots, 10$ separately. What do these graphs show regarding accuracy of interpolation and extrapolation?

21. WRITING PROJECT. Comparison of interpolation methods. List 4–5 ideas that you feel are most important in this section. Arrange them in best logical order. Discuss them in a 2–3 page report.

19.4 Spline Interpolation

Given data (function values, points in the xy-plane) $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$ can be interpolated by a polynomial $P_n(x)$ of degree n or less so that the curve of $P_n(x)$ passes through these $n + 1$ points (x_j, f_j); here $f_0 = f(x_0), \cdots, f_n = f(x_n)$, See Sec. 19.3.

Now if n is large, there may be trouble: $P_n(x)$ may tend to oscillate for x between the **nodes** $x_0, \cdots, x_n$. Hence we must be prepared for ***numeric instability*** (Sec. 19.1). Figure 434 shows a famous example by C. Runge[3] for which the maximum error even approaches ∞ as $n \to \infty$ (with the nodes kept equidistant and their number increased). Figure 435 illustrates the increase of the oscillation with n for some other function that is piecewise linear.

Those undesirable oscillations are avoided by the method of splines initiated by I. J. Schoenberg in 1946 (*Quarterly of Applied Mathematics* **4**, pp. 45–99, 112–141). This method is widely used in practice. It also laid the foundation for much of modern **CAD (computer-aided design)**. Its name is borrowed from a *draftman's spline*, which is an elastic rod bent to pass through given points and held in place by weights. The mathematical idea of the method is as follows:

[3]CARL RUNGE (1856–1927), German mathematician, also known for his work on ODEs (Sec. 21.1).

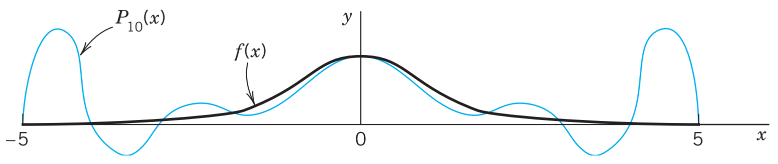

Fig. 434. Runge's example $f(x) = 1/(1 + x^2)$ and interpolating polynomial $P_{10}(x)$

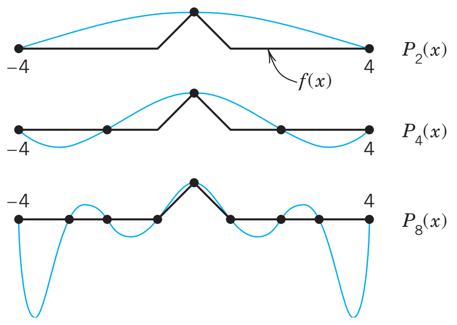

Fig. 435. Piecewise linear function $f(x)$ and interpolation polynomials of increasing degrees

Instead of using a single high-degree polynomial P_n over the entire interval $a \leqq x \leqq b$ in which the nodes lie, that is,

$$a = x_0 < x_1 < \cdots < x_n = b, \tag{1}$$

we use n low-degree, e.g., cubic, polynomials

$$q_0(x), \qquad q_1(x), \qquad \cdots, \qquad q_{n-1}(x),$$

one over each subinterval between adjacent nodes, hence q_0 from x_0 to x_1, then q_1 from x_1 to x_2, and so on. From this we compose an interpolation function $g(x)$, called a **spline**, by fitting these polynomials together into a single continuous curve passing through the data points, that is,

$$g(x_0) = f(x_0) = f_0, \qquad g(x_1) = f(x_1) = f_1, \qquad \cdots, \qquad g(x_n) = f(x_n) = f_n. \tag{2}$$

Note that $g(x) = q_0(x)$ when $x_0 \leqq x \leqq x_1$, then $g(x) = q_1(x)$ when $x_1 \leqq x \leqq x_2$, and so on, according to our construction of g.

Thus **spline interpolation** is *piecewise polynomial interpolation.*

The simplest q_j's would be linear polynomials. However, the curve of a piecewise linear continuous function has corners and would be of little interest in general—think of designing the body of a car or a ship.

We shall consider cubic splines because these are the most important ones in applications. By definition, a **cubic spline** $g(x)$ interpolating given data $(x_0, f_0), \cdots, (x_n, f_n)$ is a continuous function on the interval $a = x_0 \leqq x \leqq x_n = b$ that has continuous first and second derivatives and satisfies the interpolation condition (2); furthermore, between adjacent nodes, $g(x)$ is given by a polynomial $q_j(x)$ of degree 3 or less.

We claim that there is such a cubic spline. And if in addition to (2) we also require that

$$g'(x_0) = k_0, \qquad g'(x_n) = k_n \tag{3}$$

(given tangent directions of $g(x)$ at the two endpoints of the interval $a \leqq x \leqq b$), then we have a uniquely determined cubic spline. This is the content of the following existence and uniqueness theorem, whose proof will also suggest the actual determination of splines. (Condition (3) will be discussed after the proof.)

THEOREM 1

Existence and Uniqueness of Cubic Splines

Let $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$ *with given (arbitrarily spaced)* x_j *[see* (1)*] and given* $f_j = f(x_j), j = 0, 1, \cdots, n$. *Let* k_0 *and* k_n *be any given numbers. Then there is one and only one cubic spline* $g(x)$ *corresponding to* (1) *and satisfying* (2) *and* (3).

PROOF By definition, on every subinterval I_j given by $x_j \leqq x \leqq x_{j+1}$, the spline $g(x)$ must agree with a polynomial $q_j(x)$ of degree not exceeding 3 such that

$$q_j(x_j) = f(x_j), \qquad q_j(x_{j+1}) = f(x_{j+1}) \qquad (j = 0, 1, \cdots, n-1). \tag{4}$$

For the derivatives we write

$$q_j'(x_j) = k_j, \qquad q_j'(x_{j+1}) = k_{j+1} \qquad (j = 0, 1, \cdots, n-1) \tag{5}$$

with k_0 and k_n given and $k_1, \cdots, k_{n-1}$ to be determined later. Equations (4) and (5) are four conditions for each $q_j(x)$. By direct calculation, using the notation

$$c_j = \frac{1}{h_j} = \frac{1}{x_{j+1} - x_j} \qquad (j = 0, 1, \cdots, n-1) \tag{6*}$$

we can verify that the unique cubic polynomial $q_j(x)$ $(j = 0, 1, \cdots, n-1)$ satisfying (4) and (5) is

$$\begin{aligned} q_j(x) = {} & f(x_j)c_j^2(x - x_{j+1})^2[1 + 2c_j(x - x_j)] \\ & + f(x_{j+1})c_j^2(x - x_j)^2[1 - 2c_j(x - x_{j+1})] \\ & + k_j c_j^2(x - x_j)(x - x_{j+1})^2 \\ & + k_{j+1}c_j^2(x - x_j)^2(x - x_{j+1}). \end{aligned} \tag{6}$$

Differentiating twice, we obtain

$$q_j''(x_j) = -6c_j^2 f(x_j) + 6c_j^2 f(x_{j+1}) - 4c_j k_j - 2c_j k_{j+1} \tag{7}$$

$$q_j''(x_{j+1}) = 6c_j^2 f(x_j) - 6c_j^2 f(x_{j+1}) + 2c_j k_j + 4c_j k_{j+1}. \tag{8}$$

By definition, $g(x)$ has continuous second derivatives. This gives the conditions

$$q_{j-1}''(x_j) = q_j''(x_j) \qquad (j = 1, \cdots, n-1).$$

If we use (8) with j replaced by $j - 1$, and (7), these $n - 1$ equations become

$$c_{j-1}k_{j-1} + 2(c_{j-1} + c_j)k_j + c_jk_{j+1} = 3[c_{j-1}^2\nabla f_j + c_j^2\nabla f_{j+1}] \tag{9}$$

where $\nabla f_j = f(x_j) - f(x_{j-1})$ and $\nabla f_{j+1} = f(x_{j+1}) - f(x_j)$ and $j = 1, \cdots, n - 1$, as before. This linear system of $n - 1$ equations has a unique solution $k_1, \cdots, k_{n-1}$ since the coefficient matrix is strictly diagonally dominant (that is, in each row the (positive) diagonal entry is greater than the sum of the other (positive) entries). Hence the determinant of the matrix cannot be zero (as follows from Theorem 3 in Sec. 20.7), so that we may determine unique values $k_1, \cdots, k_{n-1}$ of the first derivative of $g(x)$ at the nodes. This proves the theorem. ■

Storage and Time Demands in solving (9) are modest, since the matrix of (9) is **sparse** (has few nonzero entries) and **tridiagonal** (may have nonzero entries only on the diagonal and on the two adjacent "parallels" above and below it). Pivoting (Sec. 7.3) is not necessary because of that dominance. This makes splines efficient in solving large problems with thousands of nodes or more. For some literature and some critical comments, see *American Mathematical Monthly* **105** (1998), 929–941.

Condition (3) includes the **clamped conditions**

$$g'(x_0) = f'(x_0), \qquad g'(x_n) = f'(x_n), \tag{10}$$

in which the tangent directions $f'(x_0)$ and $f'(x_n)$ at the ends are given. Other conditions of practical interest are the **free** or **natural conditions**

$$g''(x_0) = 0, \qquad g''(x_n) = 0 \tag{11}$$

(geometrically: zero curvature at the ends, as for the draftman's spline), giving a **natural spline**. These names are motivated by Fig. 293 in Problem Set 12.3.

Determination of Splines. Let k_0 and k_n be given. Obtain $k_1, \cdots, k_{n-1}$ by solving the linear system (9). Recall that the spline $g(x)$ to be found consists of n cubic polynomials $q_0, \cdots, q_{n-1}$. We write these polynomials in the form

$$q_j(x) = a_{j0} + a_{j1}(x - x_j) + a_{j2}(x - x_j)^2 + a_{j3}(x - x_j)^3 \tag{12}$$

where $j = 0, \cdots, n - 1$. Using Taylor's formula, we obtain

$$\begin{aligned}
a_{j0} &= q_j(x_j) = f_j && \text{by (2)},\\
a_{j1} &= q_j'(x_j) = k_j && \text{by (5)},\\
a_{j2} &= \frac{1}{2}\, q_j''(x_j) = \frac{3}{h_j^2}\,(f_{j+1} - f_j) - \frac{1}{h_j}\,(k_{j+1} + 2k_j) && \text{by (7)},\\
a_{j3} &= \frac{1}{6}\, q_j'''(x_j) = \frac{2}{h_j^3}\,(f_j - f_{j+1}) + \frac{1}{h_j^2}\,(k_{j+1} + k_j)
\end{aligned} \tag{13}$$

with a_{j3} obtained by calculating $q_j''(x_{j+1})$ from (12) and equating the result to (8), that is,

$$q_j''(x_{j+1}) = 2a_{j2} + 6a_{j3}h_j = \frac{6}{h_j^2}(f_j - f_{j+1}) + \frac{2}{h_j}(k_j + 2k_{j+1}),$$

and now subtracting from this $2a_{j2}$ as given in (13) and simplifying.

Note that for ***equidistant nodes*** of distance $h_j = h$ we can write $c_j = c = 1/h$ in (6*) and have from (9) simply

$$\textbf{(14)} \qquad \boxed{k_{j-1} + 4k_j + k_{j+1} = \frac{3}{h}(f_{j+1} - f_{j-1})} \qquad (j = 1, \cdots, n-1).$$

EXAMPLE 1 **Spline Interpolation. Equidistant Nodes**

Interpolate $f(x) = x^4$ on the interval $-1 \leqq x \leqq 1$ by the cubic spline $g(x)$ corresponding to the nodes $x_0 = -1$, $x_1 = 0$, $x_2 = 1$ and satisfying the clamped conditions $g'(-1) = f'(-1)$, $g'(1) = f'(1)$.

Solution. In our standard notation the given data are $f_0 = f(-1) = 1, f_1 = f(0) = 0, f_2 = f(1) = 1$. We have $h = 1$ and $n = 2$, so that our spline consists of $n = 2$ polynomials

$$q_0(x) = a_{00} + a_{01}(x+1) + a_{02}(x+1)^2 + a_{03}(x+1)^3 \qquad (-1 \leqq x \leqq 0),$$
$$q_1(x) = a_{10} + a_{11}x + a_{12}x^2 + a_{13}x^3 \qquad (0 \leqq x \leqq 1).$$

We determine the k_j from (14) (equidistance!) and then the coefficients of the spline from (13). Since $n = 2$, the system (14) is a single equation (with $j = 1$ and $h = 1$)

$$k_0 + 4k_1 + k_2 = 3(f_2 - f_0).$$

Here $f_0 = f_2 = 1$ (the value of x^4 at the ends) and $k_0 = -4$, $k_2 = 4$, the values of the derivative $4x^3$ at the ends -1 and 1. Hence

$$-4 + 4k_1 + 4 = 3(1-1) = 0, \qquad k_1 = 0.$$

From (13) we can now obtain the coefficients of q_0, namely, $a_{00} = f_0 = 1$, $a_{01} = k_0 = -4$, and

$$a_{02} = \frac{3}{1^2}(f_1 - f_0) - \frac{1}{1}(k_1 + 2k_0) = 3(0-1) - (0-8) = 5$$

$$a_{03} = \frac{2}{1^3}(f_0 - f_1) + \frac{1}{1^2}(k_1 + k_0) = 2(1-0) + (0-4) = -2.$$

Similarly, for the coefficients of q_1 we obtain from (13) the values $a_{10} = f_1 = 0$, $a_{11} = k_1 = 0$, and

$$a_{12} = 3(f_2 - f_1) - (k_2 + 2k_1) = 3(1-0) - (4+0) = -1$$
$$a_{13} = 2(f_1 - f_2) + (k_2 + k_1) = 2(0-1) + (4+0) = 2.$$

This gives the polynomials of which the spline $g(x)$ consists, namely,

$$g(x) = \begin{cases} q_0(x) = 1 - 4(x+1) + 5(x+1)^2 - 2(x+1)^3 = -x^2 - 2x^3 & \text{if} \quad -1 \leqq x \leqq 0 \\ q_1(x) = -x^2 + 2x^3 & \text{if} \quad 0 \leqq x \leqq 1. \end{cases}$$

Figure 436 shows $f(x)$ and this spline. Do you see that we could have saved over half of our work by using symmetry? ■

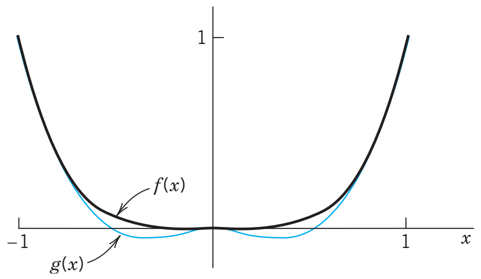

Fig. 436. Function $f(x) = x^4$ and cubic spline $g(x)$ in Example 1

EXAMPLE 2 Natural Spline. Arbitrarily Spaced Nodes

Find a spline approximation and a polynomial approximation for the curve of the cross section of the circular-shaped Shrine of the Book in Jerusalem shown in Fig. 437.

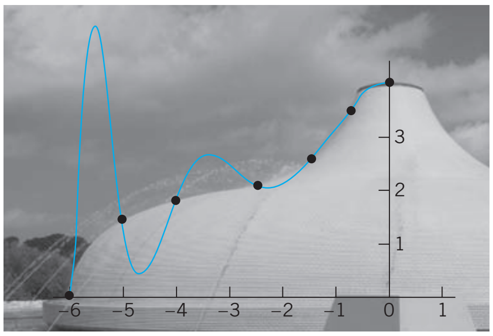

Fig. 437. Shrine of the Book in Jerusalem (Architects F. Kissler and A. M. Bartus)

Solution. Thirteen points, about equally distributed along the contour (not along the x-axis!), give these data:

x_j	−5.8	−5.0	−4.0	−2.5	−1.5	−0.8	0	0.8	1.5	2.5	4.0	5.0	5.8
f_j	0	1.5	1.8	2.2	2.7	3.5	3.9	3.5	2.7	2.2	1.8	1.5	0

The figure shows the corresponding interpolation polynomial of 12th degree, which is useless because of its oscillation. (Because of roundoff your software will also give you small error terms involving odd powers of x.) The polynomial is

$$P_{12}(x) = 3.9000 - 0.65083x^2 + 0.033858x^4 + 0.011041x^6 - 0.0014010x^8$$
$$+ 0.000055595x^{10} - 0.00000071867x^{12}.$$

The spline follows practically the contour of the roof, with a small error near the nodes −0.8 and 0.8. The spline is symmetric. Its six polynomials corresponding to positive x have the following coefficients of their representations (12). (Note well that (12) is in terms of powers of $x - x_j$, not x!)

j	x-interval	a_{j0}	a_{j1}	a_{j2}	a_{j3}
0	0.0...0.8	3.9	0.00	−0.61	−0.015
1	0.8...1.5	3.5	−1.01	−0.65	0.66
2	1.5...2.5	2.7	−0.95	0.73	−0.27
3	2.5...4.0	2.2	−0.32	−0.091	0.084
4	4.0...5.0	1.8	−0.027	0.29	−0.56
5	5.0...5.8	1.5	−1.13	−1.39	0.58

PROBLEM SET 19.4

1. **WRITING PROJECT. Splines.** In your own words, and using as few formulas as possible, write a short report on spline interpolation, its motivation, a comparison with polynomial interpolation, and its applications.

2–9 VERIFICATIONS. DERIVATIONS. COMPARISONS

2. **Individual polynomial q_j.** Show that $q_j(x)$ in (6) satisfies the interpolation condition (4) as well as the derivative condition (5).
3. Verify the differentiations that give (7) and (8) from (6).
4. **System for derivatives.** Derive the basic linear system (9) for $k_1, \cdots, k_{n-1}$ as indicated in the text.
5. **Equidistant nodes.** Derive (14) from (9).
6. **Coefficients.** Give the details of the derivation of a_{j2} and a_{j3} in (13).
7. Verify the computations in Example 1.
8. **Comparison.** Compare the spline g in Example 1 with the quadratic interpolation polynomial over the whole interval. Find the maximum deviations of g and p_2 from f. Comment.
9. **Natural spline condition.** Using the given coefficients, verify that the spline in Example 2 satisfies $g''(x) = 0$ at the ends.

10–16 DETERMINATION OF SPLINES

Find the cubic spline $g(x)$ for the given data with k_0 and k_n as given.

10. $f(-2) = f(-1) = f(1) = f(2) = 0, \quad f(0) = 1,$ $k_0 = k_4 = 0$
11. If we started from the piecewise linear function in Fig. 438, we would obtain $g(x)$ in Prob. 10 as the spline satisfying $g'(-2) = f'(-2) = 0, \quad g'(2) = f'(2) = 0.$ Find and sketch or graph the corresponding interpolation polynomial of 4th degree and compare it with the spline. Comment.

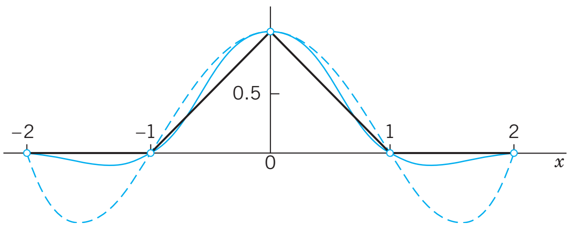

Fig. 438. Spline and interpolation polynomial in Probs. 10 and 11

12. $f_0 = f(0) = 1, \quad f_1 = f(2) = 9, \quad f_2 = f(4) = 41,$ $f_3 = f(6) = 41, \quad k_0 = 0, \quad k_3 = -12$
13. $f_0 = f(0) = 1, \quad f_1 = f(1) = 0, \quad f_2 = f(2) = -1,$ $f_3 = f(3) = 0, \quad k_0 = 0, \quad k_3 = -6$
14. $f_0 = f(0) = 2, \quad f_1 = f(1) = 3, \quad f_2 = f(2) = 8,$ $f_3 = f(3) = 12, \quad k_0 = k_3 = 0$
15. $f_0 = f(0) = 4, \quad f_1 = f(2) = 0, \quad f_2 = f(4) = 4,$ $f_3 = f(6) = 80, \quad k_0 = k_3 = 0$
16. $f_0 = f(0) = 2, \quad f_1 = f(2) = -2, \quad f_2 = f(4) = 2,$ $f_3 = f(6) = 78, \quad k_0 = k_3 = 0$. Can you obtain the answer from that of Prob. 15?
17. If a cubic spline is three times continuously differentiable (that is, it has continuous first, second, and third derivatives), show that it must be a single polynomial.
18. **CAS EXPERIMENT. Spline versus Polynomial.** If your CAS gives natural splines, find the natural splines when x is integer from $-m$ to m, and $y(0) = 1$ and all other y equal to 0. Graph each such spline along with the interpolation polynomial p_{2m}. Do this for $m = 2$ to 10 (or more). What happens with increasing m?
19. **Natural conditions.** Explain the remark after (11).
20. **TEAM PROJECT. Hermite Interpolation and Bezier Curves.** In **Hermite interpolation** we are looking for a polynomial $p(x)$ (of degree $2n + 1$ or less) such that $p(x)$ and its derivative $p'(x)$ have given values at $n + 1$ nodes. (More generally, $p(x), p'(x), p''(x), \cdots$ may be required to have given values at the nodes.)

    (a) Curves with given endpoints and tangents. Let C be a curve in the xy-plane parametrically represented by $\mathbf{r}(t) = [x(t), y(t)], 0 \leqq t \leqq 1$ (see Sec. 9.5). Show that for given initial and terminal points of a curve and given initial and terminal tangents, say,

$$\begin{aligned} A: \quad \mathbf{r}_0 &= [x(0), y(0)] \\ &= [x_0, y_0], \\ B: \quad \mathbf{r}_1 &= [x(1), y(1)] \\ &= [x_1, y_1] \\ \mathbf{v}_0 &= [x'(0), y'(0)] \\ &= [x_0', y_0'], \\ \mathbf{v}_1 &= [x'(1), y'(1)] \\ &= [x_1', y_1'] \end{aligned}$$

we can find a curve C, namely,

$$\begin{aligned} \mathbf{r}(t) = \mathbf{r}_0 &+ \mathbf{v}_0 t \\ &+ (3(\mathbf{r}_1 - \mathbf{r}_0) - (2\mathbf{v}_0 + \mathbf{v}_1))t^2 \\ &+ (2(\mathbf{r}_0 - \mathbf{r}_1) + \mathbf{v}_0 + \mathbf{v}_1)t^3; \end{aligned} \tag{15}$$

in components,

$$x(t) = x_0 + x_0' t + (3(x_1 - x_0) - (2x_0' + x_1'))t^2$$
$$+ (2(x_0 - x_1) + x_0' + x_1')t^3$$

$$y(t) = y_0 + y_0' t + (3(y_1 - y_0) - (2y_0' + y_1'))t^2$$
$$+ (2(y_0 - y_1) + y_0' + y_1')t^3.$$

Note that this is a cubic Hermite interpolation polynomial, and $n = 1$ because we have two nodes (the endpoints of C). (This has nothing to do with the Hermite polynomials in Sec. 5.8.) The two points

$$G_A\colon\ \mathbf{g}_0 = \mathbf{r}_0 + \mathbf{v}_0 = [x_0 + x_0',\ y_0 + y_0']$$

and

$$G_B\colon\ \mathbf{g}_1 = \mathbf{r}_1 - \mathbf{v}_1 = [x_1 - x_1',\ y_1 - y_1']$$

are called **guidepoints** because the segments AG_A and BG_B specify the tangents graphically. A, B, G_A, G_B determine C, and C can be changed quickly by moving the points. A curve consisting of such Hermite interpolation polynomials is called a **Bezier curve**, after the French engineer P. Bezier of the Renault Automobile Company, who introduced them in the early 1960s in designing car bodies. Bezier curves (and surfaces) are used in computer-aided design (CAD) and computer-aided manufacturing (CAM). (For more details, see Ref. [E21] in App. 1.)

(b) Find and graph the Bezier curve and its guidepoints if A: [0, 0], B: [1, 0], $\mathbf{v}_0 = [\frac{1}{2}, \frac{1}{2}]$, $\mathbf{v}_1 = [-\frac{1}{2}, -\frac{1}{4}\sqrt{3}]$.

(c) Changing guidepoints changes C. Moving guidepoints farther away results in C "staying near the tangents for a longer time." Confirm this by changing $\mathbf{v}_0$ and $\mathbf{v}_1$ in (b) to $2\mathbf{v}_0$ and $2\mathbf{v}_1$ (see Fig. 439).

(d) Make experiments of your own. What happens if you change $\mathbf{v}_1$ in (b) to $-\mathbf{v}_1$. If you rotate the tangents? If you multiply $\mathbf{v}_0$ and $\mathbf{v}_1$ by positive factors less than 1?

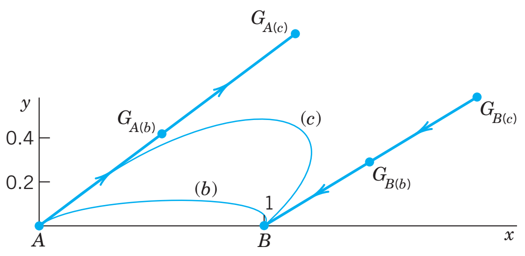

Fig. 439. Team Project 20(b) and (c): Bezier curves

19.5 Numeric Integration and Differentiation

In applications, the engineer often encounters integrals that are very difficult or even impossible to solve analytically. For example, the error function, the Fresnel integrals (see Probs. 16–25 on nonelementary integrals in this section), and others cannot be evaluated by the usual methods of calculus (see App. 3, (24)–(44) for such "difficult" integrals). We then need methods from numerical analysis to evaluate such integrals. We also need numerics when the integrand of the integral to be evaluated consists of an empirical function, where we are given some recorded values of that function. Methods that address these kinds of problems are called methods of numeric integration.

Numeric integration means the numeric evaluation of integrals

$$J = \int_a^b f(x)\,dx$$

where a and b are given and f is a function given analytically by a formula or empirically by a table of values. Geometrically, J is the area under the curve of f between a and b (Fig. 440), taken with a minus sign where f is negative.

We know that if f is such that we can find a differentiable function F whose derivative is f, then we can evaluate J directly, i.e., without resorting to numeric integration, by applying the familiar formula

$$J = \int_a^b f(x)\,dx = F(b) - F(a) \qquad [F'(x) = f(x)].$$

Your CAS (Mathematica, Maple, etc.) or tables of integrals may be helpful for this purpose.

Rectangular Rule. Trapezoidal Rule

Numeric integration methods are obtained by approximating the integrand f by functions that can easily be integrated.

The simplest formula, the **rectangular rule**, is obtained if we subdivide the interval of integration $a \leqq x \leqq b$ into n subintervals of equal length $h = (b - a)/n$ and in each subinterval approximate f by the constant $f(x_j^*)$, the value of f at the midpoint x_j^* of the jth subinterval (Fig. 441). Then f is approximated by a **step function** (piecewise constant function), the n rectangles in Fig. 441 have the areas $f(x_1^*)h, \cdots, f(x_n^*)h$, and the **rectangular rule** is

$$\textbf{(1)} \qquad \boxed{J = \int_a^b f(x)\,dx \approx h[f(x_1^*) + f(x_2^*) + \cdots + f(x_n^*)]} \qquad \left(h = \frac{b-a}{n}\right).$$

The **trapezoidal rule** is generally more accurate. We obtain it if we take the same subdivision as before and approximate f by a broken line of segments (chords) with endpoints $[a, f(a)], [x_1, f(x_1)], \cdots, [b, f(b)]$ on the curve of f (Fig. 442). Then the area under the curve of f between a and b is approximated by n trapezoids of areas

$$\tfrac{1}{2}[f(a) + f(x_1)]h, \qquad \tfrac{1}{2}[f(x_1) + f(x_2)]h, \qquad \cdots, \qquad \tfrac{1}{2}[f(x_{n-1}) + f(b)]h.$$

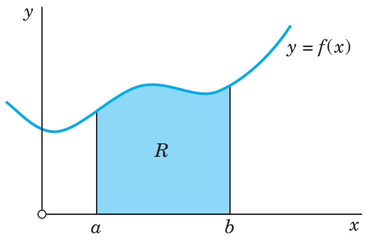

Fig. 440. Geometric interpretation of a definite integral

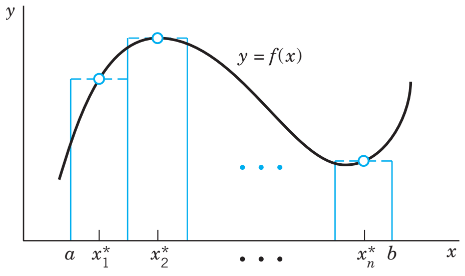

Fig. 441. Rectangular rule

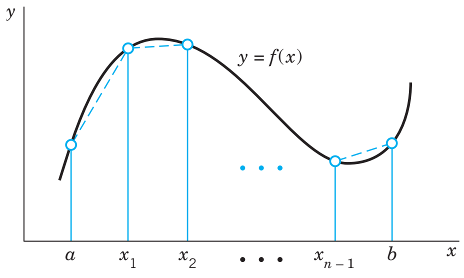

Fig. 442. Trapezoidal rule

By taking their sum we obtain the **trapezoidal rule**

$$\textbf{(2)} \qquad J = \int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right]$$

where $h = (b - a)/n$, as in (1). The x_j's and a and b are called **nodes**.

EXAMPLE 1 Trapezoidal Rule

Evaluate $J = \int_0^1 e^{-x^2}\,dx$ by means of (2) with $n = 10$.

Note that this integral cannot be evaluated by elementary calculus, but leads to the error function (see Eq. (35), App. 3).

Solution. $J \approx 0.1(0.5 \cdot 1.367879 + 6.778167) = 0.746211$ from Table 19.3. ■

Table 19.3 Computations in Example 1

j	x_j	x_j^2	$e^{-x_j^2}$	
0	0	0	1.000000	
1	0.1	0.01		0.990050
2	0.2	0.04		0.960789
3	0.3	0.09		0.913931
4	0.4	0.16		0.852144
5	0.5	0.25		0.778801
6	0.6	0.36		0.697676
7	0.7	0.49		0.612626
8	0.8	0.64		0.527292
9	0.9	0.81		0.444858
10	1.0	1.00	0.367879	
Sums			1.367879	6.778167

Error Bounds and Estimate for the Trapezoidal Rule

An error estimate for the trapezoidal rule can be derived from (5) in Sec. 19.3 with $n = 1$ by integration as follows. For a single subinterval we have

$$f(x) - p_1(x) = (x - x_0)(x - x_1)\,\frac{f''(t)}{2}$$

with a suitable t depending on x, between x_0 and x_1. Integration over x from $a = x_0$ to $x_1 = x_0 + h$ gives

$$\int_{x_0}^{x_0+h} f(x)\,dx - \frac{h}{2}\,[f(x_0) + f(x_1)] = \int_{x_0}^{x_0+h} (x - x_0)(x - x_0 - h)\,\frac{f''(t(x))}{2}\,dx.$$

Setting $x - x_0 = v$ and applying the mean value theorem of integral calculus, which we can use because $(x - x_0)(x - x_0 - h)$ does not change sign, we find that the right side equals

$$(3^*) \qquad \int_0^h v(v-h)\,dv\,\frac{f''(\tilde{t})}{2} = \left(\frac{h^3}{3} - \frac{h^3}{2}\right)\frac{f''(\tilde{t})}{2} = -\frac{h^3}{12}\,f''(\tilde{t})$$

where $\tilde{t}$ is a (suitable, unknown) value between x_0 and x_1. This is the error for the trapezoidal rule with $n = 1$, often called the **local error**.

Hence the **error** ϵ of (2) with any n is the sum of such contributions from the n subintervals; since $h = (b-a)/n$, $nh^3 = n(b-a)^3/n^3$, and $(b-a)^2 = n^2h^2$, we obtain

$$\textbf{(3)} \qquad \epsilon = -\frac{(b-a)^3}{12n^2}\,f''(\hat{t}) = -\frac{b-a}{12}\,h^2 f''(\hat{t})$$

with (suitable, unknown) $\hat{t}$ between a and b.

Because of (3) the trapezoidal rule (2) is also written

$$(2^*) \quad J = \int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(a) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right] - \frac{b-a}{12}\,h^2 f''(\hat{t}).$$

Error Bounds are now obtained by taking the largest value for f'', say, M_2, and the smallest value, M_2^*, in the interval of integration. Then (3) gives (note that K is negative)

$$\textbf{(4)} \qquad KM_2 \leqq \epsilon \leqq KM_2^* \quad \text{where} \quad K = -\frac{(b-a)^3}{12n^2} = -\frac{b-a}{12}\,h^2.$$

Error Estimation by Halving *h* is advisable if f'' is very complicated or unknown, for instance, in the case of experimental data. Then we may apply the Error Principle of Sec. 19.1. That is, we calculate by (2), first with h, obtaining, say, $J = J_h + \epsilon_h$, and then with $\frac{1}{2}h$, obtaining $J = J_{h/2} + \epsilon_{h/2}$. Now if we replace h^2 in (3) with $(\frac{1}{2}h)^2$, the error is multiplied by $\frac{1}{4}$. Hence $\epsilon_{h/2} \approx \frac{1}{4}\epsilon_h$ (not exactly because $\hat{t}$ may differ). Together, $J_{h/2} + \epsilon_{h/2} = J_h + \epsilon_h \approx J_h + 4\epsilon_{h/2}$. Thus $J_{h/2} - J_h = (4-1)\epsilon_{h/2}$. Division by 3 gives the error formula for $J_{h/2}$

$$\textbf{(5)} \qquad \epsilon_{h/2} \approx \tfrac{1}{3}\,(J_{h/2} - J_h).$$

EXAMPLE 2 Error Estimation for the Trapezoidal Rule by (4) and (5)

Estimate the error of the approximate value in Example 1 by (4) and (5).

Solution. **(A) *Error bounds by* (4).** By differentiation, $f''(x) = 2(2x^2 - 1)e^{-x^2}$. Also, $f'''(x) > 0$ if $0 < x < 1$, so that the minimum and maximum occur at the ends of the interval. We compute $M_2 = f''(1) = 0.735759$ and $M_2^* = f''(0) = -2$. Furthermore, $K = -1/1200$, and (4) gives

$$-0.000614 \leqq \epsilon \leqq 0.001667.$$

Hence the exact value of J must lie between

$$0.746211 - 0.000614 = 0.745597 \qquad \text{and} \qquad 0.746211 + 0.001667 = 0.747878.$$

Actually, $J = 0.746824$, exact to 6D.

(B) ***Error estimate by*** **(5).** $J_h = 0.746211$ in Example 1. Also,

$$J_{h/2} = 0.05\left[\sum_{j=1}^{19} e^{-(j/20)^2} + \frac{1}{2}(1 + 0.367879)\right] = 0.746671.$$

Hence $\epsilon_{h/2} = \frac{1}{3}(J_{h/2} - J_h) = 0.000153$ and $J_{h/2} + \epsilon_{h/2} = 0.746824$, exact to 6D. ■

Simpson's Rule of Integration

Piecewise constant approximation of f led to the rectangular rule (1), piecewise linear approximation to the trapezoidal rule (2), and piecewise quadratic approximation will lead to Simpson's rule, which is of great practical importance because it is sufficiently accurate for most problems, but still sufficiently simple.

To derive Simpson's rule, we divide the interval of integration $a \leqq x \leqq b$ into an ***even number*** of equal subintervals, say, into $n = 2m$ subintervals of length $h = (b - a)/(2m)$, with endpoints $x_0 (= a), x_1, \cdots, x_{2m-1}, x_{2m} (= b)$; see Fig. 443. We now take the first two subintervals and approximate $f(x)$ in the interval $x_0 \leqq x \leqq x_2 = x_0 + 2h$ by the Lagrange polynomial $p_2(x)$ through $(x_0, f_0), (x_1, f_1), (x_2, f_2)$, where $f_j = f(x_j)$. From (3) in Sec. 19.3 we obtain

$$(6) \quad p_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f_2.$$

The denominators in (6) are $2h^2$, $-h^2$, and $2h^2$, respectively. Setting $s = (x - x_1)/h$, we have

$$x - x_1 = sh, \qquad x - x_0 = x - (x_1 - h) = (s + 1)h$$
$$x - x_2 = x - (x_1 + h) = (s - 1)h$$

and we obtain

$$p_2(x) = \tfrac{1}{2}s(s - 1)f_0 - (s + 1)(s - 1)f_1 + \tfrac{1}{2}(s + 1)sf_2.$$

We now integrate with respect to x from x_0 to x_2. This corresponds to integrating with respect to s from -1 to 1. Since $dx = h\,ds$, the result is

$$(7^*) \qquad \int_{x_0}^{x_2} f(x)\,dx \approx \int_{x_0}^{x_2} p_2(x)\,dx = h\left(\frac{1}{3} f_0 + \frac{4}{3} f_1 + \frac{1}{3} f_2\right).$$

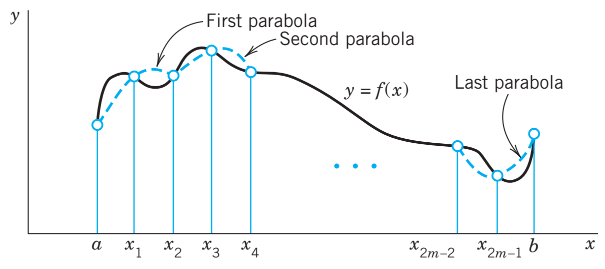

Fig. 443. Simpson's rule

A similar formula holds for the next two subintervals from x_2 to x_4, and so on. By summing all these m formulas we obtain **Simpson's rule**[4]

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\,(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m}), \tag{7}$$

where $h = (b - a)/(2m)$ and $f_j = f(x_j)$. Table 19.4 shows an algorithm for Simpson's rule.

Table 19.4 Simpson's Rule of Integration

```
ALGORITHM SIMPSON ($a, b, m, f_0, f_1, \cdots, f_{2m}$)

This algorithm computes the integral $J = \int_a^b f(x)\,dx$ from given values $f_j = f(x_j)$ at
equidistant $x_0 = a, x_1 = x_0 + h, \cdots, x_{2m} = x_0 + 2mh = b$ by Simpson's rule (7),
where $h = (b - a)/(2m)$.

    INPUT:   $a, b, m, f_0, \cdots, f_{2m}$

    OUTPUT:  Approximate value $\tilde{J}$ of $J$

    Compute  $s_0 = f_0 + f_{2m}$
             $s_1 = f_1 + f_3 + \cdots + f_{2m-1}$
             $s_2 = f_2 + f_4 + \cdots + f_{2m-2}$
             $h = (b - a)/2m$
             $\tilde{J} = \frac{h}{3}(s_0 + 4s_1 + 2s_2)$

    OUTPUT $\tilde{J}$. Stop.

End SIMPSON
```

Error of Simpson's Rule (7). If the fourth derivative $f^{(4)}$ exists and is continuous on $a \leqq x \leqq b$, the **error** of (7), call it ϵ_S, is

$$\epsilon_S = -\frac{(b-a)^5}{180\,(2m)^4}\,f^{(4)}(\hat{t}) = -\frac{b-a}{180}\,h^4 f^{(4)}(\hat{t}); \tag{8}$$

here $\hat{t}$ is a suitable unknown value between a and b. This is obtained similarly to (3). With this we may also write Simpson's rule (7) as

$$\int_a^b f(x)\,dx = \frac{h}{3}\,(f_0 + 4f_1 + \cdots + f_{2m}) - \frac{b-a}{180}\,h^4 f^{(4)}(\hat{t}). \tag{7**}$$

[4]THOMAS SIMPSON (1710–1761), self-taught English mathematician, author of several popular textbooks. Simpson's rule was used much earlier by Torricelli, Gregory (in 1668), and Newton (in 1676).

Error Bounds. By taking for $f^{(4)}$ in (8) the maximum M_4 and minimum M_4^* on the interval of integration we obtain from (8) the error bounds (note that C is negative)

$$CM_4 \leqq \epsilon_S \leqq CM_4^* \qquad \text{where} \qquad C = -\frac{(b-a)^5}{180(2m)^4} = -\frac{b-a}{180}h^4. \tag{9}$$

Degree of Precision (DP) *of an integration formula.* This is the maximum degree of arbitrary polynomials for which the formula gives exact values of integrals over any intervals.

Hence for the trapezoidal rule,

$$\text{DP} = 1$$

because we approximate the curve of f by portions of straight lines (linear polynomials).

For Simpson's rule we might expect DP $= 2$ (why?). Actually,

$$\text{DP} = 3$$

by (9) because $f^{(4)}$ is identically zero for a cubic polynomial. This makes Simpson's rule sufficiently accurate for most practical problems and accounts for its popularity.

Numeric Stability *with respect to rounding* is another important property of Simpson's rule. Indeed, for the sum of the roundoff errors ϵ_j of the $2m + 1$ values f_j in (7) we obtain, since $h = (b-a)/2m$,

$$\frac{h}{3}\,|\epsilon_0 + 4\epsilon_1 + \cdots + \epsilon_{2m}| \leqq \frac{b-a}{3.2m}\,6mu = (b-a)u$$

where u is the rounding unit ($u = \frac{1}{2} \cdot 10^{-6}$ if we round off to 6D; see Sec. 19.1). Also $6 = 1 + 4 + 1$ is the sum of the coefficients for a pair of intervals in (7); take $m = 1$ in (7) to see this. The bound $(b-a)u$ is independent of m, so that it cannot increase with increasing m, that is, with decreasing h. This proves stability. ■

Newton–Cotes Formulas. We mention that the trapezoidal and Simpson rules are special *closed Newton–Cotes formulas*, that is, integration formulas in which $f(x)$ is interpolated at equally spaced nodes by a polynomial of degree n ($n = 1$ for trapezoidal, $n = 2$ for Simpson), and **closed** means that a and b are nodes ($a = x_0$, $b = x_n$). $n = 3$ and higher n are used occasionally. From $n = 8$ on, some of the coefficients become negative, so that a positive f_j could make a negative contribution to an integral, which is absurd. For more on this topic see Ref. [E25] in App. 1.

EXAMPLE 3 Simpson's Rule. Error Estimate

Evaluate $J = \int_0^1 e^{-x^2}dx$ by Simpson's rule with $2m = 10$ and estimate the error.

Solution. Since $h = 0.1$, Table 19.5 gives

$$J \approx \frac{0.1}{3}(1.367879 + 4 \cdot 3.740266 + 2 \cdot 3.037901) = 0.746825.$$

Estimate of error. Differentiation gives $f^{(4)}(x) = 4(4x^4 - 12x^2 + 3)e^{-x^2}$. By considering the derivative $f^{(5)}$ of $f^{(4)}$ we find that the largest value of $f^{(4)}$ in the interval of integration occurs at 0 and the smallest value at $x^* = (2.5 - 0.5\sqrt{10})^{1/2}$. Computation gives the values $M_4 = f^{(4)}(0) = 12$ and $M_4^* = f^{(4)}(x^*) = -7.419$. Since $2m = 10$ and $b - a = 1$, we obtain $C = -1/1800000 = -0.00000056$. Therefore, from (9),

$$-0.000007 \leqq \epsilon_s \leqq 0.000005.$$

Hence J must lie between $0.746825 - 0.000007 = 0.746818$ and $0.746825 + 0.000005 = 0.746830$, so that at least four digits of our approximate value are exact. Actually, the value 0.746825 is exact to 5D because $J = 0.746824$ (exact to 6D).

Thus our result is much better than that in Example 1 obtained by the trapezoidal rule, whereas the number of operations is nearly the same in both cases. ■

Table 19.5 Computations in Example 3

j	x_j	x_j^2	$e^{-x_j^2}$		
0	0	0	1.000000		
1	0.1	0.01		0.990050	
2	0.2	0.04			0.960789
3	0.3	0.09		0.913931	
4	0.4	0.16			0.852144
5	0.5	0.25		0.778801	
6	0.6	0.36			0.697676
7	0.7	0.49		0.612626	
8	0.8	0.64			0.527292
9	0.9	0.81		0.444858	
10	1.0	1.00	0.367879		
Sums			1.367879	3.740266	3.037901

Instead of picking an $n = 2m$ and then estimating the error by (9), as in Example 3, it is better to require an accuracy (e.g., 6D) and then determine $n = 2m$ from (9).

EXAMPLE 4 Determination of $n = 2m$ in Simpson's Rule from the Required Accuracy

What n should we choose in Example 3 to get 6D-accuracy?

Solution. Using $M_4 = 12$ (which is bigger in absolute value than M_4^*, we get from (9), with $b - a = 1$ and the required accuracy,

$$|CM_4| = \frac{12}{180(2m)^4} = \frac{1}{2} \cdot 10^{-6}, \qquad \text{thus} \qquad m = \left[\frac{2 \cdot 10^6 \cdot 12}{180 \cdot 2^4}\right]^{1/4} = 9.55.$$

Hence we should choose $n = 2m = 20$. Do the computation, which parallels that in Example 3.

Note that the error bounds in (4) or (9) may sometimes be loose, so that in such a case a smaller $n = 2m$ may already suffice. ■

Error Estimation for Simpson's Rule by Halving h. The idea is the same as in (5) and gives

(10)
$$\epsilon_{h/2} \approx \tfrac{1}{15}(J_{h/2} - J_h).$$

J_h is obtained by using h and $J_{h/2}$ by using $\frac{1}{2}h$, and $\epsilon_{h/2}$ is the error of $J_{h/2}$.

Derivation. In (5) we had $\frac{1}{3}$ as the reciprocal of $3 = 4 - 1$ and $\frac{1}{4} = (\frac{1}{2})^2$ resulted from h^2 in (3) by replacing h with $\frac{1}{2}h$. In (10) we have $\frac{1}{15}$ as the reciprocal of $15 = 16 - 1$ and $\frac{1}{16} = (\frac{1}{2})^4$ results from h^4 in (8) by replacing h with $\frac{1}{2}h$.

EXAMPLE 5 Error Estimation for Simpson's Rule by Halving

Integrate $f(x) = \frac{1}{4}\pi x^4 \cos \frac{1}{4}\pi x$ from 0 to 2 with $h = 1$ and apply (10).

Solution. The exact 5D-value of the integral is $J = 1.25953$. Simpson's rule gives

$$\begin{aligned} J_h &= \tfrac{1}{3}[f(0) + 4f(1) + f(2)] = \tfrac{1}{3}(0 + 4 \cdot 0.555360 + 0) = 0.740480, \\ J_{h/2} &= \tfrac{1}{6}\,[f(0) + 4f(\tfrac{1}{2}) + 2f(1) + 4f(\tfrac{3}{2}) + f(2)] \\ &= \tfrac{1}{6}[0 + 4 \cdot 0.045351 + 2 \cdot 0.555361 + 4 \cdot 1.521579 + 0] = 1.22974. \end{aligned}$$

Hence (10) gives $\epsilon_{h/2} = \frac{1}{15}(1.22974 - 0.74048) = 0.032617$ and thus $J \approx J_{h/2} + \epsilon_{h/2} = 1.26236$, with an error -0.00283 which is less in absolute value than $\frac{1}{10}$ of the error 0.02979 of $J_{h/2}$. Hence the use of (10) was well worthwhile. ■

Adaptive Integration

The idea is to adapt step h to the variability of $f(x)$. That is, where f varies but little, we can proceed in large steps without causing a substantial error in the integral, but where f varies rapidly, we have to take small steps in order to stay everywhere close enough to the curve of f.

Changing h is done systematically, usually by halving h, and automatically (not "by hand") depending on the size of the (estimated) error over a subinterval. The subinterval is halved if the corresponding error is still too large, that is, larger than a given **tolerance** TOL (maximum admissible absolute error), or is not halved if the error is less than or equal to TOL (or doubled if the error is very small).

Adapting is one of the techniques typical of modern software. In connection with integration it can be applied to various methods. We explain it here for Simpson's rule. In Table 19.6 an asterisk means that for that subinterval, TOL has been reached.

EXAMPLE 6 Adaptive Integration with Simpson's Rule

Integrate $f(x) = \frac{1}{4}\pi x^4 \cos \frac{1}{4}\pi x$ from $x = 0$ to 2 by adaptive integration and with Simpson's rule and TOL[0, 2] = 0.0002.

Solution. Table 19.6 shows the calculations. Figure 444 shows the integrand $f(x)$ and the adapted intervals used. The first two intervals ([0, 0.5], [0.5, 1.0]) have length 0.5, hence $h = 0.25$ [because we use $2m = 2$ subintervals in Simpson's rule (7**)]. The next two intervals ([1.00, 1.25], [1.25, 1.50]) have length 0.25 (hence $h = 0.125$) and the last four intervals have length 0.125. *Sample computations.* For 0.740480 see Example 5. Formula (10) gives $(0.123716 - 0.122794)/15 = 0.000061$. Note that 0.123716 refers to [0, 0.5] and [0.5, 1], so that we must subtract the value corresponding to [0, 1] in the line before. Etc. TOL[0, 2] = 0.0002 gives 0.0001 for subintervals of length 1, 0.00005 for length 0.5, etc. The value of the integral obtained is the sum of the values marked by an asterisk (for which the error estimate has become less than TOL). This gives

$$J \approx 0.123716 + 0.528895 + 0.388263 + 0.218483 = 1.25936.$$

The exact 5D-value is $J = 1.25953$. Hence the error is 0.00017. This is about 1/200 of the absolute value of that in Example 5. Our more extensive computation has produced a much better result. ■

Table 19.6 Computations in Example 6

Interval	Integral	Error (10)	TOL	Comment
[0, 2]	0.740480		0.0002	
[0, 1]	0.122794			
[1, 2]	1.10695			
	Sum = 1.22974	0.032617	0.0002	Divide further
[0.0, 0.5]	0.004782			
[0.5, 1.0]	0.118934			
	Sum = 0.123716*	0.000061	0.0001	TOL reached
[1.0, 1.5]	0.528176			
[1.5, 2.0]	0.605821			
	Sum = 1.13300	0.001803	0.0001	Divide further
[1.00, 1.25]	0.200544			
[1.25, 1.50]	0.328351			
	Sum = 0.528895*	0.000048	0.00005	TOL reached
[1.50, 1.75]	0.388235			
[1.75, 2.00]	0.218457			
	Sum = 0.606692	0.000058	0.00005	Divide further
[1.500, 1.625]	0.196244			
[1.625, 1.750]	0.192019			
	Sum = 0.388263*	0.000002	0.000025	TOL reached
[1.750, 1.875]	0.153405			
[1.875, 2.000]	0.065078			
	Sum = 0.218483*	0.000002	0.000025	TOL reached

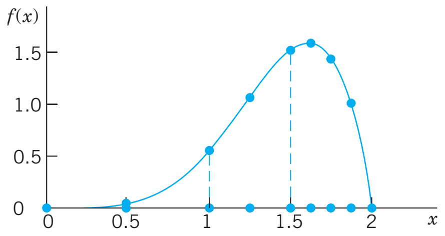

Fig. 444. Adaptive integration in Example 6

Gauss Integration Formulas Maximum Degree of Precision

Our integration formulas discussed so far use function values at *predetermined* (equidistant) x-values (nodes) and give exact results for polynomials not exceeding a certain degree [called the *degree of precision*; see after (9)]. But we can get much more accurate integration formulas as follows. We set

$$\int_{-1}^{1} f(t)\,dt \approx \sum_{j=1}^{n} A_j f_j \qquad [f_j = f(t_j)] \tag{11}$$

with fixed n, and $t = \pm 1$ obtained from $x = a, b$ by setting $x = \frac{1}{2}[a(t-1) + b(t+1)]$. Then we determine the n coefficients $A_1, \cdots, A_n$ and n nodes $t_1, \cdots, t_n$ so that (11) gives exact results for polynomials of degree k as high as possible. Since $n + n = 2n$ is the number of coefficients of a polynomial of degree $2n - 1$, it follows that $k \leqq 2n - 1$.

Gauss has shown that exactness for polynomials of degree not exceeding $2n - 1$ (instead of $n - 1$ for predetermined nodes) can be attained, and he has given the location of the t_j(= the jth zero of the Legendre polynomial P_n in Sec. 5.3) and the coefficients A_j which depend on n but not on $f(t)$, and are obtained by using Lagrange's interpolation polynomial, as shown in Ref. [E5] listed in App. 1. With these t_j and A_j, formula (11) is called a **Gauss integration formula** or *Gauss quadrature formula*. Its degree of precision is $2n - 1$, as just explained. Table 19.7 gives the values needed for $n = 2, \cdots, 5$. (For larger n, see pp. 916–919 of Ref. [GenRef1] in App. 1.)

Table 19.7 Gauss Integration: Nodes t_j and Coefficients A_j

n	Nodes t_j	Coefficients A_j	Degree of Precision
2	−0.5773502692	1	3
	0.5773502692	1	
3	−0.7745966692	0.5555555556	5
	0	0.8888888889	
	0.7745966692	0.5555555556	
4	−0.8611363116	0.3478548451	7
	−0.3399810436	0.6521451549	
	0.3399810436	0.6521451549	
	0.8611363116	0.3478548451	
5	−0.9061798459	0.2369268851	9
	−0.5384693101	0.4786286705	
	0	0.5688888889	
	0.5384693101	0.4786286705	
	0.9061798459	0.2369268851	

EXAMPLE 7 Gauss Integration Formula with $n = 3$

Evaluate the integral in Example 3 by the Gauss integration formula (11) with $n = 3$.

Solution. We have to convert our integral from 0 to 1 into an integral from -1 to 1. We set $x = \frac{1}{2}(t + 1)$. Then $dx = \frac{1}{2}\,dt$, and (11) with $n = 3$ and the above values of the nodes and the coefficients yields

$$\int_0^1 \exp(-x^2)\,dx = \frac{1}{2}\int_{-1}^{1} \exp\left(-\frac{1}{4}(t+1)^2\right)dt$$

$$\approx \frac{1}{2}\left[\frac{5}{9}\exp\left(-\frac{1}{4}\left(1-\sqrt{\frac{3}{5}}\right)^2\right) + \frac{8}{9}\exp\left(-\frac{1}{4}\right) + \frac{5}{9}\exp\left(-\frac{1}{4}\left(1+\sqrt{\frac{3}{5}}\right)^2\right)\right] = 0.746815$$

(exact to 6D: 0.746825), which is almost as accurate as the Simpson result obtained in Example 3 with a much larger number of arithmetic operations. With 3 function values (as in this example) and Simpson's rule we would get $\frac{1}{6}(1 + 4e^{-0.25} + e^{-1}) = 0.747180$, with an error over 30 times that of the Gauss integration. ■

EXAMPLE 8 Gauss Integration Formula with $n = 4$ and 5

Integrate $f(x) = \frac{1}{4}\pi x^4 \cos \frac{1}{4}\pi x$ from $x = 0$ to 2 by Gauss. Compare with the adaptive integration in Example 6 and comment.

Solution. $x = t + 1$ gives $f(t) = \frac{1}{4}\pi(t+1)^4 \cos(\frac{1}{4}\pi(t+1))$, as needed in (11). For $n = 4$ we calculate (6S)

$$\begin{aligned} J &\approx A_1 f_1 + \cdots + A_4 f_4 = A_1(f_1 + f_4) + A_2(f_2 + f_3) \\ &= 0.347855(0.000290309 + 1.02570) + 0.652145(0.129464 + 1.25459) = 1.25950. \end{aligned}$$

The error is 0.00003 because $J = 1.25953$ (6S). Calculating with 10S and $n = 4$ gives the same result; so the error is due to the formula, not rounding. For $n = 5$ and 10S we get $J \approx 1.259526185$, too large by the amount 0.000000250 because $J = 1.259525935$ (10S). The accuracy is impressive, particularly if we compare the amount of work with that in Example 6. ■

Gauss integration is of considerable practical importance. Whenever the integrand f is given by a formula (not just by a table of numbers) or when experimental measurements can be set at times t_j (or whatever t represents) shown in Table 19.7 or in Ref. [GenRef1], then the great accuracy of Gauss integration outweighs the disadvantage of the complicated t_j and A_j (which may have to be stored). Also, Gauss coefficients A_j are positive for all n, in contrast with some of the Newton–Cotes coefficients for larger n.

Of course, there are frequent applications with equally spaced nodes, so that Gauss integration does not apply (or has no great advantage if one first has to get the t_j in (11) by interpolation).

Since the endpoints -1 and 1 of the interval of integration in (11) are not zeros of P_n, they do not occur among $t_0, \cdots, t_n$, and the Gauss formula (11) is called, therefore, an **open formula**, in contrast with a **closed formula**, in which the endpoints of the interval of integration are t_0 and t_n. [For example, (2) and (7) are closed formulas.]

Numeric Differentiation

Numeric differentiation is the computation of values of the derivative of a function f from given values of f. Numeric differentiation should be avoided whenever possible. Whereas *integration* is a smoothing process and is not very sensitive to small inaccuracies in function values, *differentiation* tends to make matters rough and generally gives values of f' that are much less accurate than those of f. The difficulty with differentiation is tied in with the definition of the derivative, which is the limit of the difference quotient, and, in that quotient, you usually have the difference of a large quantity divided by a small quantity. This can cause numerical instability. While being aware of this caveat, we must still develop basic differentiation formulas for use in numeric solutions of differential equations.

We use the notations $f'_j = f'(x_j)$, $f''_j = f''(x_j)$, etc., and may obtain rough approximation formulas for derivatives by remembering that

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

This suggests

$$f'_{1/2} \approx \frac{\delta f_{1/2}}{h} = \frac{f_1 - f_0}{h}. \tag{12}$$

Similarly, for the second derivative we obtain

$$f''_1 \approx \frac{\delta^2 f_1}{h^2} = \frac{f_2 - 2f_1 + f_0}{h^2}, \qquad \text{etc.} \tag{13}$$

More accurate approximations are obtained by differentiating suitable Lagrange polynomials. Differentiating (6) and remembering that the denominators in (6) are $2h^2$, $-h^2$, $2h^2$, we have

$$f'(x) \approx p'_2(x) = \frac{2x - x_1 - x_2}{2h^2} f_0 - \frac{2x - x_0 - x_2}{h^2} f_1 + \frac{2x - x_0 - x_1}{2h^2} f_2.$$

Evaluating this at x_0, x_1, x_2, we obtain the "three-point formulas"

$$\begin{aligned} &\text{(a)} \quad f'_0 \approx \frac{1}{2h}(-3f_0 + 4f_1 - f_2), \\ &\text{(b)} \quad f'_1 \approx \frac{1}{2h}(-f_0 + f_2), \\ &\text{(c)} \quad f'_2 \approx \frac{1}{2h}(f_0 - 4f_1 + 3f_2). \end{aligned} \tag{14}$$

Applying the same idea to the Lagrange polynomial $p_4(x)$, we obtain similar formulas, in particular,

$$f'_2 \approx \frac{1}{12h}(f_0 - 8f_1 + 8f_3 - f_4). \tag{15}$$

Some examples and further formulas are included in the problem set as well as in Ref. [E5] listed in App. 1.

PROBLEM SET 19.5

1–6 RECTANGULAR AND TRAPEZOIDAL RULES

1. **Rectangular rule.** Evaluate the integral in Example 1 by the rectangular rule (1) with subintervals of length 0.1. Compare with Example 1. (6S-exact: 0.746824)
2. **Bounds for (1).** Derive a formula for lower and upper bounds for the rectangular rule. Apply it to Prob. 1.
3. **Trapezoidal rule.** To get a feel for increase in accuracy, integrate x^2 from 0 to 1 by (2) with $h = 1, 0.5, 0.25, 0.1$.
4. **Error estimation by halfing.** Integrate $f(x) = x^4$ from 0 to 1 by (2) with $h = 1, h = 0.5, h = 0.25$ and estimate the error for $h = 0.5$ and $h = 0.25$ by (5).
5. **Error estimation.** Do the tasks in Prob. 4 for $f(x) = \sin \frac{1}{2}\pi x$.

6. Stability. Prove that the trapezoidal rule is stable with respect to rounding.

7–15 SIMPSON'S RULE

Evaluate the integrals $A = \int_1^2 \frac{dx}{x}$, $B = \int_0^{0.4} xe^{-x^2}\,dx$, $J = \int_0^1 \frac{dx}{1+x^2}$ by Simpson's rule with $2m$ as indicated, and compare with the exact value known from calculus.

7. A, $2m = 4$

8. A, $2m = 10$

9. B, $2m = 4$

10. B, $2m = 10$

11. J, $2m = 4$

12. J, $2m = 10$

13. Error estimate. Compute the integral J by Simpson's rule with $2m = 8$ and use the value and that in Prob. 11 to estimate the error by (10).

14. Error bounds and estimate. Integrate e^{-x} from 0 to 2 by (7) with $h = 1$ and with $h = 0.5$. Give error bounds for the $h = 0.5$ value and an error estimate by (10).

15. Given TOL. Find the smallest n in computing A (see Probs. 7 and 8) such that 5S-accuracy is guaranteed **(a)** by (4) in the use of (2), **(b)** by (9) in the use of (7).

16–21 NONELEMENTARY INTEGRALS

The following integrals cannot be evaluated by the usual methods of calculus. Evaluate them as indicated. Compare your value with that possibly given by your CAS. Si(x) is the sine integral. S(x) and C(x) are the Fresnel integrals. See App. A3.1. They occur in optics.

$$\mathrm{Si}(x) = \int_0^x \frac{\sin x^*}{x^*}\,dx^*,$$

$$\mathrm{S}(x) = \int_0^x \sin(x^{*2})\,dx^*, \quad \mathrm{C}(x) = \int_0^x \cos(x^{*2})\,dx^*$$

16. Si(1) by (2), $n = 5$, $n = 10$, and apply (5).

17. Si(1) by (7), $2m = 2$, $2m = 4$

18. Obtain a better value in Prob. 17. *Hint.* Use (10).

19. Si(1) by (7), $2m = 10$

20. S(1.25) by (7), $2m = 10$

21. C(1.25) by (7), $2m = 10$

22–25 GAUSS INTEGRATION

Integrate by (11) with $n = 5$:

22. $\cos x$ from 0 to $\frac{1}{2}\pi$

23. xe^{-x} from 0 to 1

24. $\sin(x^2)$ from 0 to 1.25

25. $\exp(-x^2)$ from 0 to 1

26. TEAM PROJECT. Romberg Integration (W. Romberg, *Norske Videnskab. Trondheim, Fφrh.* 28, Nr. 7, 1955). This method uses the trapezoidal rule and gains precision stepwise by halving h and adding an error estimate. Do this for the integral of $f(x) = e^{-x}$ from $x = 0$ to $x = 2$ with TOL $= 10^{-3}$, as follows.

Step 1. Apply the trapezoidal rule (2) with $h = 2$ (hence $n = 1$) to get an approximation J_{11}. Halve h and use (2) to get J_{21} and an error estimate

$$\epsilon_{21} = \frac{1}{2^2 - 1}(J_{21} - J_{11}).$$

If $|\epsilon_{21}| \leqq$ TOL, stop. The result is $J_{22} = J_{21} + \epsilon_{21}$.

Step 2. Show that $\epsilon_{21} = -0.066596$, hence $|\epsilon_{21}| >$ TOL and go on. Use (2) with $h/4$ to get J_{31} and add to it the error estimate $\epsilon_{31} = \frac{1}{3}(J_{31} - J_{21})$ to get the better $J_{32} = J_{31} + \epsilon_{31}$. Calculate

$$\epsilon_{32} = \frac{1}{2^4 - 1}(J_{32} - J_{22}) = \frac{1}{15}(J_{32} - J_{22}).$$

If $|\epsilon_{32}| \leqq$ TOL, stop. The result is $J_{33} = J_{32} + \epsilon_{32}$. (Why does $2^4 = 16$ come in?) Show that we obtain $\epsilon_{32} = -0.000266$, so that we can stop. Arrange your J- and ϵ-values in a kind of "difference table."

$$\begin{tikzcd}
J_{11} \arrow[dr] & & & & \\
& \epsilon_{21} \arrow[dr] & & & \\
J_{21} \arrow[ur] \arrow[dr] & & J_{22} \arrow[dr] & & \\
& \epsilon_{31} \arrow[dr] & & \epsilon_{32} \arrow[dr] & \\
J_{31} \arrow[ur] & & J_{32} \arrow[ur] & & J_{33}
\end{tikzcd}$$

If $|\epsilon_{32}|$ were greater than TOL, you would have to go on and calculate in the next step J_{41} from (2) with $h = \frac{1}{4}$; then

$$\begin{aligned}
J_{42} &= J_{41} + \epsilon_{41} &\quad \text{with} \quad \epsilon_{41} &= \tfrac{1}{3}(J_{41} - J_{31})\\
J_{43} &= J_{42} + \epsilon_{42} &\quad \text{with} \quad \epsilon_{42} &= \tfrac{1}{15}(J_{42} - J_{32})\\
J_{44} &= J_{43} + \epsilon_{43} &\quad \text{with} \quad \epsilon_{43} &= \tfrac{1}{63}(J_{43} - J_{33})
\end{aligned}$$

where $63 = 2^6 - 1$. (How does this come in?)

Apply the Romberg method to the integral of $f(x) = \frac{1}{4}\pi x^4 \cos\frac{1}{4}\pi x$ from $x = 0$ to 2 with TOL $= 10^{-4}$.

27–30 DIFFERENTIATION

27. Consider $f(x) = x^4$ for $x_0 = 0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6, x_4 = 0.8$. Calculate f_2' from (14a), (14b), (14c), (15). Determine the errors. Compare and comment.

28. A "**four-point formula**" for the derivative is

$$f_2' \approx \frac{1}{6h}\,(-2f_1 - 3f_2 + 6f_3 - f_4).$$

Apply it to $f(x) = x^4$ with $x_1, \cdots, x_4$ as in Prob. 27, determine the error, and compare it with that in the case of (15).

29. The derivative $f'(x)$ can also be approximated in terms of first-order and higher order differences (see Sec. 19.3):

$$f'(x_0) \approx \frac{1}{h}\left(\Delta f_0 - \frac{1}{2}\Delta^2 f_0 + \frac{1}{3}\Delta^3 f_0 - \frac{1}{4}\Delta^4 f_0 + - \cdots\right).$$

Compute $f'(0.4)$ in Prob. 27 from this formula, using differences up to and including first order, second order, third order, fourth order.

30. Derive the formula in Prob. 29 from (14) in Sec. 19.3.

CHAPTER 19 REVIEW QUESTIONS AND PROBLEMS

1. What is a numeric method? How has the computer influenced numerics?

2. What is an error? A relative error? An error bound?

3. Why are roundoff errors important? State the rounding rules.

4. What is an algorithm? Which of its properties are important in software implementation?

5. What do you know about stability?

6. Why is the selection of a *good* method at least as important on a large computer as it is on a small one?

7. Can the Newton (–Raphson) method diverge? Is it fast? Same questions for the bisection method.

8. What is fixed-point iteration?

9. What is the advantage of Newton's interpolation formulas over Lagrange's?

10. What is spline interpolation? Its advantage over polynomial interpolation?

11. List and compare the integration methods we have discussed.

12. How did we use an interpolation polynomial in deriving Simpson's rule?

13. What is adaptive integration? Why is it useful?

14. In what sense is Gauss integration optimal?

15. How did we obtain formulas for numeric differentiation?

16. Write -46.9028104, 0.000317399, $54/7$, $-890/3$ in floating-point form with 5S (5 significant digits, properly rounded).

17. Compute $(5.346 - 3.644)/(3.444 - 3.055)$ as given and then rounded stepwise to 3S, 2S, 1S. Comment. ("Stepwise" means rounding the rounded numbers, not the given ones.)

18. Compute $0.38755/(5.6815 - 0.38419)$ as given and then rounded stepwise to 4S, 3S, 2S, 1S. Comment.

19. Let 19.1 and 25.84 be correctly rounded. Find the shortest interval in which the sum s of the true (unrounded) numbers must lie.

20. Do the same task as in Prob. 19 for the difference $3.2 - 6.29$.

21. What is the relative error of $n\tilde{a}$ in terms of that of $\tilde{a}$?

22. Show that the relative error of $\tilde{a}^2$ is about twice that of $\tilde{a}$.

23. Solve $x^2 - 40x + 2 = 0$ in two ways (cf. Sec. 19.1). Use 4S-arithmetic.

24. Solve $x^2 - 100x + 1 = 0$. Use 5S-arithmetic.

25. Compute the solution of $x^4 = x + 0.1$ near $x = 0$ by transforming the equation algebraically to the form $x = g(x)$ and starting from $x_0 = 0$.

26. Solve $\cos x = x^2$ by Newton's method, starting from $x = 0.5$.

27. Solve Prob. 25 by bisection (3S-accuracy).

28. Compute $\sinh 0.4$ from $\sinh 0$, $\sinh 0.5 = 0.521$, $\sinh 1.0 = 1.175$ by quadratic interpolation.

29. Find the cubic spline for the data $f(0) = 0$, $f(1) = 0$, $f(2) = 4$, $k_0 = -1$, $k_2 = 5$.

30. Find the cubic spline q and the interpolation polynomial p for the data (0, 0), (1, 1), (2, 6), (3, 10), with $q'(0) = 0$, $q'(3) = 0$ and graph p and q on common axes.

31. Compute the integral of x^3 from 0 to 1 by the trapezoidal rule with $n = 5$. What error bounds are obtained from (4) in Sec. 19.5? What is the actual error of the result?

32. Compute the integral of $\cos(x^2)$ from 0 to 1 by Simpson's rule with $2m = 4$.

33. Solve Prob. 32 by Gauss integration with $n = 3$ and $n = 5$.

34. Compute $f'(0.2)$ for $f(x) = x^3$ using (14b) in Sec. 19.5 with **(a)** $h = 0.2$, **(b)** $h = 0.1$. Compare the accuracy.

35. Compute $f''(0.2)$ for $f(x) = x^3$ using (13) in Sec. 19.5 with **(a)** $h = 0.2$, **(b)** $h = 0.1$.

SUMMARY OF CHAPTER 19 Numerics in General

In this chapter we discussed concepts that are relevant throughout numeric work as a whole and methods of a general nature, as opposed to methods for linear algebra (Chap. 20) or differential equations (Chap. 21).

In scientific computations we use the ***floating-point*** representation of numbers (Sec. 19.1); fixed-point representation is less suitable in most cases.

Numeric methods give approximate values $\tilde{a}$ of quantities. The **error** ϵ of $\tilde{a}$ is

$$\epsilon = a - \tilde{a} \qquad \text{(Sec. 19.1)} \tag{1}$$

where a is the exact value. The *relative error* of $\tilde{a}$ is ϵ/a. Errors arise from rounding, inaccuracy of measured values, truncation (that is, replacement of integrals by sums, series by partial sums), and so on.

An algorithm is called **numerically stable** if small changes in the initial data give only correspondingly small changes in the final results. Unstable algorithms are generally useless because errors may become so large that results will be very inaccurate. The numeric instability of algorithms must not be confused with the mathematical instability of problems ("*ill-conditioned problems*," Sec. 19.2).

Fixed-point iteration is a method for solving equations $f(x) = 0$ in which the equation is first transformed algebraically to $x = g(x)$, an initial guess x_0 for the solution is made, and then approximations $x_1, x_2, \cdots$, are successively computed by iteration from (see Sec. 19.2)

$$x_{n+1} = g(x_n) \qquad (n = 0, 1, \cdots). \tag{2}$$

Newton's method for solving equations $f(x) = 0$ is an iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{(Sec. 19.2).} \tag{3}$$

Here x_{n+1} is the x-intercept of the tangent of the curve $y = f(x)$ at the point x_n. This method is of second order (Theorem 2, Sec. 19.2). If we replace f' in (3) by a difference quotient (geometrically: we replace the tangent by a secant), we obtain the **secant method;** see (10) in Sec. 19.2. For the *bisection method* (which converges slowly) and the *method of false position,* see Problem Set 19.2.

Polynomial interpolation means the determination of a polynomial $p_n(x)$ such that $p_n(x_j) = f_j$, where $j = 0, \cdots, n$ and $(x_0, f_0), \cdots, (x_n, f_n)$ are measured or observed values, values of a function, etc. $p_n(x)$ is called an *interpolation polynomial.* For given data, $p_n(x)$ of degree n (or less) is unique. However, it can be written in different forms, notably in **Lagrange's form** (4), Sec. 19.3, or in **Newton's divided difference form** (10), Sec. 19.3, which requires fewer operations. For regularly spaced $x_0, x_1 = x_0 + h, \cdots, x_n = x_0 + nh$ the latter becomes **Newton's forward difference formula** (formula (14) in Sec. 19.3):

$$(4) \qquad f(x) \approx p_n(x) = f_0 + r\,\Delta f_0 + \cdots + \frac{r(r-1)\cdots(r-n+1)}{n!}\,\Delta^n f_0$$

where $r = (x - x_0)/h$ and the forward differences are $\Delta f_j = f_{j+1} - f_j$ and

$$\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j \qquad (k = 2, 3, \cdots).$$

A similar formula is *Newton's backward difference interpolation formula* (formula (18) in Sec. 19.3).

Interpolation polynomials may become numerically unstable as n increases, and instead of interpolating and approximating by a single high-degree polynomial it is preferable to use a cubic **spline** $g(x)$, that is, a twice continuously differentiable interpolation function [thus, $g(x_j) = f_j$], which in each subinterval $x_j \leqq x \leqq x_{j+1}$ consists of a cubic polynomial $q_j(x)$; see Sec. 19.4.

Simpson's rule of numeric integration is [see (7), Sec. 19.5]

$$(5) \qquad \int_a^b f(x)\,dx \approx \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m})$$

with equally spaced nodes $x_j = x_0 + jh, j = 1, \cdots, 2m, h = (b - a)/(2m)$, and $f_j = f(x_j)$. It is simple but accurate enough for many applications. Its degree of precision is DP = 3 because the error (8), Sec. 19.5, involves h^4. A more practical error estimate is (10), Sec. 19.5,

$$\epsilon_{h/2} = \tfrac{1}{15}(J_{h/2} - J_h),$$

obtained by first computing with step h, then with step $h/2$, and then taking $\frac{1}{15}$ of the difference of the results.

Simpson's rule is the most important of the **Newton–Cotes formulas**, which are obtained by integrating Lagrange interpolation polynomials, linear ones for the **trapezoidal rule** (2), Sec. 19.5, quadratic for Simpson's rule, cubic for the *three-eights rule* (see the Chap. 19 Review Problems), etc.

Adaptive integration (Sec. 19.5, Example 6) is integration that adjusts (*"adapts"*) the step (automatically) to the variability of $f(x)$.

Romberg integration (Team Project 26, Problem Set 19.5) starts from the trapezoidal rule (2), Sec. 19.5, with $h, h/2, h/4$, etc. and improves results by systematically adding error estimates.

Gauss integration (11), Sec. 19.5, is important because of its great accuracy (DP = $2n - 1$, compared to Newton–Cotes's DP = $n - 1$ or n). This is achieved by an optimal choice of the nodes, which are not equally spaced; see Table 19.7, Sec. 19.5.

Numeric differentiation is discussed at the end of Sec. 19.5. (Its main application (to differential equations) follows in Chap. 21.)