Chapter 8
Numerical Differentiation and Integration

8.1 Introduction

The concepts of differentiation and integration are fundamental to calculus. The problem of computing a rate of change, for example velocity from position, and the geometric task of calculating an area beneath the graph of a positive a function are, perhaps, the most common applications of these concepts. Although both ideas are well understood and simple to compute, the basic mathematical definitions of these fundamental concepts are often forgotten.

The derivative and the definite integral are both defined in terms of special limits. As we investigate differentiation and integration from a numerical point of view it is important to recall the basic definitions. The derivative of $f(x)$, denoted $f'(x)$, is defined by the limit of a quotient of differences. The standard form is

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{(x+h) - x} \;=\; \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}. \tag{8.1}$$

There are various equivalent definitions that may be useful in certain applications, for example, if f' exists,

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h} \tag{8.2}$$

is a definition centered at x.

The definite integral of $f(x)$ from a to b, denoted $\int_a^b f(x)dx$, is often defined as the limit of a Riemann sum

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(\xi_i)\Delta x \tag{8.3}$$

where $\Delta x = (b-a)/n$ and ξ_i is a value in the i^{th} subinterval of $[\,a,\,b\,]$ specified by the inequality $a + (i-1)\Delta x \le \xi_i \le a + i\Delta x$.

Limit definitions such as (8.1), (8.2) and (8.3) frequently provide the basis for numerical approximations. For example, using (8.2), the derivative of $f(x)$ at x_1 may be approximated by simply deleting the limit and selecting a small value for h.

$$f'(x_1) \approx \frac{f(x_1+h) - f(x_1-h)}{2h}. \tag{8.4}$$

The geometric implication of (8.4) should be clear. The slope of a line between the points

$(x_1 + h,\ f(x_1 + h))$ and $(x_1 - h,\ f(x_1 - h))$ will approximate the derivative. The definite integral may be approximated by the sum $\sum_{i=1}^{n} f(\xi_i)\Delta x$ wherein each product $f(\xi_i)\Delta x$ may be interpreted, assuming that $f(x)$ is positive, as the area of a rectangle with base Δx and height $f(\xi_i)$.

8.2 Derivatives and Numerical Approximations

It is important to understand that there are two underlying themes in the numerical approximation of derivatives. On one hand, we may have a data set and desire to approximate a rate of change at some point, say $\overline{x}$. Our previous work with interpolating polynomials, splines and least squares models has demonstrated how these different approaches may be used to estimate derivatives. We simply compute $P_n'(\overline{x})$, $S'(\overline{x})$ or $F'(\overline{x})$. All three methods will provide estimates of the rate of change – some good and some not so good.

A second point of view focuses on deriving formulas to approximate derivatives. For example, using (8.4), the differential equation $y' = x - y$ at $x = x_i$, $y'(x_i) = x_i - y(x_i)$, may be approximated by a *difference* equation,

$$\frac{y(x_i + h) - y(x_i - h)}{2h} = x_i - y(x_i). \tag{8.5}$$

Equation (8.5) represents a *discrete* model for the continuous differential equation. Using the notation $x_i = x_0 + ih$ and $y_i = y(x_i)$, equation (8.5) becomes $y_{i+1} = y_{i-1} + 2h(x_i - y_i)$. The solution to this difference equation will model the solution of the differential equation. If you are familiar with analytic solution methods for differential equations, the procedures to solve difference equations are similar.

Lagrange Polynomial Models

Lagrange interpolating polynomials, discussed in Section 6.2, may be used to compute approximations for derivatives. For example, the derivative of a linear interpolating polynomial $P_1(x)$, (6.10), gives the expected difference result $P_1'(x) = \frac{y_2 - y_1}{x_2 - x_1}$. The MATLAB command **diff** may be used to compute this quotient. Convince yourself that the following MATLAB commands will compute four difference quotients each of the form $\frac{y_{i+1} - y_i}{x_{i+1} - x_i}$.

```
>> x=[1 2 3 4 5];y=[3 7 -1 2 6];
>> diff(y)./diff(x)
ans =
   4   -8    3    4
```

A more complicated formula follows from a quadratic interpolating polynomial, see (6.12), $P_2(x) = l_1(x)y_1 + l_2(x)y_2 + l_3(x)y_3$. The important factors in $P_2'(x)$ are the derivatives of the quadratic $l_i(x)$'s defined in Chapter 6. Differentiating the quadratic Lagrange terms we find

$$l_1'(x) = \frac{(x-x_2)+(x-x_3)}{(x_1-x_2)(x_1-x_3)},$$

$$l_2'(x) = \frac{(x-x_1)+(x-x_3)}{(x_2-x_1)(x_2-x_3)}, \tag{8.6}$$

and $$l_3'(x) = \frac{(x-x_1)+(x-x_2)}{(x_3-x_1)(x_3-x_2)}.$$

The expressions in (8.6) may be used with data values, x_1, x_2, and x_3, to construct an approximation for the derivative. If $x = \overline{x}$, $P_2'(\overline{x})$ provides a numerical estimate of the derivative using the formula

$$P_2'(\overline{x}) \;=\; l_1'(\overline{x})y_1 + l_2'(\overline{x})y_2 + l_3'(\overline{x})y_3. \tag{8.7}$$

One possible use of equation (8.7) is the computation of endpoint slopes for use in a clamped spline. See (6.43). Given the luxury of preselecting $x_1, x_2,$ and x_3 difference formulas for the derivative will follow. For example, with $x_1 = \overline{x}$, $x_2 = \overline{x} + h$, and $x_3 = \overline{x} + 2h$, (8.7) becomes

$$P_2'(\overline{x}) \;=\; \frac{-3h}{(-h)(-2h)}y_1 + \frac{-2h}{(h)(-h)}y_2 + \frac{-h}{(2h)(h)}y_3$$

or $$P_2'(\overline{x}) \;= P_2'(x_1) \;= -\frac{3}{2h}y_1 + \frac{4}{2h}y_2 - \frac{1}{2h}y_3 = \frac{-3y_1 + 4y_2 - 1y_3}{2h}. \tag{8.8}$$

Equation (8.8) is called a *forward* difference approximation for the first derivative. With $h > 0$, the values for x_2 and x_3 are to the right or forward of x_1. Since the coefficients (weighting values) in the numerator, $-3,\ 4$ and $-1,$ sum to zero, the numerator will also approach zero as h approaches zero producing the familiar undetermined form $0/0$.

Extending the development begun in Section 6.2, the symbolic capability of MATLAB may be used to derive the forward difference formula given in (8.8). Using the data points (x_1, y_1), (x_2, y_2) and (x_3, y_3), the following MATLAB commands construct, differentiate and evaluate a symbolic quadratic interpolating polynomial, denoted **P2**. Recall that **v** represents the Vandermonde matrix.

```
>> syms h x x1 x2 x3 y1 y2 y3
>> v = [x1^2, x1, 1; x2^2, x2, 1; x3^2, x3, 1];
>> P2 = [x^2, x, 1]*inv(v)*[y1; y2; y3];
>> P2prime = diff(P2);
>> P2primeatx1 = subs(P2prime,x,x1);
>> factor(subs(P2primeatx1,[x2, x3],[x1+h, x1+2*h]))
ans =
-1/2*(3*y1-4*y2+y3)/h
```

As a second example, set $x_2 = \overline{x}$, $x_1 = \overline{x} - h$, and $x_3 = \overline{x} + h$. In this case (8.7) becomes

$$P_2'(\overline{x}) = P_2'(x_2) = \frac{-h}{(-h)(-2h)}y_1 + \frac{0}{(h)(-h)}y_2 + \frac{h}{(2h)(h)}y_3 = \frac{-1y_1 + 1y_3}{2h}. \quad (8.9)$$

Using MATLAB we may verify the result given in (8.9).

```
>> P2primeatx2 = subs(P2prime, x, x2);
>> factor(subs(P2primeatx2, [x1, x3], [x2-h, x2+h]))
ans =
-1/2*(y1-y3)/h
```

Equation (8.9) defines a *central* difference approximation for the first derivative and represents the slope of a line between the points $(\overline{x} - h, y_1)$ and $(\overline{x} + h, y_3)$. Note that values on either side of $x_2 = \overline{x}$ are used in the quotient. Although approximations (8.8) and (8.9) are based on equally spaced values, it is important to remember that the general form of (8.7) may be used with any values of $x_1, x_2,$ and x_3.

The second derivative of $P_2(x)$ is a constant on the interval $[x_1, x_3]$. Using equally spaced data, $x_1 = \overline{x}$, $x_2 = \overline{x} + h$, and $x_3 = \overline{x} + 2h$, we find

```
>> factor(subs(diff(P2, 2), [x2, x3], [x1+h, x1+2*h]))
ans =
(y1-2*y2+y3)/h^2
```

that is,
$$P_2''(x) = \frac{y_1 - 2y_2 + y_3}{h^2}. \quad (8.10)$$

Once again, the weights in the numerator of (8.10) sum to zero. This property of difference approximations for derivatives will be explained later in the section.

Since $P_2(x)$ requires three points, the formulas given in (8.8), (8.9) and (8.10) are sometimes called three term approximations for derivatives. Additional points and higher order interpolating polynomials lead to various formulas for difference approximations. The problems at the end of this chapter will provide additional examples.

Taylor Polynomial Models

Taylor polynomials may also be used to construct difference approximations of derivatives. The defining equations for $T_n(x)$ and $R_n(x)$, from (3.4) and (3.8), give the Taylor expression for $f(x)$ centered at a:

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(a)(x-a)^k + \frac{1}{(n+1)!} f^{(n+1)}(c)(x-a)^{n+1}, \quad (8.11)$$

where c is an unknown between a and x.

To simplify our initial efforts, we begin with $n = 1$.

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(c)(x-a)^2 \tag{8.12}$$

The choices $a = x_i$ and $x = x_{i+1}$ lead to

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)f'(x_i) + \tfrac{1}{2}(x_{i+1} - x_i)^2 f''(c). \tag{8.13}$$

Solving for $f'(x_i)$ gives

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} - \frac{1}{2}(x_{i+1} - x_i)f''(c). \tag{8.14}$$

It should be clear that the difference approximation $f'(x_i) \approx \frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}$ with $i = 1$, $f'(x_1) \approx \frac{f(x_2)-f(x_1)}{x_2-x_1}$, corresponds to the difference formula given by $P_1'(x) = \frac{y_2-y_1}{x_2-x_1}$. In both cases, the slope of a secant line is used to approximate the derivative. The Taylor result, (8.14), includes an error term, $-\frac{1}{2}(x_{i+1} - x_i)f''(c)$. Note that the length of the interval, $x_{i+1} - x_i$, and the concavity of the function (measured by the second derivative) are the important factors in the error expression. If the concavity is small, the graph of $f(x)$ resembles a line and the forward difference $\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}$ gives a good approximation to the derivative at x_i.

A second choice of values in (8.12), $a = x_i$ and $x = x_{i-1}$, results in a *backward* difference model

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} + \frac{1}{2}(x_i - x_{i-1})f''(\overline{c}). \tag{8.15}$$

Both (8.14) and (8.15) may be simplified using $x_{i+1} = x_i + h$ and $x_{i-1} = x_i - h$. Note that the error terms will depend on h the spacing between the values. In the remainder of this section, we assume that the data is uniformly spaced.

Numerous formulas may be derived from the Taylor expression (8.11). Particular choices for n, a and x lead to specific models. For example, with $n = 2$ we have the four term Taylor expression

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{3!}f'''(c)(x-a)^3. \tag{8.16}$$

With $a = x_i$ and $x = x_i + h$, (8.16) becomes

$$f(x_i + h) = f(x_i) + hf'(x_i) + \frac{1}{2}h^2 f''(x_i) + \frac{1}{3!}h^3 f'''(c). \tag{8.17}$$

Replacing h by $-h$ gives

$$f(x_i - h) = f(x_i) - hf'(x_i) + \frac{1}{2}h^2 f''(x_i) - \frac{1}{3!}h^3 f'''(\overline{c}). \tag{8.18}$$

Subtracting (8.18) from (8.17) will eliminate $f(x_i)$ and $f''(x_i)$ so that

$$f(x_i + h) - f(x_i - h) = 2hf'(x_i) + \frac{h^3}{6}[f'''(c) + f'''(\overline{c})].$$

Solving for $f'(x_i)$ produces the central difference model for the first derivative together with the error term:

$$f'(x_i) = \frac{1}{2h}[f(x_i + h) - f(x_i - h)] - \frac{h^2}{6}\frac{1}{2}[f'''(c) + f'''(\overline{c})].$$

The expression $\frac{1}{2}[f'''(c) + f'''(\overline{c})]$ in the error term is an average to two unknown values of the third derivative. Using the *Intermediate Value Theorem* from calculus we may show that $\frac{1}{2}[f'''(c) + f'''(\overline{c})] = f'''(\xi)$ where the unknown ξ is somewhere between $x_i - h$ and $x_i + h$. Thus, the final expression becomes

$$f'(x_i) = \frac{1}{2h}[f(x_i + h) - f(x_i - h)] - \frac{h^2}{6}f'''(\xi). \tag{8.19}$$

Equation (8.19) shows that the error in a central difference approximation is proportional to h^2. This is superior to the forward and backward difference error results which are proportional to h^1. See (8.14) and (8.15). Wherever possible, the first derivative should be approximated with the central difference formula $f'(x_i) \approx \frac{1}{2h}[f(x_i + h) - f(x_i - h)]$.

Let's use MATLAB to illustrate the use of the three difference approximations. Consider the derivative of $\sin(x)$ at $x = 1.12$, i.e., $\cos(1.12)$. The following commands will produce five data points on the interval $[\,1.10, 1.14\,]$ with $x_3 = 1.12$.

```
>> x=linspace(1.10,1.14,5); y=sin(x);
```

Forward and backward difference approximations are

```
>> FwdDiff=(y(4)-y(3))/.01
FwdDiff =
  4.3117e-001
>> BkwDiff=(y(3)-y(2))/.01
BkwDiff =
  4.4018e-001
```

For comparison, the true value of the derivative is given by

```
>> True=cos(x(3))
True =
  4.3568e-001
```

Using the error terms in (8.14) and (8.15) you should be able to show, for this example, that the forward difference approximation is too low and, conversely, the backward difference approximation is too high as is seen in the preceding numerical results. From (8.19), the central difference approximation is

```
>> CtrDiff=(y(4)-y(2))/.02
CtrDiff =
  4.3568e-001
```

```
>> Err=True-CtrDiff
Err =
  7.2613e-006
```

The value **Err** reveals that the central difference approximation is more accurate but still low. The error term in (8.19) will allow us to bound the error. Since the third derivative of $\sin(x)$ is $-\cos(x)$, we have

$$-\frac{h^2}{6}f^{(3)}(\xi) = -\frac{(.01)^2}{6}(-\cos(\xi)) \text{ where } 1.10 \le \xi \le 1.14.$$

It is a simple task to show that the error term is positive and less than $\frac{(.01)^2}{6}\cos(1.10)$ $\approx 7.5599 \cdot 10^{-6}$ on the interval $1.10 \le \xi \le 1.14$. The actual error $7.2613 \cdot 10^{-6}$ is slightly smaller than the bound. An error analysis of this type works well for known functions. A similar analysis for experimental data is not possible. (Why?)

Other formulas may be derived by thoughtful use of Taylor expansions. As an example we will derive a three term forward difference approximation. Consider (8.16) with the error term deleted

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2.$$

Replacing a with x_i and x with $x_i + h$,

$$f(x_i + h) \approx f(x_i) + hf'(x_i) + \frac{1}{2}h^2 f''(x_i),$$

and then a second time with h replaced by $2h$ gives

$$f(x_i + 2h) \approx f(x_i) + 2hf'(x_i) + \frac{1}{2}(2h)^2 f''(x_i).$$

Forming the difference $f(x_i + 2h) - 4f(x_i + h)$ will eliminate the second derivative $f''(x_i)$ as follows:

$$f(x_i + 2h) - 4f(x_i + h) \approx -3f(x_i) - 2hf'(x_i).$$

One last step gives a three term forward difference formula for the first derivative

$$f'(x_i) \approx \frac{1}{2h}\Big[-3f(x_i) + 4f(x_i + h) - 1f(x_i + 2h)\Big]. \qquad (8.20)$$

Equation (8.20) corresponds to the Lagrange form given in (8.8).

Second derivatives

So far our efforts have focused on the first derivative. Taylor methods may be used to model higher order derivatives. Using the expressions from (8.17) and (8.18),

$$f(x_i + h) \approx f(x_i) + hf'(x_i) + \frac{1}{2}h^2 f''(x_i)$$

and $$f(x_i - h) \approx f(x_i) - hf'(x_i) + \frac{1}{2}h^2 f''(x_i),$$

a formula for the second derivative may be obtained by addition:

$$f(x_i + h) + f(x_i - h) \approx 2f(x_i) + h^2 f''(x_i).$$

Solving for the second derivative gives a central difference approximation for the second derivative:

$$f''(x_i) \approx \frac{1}{h^2}[f(x_i + h) - 2f(x_i) + f(x_i - h)] \tag{8.21}$$

Undetermined Coefficients

There is a third approach for developing difference formulas. For example, the three term forward difference formula for the first derivative given in (8.20) is a linear combination of three values of $f(x)$. We may construct a four term forward difference approximation for $f'(x_i)$ as follows:

$$f'(x_i) \approx Af(x_i) + Bf(x_i + h) + Cf(x_i + 2h) + Df(x_i + 3h), \tag{8.22}$$

where the parameters A, B, C and D are unknown, the undetermined coefficients.

Using Taylor expansions centered at x_i for the second, third and fourth terms in (8.22) we find

$$\begin{aligned} f'(x_i) \approx\ & Af(x_i) \\ & + B\Big[f(x_i) + hf'(x_i) + \frac{1}{2}h^2 f''(x_i) + \frac{1}{6}h^3 f'''(x_i) + \dots\Big] \\ & + C\Big[f(x_i) + 2hf'(x_i) + \frac{1}{2}4h^2 f''(x_i) + \frac{1}{6}8h^3 f'''(x_i) + \dots\Big] \\ & + D\Big[f(x_i) + 3hf'(x_i) + \frac{1}{2}9h^2 f''(x_i) + \frac{1}{6}27h^3 f'''(x_i) + \dots\Big]. \end{aligned}$$

Collecting terms on like powers of h gives

$$\begin{aligned} f'(x_i) \approx\ & \Big[A + B + C + D\Big]f(x_i) \\ & + h\Big[B + 2C + 3D\Big]f'(x_i) \\ & + \tfrac{1}{2}h^2\Big[B + 4C + 9D\Big]f''(x_i) \\ & + \tfrac{1}{6}h^3\Big[B + 8C + 27D\Big]f'''(x_i) + \dots \end{aligned} \tag{8.23}$$

We may ensure that a difference approximation derived from the right hand side of (8.23) is close to the derivative $f'(x_i)$ by imposing the following four conditions:

$$
\begin{aligned}
A + B + C + D &= 0 \\
h[B + 2C + 3D] &= 1 \\
\frac{1}{2}h^2[B + 4C + 9D] &= 0 \\
\frac{1}{6}h^3[B + 8C + 27D] &= 0
\end{aligned}
$$

The first equation forces the multiplier of $f(x_i)$ to zero. This will guarantee that the weights will sum to zero as observed in earlier difference approximations. In matrix form the equations are

$$
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 8 & 27 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 1/h \\ 0 \\ 0 \end{bmatrix} \tag{8.24}
$$

Using the symbolic capability of MATLAB the solution of (8.24) is

```
>> syms h
>> sym([1 1 1 1;0 1 2 3;0 1 4 9;0 1 8 27])\[0;1/h;0;0]
ans =
[ -11/6/h]
[    3/h]
[ -3/2/h]
[  1/3/h]
```

With a common denominator of $6h$, our four term approximation becomes

$$
f'(x_i) \approx \frac{1}{6h}[-11f(x_i) + 18f(x_i + h) - 9f(x_i + 2h) + 2f(x_i + 3h)]. \tag{8.25}
$$

Error terms associated with the Taylor expansions used to derive (8.23) will depend on h^4, $f^{(4)}$ and the undetermined coefficients. Since the undetermined coefficients are proportional to $1/h$, the error in (8.25) is actually proportional to h^3.

All three procedures, Lagrange interpolating polynomials, Taylor expansions and undetermined coefficients, may be used to derive difference approximations. For reference, some standard formulas are summarized below. Errors proportional to h^n are expressed using the *big oh* notation, $\mathcal{O}(h^n)$.

First Derivative - Forward Differences:

$$
f'(x_i) = \frac{1}{h}[f(x_i + h) - f(x_i)] + \mathcal{O}(h)
$$

$$
f'(x_i) = \frac{1}{2h}[-f(x_i + 2h) + 4f(x_i + h) - 3f(x_i)] + \mathcal{O}(h^2)
$$

$$f'(x_i) = \frac{1}{6h}[2f(x_i + 3h) - 9f(x_i + 2h) + 18f(x_i + h) - 11f(x_i)] + \mathcal{O}(h^3)$$

First Derivative - Central Difference:

$$f'(x_i) = \frac{1}{2h}[f(x_i + h) - f(x_i - h)] + \mathcal{O}(h^2)$$

First Derivative - Backward Differences:

$$f'(x_i) = \frac{1}{h}[f(x_i) - f(x_i - h)] + \mathcal{O}(h)$$

$$f'(x_i) = \frac{1}{2h}[3f(x_i) - 4f(x_i - h) + f(x_i - 2h)] + \mathcal{O}(h^2)$$

Second Derivative - Forward Difference:

$$f''(x_i) = \frac{1}{h^2}[f(x_i + 2h) - 2f(x_i + h) + f(x_i)] + \mathcal{O}(h)$$

Second Derivative - Central Difference:

$$f''(x_i) = \frac{1}{h^2}[f(x_i + h) - 2f(x_i) + f(x_i - h)] + \mathcal{O}(h^2)$$

There are comparable difference formulas for partial derivatives that are used in the numerical solution of partial differential equations. Advanced texts on numerical analysis will provide the specific formulas.

8.3 Numerical Integration - Introduction

The computation of a definite integral, $I = \int_a^b f(x)dx$, is usually accomplished using the *Fundamental Theorem of Integral Calculus*, $I = F(b) - F(a)$, where $F(x)$ is an antiderivative of $f(x)$ so that $F'(x) = f(x)$. The theorem assumes that the antiderivative can be found. Unfortunately, there are cases where $F(x)$ cannot be determined in closed form. In these cases, a numerical method will be necessary to compute an approximate value for the integral. A standard example that cannot be evaluated by the Fundamental Theorem is the integral $\int_a^b e^{-x^2}dx$. In other words, the antiderivative $\int e^{-x^2}dx$ cannot be expressed in a simple form.

Current mathematical software is designed to apply the Fundamental Theorem to most problems; however, difficulties do arise in special cases. There are exceptional problems where

the computation of an antiderivative is possible provided extra, time consuming, steps are taken by the user before the software may finish the task. For many of these exceptional cases, a fast numerical approximation may be the preferred choice.

Numerical approximation schemes, sometimes called quadrature formulas or rules, usually resemble the Riemann sum in (8.3). They have the general form

$$Q_n = \sum_{i=1}^{n} w_i f(\xi_i), \tag{8.26}$$

where the w_i's (weights) and ξ_i's (values in the interval [a, b]) are specified by a particular rule. The goal is to specify the w_i's and ξ_i's so that Q_n approximates the true value of the definite integral. As may be expected, the choice of n is related to accuracy.

Many quadrature rules are derived by approximating the integrand, $f(x)$, with a known function that is simple to integrate. In the following sections, we will discuss the *Trapezoidal Rule* (replace the integrand $f(x)$ with a linear interpolating polynomial) and *Simpson's Rule* (replace the integrand with a quadratic interpolating polynomial). Another approach, *Gaussian quadrature*, is designed to be exact provided the integrand is a polynomial.

8.4 The Trapezoidal Rule

As noted above, the Trapezoidal Rule uses a linear interpolating polynomial, $P_1(x)$, to approximate the integrand. In other words, $I = \int_a^b f(x)dx \approx T_1 = \int_a^b P_1(x)dx$. Figure 8.1 shows a graph of $P_1(x)$ interpolating $f(x)$ at x_1 and x_2.

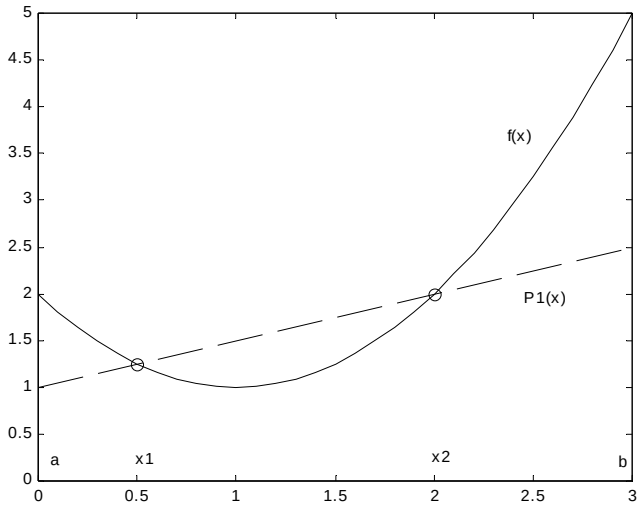

Figure 8.1 Trapezoidal Rule

The trapezoidal area beneath $P_1(x)$ motivates the name of the method. Using (6.10), we find

$$T_1 = \int_a^b P_1(x)dx = \int_a^b \Big[\Big(\frac{x-x_2}{x_1-x_2}\Big)f(x_1) + \Big(\frac{x-x_1}{x_2-x_1}\Big)f(x_2)\Big]dx$$

or

$$T_1 = f(x_1)\int_a^b \Big(\frac{x-x_2}{x_1-x_2}\Big)dx + f(x_2)\int_a^b \Big(\frac{x-x_1}{x_2-x_1}\Big)dx. \tag{8.27}$$

It should be clear that the numerical value of T_1 depends on the location of the interpolation points, x_1 and x_2, in the interval $[\,a,\,b\,]$. To avoid numerous trapezoidal rules, convention requires that the interpolation points correspond to the limits of integration; thus, $x_1 = a$ and $x_2 = b$. With this choice, it is a simple matter to arrive at the Trapezoidal Rule where $h = b - a$.

$$T_1 = \frac{1}{2}h\Big[f(a) + f(b)\Big]. \tag{8.28}$$

A simple application of the Trapezoidal Rule seldom provides a satisfactory approximation. One way to improve accuracy begins with a partition of the interval $[\,a,\,b\,]$ into multiple subintervals. For convenience, we may subdivide the interval into n subintervals each of length $h = (b-a)/n$ so that the original problem $\int_a^b f(x)dx$ is converted into a sum of integrals

$$\int_a^b f(x)dx = \sum_{i=1}^{n}\int_{x_i}^{x_{i+1}} f(x)dx. \tag{8.29}$$

where $x_i = a + (i-1)h$. We now apply the Trapezoidal template, (8.28), to each integral in (8.29) to arrive at the *composite* Trapezoidal Rule for n subintervals

$$T_n = \sum_{i=1}^{n}\frac{1}{2}h\Big[f(x_i) + f(x_{i+1})\Big]. \tag{8.30}$$

With $f_i = f(x_i)$, (8.30) may be written as

$$T_n = \frac{1}{2}h\Big[f_1 + 2f_2 + 2f_3 + \ldots + 2f_n + f_{n+1}\Big]. \tag{8.31}$$

Note that the composite formula T_n actually uses $n+1$ function evaluations. To illustrate the use of the composite formula, we approximate the integral $\int_\pi^{2\pi}\frac{\sin(x)}{x}dx$ using $n = 4$ subintervals. With $h = \frac{\pi}{4}$,

$$T_4 = \frac{1}{2}\frac{\pi}{4}\Big[1\frac{\sin(\pi)}{\pi} + 2\frac{\sin(5\pi/4)}{5\pi/4} + 2\frac{\sin(6\pi/4)}{6\pi/4} + 2\frac{\sin(7\pi/4)}{7\pi/4} + 1\frac{\sin(2\pi)}{2\pi}\Big] \approx -.409.$$

The example integral may be evaluated using **sinint** a special MATLAB command to evaluate the so-called *Sine Integral* function, $\mathrm{Si}(x) = \int_0^x \frac{\sin(t)}{t}dt$. Using properties of definite integrals

$$\int_{\pi}^{2\pi} \frac{\sin(x)}{x} dx = \int_{0}^{2\pi} \frac{\sin(x)}{x} dx - \int_{0}^{\pi} \frac{\sin(x)}{x} dx = \text{Si}(2\pi) - \text{Si}(\pi).$$

>> sinint(2*pi)-sinint(pi)
ans =
-4.3379e-001

The value for T_4 is reasonable but not too accurate.

Intuitively, we expect that increasing the number of subintervals will produce a better value and that $\lim_{n\to\infty} T_n = I$. In theory, the mathematical Trapezoidal Rule will converge to the true value of the integral; however, implementing the Trapezoidal Rule on a computer introduces errors as the result of a finite word length. In other words, an estimate of T_n, denoted $\widehat{T}_n$, is actually calculated. At some point, the accumulated error from numerous computations may give misleading information about the value of the integral.

It seems reasonable to approximate I by computing a sequence of values $\{\widehat{T}_1, \widehat{T}_2, \widehat{T}_3, \widehat{T}_4, \dots\}$. As we shall see, there are advantages to computing the sequence of values $\{\widehat{T}_2, \widehat{T}_4, \widehat{T}_8, \widehat{T}_{16}, \dots\}$ wherein the number of subintervals is doubled each time. One advantage is that some values of f may be reused in subsequent steps; thus, there is no need to re-compute all values.

Consider the following expressions

$$T_2 = \frac{1}{2}\frac{(b-a)}{2}\Big[f(a) + 2f(a+\frac{h}{2}) + f(b)\Big]$$

and

$$T_4 = \frac{1}{2}\frac{(b-a)}{4}\Big[f(a) + 2f(a+\frac{h}{4}) + 2f(a+\frac{2h}{4}) + 2f(a+\frac{3h}{4}) + f(b)\Big].$$

As we move from T_2 to T_4 some values of f may be reused. For example, $f(a+\frac{h}{2})$ and $f(a+\frac{2h}{4})$ are the same. Rewriting these expressions using $x_i = a + (i-1)h$ gives

$$T_2 = \frac{1}{2}\frac{(b-a)}{2}\Big[f(a) + 2f(x_2) + f(b)\Big]$$

and

$$T_4 = \frac{1}{2}\frac{(b-a)}{4}\Big[f(a) + 2f(x_2) + 2f(x_3) + 2f(x_4) + f(b)\Big].$$

Note that $f(a)$ and $f(b)$ need not be re-computed and that $f(x_3)$ in T_4 is identical to $f(x_2)$ in T_2. Only new values for $f(x_2)$ and $f(x_4)$ are needed. An economical structure exploiting doubling is given by

$$T_n = \frac{1}{2}\frac{(b-a)}{n}\Big[f(a) + f(b) + 2(f_2 + f_4 + \dots) + 2(f_3 + f_5 + \dots)\Big]. \qquad (8.32)$$

where terms with even and odd subscript indices have been grouped together. The reason for this grouping will become clear in the next section.

8.5 Simpson's Rule

Simpson's Rule, named after Thomas Simpson (1727-1761), uses a quadratic interpolating polynomial, $P_2(x)$, to approximate the integrand. In other words, $I = \int_a^b f(x)dx \approx S_2 = \int_a^b P_2(x)dx$. Again, the choice of interpolation points on the interval $[\,a,\,b\,]$ is an important factor. Following convention we set $x_1 = a, x_2 = \frac{1}{2}(a+b) = c$ (the midpoint), and $x_3 = b$. Using (6.12) we find

$$S_2 = \int_a^b P_2(x)dx = \int_a^b \Big[\frac{(x-b)(x-c)}{(a-b)(a-c)}f(a) + \frac{(x-a)(x-b)}{(c-a)(c-b)}f(c) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(b)\Big]dx.$$

Integrating each term gives

$$S_2 = f(a)\int_a^b \frac{(x-b)(x-c)}{(a-b)(a-c)}dx + f(c)\int_a^b \frac{(x-a)(x-b)}{(c-a)(c-b)}dx + f(b)\int_a^b \frac{(x-a)(x-c)}{(b-a)(b-c)}dx.$$

With $c = a + h$ and $b = a + 2h$, MATLAB may be employed to evaluate the integrals. In each case the results will be expressed in terms of $h = \frac{1}{2}(b-a)$.

```
>> syms a b c h x
>> term1 = factor(subs(int((x-b)*(x-c)/((a-b)*(a-c)),a,b),[c,b],[a+h,a+2*h]))
term1 =
1/3*h
>> term2 = factor(subs(int((x-a)*(x-b)/((c-a)*(c-b)),a,b),[c,b[,[a+h,a+2*h]))
term2 =
4/3*h
>> term3 = factor(subs(int((x-a)*(x-c)/((b-a)*(b-c)),a,b),[c,b],[a+h,a+2*h]))
term3 =
1/3*h
```

Study the syntax carefully. Three separate MATLAB commands, **int**, **subs** and **factor**, are nested together to compute the integrals, substitute for c and b, and factor the results. Substituting the integral values gives Simpson's Rule.

$$S_2 = \frac{1}{3}h\Big[f(a) + 4f(c) + f(b)\Big] \tag{8.33}$$

The subscript on S reflects the fact that there are two subintervals, each of length h.

The composite Simpson's Rule may be developed by subdividing the interval $[\,a,\,b\,]$ into an <u>even</u> number of subintervals. The integral $\int_a^b f(x)dx$ is now a sum of $n/2$ integrals.

$$\int_a^b f(x)dx = \sum_{i=1}^{n/2} \int_{x_{2i-1}}^{x_{2i+1}} f(x)dx \tag{8.34}$$

where $x_i = a + (i-1)h$. We now apply the Simpson template, (8.33), to each integral in (8.34) to arrive at the composite formula for n subintervals

$$S_n = \sum_{i=1}^{n/2} \frac{1}{3}h\Big[f(x_{2i-1}) + 4f(x_{2i}) + f(x_{2i+1})\Big]. \tag{8.35}$$

To better understand the summation in (8.35), let's write out a few terms

$$\begin{aligned} S_n = &\frac{1}{3}h\Big[f(x_1) + 4f(x_2) + f(x_3)\Big] \\ &+ \frac{1}{3}h\Big[f(x_3) + 4f(x_4) + f(x_5)\Big] \\ &+ \frac{1}{3}h\Big[f(x_5) + 4f(x_6) + f(x_7)\Big] \\ &\vdots \\ &+ \frac{1}{3}h\Big[f(x_{n-1}) + 4f(x_n) + f(x_{n+1})\Big]. \end{aligned}$$

Changing to subscript notation and regrouping terms gives the composite rule

$$S_n = \frac{1}{3}\frac{(b-a)}{n}\Big[f(a) + f(b) + 4(f_2 + f_4 + \ldots + f_n) + 2(f_3 + f_5 + \ldots + f_{n+1})\Big]. \tag{8.36}$$

The differences between (8.36) and the composite Trapezoidal Rule, (8.32), are the fractional multiplier in front, $\frac{1}{3}$versus $\frac{1}{2}$, and the multiplier of the even terms, 4 versus 2.

Let's re-compute the example integral of the previous section, $\int_\pi^{2\pi} \frac{\sin(x)}{x}dx$, using Simpson's Rule, (8.36), with four subintervals.

$$S_4 = \frac{1}{3}\frac{\pi}{4}\Big[\frac{\sin(\pi)}{\pi} + \frac{\sin(2\pi)}{2\pi} + 4\big(\frac{\sin(5\pi/4)}{5\pi/4} + \frac{\sin(7\pi/4)}{7\pi/4}\big) + 2\frac{\sin(6\pi/4)}{6\pi/4}\Big] \approx -.434$$

The Simpson result, S_4, is much closer to the actual value of $-.43379$.

The theoretical comments about the Trapezoidal Rule also apply to Simpson's Rule. Increasing the number of subintervals will produce a better estimate and $\lim_{n\to\infty} S_n = I$. As a practical issue we actually compute an estimate of S_n, say $\widehat{S}_n$. As noted earlier, there are advantages to computing a sequence of values $\{\widehat{S}_2, \widehat{S}_4, \widehat{S}_8, \widehat{S}_{16}, \ldots\}$ wherein the number of subintervals is doubled each time. In fact, (8.36) is structured to exploit the doubling scheme.

The following M-file will compute the composite results for both the Trapezoidal Rule and Simpson's Rule given in (8.32) and (8.36).

M-file trapsimp.m

```
function trapsimp(fun,a,b)
%TRAPSIMP Trapezoidal and Simpson's rule
% fun = the integrand in an M-file
% a, b = limits of integration
% n max  = 2^10 = 1024; may be changed
send = feval(fun,a)+feval(fun,b);
sodd = 0; seve = 0;
fprintf('    n                    Trap                    Simp\n')
for j = 2:11
  n = 2^(j-1);
  h = (b-a)/n;
  sodd = sodd+seve;
  seve = 0;
  for k = 1:2:n-1
        x = a+k*h;
        seve = seve+feval(fun,x);
  end
  % Trapezoidal rule
  tr = h*(send+2*seve+2*sodd)/2;
  % Simpson's rule
  sp = h*(send+4*seve+2*sodd)/3;
  output = [n tr sp];
  fprintf('%6.0f \t %8.4e \t %8.4e \n',output)
end
```

Note that the endpoint values are only computed once: **send = feval(fun,a)+feval(fun,b)**. The critical part of the code is the for-end loop that computes new even function values when the number of subintervals is doubled.

```
  for k = 1:2:n-1
        x = a+k*h;
        seve = seve+feval(fun,x);
  end
```

To illustrate, set $n = 8$. The counter k will take on values 1, 3, 5 and 7 leading to $x = a + h$, $a + 3h$, $a + 5h$ and $a + 7h$. Convince yourself that these values are x_2, x_4, x_6 and x_8. These numbers are used to compute the new even values as we move from $\widehat{T}_4$ to $\widehat{T}_8$ and $\widehat{S}_4$ to $\widehat{S}_8$.

As an example consider the integral

$$I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx.$$

I represents the probability that x is within one standard deviation of the mean of a normal

distribution with a standard deviation equal to one. The normal distribution is often called the bell-shaped curve. Output from **trapsimp** is

>> trapsimp('fct1',-1,1)

n	Trap	Simp
2	6.4091e-001	6.9324e-001
4	6.7252e-001	6.8306e-001
8	6.8016e-001	6.8271e-001
16	6.8206e-001	6.8269e-001
32	6.8253e-001	6.8269e-001
64	6.8265e-001	6.8269e-001
128	6.8268e-001	6.8269e-001
256	6.8269e-001	6.8269e-001
512	6.8269e-001	6.8269e-001
1024	6.8269e-001	6.8269e-001

The values suggest that the integral is approximately 0.68269. Correct to ten decimal places, $I \approx 0.68268\,94921$. By observing how quickly the values for **Trap** and **Simp** approach the displayed 0.68269 it appears that Simpson's Rule is the better choice for this integral.

8.6 Open Quadrature Rules

The composite Trapezoidal and Simpson's Rules provide simple and economical algorithms to approximate definite integrals. Both are examples of *closed* Newton-Cotes quadrature formulas in that the endpoint values, $f(a)$ and $f(b)$, are part of the results. So-called *open* quadrature formulas avoid endpoint values.

For example, a simple midpoint formula approximates the integrand with a constant leading to a *rectangular* rule. In other words the integral $I = \int_a^b f(x)dx$ may be approximated by the simple formula

$$M_1 = (b-a)f(\frac{1}{2}(a+b)). \tag{8.37}$$

that represents the area of a rectangle with base, $b-a$, and height, $f(\frac{1}{2}(a+b))$. Generalizing this result to n subintervals is left to the problems at the end of the chapter.

The Trapezoidal Rule, (8.28), assumes that the interpolation points for $P_1(x)$ have been located at the ends of an interval. An open trapezoidal rule may be derived by selecting the interpolation points interior to the interval $[\,a,\,b\,]$. Recall Figure 8.1. One standard open formula is based on subdividing the interval into thirds. With $h = \frac{1}{3}(b-a)$, the points $x_1 = a+h$ and $x_2 = a+2h$ will provide the data for $P_1(x)$. The quadrature formula will resemble the Trapezoidal Rule, (8.28), and is given by

$$\frac{1}{2}(b-a)\Big[f(a+h)+f(a+2h)\Big] \tag{8.38}$$

Advanced texts in numerical analysis will provide further details for open Newton-Cotes quadrature formulas. The open formulas are often used to estimate values of convergent improper integrals where an endpoint function value is undefined.

8.7 Error Analysis for the Trapezoidal Rule

The error expression for interpolating polynomials, (6.19), provides the starting point for error analysis of many quadrature formulas. Including the error term for $P_1(x)$, see (6.20), gives

$$I = \int_a^b f(x)dx = \int_a^b \Big[P_1(x) + \frac{(x-a)(x-b)}{2} f''(c)\Big]dx$$

or

$$I = T_1 + \int_a^b \frac{(x-a)(x-b)}{2} f''(c)dx. \tag{8.39}$$

The integral in (8.39) is confounded by the unknown value of c in the second derivative where c depends on the integration variable x. Using the so-called *Weighted Mean Value Theorem* for definite integrals leads to

$$I - T_1 = E_1^T = f''(\eta)\int_a^b \frac{(x-a)(x-b)}{2}dx$$

where η is now an unknown number in the interval $[\,a,\,b\,]$. A simple integration

```
>> syms a b x
>> factor(int((x-a)*(x-b)/2,a,b))
ans =
1/12*(a-b)^3
```

gives the standard error formula for the Trapezoidal Rule

$$E_1^T = -\frac{(b-a)^3}{12} f''(\eta) = -\frac{h^3}{12} f''(\eta). \tag{8.40}$$

It is a simple task to show that the error formula for the composite Trapezoidal Rule with n subintervals is

$$E_n^T = -\sum_{i=1}^{n} \frac{h^3}{12} f''(\eta_i), \tag{8.41}$$

where $x_i \le \eta_i \le x_{i+1}$ and $h = (b-a)/n$.

Using the Intermediate Value Theorem, (8.41) may be written as $E_n^T = -\frac{h^3}{12} n f''(\lambda)$, where λ is yet another unknown number in $[\,a,\,b\,]$. With $h = (b-a)/n$, the error formula may be structured to stress h (the length of a subinterval) or n (the number of subintervals). In other words,

$$E_n^T = -\frac{h^2}{12}(b-a)f''(\lambda) = -\frac{(b-a)^3}{12n^2}f''(\lambda). \tag{8.42}$$

The composite error formula shows that $\lim_{n\to\infty} E_n^T = 0$, so that T_n will converge to I provided f'' is a bounded function on $[\,a,\,b\,]$. Note also that the error depends on the concavity of the integrand as measured by the second derivative.

If a quadrature formula integrates an m^{th} degree polynomial exactly, we say that the formula has a *precision* of m. Since the error formula (8.42) contains the second derivative, the precision of the Trapezoidal Rule is one.

To illustrate the use of (8.42) recall the example in Section 8.5, $I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\,dx$. Suppose we wish to approximate I with the composite Trapezoidal Rule and at the same time ask that the error be less than a prescribed tolerance, TOL. In other words, we require that $|E^T| = \frac{(b-a)^3}{12n^2}|f''(\lambda)| \le$ TOL. Since λ is unknown, we are faced with a similar problem encountered in the error analysis of Taylor polynomials – determine the maximum value of $|f''(\lambda)|$ on the interval $[\,a,\,b\,]$. In other words, solve the following inequality for n.

$$\frac{(b-a)^3}{12n^2} \max_{[\,a,b\,]} |\,f''(\lambda)\,| \le \text{TOL} \tag{8.43}$$

In many cases a graphical approach will provide satisfactory information about the maximum value. The following MATLAB commands will plot the absolute value of the second derivative of $e^{-\frac{1}{2}x^2}$ on the interval $[\,-1, 1\,]$.

```
>> syms x
>> ezplot(abs(diff(exp(-x*x/2),2)),[-1,1])
```

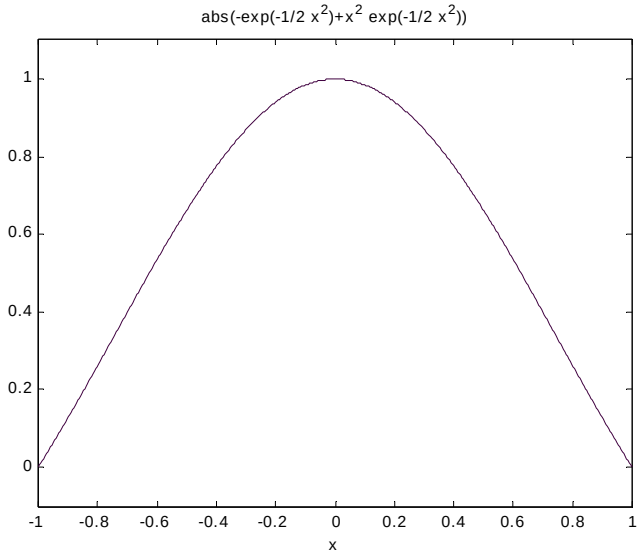

Figure 8.2 Graphical Maximum

Based on the graph, the absolute value of $(e^{-\frac{1}{2}x^2})''$ has a maximum value of 1 at $x = 0$. Substituting into (8.43) we find

$$\frac{(1-(-1))^3}{12n^2}\ \frac{1}{\sqrt{2\pi}}\ 1 \leq \text{TOL}.$$

With TOL $= 0.0001$, solution of the inequality $\frac{8}{12n^2\sqrt{2\pi}} \leq 0.0001$ is $n \geq 52$. Since a maximum value was used in the analysis, the prediction of 52 intervals is actually larger than is actually necessary. Using the maximum value of $|f''|$ is an inherently conservative approach; however, the procedure will guarantee that $|I - T_n| \leq$ TOL. Using the Trapezoidal Rule data from Section 8.5 with error values appended we see that 32 intervals comes very close to meeting the specified error tolerance; whereas, 64 intervals is more than necessary.

n	**Trap**	**Simp**	$I - T_n$
16	**6.8206e-001**	**6.8269e-001**	0.00062
32	**6.8253e-001**	**6.8269e-001**	0.00015
64	**6.8265e-001**	**6.8269e-001**	0.00003

Since trapsimp.m does not provide all values of n we may only estimate that an n somewhat greater than 32 and less than 64 will be satisfactory. The result of our analysis, 52 intervals seems reasonable.

Asymptotic Error Formula for the Trapezoidal Rule

As an alternative to (8.43), we return to the error summation given in (8.41). $E_n^T = -\sum_{i=1}^{n} \frac{h^3}{12} f''(\eta_i)$ or $E_n^T = -\frac{h^2}{12} \sum_{i=1}^{n} f''(\eta_i)h$. The Riemann sum in the last expression may be approximated by a definite integral, leading to the *asymptotic* (large n) error formula for the composite Trapezoidal Rule as follows:

$$E_n^T \approx -\frac{h^2}{12} \int_a^b f''(x)dx = -\frac{h^2}{12}\Big[f'(b) - f'(a)\Big]. \tag{8.44}$$

In many cases the asymptotic error formula will provide a more realistic estimate of the number of intervals needed to meet an error tolerance. The asymptotic result leads to the inequality

$$\frac{(b-a)^2}{12n^2} |\, f'(b) - f'(a)\,| \leq \text{TOL} \tag{8.45}$$

For our example, the asymptotic inequality (8.45) results in

$$\frac{2^2}{12n^2} |f'(1) - f'(-1)| \leq 0.0001.$$

A simple computation will show that n should be larger than 41. This value seems consistent with the numerical results shown above.

Outcomes from the Error Formula

Computing estimates of the number of intervals needed to reach an error tolerance is not the only application of the Trapezoidal error formula. In Section 8.4 we noted a computational advantage obtained by doubling the number of intervals. Using (8.42) we may show

$$\frac{E_n^T}{E_{2n}^T} = \frac{I - T_n}{I - T_{2n}} = 4\frac{f''(\lambda)}{f''(\mu)}.$$

As n becomes large this ratio should approach four. In other words, doubling the number of subintervals should cause the error to drop by a factor of four, approximately. Provided n is not so large that the accumulated error becomes significant, a similar result holds for the computed estimates

$$\frac{I - \widehat{T}_n}{I - \widehat{T}_{2n}} \approx 4. \tag{8.46}$$

The expression in (8.46) provides two interesting and important results. Using $I - \widehat{T}_n \approx 4(I - \widehat{T}_{2n})$ we may rearrange terms to give

$$I - \widehat{T}_{2n} \approx \frac{1}{3}(\widehat{T}_{2n} - \widehat{T}_n). \tag{8.47}$$

In other words, an estimate for the error in $\widehat{T}_{2n}$ may be determined by calculating $\frac{1}{3}(\widehat{T}_{2n} - \widehat{T}_n)$. The ability to estimate errors is an important aspect of quadrature programs. Using the theoretical error formula, (8.42), is not easy; however, the error estimate, (8.47), is simple to compute. A quadrature program implementing the Trapezoidal Rule can be terminated when the estimated error is less than some prescribed tolerance, i.e., $\frac{1}{3}(\widehat{T}_{2n} - \widehat{T}_n) \leq$ TOL.

Solving the relation $I - \widehat{T}_n \approx 4(I - \widehat{T}_{2n})$ for I gives the second result.

$$I \approx \frac{1}{3}(4\widehat{T}_{2n} - \widehat{T}_n). \tag{8.48}$$

The right hand side is often denoted $\widehat{R}_{2n}$ and is called a *Richardson* extrapolation formula. Typically, the numerical value of $\widehat{R}_{2n} = \frac{1}{3}(4\widehat{T}_{2n} - \widehat{T}_n)$ provides a more accurate estimate for I than $\widehat{T}_{2n}$ itself. The following data from a MATLAB program similar to trapsimp.m provides a wealth of information about our example integral, $I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$.

>> trap('fct1',-1,1,true)

n	Trap	Err Ratio	Error	Est Error	Rich	RichErr
2	6.409e-001	4.757e+000	4.178e-002	5.232e-002	6.932e-001	-1.055e-002
4	6.725e-001	4.109e+000	1.017e-002	1.054e-002	6.831e-001	-3.686e-004
8	6.802e-001	4.026e+000	2.526e-003	2.547e-003	6.827e-001	-2.148e-005
16	6.821e-001	4.006e+000	6.305e-004	6.318e-004	6.827e-001	-1.320e-006
32	6.825e-001	4.002e+000	1.576e-004	1.576e-004	6.827e-001	-8.220e-008
64	6.827e-001	4.000e+000	3.938e-005	3.939e-005	6.827e-001	-5.167e-009
128	6.827e-001	4.000e+000	9.846e-006	9.846e-006	6.827e-001	-3.576e-010

256	**6.827e-001**	**4.000e+000**	**2.461e-006**	**2.461e-006**	**6.827e-001**	**-5.712e-011**
512	**6.827e-001**	**4.000e+000**	**6.153e-007**	**6.154e-007**	**6.827e-001**	**-3.834e-011**
1024	**6.827e-001**	**4.001e+000**	**1.538e-007**	**1.538e-007**	**6.827e-001**	**-3.716e-011**

The output reveals that the error ratio is approximately four and that the estimated error is very close to the actual error with results improving as n increases. Using 16 subintervals as an example, we see that the error in the Richardson formula, $|I - \widehat{R}_{16}| = 1.32 \cdot 10^{-6}$, is smaller than the error in the corresponding Trapezoidal Rule, $|I - \widehat{T}_{16}| = 6.305 \cdot 10^{-4}$. Since the error $|I - \widehat{T}_{256}| = 2.461 \cdot 10^{-6}$ we observe that $\widehat{R}_{16}$ (17 integrand evaluations) corresponds to $\widehat{T}_{256}$ (257 integrand evaluations). This is a rather dramatic savings in the number of function evaluations needed to compute the integral.

Summary of Results for Simpson's Rule

All of the error analysis presented earlier in this section may be repeated, with some complications, for Simpson's Rule. The results are summarized below.

Standard error formula:

$$E_2^S = -\frac{(b-a)^5}{2880} f^{(4)}(\eta) = -\frac{h^5}{90} f^{(4)}(\eta), \text{ where } h = (b-a)/2 \tag{8.49}$$

Composite error formula (n is even):

$$E_n^S = -\frac{h^4(b-a)}{180} f^{(4)}(\lambda) = -\frac{(b-a)^5}{180n^4} f^{(4)}(\lambda), \text{ where } h = (b-a)/n \tag{8.50}$$

Asymptotic error formula:

$$E_n^S \approx -\frac{h^4}{180}\left[f^{(3)}(b) - f^{(3)}(a)\right] \tag{8.51}$$

Error Ratio:

$$\frac{E_n^S}{E_{2n}^S} = \frac{I - S_n}{I - S_{2n}} = 16\frac{f^{(4)}(\lambda)}{f^{(4)}(\mu)} \quad \text{and} \quad \frac{I - \widehat{S}_n}{I - \widehat{S}_{2n}} \approx 16 \tag{8.52}$$

Error estimate:

$$I - \widehat{S}_{2n} \approx \frac{1}{15}(\widehat{S}_{2n} - \widehat{S}_n) \tag{8.53}$$

Richardson extrapolation:

$$I \approx \frac{1}{15}(16\widehat{S}_{2n} - \widehat{S}_n) \tag{8.54}$$

You may wish to compare the corresponding formulas for both the Trapezoidal and Simpson's Rule side by side. For example, since the error in Simpson's Rule depends on the fourth derivative, the precision will be three.

To illustrate some of the preceding formulas, let's return to $I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$. Suppose we wish to approximate I with the composite Simpson's Rule and at the same time ask that the error be less than a prescribed tolerance, TOL $= 10^{-6}$. In other words, we require

$$| E_n^S | = | \frac{(b-a)^5}{180n^4} f^{(4)}(\lambda) | \leq 10^{-6} \tag{8.55}$$

The following MATLAB commands will plot the absolute value of the fourth derivative of $e^{-\frac{1}{2}x^2}$ on the interval $[-1, 1]$ showing a maximum value of three.

```
>> syms x
>> f = exp(-x*x/2);
>> ezplot(abs(diff(f,4)),[-1,1]), grid on
```

Substituting into (8.55) gives

$$\frac{(1-(-1))^5}{180n^4} \ \frac{1}{\sqrt{2\pi}}\, 3 \leq 10^{-6} \ \text{ or } \ n \geq 22.$$

The following MATLAB output shows $n, S_n,\ E_n^S$ and the estimated error from (8.53). The data shows that the predicted value of at least 22 intervals is in the ballpark. Note also that as n increases the estimated error is close to the true error.

```
>> true = .6826894921;
>> simp('fct1',-1,1,true)
```

n	Simp	Error	Est Error
2	6.9324e-001	-1.0547e-002	4.6216e-002
4	6.8306e-001	-3.6861e-004	-6.7858e-004
8	6.8271e-001	-2.1484e-005	-2.3142e-005
16	6.8269e-001	-1.3202e-006	-1.3442e-006
32	6.8269e-001	-8.2200e-008	-8.2532e-008

8.8 Adaptive Quadrature and MATLAB

MATLAB's command for numerical integration, **quad**, is based on an "adaptive recursive Simpson's Rule." The use is as follows:

```
>> I = quad('fct1',a,b,TOL)
```

where the integrand is in the M-file fct1.m. If an integration tolerance is omitted the default is 0.001. See >> **help quad**. The procedures presented in Sections 8.4 and 8.5 simply increase the number of subintervals, uniformly, in $[\,a,\,b\,]$ without considering the nature of the integrand. An *adaptive* quadrature scheme will adjust the number of subintervals on various parts of $[\,a,\,b\,]$ depending on the behavior of the integrand.

Any quadrature method may be used in an adaptive procedure; however, computer implementation requires the capability to estimate the integration error as the number of subintervals is increased. Since **quad** uses Simpson's Rule, the error estimate, (8.53), $I - \widehat{S}_{2n} \approx \frac{1}{15}(\widehat{S}_{2n} - \widehat{S}_n)$ is important. In particular, with $n = 2$, we may estimate the error in $\widehat{S}_4$, $I - \widehat{S}_4$ with $\frac{1}{15}(\widehat{S}_4 - \widehat{S}_2)$. This formula, using two and four subintervals, is a major factor in an adaptive scheme.

Simpson's Rule two interval template is $\frac{1}{3}h\Big[f(a) + 4f(c) + f(b)\Big]$. This is the basic building block in an adaptive scheme. We begin by denoting $\widehat{S}_2$ on the interval $[a, b]$ by $Q(a, b, 2)$ where the arguments of Q are the limits of integration and the number of subintervals. $Q(a, b, 2)$ is one application of the basic Simpson's Rule. Doubling the intervals to four, we apply Simpson's Rule once on the interval $[a, \frac{a+b}{2}]$ and again on the interval $[\frac{a+b}{2}, b]$. Adding the results together gives the same result as the composite Simpson's Rule with 4 subintervals. The result is $\widehat{S}_4 = Q(a, \frac{a+b}{2}, 2) + Q(\frac{a+b}{2}, b, 2)$. To keep track of the interval and number of subintervals, denote this result by $Q(a, b, 4)$. Using TOL as our integration tolerance, we test the accuracy of $Q(a, b, 4)$ as follows:

$$\text{If } |\,Q(a, b, 4) - Q(a, b, 2)\,| \le 15 \cdot \text{TOL, accept } I \approx Q(a, b, 4). \tag{8.56}$$

If the inequality is not satisfied, we subdivide and test again. The subdivision will result in four, two subinterval Simpson's Rules as follows:

$$Q(a, \tfrac{a+b}{4}, 2) \qquad Q(\tfrac{a+b}{4}, \tfrac{a+b}{2}, 2) \qquad Q(\tfrac{a+b}{2}, \tfrac{3(a+b)}{4}, 2) \qquad Q(\tfrac{3(a+b)}{4}, b, 2)$$

Adding the first and second terms and the third and fourth terms gives

$$Q(a, \frac{a+b}{2}, 4) = Q(a, \frac{a+b}{4}, 2) + Q(\frac{a+b}{4}, \frac{a+b}{2}, 2)$$

$$Q(\frac{a+b}{2}, b, 4) = Q(\frac{a+b}{2}, \frac{3(a+b)}{4}, 2) + Q(\frac{3(a+b)}{4}, b, 2)$$

Assuming that the error is uniformly distributed over the entire interval $[\,a,\,b\,]$, we use TOL$/2$ in the following tests.

$$\text{If } |Q(a, \frac{a+b}{2}, 4) - Q(a, \frac{a+b}{2}, 2)| \le 15 \cdot \text{TOL}/2, \int_a^{\frac{a+b}{2}} f(x)dx \approx Q(a, \frac{a+b}{2}, 4). \tag{8.57}$$

$$\text{If } |Q(\frac{a+b}{2}, b, 4) - Q(\frac{a+b}{2}, b, 2)| \le 15 \cdot \text{TOL}/2, \int_{\frac{a+b}{2}}^{b} f(x)dx \approx Q(\frac{a+b}{2}, b, 4). \tag{8.58}$$

An example will clarify the adaptive procedure. Let $f(x) = 2e^{-5x^2} + 0.5x$ on the interval $0 \le x \le 2$ and suppose an error tolerance of 0.001 is specified for the integral $\int_0^2 f(x)dx$. A graph of $f(x)$, see Figure 8.3, shows that the integrand changes rapidly on the first half of the interval and is close to a line on the second half. As a first step the M-file trapsimp.m in Section 8.5 may be used to compute the Simpson's Rule values for $n = 2$ and 4 subintervals.

>> trapsimp('fct1',0,2)

n	Trap	Simp
2	2.0135e+000	1.6846e+000
4	1.7933e+000	1.7198e+000

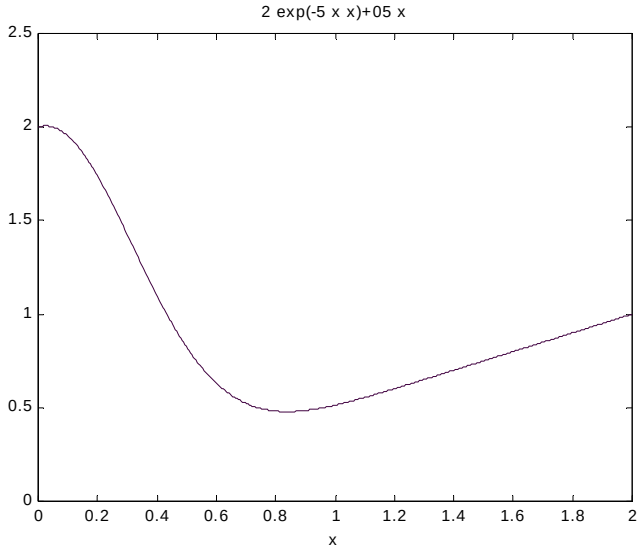

Figure 8.3 The Integrand $f(x) = 2e^{-5x^2} + 0.5x$

Using the expression in (8.56) we may compare the Simpson's Rule data as follows:

$$| Q(0,2,4) - Q(0,2,2) | = | 1.7198 - 1.6846 | = 0.0352.$$

Since 0.0352 is not less than $15 \cdot 0.001 = 0.015$, subdivide the interval $[\,0,\ 2\,]$ and try again. The intervals to be examined are now $[\ 0, 1]$ and $[\ 1, 2\ ]$. Consider $[\,1,\ 2\,]$ first and use the expression in (8.58). We find

>> trapsimp('fct1',1,2)

n	Trap	Simp
2	7.5338e-001	7.5226e-001
4	7.5189e-001	7.5140e-001

$$| Q(1,2,4) - Q(1,2,2) | = | 0.75140 - 0.75226 | = 0.00086.$$

Since this is less than $15 \cdot \text{TOL}/2 = 0.0075$, we accept the approximation $Q(1, 2, 4)$ as within tolerance for the value of the integral $\int_1^2 f(x)dx$, in other words

$$\int_1^2 f(x)dx \approx Q(1, 2, 4).$$

Using (8.57), consider $[\,0,\ 1\,]$ next.

>> trapsimp('fct1',0,1)

n	Trap	Simp
2	**1.0399e+000**	**9.6759e-001**
4	**1.0408e+000**	**1.0411e+000**

$$|Q(0, 1, 4) - Q(0, 1, 2)| = |\,1.0411 - 0.96759\,| = 0.07351.$$

This is not less than 0.0075, so we need to subdivide the interval $[\,0,\ 1\,]$ once more. Consider the interval $[\,0, 0.5\,]$ and the new tolerance $15 \cdot \text{TOL}/4 = 0.00375$.

>> trapsimp('fct1',0,.5)

n	Trap	Simp
2	**7.4993e-001**	**7.6466e-001**
4	**7.6119e-001**	**7.6494e-001**

$$|\,Q(0, 0.5, 4) - Q(0, 0.5, 2)\,| = |\,0.76494 - 0.76466\,| = 0.00028.$$

This is within the prescribed tolerance, so we accept $Q(0, .5, 4)$ for $\int_0^{0.5} f(x)dx$. Finally consider the interval $[\,0.5, 1\,]$ with the same error tolerance 0.00375. Here we find

>> trapsimp('fct1',.5,1)

n	Trap	Simp
2	**2.9084e-001**	**2.7641e-001**
4	**2.8006e-001**	**2.7647e-001**

$$|\,Q(0.5, 1, 4) - Q(0.5, 1, 2)\,| = |\,0.27647 - 0.27641\,| = 0.00006.$$

This is also within tolerance and we accept the value of $Q(0.5, 1, 4)$ for $\int_{0.5}^1 f(x)dx$. The final approximation of the integral $\int_0^2 f(x)dx$ is given by the sum of three terms

$$\int_0^{0.5} f(x)dx + \int_{0.5}^1 f(x)dx + \int_1^2 f(x)dx =$$

$$Q(0, 0.5, 4) + Q(0.5, 1, 4) + Q(1, 2, 4) =$$

$$0.76494 + 0.27647\ + 0.75140 = 1.79281.$$

The three integrals require that f be evaluated at the nodes $\{0,\ 0.125, 0\,.25, 0\,.375,\ 0.5, 0\,.625,\ 0.75,\ 0.875,\ 1,\ 1.25,\ 1.5,\ 1.75,\ 2\}$. Observe that there are fewer function evaluations between 1 and 2 than between 0 and 1. The quadrature procedure has adapted to the way in which the

integrand changes. Figure 8.4 shows where the integrand has been evaluated. Nodes are shown as asterisks at the bottom of the figure.

Recall that the error in estimating the integral with $\widehat{S}_4$ is approximately $\frac{1}{15}(\widehat{S}_4 - \widehat{S}_2)$. Although we accepted the estimate if $\frac{1}{15}(\widehat{S}_4 - \widehat{S}_2)$ was less than the tolerance, an adaptive quadrature routine using Simpson's Rule may use a more conservative decision rule obtained by replacing the theoretical value of 15 with a smaller value, say 10. The rule becomes: accept when $\left|\widehat{S}_4 - \widehat{S}_2\right| < 10 \cdot$ TOL. This may necessitate extra subdivisions but should provide more assurance of the desired tolerance level.

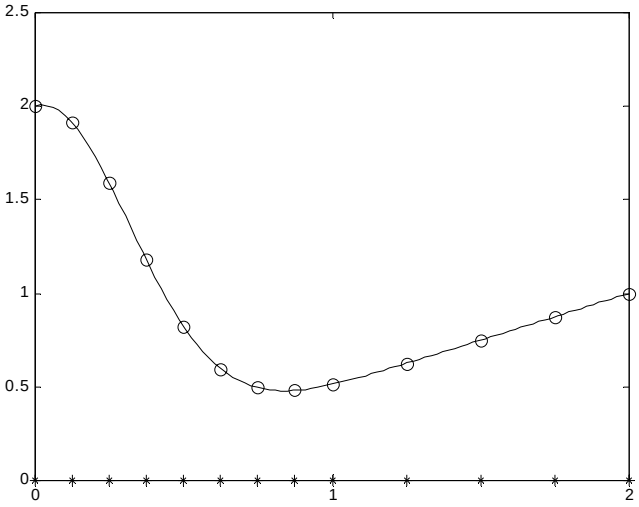

Figure 8.4 Adaptive Integration Points

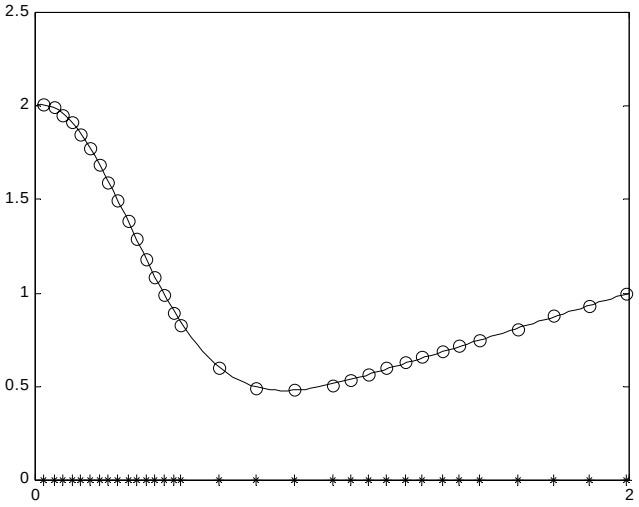

Figure 8.5 Integration Points Used by **quad**

MATLAB's **quad** is a more sophisticated adaptive algorithm. The numerical result for our example is given by

```
>> quad('fct',0,2,.001)
ans =
  1.7926e+000
```

An optional fifth argument will provide information on integrand evaluations. See >> **help quad**. The 32 integration points used for a tolerance of 0.001 are shown in Figure 8.5. Again, integration nodes are shown at the bottom of the figure.

8.9 Gaussian Quadrature

Recall that a quadrature scheme has precision n if it integrates polynomials of degree n exactly. In Sections 8.4 and 8.5 the precision's of the Trapezoidal Rule and Simpson's Rule, one and three respectively, were outcomes of the error analysis. Gaussian quadrature formulas are derived with exact integration of polynomials as the guiding principle. We approximate integrals of the form $\int_{-1}^{1} f(x)dx$ with an n-point Gaussian formula G_n,

$$\int_{-1}^{1} f(x)dx \approx G_n = \sum_{i=1}^{n} w_i f(\xi_i), \tag{8.59}$$

where the w_i's (weights) and ξ_i's (x values in the interval $[-1, 1]$) are found by requiring that G_n be exact if $f(x)$ is a polynomial of some specified degree.

We begin with a one point formula, G_1,

$$\int_{-1}^{1} f(x)dx \approx G_1 = w_1 f(\xi_1).$$

Since there are two unknown parameters in G_1, w_1 and ξ_1, and two coefficients in a linear polynomial, it seems plausible that G_1may have a precision of one. With $f(x) = mx + b$, where m and b may be any values, we attempt to solve the problem $I = G_1$ or

$$\int_{-1}^{1} (mx + b)dx = w_1(m\xi_1 + b).$$

Integrating and distributing w_1 gives $2b = w_1 m\xi_1 + w_1 b$. This equation must hold for all arbitrary values of m and b; thus, we equate corresponding terms on both sides.

$$\begin{aligned} m \text{ terms:} \quad & 0 = w_1\xi_1 \\ b \text{ terms:} \quad & 2 = w_1 \end{aligned}$$

We observe that $w_1 = 2$ and $\xi_1 = 0$ so that the one point approximation is

$$\int_{-1}^{1} f(x)dx \approx G_1 = 2f(0). \tag{8.60}$$

For example, $\int_{-1}^{1} e^x dx \approx G_1 = 2e^0 = 2$ compared to the exact value $e - e^{-1} \approx 2.35$.

A two point formula, G_2, is the next step.

$$\int_{-1}^{1} f(x)dx \approx G_2 = w_1 f(\xi_1) + w_2 f(\xi_2).$$

There are four unknown parameters in G_2 and four coefficients in a cubic polynomial; again, it seems plausible that G_2 may have a precision of three. Replacing $f(x)$ by the cubic polynomial $ax^3 + bx^2 + cx + d$ gives

$$\int_{-1}^{1} (ax^3 + bx^2 + cx + d)dx = w_1(a\xi_1^3 + b\xi_1^2 + c\xi_1 + d) + w_2(a\xi_2^3 + b\xi_2^2 + c\xi_2 + d).$$

Integrating

```
>> syms a b c d x
>> int(a*x^3+b*x^2+c*x+d,-1,1)
ans =
2/3*b+2*d
```

leads to

$$0 \cdot a + \frac{2}{3} \cdot b + 0 \cdot c + 2 \cdot d = w_1(a\xi_1^3 + b\xi_1^2 + c\xi_1 + d) + w_2(a\xi_2^3 + b\xi_2^2 + c\xi_2 + d).$$

As above, this equation must be true for arbitrary values of a, b, c and d resulting in four equations.

$$\begin{array}{ll} a \text{ terms:} & 0 = w_1\xi_1^3 + w_2\xi_2^3 \\ b \text{ terms:} & \frac{2}{3} = w_1\xi_1^2 + w_2\xi_2^2 \\ c \text{ terms:} & 0 = w_1\xi_1 + w_2\xi_2 \\ d \text{ terms:} & 2 = w_1 + w_2 \end{array} \tag{8.61}$$

Equation (8.61) is a *nonlinear* system. Our efforts in Chapter 5 have not prepared us for this situation. There are numerical methods to solve nonlinear systems; however, they are beyond the scope of this text. In this instance we shall rely on our intuition and trial and error. The last line in (8.61), the d terms, indicates that the sum of the weights is two. It seems reasonable that both terms in G_2 should be weighted equally so we set $w_1 = 1$ and $w_2 = 1$. In turn, the third line, c terms, will give $\xi_2 = -\xi_1$ which shows $\frac{2}{3} = 2\xi_1^2$ or $\xi_1 = \sqrt{3}/3$. Using these values the two point formula becomes

$$\int_{-1}^{1} f(x)dx \approx G_2 = 1f(\sqrt{3}/3) + 1f(-\sqrt{3}/3). \tag{8.62}$$

Data for Gaussian quadrature formulas is tabulated in various handbooks. For example, the three point formula is

$$\int_{-1}^{1} f(x)dx \approx G_3 = \frac{5}{9}f(\sqrt{(3/5)}) + \frac{8}{9}f(0) + \frac{5}{9}f(-\sqrt{(3/5)}). \tag{8.63}$$

Usually, definite integrals are not in the standard Gaussian form $\int_{-1}^{1} f(x)dx$. In order to integrate $\int_a^b f(x)dx$ using the Gaussian formulas it will be necessary to make a change of variable so that the limits of integration are changed. Assuming that $x = \alpha t + \beta$, we require that $x = a$ correspond to $t = -1$ and $x = b$ correspond to $t = 1$. With these requirements $\alpha = (b-a)/2$ and $\beta = (a+b)/2$. Note also that $dx = \alpha dt$. In other words

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}t + \frac{a+b}{2})dt.$$

In general, the n-point Gaussian quadrature formula has the form

$$\int_a^b f(x)dx \approx G_n = \frac{b-a}{2} \cdot \sum_{i=1}^{n} w_i f(\frac{b-a}{2}\xi_i + \frac{a+b}{2}). \tag{8.64}$$

As an example, suppose we wish to approximate the integral $\int_1^5 \ln(x)dx = 5\ln(5) - 4 \approx 4.04719$ with Gaussian quadrature. Transforming the limits of integration with $x = 2t + 3$ gives

$$\int_1^5 \ln(x)dx = 2 \cdot \int_{-1}^{1} \ln(2t+3)dt.$$

Using the Gaussian formulas (8.60), (8.62) and (8.63) we find

$$\begin{aligned}
G_1 &= 2 \cdot \Big[2 \cdot \ln(3)\Big] && \approx 4.39445 \\
G_2 &= 2 \cdot \Big[1 \cdot \ln(2\tfrac{\sqrt{3}}{3} + 3) + 1 \cdot \ln(-2\tfrac{\sqrt{3}}{3} + 3\Big] && \approx 4.07375 \\
G_3 &= 2 \cdot \Big[\tfrac{5}{9} \cdot \ln(2\sqrt{\tfrac{3}{5}} + 3) + \tfrac{8}{9} \cdot \ln(3) + \tfrac{5}{9} \cdot \ln(-2\sqrt{\tfrac{3}{5}} + 3)\Big] && \approx 4.04983
\end{aligned}$$

As the number of points increases the accuracy of the Gaussian formulas improves. Note the economy of the procedure. For this example only three integrand evaluations results in two decimal places of accuracy. As a result of their simple form Gaussian quadrature formulas are often embedded in more advanced numerical methods that require integration of polynomials. One common application is the *finite element* method used to solve partial differential equations.

8.10 Integration of Numerical Data

The quadrature methods presented in this chapter have been based on the known functional form of the integrand $f(x)$. The Trapezoidal Rule, Simpson's Rule and Gaussian Quadrature are all based on evaluating the integrand at specific values. Given a set of $n+1$ data points, (x_i, y_i), describing the integrand, these formulas may or may not be applicable.

The composite Trapezoidal rule was based on

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx.$$

where $x_1 = a$ and $x_{n+1} = b$. Using the data to construct a piece wise linear model for the integrand leads to the approximation of the integrals as follows:

$$\sum_{i=1}^{n} \frac{1}{2}(x_{i+1} - x_i)(y_i + y_{i+1}). \tag{8.65}$$

You should recognize $\frac{1}{2}(x_{i+1} - x_i)(y_i + y_{i+1})$ as the area of a trapezoid. If the x_i's are equally spaced and the number of intervals is even, the composite Simpson Rule, (8.35), provides an approximation of the integral. Function values are replaced by the data values to give

$$\sum_{i=1}^{n/2} \frac{1}{3}h\Big[y_{2i-1} + 4y_{2i} + y_{2i+1}\Big].$$

Our previous work with interpolating polynomials, splines and least squares models have demonstrated various procedures to model data with continuous functions. Integration of these continuous functions may be used to approximate a definite integral.

For example, consider the data of Section 6.2 and the interpolating polynomial (6.18). We may approximate the definite integral specified by the data with $\int_2^6 P_4(x)dx$. Keep in mind that the antiderivative of a fourth degree polynomial will be a fifth degree polynomial. We need only to compute the coefficients of the fifth degree polynomial. See **c4int** in the following MATLAB code. Note that the constant of integration has been set to zero.

```
c4 =
 -7.1857e-002  1.0941e+000 -5.9312e+000  1.3783e+001 -9.8221e+000
>> m=length(c4);
>> for j = 1:m
c4int(j) = c4(j)/(m+1-j);
end
>> c4int(m+1) = 0;
>> c4int
c4int =
 -1.4371e-002  2.7354e-001 -1.9771e+000  6.8917e+000 -9.8221e+000            0
>> polyval(c4int,6)-polyval(c4int,2)  %The Fundamental Theorem
ans =
  8.8452e+000
```

The value for **ans** provides a reasonable approximation for the area beneath $P_4(x)$. See Figure 6.1. Since the actual integrand is unknown, the accuracy of the approximation is also unknown.

In Chapter 6 cubic interpolating splines eliminated the oscillatory problems associated with some interpolating polynomials. It seems plausible that integrating a spline may provide a good numerical approximation. The key equation is

$$S_j(x) = a_j(x - x_j)^3 + b_j(x - x_j)^2 + c_j(x - x_j) + d_j,\ x_j \le x\ \le\ x_{j+1}. \qquad (6.25)$$

Integrating (6.25) gives

$$\int_{x_j}^{x_{j+1}} S_j(x)dx = \frac{1}{4}a_j(h_j)^4 + \frac{1}{3}b_j(h_j)^3 + \frac{1}{2}c_j(h_j)^2 + d_j h_j. \qquad (8.66)$$

where $h_j = x_{j+1} - x_j$.

As we can see, the spline coefficients for the j^{th} subinterval and the length of the subinterval, h_j, are the important factors in the integration. Summing over each subinterval, our composite approximation is

$$\int_a^b S(x)dx = \sum_{j=1}^{n}\Big[\frac{1}{4}a_j(h_j)^4 + \frac{1}{3}b_j(h_j)^3 + \frac{1}{2}c_j(h_j)^2 + d_j h_j\Big]. \qquad (8.67)$$

To compute (8.67) we need to extract the spline coefficients from **spline(x,y)**, determine the lengths of the n intervals and evaluate the summation.

Extracting the coefficients requires the MATLAB commands

```
>> spdata = spline(x,y);
>> [breaks,spcoef,pieces,order,dim] = unmkpp(spdata);
```

Recall that the a_j's are in the first column of **spcoef**, the b_j's in the second column and so forth.

MATLAB's **diff(*x*)** is $[x_2 - x_1, x_3 - x_2, \ldots, x_{n+1} - x_n] = [h_1, h_2, \ldots, h_n]$. In other words, the vector ***h*** contains the lengths of the n subintervals.

```
>> h = diff(x);
```

Equation (8.67) may be evaluated with careful use of array operations. For example, the n products in the last term, $d_i h_i$, are contained in the vector **spcoef(:,4).*h**.

Including all terms, the integral of the spline is given by

```
>> sum(spcoef(:,1).*h.^4/4+spcoef(:,2).*h.^3/3+spcoef(:,3).*h.^2/2 ...
+spcoef(:,4).*h)
```

8.11 Problems

8-1. Verify $P_2''(x) = \frac{y_1 - 2y_2 + y_3}{h^2}$, (8.10), by computing the second derivative of $P_2(x) = l_1(x)y_1 + l_2(x)y_2 + l_3(x)y_3$ with $x_1 = \overline{x},\ x_2 = \overline{x} + h$, and $x_3 = \overline{x} + 2h$.

8-2. Consider the first derivative of the following functions at $x = 1$. Use the numerical formulas at the end of Section 8.2 to approximate the first derivative using $h = 0.1,\ 0.05$ and 0.025. Compute the errors in your approximations using the exact value of the

derivative. Display the results, derivatives and errors, in table form. Plot the functions near $x = 1$ to see if your answers are reasonable.

a. $f(x) = e^{-2x}$
b. $f(x) = \sin(\sqrt{x})$. Use radians.
c. $f(x) = x \sin(2x)$. Use radians.

8-3. Recall that $f(x) = |x|$ is not differentiable at zero.
a. Compute the (first) forward difference at zero using $h = \pm 0.1,\ \pm 0.01,\ \pm 0.001$.
b. Compute the central difference for $h = 0.1, 0.01, 0.001$.
c. Give an explanation of your results based on the slope of the graph of f.
d. What conclusion(s) can you draw from the data you have generated.

8-4. Use a method of your choice, Lagrange, Taylor or undetermined coefficients, to find a backward difference approximation to the second derivative of $f(x)$ at x_i.

8-5. Consider the second derivative of $f(x) = e^{-2x}$ at $x = 1$. Use the numerical formulas at the end of Section 8.2 to approximate the second derivative using $h = 0.1,\ 0.05$ and 0.025. Compute the errors in your approximations and build tables to display the results.

8-6. Construct the cubic interpolating polynomial $P_3(x)$ for the data ($x_1,\ y_1$), ($x_1 + h,\ y_2$), ($x_1 + 2h,\ y_3$) and ($x_1 + 3h,\ y_4$). Simplify the denominators of each term and then compute the derivative of the cubic interpolating polynomial. Remember that the product rule is $\frac{d}{dx}[f(x)g(x)h(x)] = f'(x)g(x)h(x) + f(x)g'(x)h(x) + f(x)g(x)h'(x)$.

a. Compute $P_3'(x)$ at $x = x_1$ and compare your result to (8.25)
b. Compute $P_3'(x)$ at $x = x_1 + 3h$. What is an appropriate name for the formula?

8-7. Use Taylor expansions to verify the three term backward difference approximation for the first derivative.

8-8. Construct a four term backward difference approximation for $f'(x_i)$ in the form $f'(x_i) \approx Af(x_i) + Bf(x_i - h) + Cf(x_i - 2h) + Df(x_i - 3h)$ using the method of undetermined coefficients. Compare your result to the forward difference formula.

8-9. Construct a four term forward difference approximation for $f''(x_i)$ in the form $f''(x_i) \approx Af(x_i) + Bf(x_i + h) + Cf(x_i + 2h) + Df(x_i + 3h)$ using the method of undetermined coefficients. Hint: See (8.22) and (8.23).

8-10. If you have four data points in the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, write the MATLAB commands that will evaluate the first and second derivatives of the interpolating polynomial at x_1.

8-11. Consider $I = \int_{2.5}^{4} \sin(x^2/5)\, dx$.
a. Approximate I with the composite Trapezoidal rule with $n = 4$
b. Bound the error in T_4.

8-12. Repeat Problem 11 with Simpson's rule.

8-13. If $I = \int_{2.5}^{4} \sin(x^2/5)\, dx$ is to be approximated with an error of no more than 0.0001, determine the number of intervals, n, to accomplish the task.
a. Use the Trapezoidal rule
b. Use Simpson's rule

8-14. Modify the M-file trapsimp.m in two ways. First, allow the true value of an integral to be an input. Second, compute and display the actual errors for both the Trapezoidal Rule and Simpson's Rule. Test your file with the text example $I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$.

8-15. Given $I = \int_{1}^{4} \sin(\sqrt{x}) dx$.
a. Determine the exact value of I. Recall MATLAB's **int.**
b. Find the number of intervals needed to approximate I with an error tolerance of 0.000005 using the Trapezoidal Rule and Simpson's Rule.
c. Repeat part b) using the asymptotic error formulas.
d. Using the modified trapsimp.m compare your predictions in parts b and c with the output data.

8-16. Recall the midpoint template $M_1 = (b-a) f(\frac{1}{2}(a+b))$, (8.37). Construct an expression for the composite midpoint rule M_n.

8-17. Verify the open Trapezoidal Rule given in (8.38).

8-18. Consider $I = \int_{a}^{b} f(x) dx$. Approximate $f(x)$ with a third degree interpolating polynomial $P_3(x)$ using the four values of x: $a,\ a+h,\ a+2h$ and $a+3h$. Using MATLAB integrate each term symbolically from a to $b = a+3h$. Write the quadrature formula that approximates I. The result is called Simpson's 3/8 Rule.

8-19. Using $\frac{I - \widehat{S}_n}{I - \widehat{S}_{2n}} \approx 16$, derive (8.53) and (8.54).

8-20. Use the Gaussian formulas $G_1,\ G_2$ and G_3 to approximate $I = \int_{2.5}^{4} \sin(x^2/5)\, dx$.

8-21. A three point Gaussian formula is given by $G_3 = w_1 f(\xi_1) + w_2 f(\xi_2) + w_3 f(\xi_3)$. With six parameters it seems plausible that the precision of G_3 is five. Construct six equations similar to (8.61) and verify that the values given in (8.63) satisfy the equations.

8-22. Given $I = \int_{a}^{b} f(x) dx$. Write an M-file that will accept $f(x)$, a and b as inputs and return G_1, G_2 and G_3 as outputs. Test your file on $I = \int_{1}^{2} \frac{\sin(x)}{x} dx$. Use **sinint** to compute the integral and then determine the errors in the Gaussian approximations.

8-23. Explain and discuss an adaptive quadrature scheme based on the Trapezoidal Rule. Evaluate the text example in Section 8.8 using your scheme.

8-24. Consider a curve specified in parametric form.
$x(t) = (2 + \cos(t))\cos(t)$ and $y(t) = (2 + \cos(t))\sin(t)$ for $0 \leq t \leq \pi$.
Look up the formula for arc length of parametric equations in a calculus text and construct the arc length integral. MATLAB can help with the derivatives and algebra. Although the integral has a rather simple integrand there is no antiderivative. Use **quad** to compute the arc length. Explain why your result is reasonable.

8-25. Given $f(x) = 3e^{5x}$ on the interval $[\,0, 2\,]$. Use **quad** to compute the volume and surface area if $f(x)$ is rotated about the x-axis.

8-26. Consider the following data from Section 6.4:

x	2	2.5	4	4.5	6
y	1.623	1.855	2.425	2.806	3.966

Approximate the definite integral from 2 to 6 using a piece wise linear approximation, (8.65), an interpolating polynomial $\int_2^6 P_4(x)dx$, and then a spline $\int_2^6 S(x)dx$. Which procedure gives the best approximation? Explain your reasoning. See Figure 6.1.

8-27. Explain and discuss the following segment of MATLAB code.

```
>> x = linspace(1,4);y = sin(sqrt(x));
>> spdata = spline(x,y);
>> [breaks,spcoef,pieces,order,dim] = unmkpp(spdata);
>> a = spcoef(:,1);b = spcoef(:,2);c = spcoef(:,3);d = spcoef(:,4);
>> h = diff(x);SUM = 0;
>> for k = 1:length(h)
SUM = SUM+a(k)*h(k)^4/4+b(k)*h(k)^3/3+c(k)*h(k)^2/2+d(k)*h(k);
end
```

What does the final value of **SUM** represent?

8-29. Use a spline to approximate the integrand of $I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$.
Vary the number of subintervals in the spline and attempt to determine a relationship between the number of subintervals and the accuracy of the approximation.