System of nonlinear equations

Let us now consider the n-dimensional version of the same problem, namely,

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

or, using scalar notation

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

Bracketing is very difficult
Newton's method is good, but
needs a good initial guess

Newton–Raphson Method

In order to derive the Newton–Raphson method for a system of equations, we start with the Taylor series expansion of $f_i(\mathbf{x})$ about the point $\mathbf{x}$:

$$f_i(\mathbf{x}+\Delta\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\Delta x_j + O(\Delta x^2) \tag{4.5a}$$

Dropping terms of order Δx^2, we can write Eq. (4.5a) as

$$\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\,\Delta\mathbf{x} \tag{4.5b}$$

where $\mathbf{J}(\mathbf{x})$ is the *Jacobian matrix* (of size $n \times n$) made up of the partial derivatives

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{4.6}$$

Let us now assume that $\mathbf{x}$ is the current approximation of the solution of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, and let $\mathbf{x} + \Delta\mathbf{x}$ be the improved solution. To find the correction $\Delta\mathbf{x}$, we set $\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{0}$ in Eq. (4.5b). The result is a set of linear equations for $\Delta\mathbf{x}$:

$$\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}) \tag{4.7}$$

The following steps constitute Newton–Raphson method for simultaneous, nonlinear equations:

1. Estimate the solution vector $\mathbf{x}$.
2. Evaluate $\mathbf{f}(\mathbf{x})$.
3. Compute the Jacobian matrix $\mathbf{J}(\mathbf{x})$ from Eq. (4.6).
4. Set up the simultaneous equations in Eq. (4.7) and solve for $\Delta\mathbf{x}$.
5. Let $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$ and repeat steps 2–5.

finding analytical expression for J either not possible or impractical

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\mathbf{x} + \mathbf{e}_j h) - f_i(\mathbf{x})}{h}$$

■ newtonRaphson2

This function is an implementation of the Newton–Raphson method. The nested function `jacobian` computes the Jacobian matrix from the finite difference approximation in Eq. (4.8). The simultaneous equations in Eq. (4.7) are solved by Gauss elimination with row pivoting using the function `gaussPivot` listed in Section 2.5. The function subroutine `func` that returns the array **f**(**x**) must be supplied by the user.

```
function root = newtonRaphson2(func,x,tol)
% Newton-Raphson method of finding a root of simultaneous
% equations fi(x1,x2,...,xn) = 0, i = 1,2,...,n.
% USAGE: root = newtonRaphson2(func,x,tol)
% INPUT:
% func = handle of function that returns[f1,f2,...,fn].
% x    = starting solution vector [x1,x2,...,xn].
% tol  = error tolerance (default is 1.0e4*eps).
% OUTPUT:
% root = solution vector.
```

```
if nargin == 2; tol = 1.0e4*eps; end
if size(x,1) == 1; x = x'; end   % x must be column vector
for i = 1:30
    [jac,f0] = jacobian(func,x);
    if sqrt(dot(f0,f0)/length(x)) < tol
        root = x; return
    end
    dx = jac\(-f0);
    x = x + dx;
    if sqrt(dot(dx,dx)/length(x)) < tol*max(abs(x),1.0)
        root = x; return
    end
end
error('Too many iterations')
```

```
function [jac,f0] = jacobian(func,x)
% Returns the Jacobian matrix and f(x).
h = 1.0e-4;
n = length(x);
jac = zeros(n);
f0 = feval(func,x);
for i =1:n
    temp = x(i);
    x(i) = temp + h;
    f1 = feval(func,x);
    x(i) = temp;
    jac(:,i) = (f1 - f0)/h;
end
```

Example-1

Determine the points of intersection between the circle $x^2 + y^2 = 3$ and the hyperbola $xy = 1$.

Solution The equations to be solved are

$$f_1(x, y) = x^2 + y^2 - 3 = 0 \tag{a}$$

$$f_2(x, y) = xy - 1 = 0 \tag{b}$$

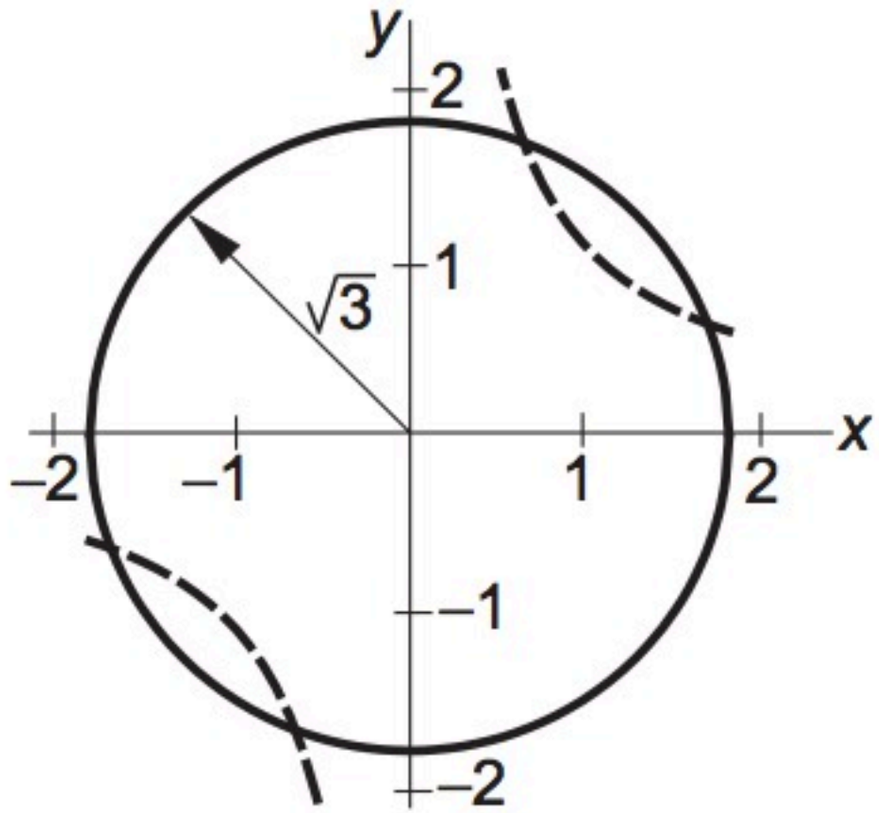
The Jacobian matrix is

$$\mathbf{J}(x, y) = \begin{bmatrix} \partial f_1/\partial x & \partial f_1/\partial y \\ \partial f_2/\partial x & \partial f_2/\partial y \end{bmatrix} = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}$$

Thus the linear equations $\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x})$ associated with the Newton–Raphson method are

$$\begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -x^2 - y^2 + 3 \\ -xy + 1 \end{bmatrix} \tag{c}$$

By plotting the circle and the hyperbola, we see that there are four points of intersection. It is sufficient, however, to find only one of these points, as the others can be deduced from symmetry. From the plot, we also get rough estimate of the coordinates of an intersection point: $x = 0.5$, $y = 1.5$, which we use as the starting values.



The computations then proceed as follows.

First iteration Substituting $x = 0.5$, $y = 1.5$ in Eq. (c), we get

$$\begin{bmatrix} 1.0 & 3.0 \\ 1.5 & 0.5 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} 0.50 \\ 0.25 \end{bmatrix}$$

the solution of which is $\Delta x = \Delta y = 0.125$. Therefore, the improved coordinates of the intersection point are

$$x = 0.5 + 0.125 = 0.625 \qquad y = 1.5 + 0.125 = 1.625$$

Second iteration Repeating the procedure using the latest values of x and y, we obtain

$$\begin{bmatrix} 1.250 & 3.250 \\ 1.625 & 0.625 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -0.031250 \\ -0.015625 \end{bmatrix}$$

which yields $\Delta x = \Delta y = -0.00694$. Thus

$$x = 0.625 - 0.006\,94 = 0.618\,06 \qquad y = 1.625 - 0.006\,94 = 1.618\,06$$

Third iteration Substitution of the latest x and y into Eq. (c) yields

$$\begin{bmatrix} 1.236\,12 & 3.23612 \\ 1.618\,06 & 0.61806 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -0.000\,116 \\ -0.000\,058 \end{bmatrix}$$

The solution is $\Delta x = \Delta y = -0.00003$, so that

$$x = 0.618\,06 - 0.000\,03 = 0.618\,03$$

$$y = 1.618\,06 - 0.000\,03 = 1.618\,03$$

Subsequent iterations would not change the results within five significant figures. Therefore, the coordinates of the four intersection points are

$$\pm(0.618\,03, 1.618\,03) \text{ and } \pm(1.618\,03, 0.618\,03)$$

Example 2

Find a solution of

$$\sin x + y^2 + \ln z - 7 = 0$$

$$3x + 2^y - z^3 + 1 = 0$$

$$x + y + z - 5 = 0$$

using `newtonRaphson2`. Start with the point (1, 1, 1).

Solution Letting $x = x_1$, $y = x_2$, and $z = x_3$, the M-file defining the function array **f(x)** is

```
function y = fex4_9(x)
% Function used in Example 4.9
y = [sin(x(1)) + x(2)^2 + log(x(3)) - 7; ...
       3*x(1) + 2^x(2) - x(3)^3 + 1;     ...
         x(1) + x(2) + x(3) - 5];
```

The solution can now be obtained with the single command

```
>> newtonRaphson2(@fex4_9,[1;1;1])
```

which results in

```
ans =
    0.5991
    2.3959
    2.0050
```

Hence the solution is $x = 0.5991$, $y = 2.3959$, and $z = 2.0050$.

Numerical Differentiation and Integration

Given the function $f(x)$, compute $d^n f/dx^n$ at given x

Numerical differentiation deals with the following problem: we are given the function $y = f(x)$ and wish to obtain one of its derivatives at the point $x = x_k$. The term "given" means that we either have an algorithm for computing the function, or possess a set of discrete data points (x_i, y_i), $i = 1, 2, \ldots, n$. In either case, we have access to a finite number of (x, y) data pairs from which to compute the derivative.

Numerical differentiation is not a particularly accurate process. It suffers from a conflict between roundoff errors (due to limited machine precision) and errors inherent in interpolation. For this reason, a derivative of a function can never be computed with the same precision as the function itself.

Using Taylor Series

The derivation of the finite difference approximations for the derivatives of $f(x)$ are based on forward and backward Taylor series expansions of $f(x)$ about x, such as

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) + \cdots \qquad \text{(a)}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) - \cdots \qquad \text{(b)}$$

$$f(x+2h) = f(x) + 2hf'(x) + \frac{(2h)^2}{2!}f''(x) + \frac{(2h)^3}{3!}f'''(x) + \frac{(2h)^4}{4!}f^{(4)}(x) + \cdots \qquad \text{(c)}$$

$$f(x-2h) = f(x) - 2hf'(x) + \frac{(2h)^2}{2!}f''(x) - \frac{(2h)^3}{3!}f'''(x) + \frac{(2h)^4}{4!}f^{(4)}(x) - \cdots \qquad \text{(d)}$$

We also record the sums and differences of the series:

$$f(x+h)+f(x-h)=2f(x)+h^2f''(x)+\frac{h^4}{12}f^{(4)}(x)+\cdots \tag{e}$$

$$f(x+h)-f(x-h)=2hf'(x)+\frac{h^3}{3}f'''(x)+\ldots \tag{f}$$

$$f(x+2h)+f(x-2h)=2f(x)+4h^2f''(x)+\frac{4h^4}{3}f^{(4)}(x)+\cdots \tag{g}$$

$$f(x+2h)-f(x-2h)=4hf'(x)+\frac{8h^3}{3}f'''(x)+\cdots \tag{h}$$

First Central Difference Approximations

Solution of Eq. (f) for $f'(x)$ is

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) - \cdots$$

Keeping only the first term on the right-hand side, we have

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) \tag{5.1}$$

which is called the *first central difference approximation* for $f'(x)$. The term $\mathcal{O}(h^2)$ reminds us that the truncation error behaves as h^2.

From Eq. (e) we obtain

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \frac{h^2}{12} f^{(4)}(x) + \cdots$$

or

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2) \tag{5.2}$$

Central difference approximations for other derivatives can be obtained from Eqs. (a)–(h) in a similar manner. For example, eliminating $f'(x)$ from Eqs. (f) and (h) and solving for $f'''(x)$ yields

$$f'''(x) = \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3} + \mathcal{O}(h^2) \qquad (5.3)$$

The approximation

$$f^{(4)}(x) = \frac{f(x+2h) - 4f(x+h) + 6f(x) - 4f(x-h) + f(x-2h)}{h^4} + \mathcal{O}(h^2) \qquad (5.4)$$

is available from Eqs. (e) and (g) after eliminating $f''(x)$. Table 5.1 summarizes the results.

	$f(x-2h)$	$f(x-h)$	$f(x)$	$f(x+h)$	$f(x+2h)$
$2hf'(x)$		-1	0	1	
$h^2 f''(x)$		1	-2	1	
$2h^3 f'''(x)$	-1	2	0	-2	1
$h^4 f^{(4)}(x)$	1	-4	6	-4	1

Table 5.1. Coefficients of central finite difference approximations of $\mathcal{O}(h^2)$

First Noncentral Finite Difference Approximations

Central finite difference approximations are not always usable. For example, consider the situation where the function is given at the n discrete points $x_1, x_2, \ldots, x_n$. Since central differences use values of the function on each side of x, we would be unable to compute the derivatives at x_1 and x_n. Clearly, there is a need for finite difference expressions that require evaluations of the function only on one side of x. These expressions are called *forward* and *backward* finite difference approximations.

Noncentral finite differences can also be obtained from Eqs. (a)–(h). Solving Eq. (a) for $f'(x)$ we get

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) - \frac{h^3}{4!}f^{(4)}(x) - \cdots$$

Keeping only the first term on the right-hand side leads to the *first forward difference approximation*

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \tag{5.5}$$

Similarly, Eq. (b) yields the *first backward difference approximation*

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h) \tag{5.6}$$

Note that the truncation error is now $\mathcal{O}(h)$, which is not as good as the $\mathcal{O}(h^2)$ error in central difference approximations.

We can derive the approximations for higher derivatives in the same manner. For example, Eqs. (a) and (c) yield

$$f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} + \mathcal{O}(h) \tag{5.7}$$

The third and fourth derivatives can be derived in a similar fashion. The results are shown in Tables 5.2a and 5.2b.

	$f(x)$	$f(x+h)$	$f(x+2h)$	$f(x+3h)$	$f(x+4h)$
$hf'(x)$	−1	1			
$h^2 f''(x)$	1	−2	1		
$h^3 f'''(x)$	−1	3	−3	1	
$h^4 f^{(4)}(x)$	1	−4	6	−4	1

Table 5.2a. Coefficients of forward finite difference approximations of $\mathcal{O}(h)$

	$f(x-4h)$	$f(x-3h)$	$f(x-2h)$	$f(x-h)$	$f(x)$
$hf'(x)$				−1	1
$h^2 f''(x)$			1	−2	1
$h^3 f'''(x)$		−1	3	−3	1
$h^4 f^{(4)}(x)$	1	−4	6	−4	1

Table 5.2b. Coefficients of backward finite difference approximations of $\mathcal{O}(h)$

Second Noncentral Finite Difference Approximations

Finite difference approximations of $\mathcal{O}(h)$ are not popular due to reasons that will be explained shortly. The common practice is to use expressions of $\mathcal{O}(h^2)$. To obtain noncentral difference formulas of this order, we have to retain more term in Taylor series. As an illustration, we will derive the expression for $f'(x)$. We start with Eqs. (a) and (c), which are

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) + \cdots$$

$$f(x+2h) = f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{4h^3}{3}f'''(x) + \frac{2h^4}{3}f^{(4)}(x) + \cdots$$

We eliminate $f''(x)$ by multiplying the first equation by 4 and subtracting it from the second equation. The result is

$$f(x+2h) - 4f(x+h) = -3f(x) - 2hf'(x) + \frac{h^4}{2}f^{(4)}(x) + \cdots$$

Therefore,

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

	$f(x)$	$f(x+h)$	$f(x+2h)$	$f(x+3h)$	$f(x+4h)$	$f(x+5h)$
$2hf'(x)$	-3	4	-1			
$h^2 f''(x)$	2	-5	4	-1		
$2h^3 f'''(x)$	-5	18	-24	14	-3	
$h^4 f^{(4)}(x)$	3	-14	26	-24	11	-2

Table 5.3a. Coefficients of forward finite difference approximations of $\mathcal{O}(h^2)$

	$f(x-5h)$	$f(x-4h)$	$f(x-3h)$	$f(x-2h)$	$f(x-h)$	$f(x)$
$2hf'(x)$				1	-4	3
$h^2 f''(x)$			-1	4	-5	2
$2h^3 f'''(x)$		3	-14	24	-18	5
$h^4 f^{(4)}(x)$	-2	11	-24	26	-14	3

Table 5.3b. Coefficients of backward finite difference approximations of $\mathcal{O}(h^2)$

Errors in Finite Difference Approximations

Observe that in all finite difference expressions the sum of the coefficients is zero. The effect on the *roundoff error* can be profound. If h is very small, the values of $f(x)$, $f(x \pm h)$, $f(x \pm 2h)$, and so forth, will be approximately equal. When they are multiplied by the coefficients in the finite difference formulas and added, several significant figures can be lost. On the other hand, we cannot make h too large, because then the *truncation error* would become excessive. This unfortunate situation has no remedy, but we can obtain some relief by taking the following precautions:

- Use double-precision arithmetic.
- Employ finite difference formulas that are accurate to at least $\mathcal{O}(h^2)$.

To illustrate the errors, let us compute the second derivative of $f(x) = e^{-x}$ at $x = 1$ from the central difference formula, Eq. (5.2). We carry out the calculations with six- and eight-digit precision, using different values of h. The results, shown in Table 5.4, should be compared with $f''(1) = e^{-1} = 0.367\,879\,44$.

h	6-digit precision	8-digit precision
0.64	0.380 610	0.380 609 11
0.32	0.371 035	0.371 029 39
0.16	0.368 711	0.368 664 84
0.08	0.368 281	0.368 076 56
0.04	0.368 75	0.367 831 25
0.02	0.37	0.3679
0.01	0.38	0.3679
0.005	0.40	0.3676
0.0025	0.48	0.3680
0.00125	1.28	0.3712

Table 5.4. $(e^{-x})''$ at $x = 1$ from central finite difference approximation

Initial error due to truncation
Later error due to round-off

Richardson Extrapolation

Richardson extrapolation is a simple method for boosting the accuracy of certain numerical procedures, including finite difference approximations (we will also use it later in numerical integration).

Suppose that we have an approximate means of computing some quantity G. Moreover, assume that the result depends on a parameter h. Denoting the approximation by $g(h)$, we have $G = g(h) + E(h)$, where $E(h)$ represents the error. Richardson extrapolation can remove the error, provided that it has the form $E(h) = ch^p$, c and p being constants. We start by computing $g(h)$ with some value of h, say, $h = h_1$. In that case we have

$$G = g(h_1) + ch_1^p \tag{i}$$

Then we repeat the calculation with $h = h_2$, so that

$$G = g(h_2) + ch_2^p \tag{j}$$

Eliminating c and solving for G, Eqs. (i) and (j) yield

$$G = \frac{(h_1/h_2)^p g(h_2) - g(h_1)}{(h_1/h_2)^p - 1} \tag{5.8}$$

which is the *Richardson extrapolation formula.* It is common practice to use $h_2 = h_1/2$, in which case Eq. (5.8) becomes

$$G = \frac{2^p g(h_1/2) - g(h_1)}{2^p - 1} \tag{5.9}$$

Let us illustrate Richardson extrapolation by applying it to the finite difference approximation of $(e^{-x})''$ at $x = 1$. We work with six-digit precision and utilize the results in Table 5.4. Since the extrapolation works only on the truncation error, we must confine h to values that produce negligible roundoff. Choosing $h_1 = 0.64$ and letting $g(h)$ be the approximation of $f''(1)$ obtained with h, we get from Table 5.4

$$g(h_1) = 0.380\,610 \qquad g(h_1/2) = 0.371\,035$$

The truncation error in central difference approximation is $E(h) = \mathcal{O}(h^2) = c_1 h^2 + c_2 h^4 + c_3 h^6 + \cdots$. Therefore, we can eliminate the first (dominant) error term if we substitute $p = 2$ and $h_1 = 0.64$ in Eq. (5.9). The result is

$$G = \frac{2^2 g(0.32) - g(0.64)}{2^2 - 1} = \frac{4(0.371\,035) - 0.380\,610}{3} = 0.367\,843$$

which is an approximation of $(e^{-x})''$ with the error $\mathcal{O}(h^4)$. Note that it is as accurate as the best result obtained with eight-digit computations in Table 5.4.

Example-1

Given the evenly spaced data points

x	0	0.1	0.2	0.3	0.4
$f(x)$	0.0000	0.0819	0.1341	0.1646	0.1797

compute $f'(x)$ and $f''(x)$ at $x = 0$ and 0.2 using finite difference approximations of $\mathcal{O}(h^2)$.

Solution From the forward difference formulas in Table 5.3a, we get

$$f'(0) = \frac{-3f(0) + 4f(0.1) - f(0.2)}{2(0.1)} = \frac{-3(0) + 4(0.0819) - 0.1341}{0.2} = 0.967$$

$$f''(0) = \frac{2f(0) - 5f(0.1) + 4f(0.2) - f(0.3)}{(0.1)^2}$$

$$= \frac{2(0) - 5(0.0819) + 4(0.1341) - 0.1646}{(0.1)^2} = -3.77$$

The central difference approximations in Table 5.1 yield

$$f'(0.2) = \frac{-f(0.1) + f(0.3)}{2(0.1)} = \frac{-0.0819 + 0.1646}{0.2} = 0.4135$$

$$f''(0.2) = \frac{f(0.1) - 2f(0.2) + f(0.3)}{(0.1)^2} = \frac{0.0819 - 2(0.1341) + 0.1646}{(0.1)^2} = -2.17$$

Example-2

Use the data in Example 5.1 to compute $f'(0)$ as accurately as you can.

Solution One solution is to apply Richardson extrapolation to finite difference approximations. We start with two forward difference approximations for $f'(0)$: one using $h = 0.2$ and the other one $h = 0.1$. Referring to the formulas of $O(h^2)$ in Table 5.3a, we get

$$g(0.2) = \frac{-3f(0) + 4f(0.2) - f(0.4)}{2(0.2)} = \frac{3(0) + 4(0.1341) - 0.1797}{0.4} = 0.8918$$

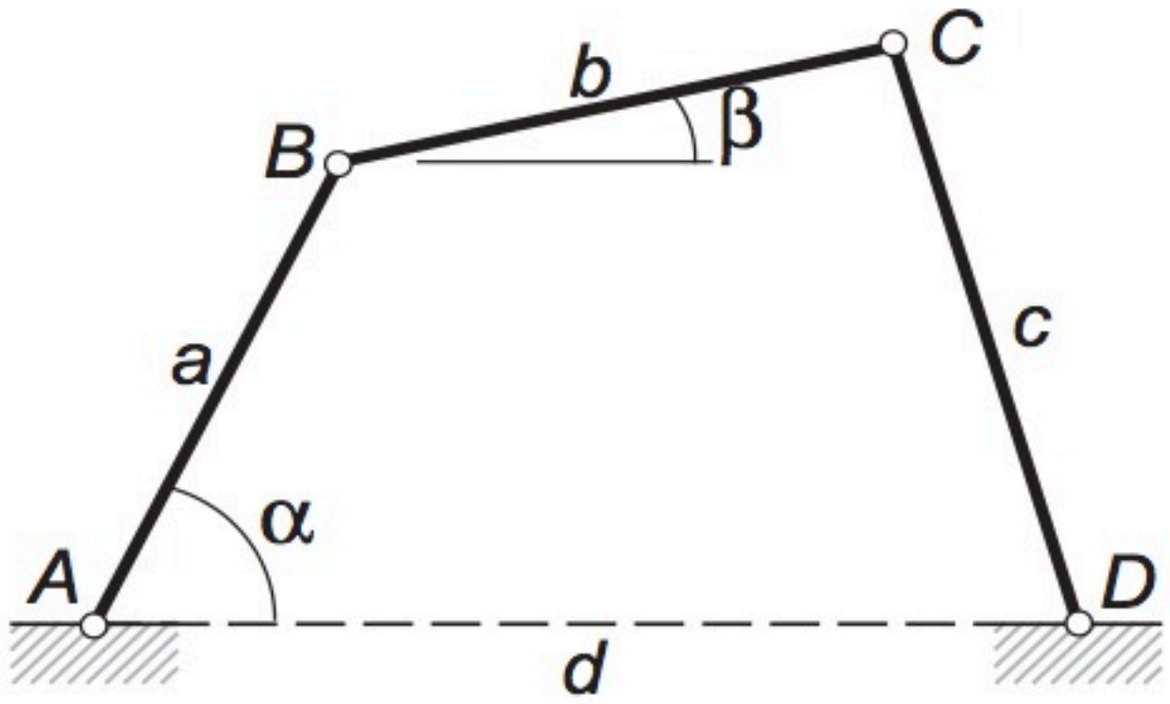
$$g(0.1) = \frac{-3f(0) + 4f(0.1) - f(0.2)}{2(0.1)} = \frac{-3(0) + 4(0.0819) - 0.1341}{0.2} = 0.9675$$

where g denotes the finite difference approximation of $f'(0)$. Recalling that the error in both approximations is of the form $E(h) = c_1h^2 + c_2h^4 + c_3h^6 + \cdots$, we can use Richardson extrapolation to eliminate the dominant error term. With $p = 2$ we obtain from Eq. (5.9)

$$f'(0) \approx G = \frac{2^2 g(0.1) - g(0.2)}{2^2 - 1} = \frac{4(0.9675) - 0.8918}{3} = 0.9927$$

which is a finite difference approximation of $O(h^4)$.

Example-3



The linkage shown has the dimensions $a = 100$ mm, $b = 120$ mm, $c = 150$ mm, and $d = 180$ mm. It can be shown by geometry that the relationship between the angles α and β is

$$(d - a\cos\alpha - b\cos\beta)^2 + (a\sin\alpha + b\sin\beta)^2 - c^2 = 0$$

For a given value of α, we can solve this transcendental equation for β by one of the root-finding methods in Chapter 4. This was done with $\alpha = 0°, 5°, 10°, \ldots, 30°$, the results being

α (deg)	0	5	10	15	20	25	30
β (rad)	1.6595	1.5434	1.4186	1.2925	1.1712	1.0585	0.9561

If link AB rotates with the constant angular velocity of 25 rad/s, use finite difference approximations of $\mathcal{O}(h^2)$ to tabulate the angular velocity $d\beta/dt$ of link BC against α.

Solution The angular speed of BC is

$$\frac{d\beta}{dt} = \frac{d\beta}{d\alpha}\frac{d\alpha}{dt} = 25\frac{d\beta}{d\alpha}\ \text{rad/s}$$

where $d\beta/d\alpha$ is computed from finite difference approximations using the data in the table. Forward and backward differences of $\mathcal{O}(h^2)$ are used at the endpoints, central differences elsewhere. Note that the increment of α is

$$h = (5\,\text{deg})\left(\frac{\pi}{180}\text{rad / deg}\right) = 0.087266\,\text{rad}$$

The computations yield

$$\dot{\beta}(0^\circ) = 25\frac{-3\beta(0^\circ) + 4\beta(5^\circ) - \beta(10^\circ)}{2h} = 25\frac{-3(1.6595) + 4(1.5434) - 1.4186}{2\,(0.087266)}$$

$$= -32.01\,\text{rad/s}$$

$$\dot{\beta}(5^\circ) = 25\frac{\beta(10^\circ) - \beta(0^\circ)}{2h} = 25\frac{1.4186 - 1.6595}{2(0.087266)} = -34.51\,\text{rad/s}$$

and so forth.

The complete set of results is

α (deg)	0	5	10	15	20	25	30
$\dot{\beta}$ (rad/s)	−32.01	−34.51	−35.94	−35.44	−33.52	−30.81	−27.86

Derivatives by Interpolation

If $f(x)$ is given as a set of discrete data points, interpolation can be a very effective means of computing its derivatives. The idea is to approximate the derivative of $f(x)$ by the derivative of the interpolant. This method is particularly useful if the data points are located at uneven intervals of x, when the finite difference approximations listed in the last section are not applicable.[10]

Polynomial Interpolant

The idea here is simple: fit the polynomial of degree $n-1$

$$P_{n-1}(x) = a_1 x^n + a_2 x^{n-1} + \cdots a_{n-1} x + a_n \qquad \text{(a)}$$

through n data points and then evaluate its derivatives at the given x. As pointed out in Section 3.2, it is generally advisable to limit the degree of the polynomial to less than six in order to avoid spurious oscillations of the interpolant. Since these oscillations are magnified with each differentiation, their effect can be devastating. In view of the above limitation, the interpolation should usually be a local one, involving no more than a few nearest-neighbor data points.

Cubic Spline Interpolant

Due to its stiffness, cubic spline is a good global interpolant; moreover, it is easy to differentiate. The first step is to determine the second derivatives k_i of the spline at the knots by solving Eqs. (3.12). This can be done with the function `splineCurv` as explained in Section 3.3. The first and second derivatives are then computed from

$$f'_{i,i+1}(x) = \frac{k_i}{6}\left[\frac{3(x-x_{i+1})^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] - \frac{k_{i+1}}{6}\left[\frac{3(x-x_i)^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] + \frac{y_i - y_{i+1}}{x_i - x_{i+1}} \tag{5.10}$$

$$f''_{i,i+1}(x) = k_i \frac{x - x_{i+1}}{x_i - x_{i+1}} - k_{i+1}\frac{x - x_i}{x_i - x_{i+1}} \tag{5.11}$$

which are obtained by differentiation of Eq. (3.10).

Example

Given the data

x	1.5	1.9	2.1	2.4	2.6	3.1
$f(x)$	1.0628	1.3961	1.5432	1.7349	1.8423	2.0397

compute $f'(2)$ and $f''(2)$ using (1) polynomial interpolation over three nearest-neighbor points, and (2) natural cubic spline interpolant spanning all the data points.

Solution of Part (1) Let the interpolant passing through the points at $x = 1.9$, 2.1, and 2.4 be $P_2(x) = a_1 + a_2 x + a_3 x^2$. The normal equations, Eqs. (3.23), of the least-squares fit are

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \end{bmatrix}$$

After substituting the data, we get

$$\begin{bmatrix} 3 & 6.4 & 13.78 \\ 6.4 & 13.78 & 29.944 \\ 13.78 & 29.944 & 65.6578 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 4.6742 \\ 10.0571 \\ 21.8385 \end{bmatrix}$$

which yields $\mathbf{a} = \begin{bmatrix} -0.7714 & 1.5075 & -0.1930 \end{bmatrix}^T$. Thus the interpolant and its derivatives are

$$P_2(x) = -0.1903x^2 + 1.5075x - 0.7714$$

$$P_2'(x) = -0.3860x + 15075$$

$$P_2''(x) = -0.3860$$

which gives us

$$f'(2) \approx P_2'(2) = -0.3860(2) + 1.50752 = 0.7355$$

$$f''(2) \approx P_2''(2) = -0.3860$$

Solution of Part (2) We must first determine the second derivatives k_i of the spline at its knots, after which the derivatives of $f(x)$ can be computed from Eqs. (5.10) and (5.11). The first part can be carried out by the following small program:

```
% Example 5.4 (Curvatures of cubic spline at the knots)
xData = [1.5; 1.9; 2.1; 2.4; 2.6; 3.1];
yData = [1.0628; 1.3961; 1.5432; 1.7349; 1.8423; 2.0397];
k = splineCurv(xData,yData)
```

The output of the program, consisting of k_1 to k_6, is

```
>> k =
         0
   -0.4258
   -0.3774
   -0.3880
   -0.5540
         0
```

Numerical Integration or Quadrature

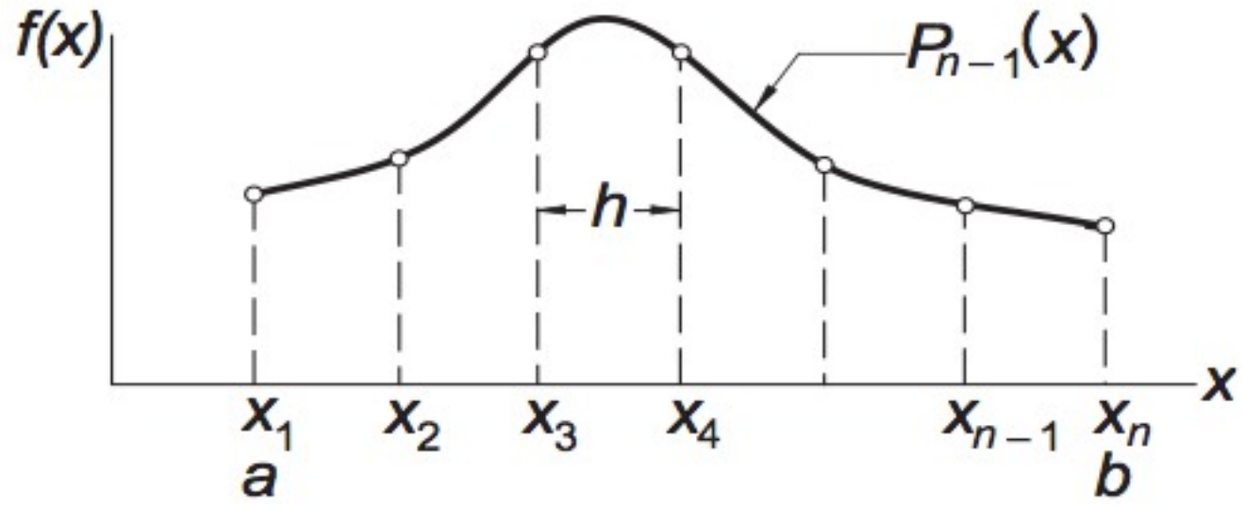
Compute $\int_a^b f(x)\,dx$, where $f(x)$ is a given function

Replace $\int_a^b f(x)\,dx$ with $I = \sum_{i=1}^{n} A_i f(x_i)$

Newton-Cotes and Gauss Quadrature

$$\int_0^1 \frac{g(x)}{\sqrt{1-x^2}}dx$$

Newton–Cotes Formulas



Consider the definite integral

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

We divide the range of integration (a,b) into n-1 equal intervals of (b-a)/(n-1)

The resultant points are $x_1, x_2, \ldots, x_n$

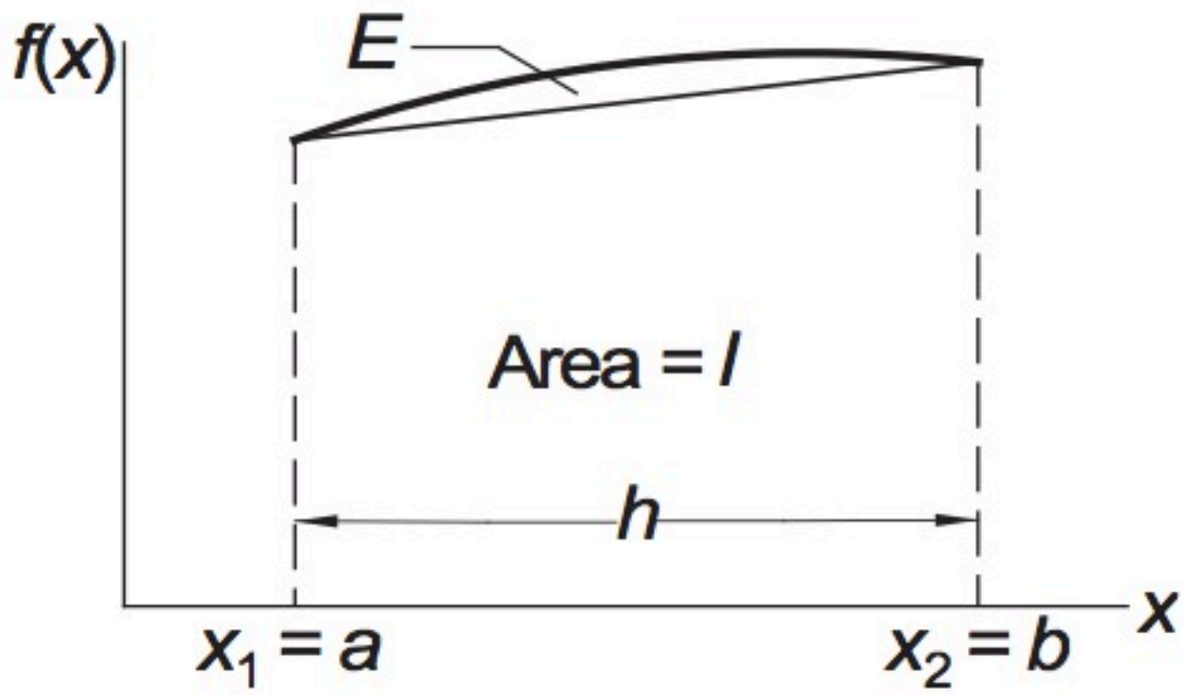
We fit a polynomial of degree n-1 through these points

$$P_{n-1}(x) = \sum_{i=1}^{n} f(x_i)\ell_i(x) \qquad I = \int_a^b P_{n-1}(x)dx = \sum_{i=1}^{n}\left[f(x_i)\int_a^b \ell_i(x)dx\right] = \sum_{i=1}^{n} A_i f(x_i)$$

$$A_i = \int_a^b \ell_i(x)dx, \quad i = 1, 2, \ldots, n$$

: *trapezoidal rule* ($n = 2$), *Simpson's rule* ($n = 3$), and *3/8 Simpson's rule* ($n = 4$)

Trapezoidal Rule



If $n = 2$, we have $\ell_1 = (x - x_2)/(x_1 - x_2) = -(x - b)/h$. Therefore,

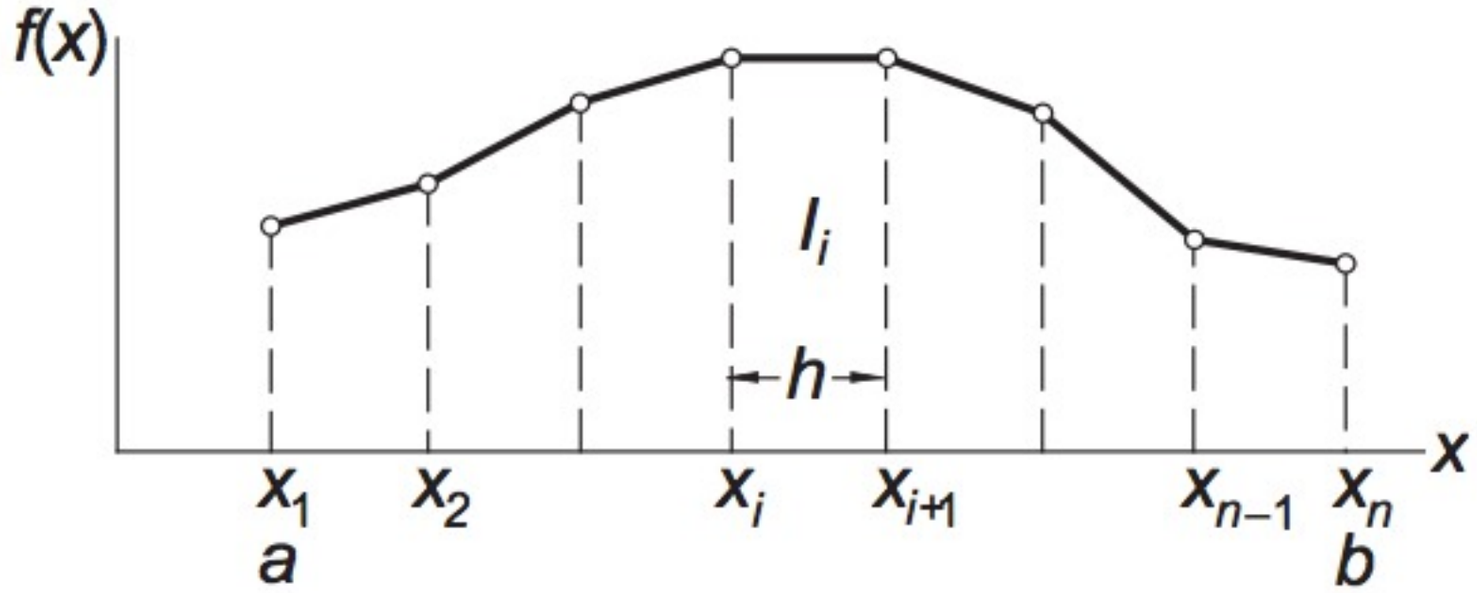
$$A_1 = -\frac{1}{h}\int_a^b (x - b)\,dx = \frac{1}{2h}(b - a)^2 = \frac{h}{2}$$

Also, $\ell_2 = (x - x_1)/(x_2 - x_1) = (x - a)/h$, so that

$$A_2 = \frac{1}{h}\int_a^b (x - a)\,dx = \frac{1}{2h}(b - a)^2 = \frac{h}{2}$$

$$I = \left[f(a) + f(b)\right]\frac{h}{2}$$

Composite Trapezoidal Rule



the region (a, b) divided into $n-1$ panels, each of width h.
we obtain for the approximate area of a typical (ith) panel

$$I_i = \left[f(x_i) + f(x_{i+1})\right] \frac{h}{2}$$

Hence total area, representing $\int_a^b f(x)\,dx$, is

$$I = \sum_{i=1}^{n-1} I_i = \left[f(x_1) + 2f(x_2) + 2f(x_3) + \cdots + 2f(x_{n-1}) + f(x_n)\right] \frac{h}{2}$$

which is the *composite trapezoidal rule.*

$$E = -\frac{(b-a)h^2}{12} f''(\xi)$$

Recursive Trapezoidal Rule

Let I_k be the integral evaluated with the composite trapezoidal rule using 2^{k-1} panels. Note that if k is increased by one, the number of panels is doubled. Using the notation

$$H = b - a$$

$k = 1, 2,$ and 3.

$k = 1$ (1 panel):

$$I_1 = \left[f(a) + f(b)\right] \frac{H}{2}$$

$k = 2$ (2 panels):

$$I_2 = \left[f(a) + 2f\left(a + \frac{H}{2}\right) + f(b)\right] \frac{H}{4} = \frac{1}{2} I_1 + f\left(a + \frac{H}{2}\right) \frac{H}{2}$$

$k = 3$ (4 panels):

$$I_3 = \left[f(a) + 2f\left(a + \frac{H}{4}\right) + 2f\left(a + \frac{H}{2}\right) + 2f\left(a + \frac{3H}{4}\right) + f(b)\right] \frac{H}{8}$$

$$= \frac{1}{2} I_2 + \left[f\left(a + \frac{H}{4}\right) + f\left(a + \frac{3H}{4}\right)\right] \frac{H}{4}$$

We can now see that for arbitrary $k > 1$ we have

$$I_k = \frac{1}{2} I_{k-1} + \frac{H}{2^{k-1}} \sum_{i=1}^{2^{k-2}} f\left[a + \frac{(2i-1)H}{2^{k-1}}\right], \quad k = 2, 3, \ldots$$

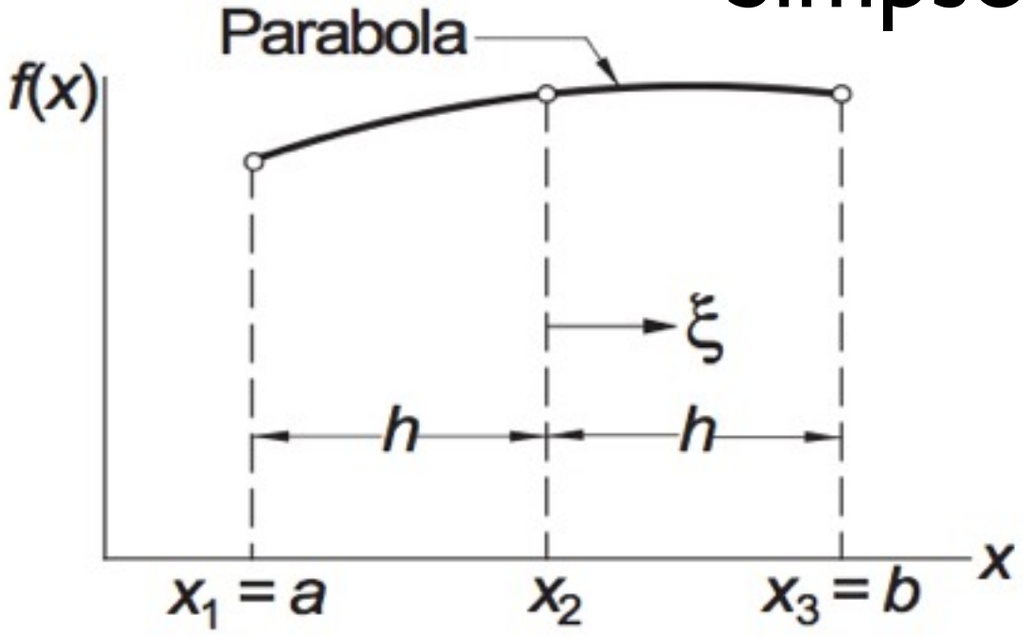
■ trapezoid

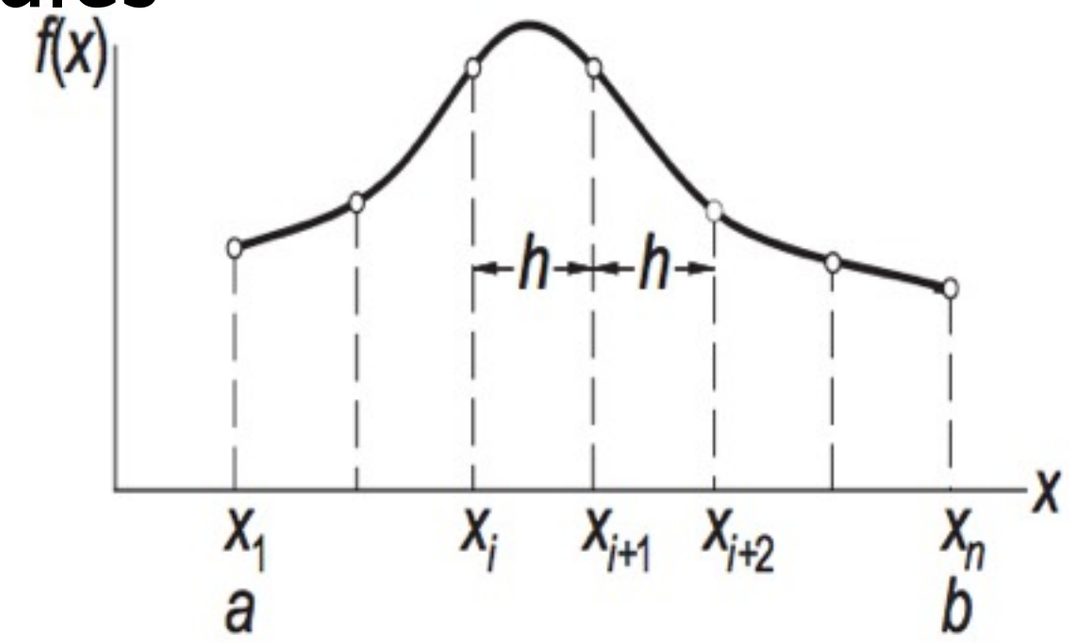
The function `trapezoid` computes $I(h)$, given $I(2h)$ using Eqs. (6.8) and (6.9). We can compute $\int_a^b f(x)\,dx$ by calling `trapezoid` repeatedly with $k = 1, 2, \ldots,$ until the desired precision is attained.

```
function Ih = trapezoid(func,a,b,I2h,k)
% Recursive trapezoidal rule.
% USAGE: Ih = trapezoid(func,a,b,I2h,k)
% func = handle of function being integrated.
% a,b  = limits of integration.
% I2h  = integral with 2^(k-1) panels.
% Ih = integral with 2^k panels.

if k == 1
    fa = feval(func,a); fb = feval(func,b);
    Ih = (fa + fb)*(b - a)/2.0;
else
    n = 2^(k -2 );       % Number of new points
    h = (b - a)/n ;      % Spacing of new points
    x = a + h/2.0;       % Coord. of 1st new point
    sum = 0.0;
    for i = 1:n
        fx = feval(func,x);
        sum = sum + fx;
        x = x + h;
    end
    Ih = (I2h + h*sum)/2.0;
end
```

Simpson's Rules





Simpson's 1/3 rule can be obtained from Newton–Cotes formulas with $n = 3$; that is, by passing a parabolic interpolant through three adjacent nodes, as shown in Fig. 6.4. The area under the parabola, which represents an approximation of $\int_a^b f(x)\,dx$, is

$$I = \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]\frac{h}{3} \qquad \text{(a)}$$

Applying Eq. (a) to two adjacent panels, we have

$$\int_{x_i}^{x_{i+2}} f(x)\,dx \approx \left[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\right]\frac{h}{3} \qquad \text{(b)}$$

$$\int_a^b f(x)dx = \sum_{i=1,3,\ldots}^{n-2}\left[\int_{x_i}^{x_{i+2}} f(x)dx\right]$$

$$\int_a^b f(x)\,dx \approx I = [f(x_1) + 4f(x_2) + 2f(x_3) + 4f(x_4) + \cdots$$

$$\cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]\frac{h}{3}$$

$$E = \frac{(b-a)h^4}{180}f^{(4)}(\xi)$$

Simpson's 1/3 rule requires the number of panels to be even. If this condition is not satisfied, we can integrate over the first (or last) three panels with *Simpson's 3/8 rule*:

$$I = \left[f(x_1) + 3f(x_2) + 3f(x_3) + f(x_4)\right] \frac{3h}{8}$$

Derive Simpson's 1/3 rule from Newton–Cotes formulas.

Solution Referring to Fig. 6.4, Simpson's 1/3 rule uses three nodes located at $x_1 = a$, $x_2 = (a+b)/2$, and $x_3 = b$. The spacing of the nodes is $h = (b-a)/2$. The cardinal functions of Lagrange's three-point interpolation are — see Section 4.2

$$\ell_1(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} \qquad \ell_2(x) = \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}$$

$$\ell_3(x) = \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}$$

The integration of these functions is easier if we introduce the variable ξ with origin at x_1. Then the coordinates of the nodes are $\xi_1 = -h$, $\xi_2 = 0$, $\xi_3 = h$, and Eq. (6.2b) becomes $A_i = \int_a^b \ell_i(x) = \int_{-h}^{h} \ell_i(\xi)\,d\xi$. Therefore,

$$A_1 = \int_{-h}^{h} \frac{(\xi-0)(\xi-h)}{(-h)(-2h)}\,d\xi = \frac{1}{2h^2}\int_{-h}^{h}(\xi^2 - h\xi)\,d\xi = \frac{h}{3}$$

$$A_2 = \int_{-h}^{h} \frac{(\xi+h)(\xi-h)}{(h)(-h)}\,d\xi = -\frac{1}{h^2}\int_{-h}^{h}(\xi^2 - h^2)\,d\xi = \frac{4h}{3}$$

$$A_3 = \int_{-h}^{h} \frac{(\xi+h)(\xi-0)}{(2h)(h)}\,d\xi = \frac{1}{2h^2}\int_{-h}^{h}(\xi^2 + h\xi)\,d\xi = \frac{h}{3}$$

$$I = \sum_{i=1}^{3} A_i f(x_i) = \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]\frac{h}{3}$$

Evaluate the bounds on $\int_0^\pi \sin(x)\,dx$ with the composite trapezoidal rule using (1) eight panels and (2) sixteen panels.

Solution of Part (1) With eight panels there are nine nodes spaced at $h = \pi/8$. The abscissas of the nodes are $x_i = (i-1)\pi/8, \quad i = 1, 2, \ldots, 9$. From Eq. (6.5) we get

$$I = \left[\sin 0 + 2\sum_{i=2}^{8} \sin\frac{i\pi}{8} + \sin\pi\right]\frac{\pi}{16} = 1.97423$$

The error is given by Eq. (6.6):

$$E = -\frac{(b-a)h^2}{12}f''(\xi) = -\frac{(\pi - 0)(\pi/8)^2}{12}(-\sin\xi) = \frac{\pi^3}{768}\sin\xi$$

where $0 < \xi < \pi$. Since we do not know the value of ξ, we cannot evaluate E, but we can determine its bounds:

$$E_{\min} = \frac{\pi^3}{768}\sin(0) = 0 \qquad E_{\max} = \frac{\pi^3}{768}\sin\frac{\pi}{2} = 0.040\,37$$

Therefore, $I + E_{\min} < \int_0^\pi \sin(x)\,dx < I + E_{\max}$, or

$$1.974\,23 < \int_0^\pi \sin(x)\,dx < 2.014\,60$$

The exact integral is, of course, $I = 2$.

Solution of Part (2) The new nodes created by doubling of panels are located at midpoints of the old panels. Their abscissas are

$$x_j = \pi/16 + (j-1)\pi/8 = (2j-1)\pi/16, \quad j = 1, 2, \ldots, 8$$

Using the recursive trapezoidal rule in Eq. (6.9b), we get

$$I = \frac{1.974\,23}{2} + \frac{\pi}{16}\sum_{j=1}^{8} \sin\frac{(2j-1)\pi}{16} = 1.993\,58$$

and the bounds on the error become (note that E is quartered when h is halved) $E_{\min} = 0$, $E_{\max} = 0.040\,37/4 = 0.010\,09$. Hence

$$1.993\,58 < \int_0^{\pi} \sin(x)\,dx < 2.003\,67$$

Estimate $\int_0^{2.5} f(x)\,dx$ from the data

x	0	0.5	1.0	1.5	2.0	2.5
$f(x)$	1.5000	2.0000	2.0000	1.6364	1.2500	0.9565

Solution We will use Simpson's rules, since they are more accurate than the trapezoidal rule. Because the number of panels is odd, we compute the integral over the first three panels by Simpson's 3/8 rule, and use the 1/3 rule for the last two panels:

$$I = \left[f(0) + 3f(0.5) + 3f(1.0) + f(1.5)\right]\frac{3(0.5)}{8}$$

$$+ \left[f(1.5) + 4f(2.0) + f(2.5)\right]\frac{0.5}{3}$$

$$= 2.8381 + 1.2655 = 4.1036$$

Use the recursive trapezoidal rule to evaluate $\int_0^{\pi} \sqrt{x}\cos x\,dx$ to six decimal places. How many function evaluations are required to achieve this result?

Solution The program listed below utilizes the function `trapezoid`. Apart from the value of the integral, it displays the number of function evaluations used in the computation.

```
% Example 6.4 (Recursive trapezoidal rule)
format long   % Display extra precision
func = @(x) (sqrt(x)*cos(x));
I2h = 0;
for k = 1:20
    Ih = trapezoid(func,0,pi,I2h,k);
    if (k > 1 && abs(Ih - I2h) < 1.0e-6)
        Integral = Ih
        No_of_func_evaluations = 2^(k-1) + 1
        return
    end
    I2h = Ih;
end
error('Too many iterations')
```

```
>> Integral =
   -0.89483166485329
No_of_func_evaluations =
        32769
```

Here is the output:

```
>> Integral =
   -0.89483166485329
No_of_func_evaluations =
        32769
```

Rounding to six decimal places, we have $\int_0^\pi \sqrt{x}\cos x\,dx = -0.894\,832$

The number of function evaluations is unusually large in this problem. The slow convergence is the result of the derivatives of $f(x)$ being singular at $x = 0$. Consequently, the error does not behave as shown in Eq. (6.7): $E = c_1 h^2 + c_2 h^4 + \cdots$, but is unpredictable. Difficulties of this nature can often be remedied by a change in variable. In this case, we introduce $t = \sqrt{x}$, so that $dt = dx/(2\sqrt{x}) = dx/(2t)$, or $dx = 2t\,dt$. Thus

$$\int_0^\pi \sqrt{x}\cos x\,dx = \int_0^{\sqrt{\pi}} 2t^2\cos t^2\,dt$$

Evaluation of the integral on the right-hand side would require 4097 function evaluations.

Romberg Integration

Romberg integration combines the composite trapezoidal rule with Richardson extrapolation (see Section 5.3). Let us first introduce the notation

$$R_{i,1} = I_i$$

where, as before, I_i represents the approximate value of $\int_a^b f(x)dx$ computed by the recursive trapezoidal rule using 2^{i-1} panels. Recall that the error in this approximation is $E = c_1 h^2 + c_2 h^4 + \cdots$, where

$$h = \frac{b-a}{2^{i-1}}$$

is the width of a panel.

Romberg integration starts with the computation of $R_{1,1} = I_1$ (one panel) and $R_{2,1} = I_2$ (two panels) from the trapezoidal rule. The leading error term $c_1 h^2$ is then eliminated by Richardson extrapolation. Using $p = 2$ (the exponent in the error term) in Eq. (5.9) and denoting the result by $R_{2,2}$, we obtain

$$R_{2,2} = \frac{2^2 R_{2,1} - R_{1,1}}{2^{2-1}} = \frac{4}{3} R_{2,1} - \frac{1}{3} R_{1,1} \qquad \text{(a)}$$

It is convenient to store the results in an array of the form

$$\begin{bmatrix} R_{1,1} & \\ R_{2,1} & R_{2,2} \end{bmatrix}$$

The next step is to calculate $R_{3,1} = I_3$ (four panels) and repeat Richardson extrapolation with $R_{2,1}$ and $R_{3,1}$, storing the result as $R_{3,2}$:

$$R_{3,2} = \frac{4}{3} R_{3,1} - \frac{1}{3} R_{2,1} \qquad \text{(b)}$$

The elements of array **R** calculated so far are

$$\begin{bmatrix} R_{1,1} & \\ R_{2,1} & R_{2,2} \\ R_{3,1} & R_{3,2} \end{bmatrix}$$

Both elements of the second column have an error of the form $c_2 h^4$, which can also be eliminated with Richardson extrapolation. Using $p = 4$ in Eq. (5.9), we get

$$R_{3,3} = \frac{2^4 R_{3,2} - R_{2,2}}{2^{4-1}} = \frac{16}{15} R_{3,2} - \frac{1}{15} R_{2,2} \qquad \text{(c)}$$

This result has an error of $\mathcal{O}(h^6)$. The array has now expanded to

$$\begin{bmatrix} R_{1,1} & & \\ R_{2,1} & R_{2,2} & \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix}$$

After another round of calculations we get

$$\begin{bmatrix} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ R_{3,1} & R_{3.2} & R_{3,3} & \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} \end{bmatrix}$$

where the error in $R_{4,4}$ is $\mathcal{O}(h^8)$. Note that the most accurate estimate of the integral is always the last diagonal term of the array. This process is continued until the difference between two successive diagonal terms becomes sufficiently small. The general extrapolation formula used in this scheme is

$$R_{i,,j} = \frac{4^{j-1} R_{i,j-1} - R_{i-1,j-1}}{4^{j-1} - 1}, \quad i > 1, \quad j = 2, 3, \ldots, i \qquad \text{(6.13a)}$$

A pictorial representation of Eq. (6.13a) is

$$\begin{array}{ccccc} \boxed{R_{i-1,j-1}} & & \searrow & & \\ & & & \alpha & \\ & & & & \searrow \\ \boxed{R_{i,j-1}} & \rightarrow & \beta & \rightarrow & \boxed{R_{i,j}} \end{array} \tag{6.13b}$$

where the multipliers α and β depend on j in the following manner

j	2	3	4	5	6
α	$-1/3$	$-1/15$	$-1/63$	$-1/255$	$-1/1023$
β	$4/3$	$16/15$	$64/63$	$256/255$	$1024/1023$

(6.13c)

The triangular array is convenient for hand computations, but computer implementation of the Romberg algorithm can be carried out within a one-dimensional array **r**. After the first extrapolation – see Eq. (a) – $R_{1,1}$ is never used again, so that it can be replaced with $R_{2,2}$. As a result, we have the array

$$\begin{bmatrix} r_1 = R_{2,2} \\ r_2 = R_{2,1} \end{bmatrix}$$

In the second extrapolation round, defined by Eqs. (b) and (c), $R_{3,2}$ overwrites $R_{2,1}$, and $R_{3,3}$ replaces $R_{2,2}$, so that the array now contains

$$\begin{bmatrix} r_1 = R_{3,3} \\ r_2 = R_{3,2} \\ r_3 = R_{3,1} \end{bmatrix}$$

and so on. In this manner, r_1 always contains the best current result. The extrapolation formula for the kth round is

$$r_j = \frac{4^{k-j} r_{j+1} - r_j}{4^{k-j} - 1}, \quad j = k-1, k-2, \ldots, 1 \tag{6.14}$$

■ romberg

The algorithm for Romberg integration is implemented in the function `romberg`. It returns the value of the integral and the required number of function evaluations. Richardson's extrapolation is performed by the subfunction `richardson`.

```
function [I,numEval] = romberg(func,a,b,tol,kMax)
% Romberg integration.
% USAGE: [I,numEval] = romberg(func,a,b,tol,kMax)
% INPUT:
% func    = handle of function being integrated.
% a,b     = limits of integration.
% tol     = error tolerance (default is 1.0e4*eps).
% kMax    = limit on the number of panel doublings
%           (default is 20).
% OUTPUT:
% I       = value of the integral.
% numEval = number of function evaluations.
if nargin < 5; kMax = 20; end
if nargin < 4; tol = 1.0e4*eps; end
r = zeros(kMax);
r(1) = trapezoid(func,a,b,0,1);
rOld = r(1);
for k = 2:kMax
    r(k) = trapezoid(func,a,b,r(k-1),k);
    r = richardson(r,k);
    if abs(r(1) - rOld) < tol
        numEval = 2^(k-1) + 1; I = r(1);
        return
    end
    rOld = r(1);
end
error('Failed to converge')

function r = richardson(r,k)
% Richardson's extrapolation in Eq. (6.14).
for j = k-1:-1:1
    c = 4^(k-j); r(j) = (c*r(j+1) - r(j))/(c-1);
end
```

Example

Show that $R_{k,2}$ in Romberg integration is identical to composite Simpson's 1/3 rule in Eq. (6.10) with 2^{k-1} panels.

Solution Recall that in Romberg integration $R_{k,1} = I_k$ denoted the approximate integral obtained by the composite trapezoidal rule with 2^{k-1} panels. Denoting the abscissas of the nodes by $x_1, x_2, \ldots, x_n$, we have from the composite trapezoidal rule in Eq. (6.5)

$$R_{k,1} = I_k = \left[f(x_1) + 2\sum_{i=2}^{n-1} f(x_i) + \frac{1}{2} f(x_n) \right] \frac{h}{2}$$

When we halve the number of panels (panel width $2h$), only the odd-numbered abscissas enter the composite trapezoidal rule, yielding

$$R_{k-1,1} = I_{k-1} = \left[f(x_1) + 2\sum_{i=3,5,\ldots}^{n-2} f(x_i) + f(x_n) \right] h$$

Applying Richardson extrapolation yields

$$\begin{aligned} R_{k,2} &= \frac{4}{3} R_{k,1} - \frac{1}{3} R_{k-1,1} \\ &= \left[\frac{1}{3} f(x_1) + \frac{4}{3} \sum_{i=2,4,\ldots}^{n-1} f(x_i) + \frac{2}{3} \sum_{i=3,5,\ldots}^{n-2} f(x_i) + \frac{1}{3} f(x_n) \right] h \end{aligned}$$

which agrees with Simpson's rule in Eq. (6.10).

Example

Use Romberg integration to evaluate $\int_0^{\pi} f(x)\,dx$, where $f(x) = \sin x$. Work with four decimal places.

Solution From the recursive trapezoidal rule in Eq. (6.9b), we get

$$R_{1,1} = I(\pi) = \frac{\pi}{2}\left[f(0) + f(\pi)\right] = 0$$

$$R_{2,1} = I(\pi/2) = \frac{1}{2}I(\pi) + \frac{\pi}{2}f(\pi/2) = 1.5708$$

$$R_{3,1} = I(\pi/4) = \frac{1}{2}I(\pi/2) + \frac{\pi}{4}\left[f(\pi/4) + f(3\pi/4)\right] = 1.8961$$

$$\begin{aligned} R_{4,1} = I(\pi/8) &= \frac{1}{2}I(\pi/4) + \frac{\pi}{8}\left[f(\pi/8) + f(3\pi/8) + f(5\pi/8) + f(7\pi/8)\right] \\ &= 1.9742 \end{aligned}$$

Using the extrapolation formulas in Eqs. (6.13), we can now construct the following table:

$$\begin{bmatrix} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ R_{3,1} & R_{3.2} & R_{3,3} & \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} \end{bmatrix} = \begin{bmatrix} 0 & & & \\ 1.5708 & 2.0944 & & \\ 1.8961 & 2.0046 & 1.9986 & \\ 1.9742 & 2.0003 & 2.0000 & 2.0000 \end{bmatrix}$$

It appears that the procedure has converged. Therefore, $\int_0^{\pi} \sin x\,dx = R_{4,4} = 2.0000$, which is, of course, the correct result.

Example

Use Romberg integration to evaluate $\int_0^{\sqrt{\pi}} 2x^2 \cos x^2 \, dx$ and compare the results with Example 6.4.

Solution We use the following program:

```
% Example 6.7 (Romberg integration)
format long
func = @(x) (2*(x^2)*cos(x^2));
[Integral,numEval] = romberg(func,0,sqrt(pi))
```

The results are

```
Integral =
  -0.894831469484l6
numEval =
   257
```

It is clear that Romberg integration is considerably more efficient than the trapezoidal rule. It required 257 function evaluations as compared to 4097 evaluations with the composite trapezoidal rule in Example 6.4.

Gaussian Integration

Gaussian Integration Formulas

We found that Newton–Cotes formulas for approximating $\int_a^b f(x)dx$ work best is $f(x)$ is a smooth function, such as a polynomial. This is also true for Gaussian quadrature. However, Gaussian formulas are also good at estimating integrals of the form

$$\int_a^b w(x)f(x)dx \tag{6.15}$$

where $w(x)$, called the *weighting function*, can contain singularities, as long as they are integrable. An example of such integral is $\int_0^1 (1+x^2)\ln x\,dx$. Sometimes infinite limits, as in $\int_0^\infty e^{-x}\sin x\,dx$, can also be accommodated.

Gaussian integration formulas have the same form as Newton–Cotes rules

$$I = \sum_{i=1}^{n} A_i f(x_i) \tag{6.16}$$

where, as before, I represents the approximation to the integral in Eq. (6.15). The difference lies in the way that the weights A_i and nodal abscissas x_i are determined. In Newton–Cotes integration the nodes were evenly spaced in (a, b), that is, their locations were predetermined. In Gaussian quadrature the nodes and weights are chosen so that Eq. (6.16) yields the *exact integral* if $f(x)$ is a polynomial of degree $2n-1$ or less; that is,

$$\int_a^b w(x)\,P_m(x)dx = \sum_{i=1}^{n} A_i\,P_m(x_i), \quad m \le 2n-1 \tag{6.17}$$

One way of determining the weights and abscissas is to substitute $P_1(x) = 1$, $P_2(x) = x, \ldots, P_{2n-1}(x) = x^{2n-1}$ in Eq. (6.17) and solve the resulting $2n$ equations

$$\int_a^b w(x)x^j\,dx = \sum_{i=1}^{n} A_i x_i^j, \quad j = 0, 1, \ldots, 2n-1$$

for the unknowns A_i and x_i, $i = 1, 2, \ldots, n$.

As an illustration, let $w(x) = e^{-x}$, $a = 0$, $b = \infty$, and $n = 2$. The four equations determining x_1, x_2, A_1, and A_2 are

$$\int_0^\infty e^{-x}dx = A_1 + A_2$$

$$\int_0^1 e^{-x}x\,dx = A_1x_1 + A_2x_{21}$$

$$\int_0^1 e^{-x}x^2dx = A_1x_1^2 + A_2x_2^2$$

$$\int_0^1 e^{-x}x^3dx = A_1x_1^3 + A_2x_2^3$$

$$x_1 = 2 - \sqrt{2} \qquad A_1 = \frac{\sqrt{2}+1}{2\sqrt{2}}$$

$$x_2 = 2 + \sqrt{2} \qquad A_2 = \frac{\sqrt{2}-1}{2\sqrt{2}}$$

$$\int_0^\infty e^{-x}f(x)dx \approx \frac{1}{2\sqrt{2}}\left[(\sqrt{2}+1)\,f\left(2-\sqrt{2}\right) + (\sqrt{2}-1)\,f\left(2+\sqrt{2}\right)\right]$$

After Evaluating

$$A_1 + A_2 = 1$$

$$A_1x_1 + A_2x_2 = 1$$

$$A_1x_1^2 + A_2x_2^2 = 2$$

$$A_1x_1^3 + A_2x_2^3 = 6$$

Abscissas and Weights for Gaussian Quadratures

We list here some classical Gaussian integration formulas. The tables of nodal abscissas and weights, covering $n = 2$ to 6, have been rounded off to six decimal places. These tables should be adequate for hand computation, but in programming you may need more precision or a larger number of nodes. In that case you should consult other references,[13] or use a subroutine to compute the abscissas and weights within the integration program.[14]

The truncation error in Gaussian quadrature

$$E = \int_a^b w(x)f(x)dx - \sum_{i=1}^{n} A_i f(x_i)$$

has the form $E = K(n)f^{(2n)}(c)$, where $a < c < b$ (the value of c is unknown; only its bounds are given). The expression for $K(n)$ depends on the particular quadrature being used. If the derivatives of $f(x)$ can be evaluated, the error formulas are useful in estimating the error bounds.

Gauss–Legendre Quadrature

$$\int_{-1}^{1} f(\xi)d\xi \approx \sum_{i=1}^{n} A_i f(\xi_i) \tag{6.26}$$

$\pm\xi_i$		A_i	$\pm\xi_i$		A_i
	$n=2$			$n=4$	
0.577 350		1.000 000	0.000 000		0.568 889
	$n=2$		0.538 469		0.478 629
0.000 000		0.888 889	0.906 180		0.236 927
0.774 597		0.555 556		$n=6$	
	$n=4$		0.238 619		0.467 914
0.339 981		0.652 145	0.661 209		0.360 762
0.861 136		0.347 855	0.932 470		0.171 324

Table 6.3

This is the most often used Gaussian integration formula. The nodes are arranged symmetrically about $\xi = 0$, and the weights associated with a symmetric pair of nodes are equal. For example, for $n = 2$, we have $\xi_1 = -\xi_2$ and $A_1 = A_2$. The truncation error in Eq. (6.26) is

$$E = \frac{2^{2n+1}\left(n!\right)^4}{(2n+1)\left[(2n)!\right]^3} f^{(2n)}(c), \quad -1 < c < 1 \tag{6.27}$$

To apply Gauss–Legendre quadrature to the integral $\int_a^b f(x)dx$, we must first map the integration range (a, b) into the "standard" range $(-1, 1)$. We can accomplish this by the transformation

$$x = \frac{b+a}{2} + \frac{b-a}{2}\xi \tag{6.28}$$

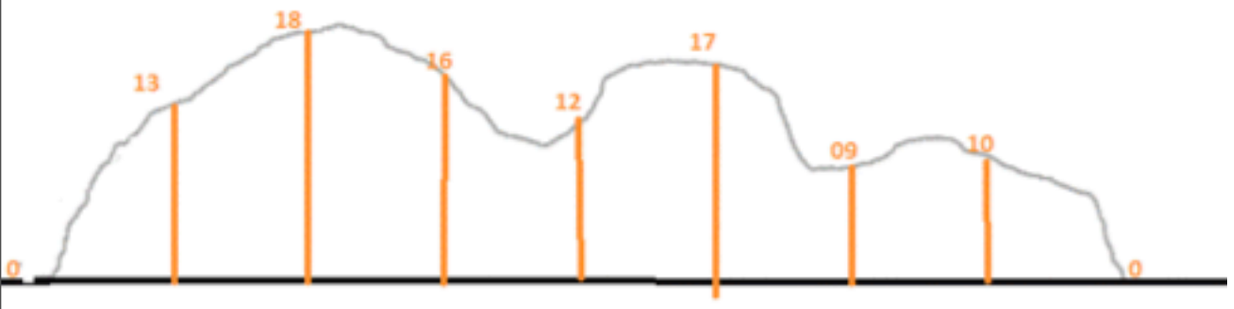
Now $dx = d\xi\,(b-a)/2$, and the quadrature becomes

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i) \tag{6.29}$$

where the abscissas x_i must be computed from Eq. (6.28). The truncation error here is

$$E = \frac{(b-a)^{2n+1}\left(n!\right)^4}{(2n+1)\left[(2n)!\right]^3} f^{(2n)}(c), \quad a < c < b \tag{6.30}$$

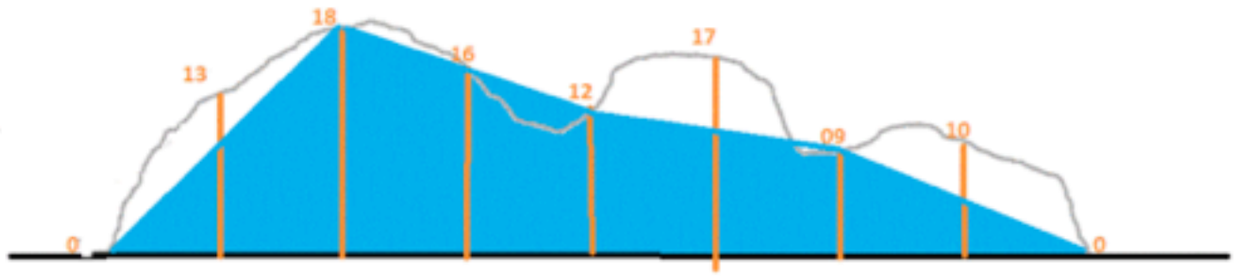
Overall length = 80
Heights are in orange



1 piece trapezoid rule estimate i
because the heights at the
endpoints are zero

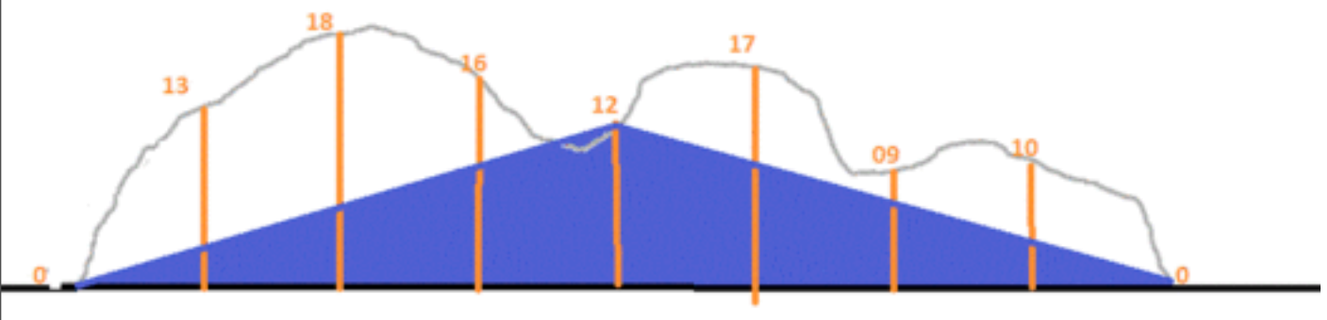
4 piece trapezoid rule
shaded area = 20(18+12+9) = 780

Overall length = 80
Heights are in orange



2 piece trapezoid rule
shaded area = 40(12) = 480

Overall length = 80
Heights are in orange



8 piece trapezoid rule
shaded area = 10(13+18+16+12+17+9+10)=950

Overall length = 80
Heights are in orange