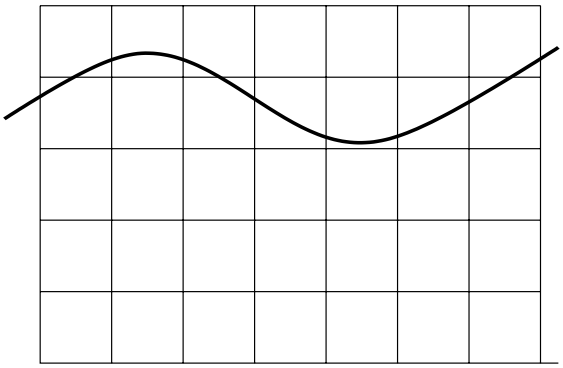
7
Numerical Integration

7.1 Introduction

In this chapter we shall discuss techniques whereby functions can be integrated; these are quite classical. We will give the derivation of the approximations and will mention the likely failings of the techniques. Two versions of the code will be given for each of the main methods, so we can start to appreciate the power of MATLAB. It is of course possible to write the code as if it were Fortran or C, but that would waste the power of the package.

Before we start we note: if we consider two points we can fit a straight line through them; with three we can fit a quadratic and with four we can fit a cubic (this was discussed in more detail in Chapter 5).

We start by breaking down the integration régime into small intervals and approximating the area below the curve by slices. This method is tantamount to counting the squares and dealing with the parts of squares at the tops of the columns in sophisticated ways.



In this case the area under the curve is approximately

$$7 \times 4 \times \text{“the area of the boxes”}$$

(the fourth row up has a few part squares as does the fifth row). We obviously need a scheme which is slightly more robust.

7.2 Integration Using Straight Lines

In this chapter our objective is to calculate the value of the integral

$$I = \int_{x=a}^{b} f(x)\,\mathrm{d}x.$$

We shall assume we can calculate (easily) the value of $f(x)$ for all values of x between a and b. This means we are dealing with a function rather than a set of data values. We shall explore the latter case in due course but at this juncture we shall use a new routine `integrand.m`:

```
%
% integrand.m
% input a set of values (x)
% output function values f(x)
%
function [f] = integrand(x)
%  Here we use f = sin(x^2) as a
% sample function.
f = sin(x.^2);
```

We shall now take a while to derive the method we are going to use to integrate the function on the grid of points. We shall use N points and as such we use the code

```
step = (b-a)/(N-1);
x = a:step:b;
f = integrand(x);
```

In order to derive the form for the integration we introduce the nomenclature that the points we have defined above are (x_j, f_j), where j runs from 1 to N. Let us consider the consecutive points (x_j, f_j) and (x_{j+1}, f_{j+1}) and approximate the curve between them by a straight line. We use the formula for the line through the points (x_1, y_1) and (x_2, y_2) which is

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1} \text{ or } y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1).$$

In our case this gives

$$f_L(x) = f_j + \frac{f_{j+1} - f_j}{h}(x - x_j),$$

where we have introduced $h = x_{j+1} - x_j$ and $f_L(x)$ the formula for the straight line (This is just a direct application of Newton's Forward Differences, (5.1).) Let us now perform the analytical integration of the function $f_L(x)$ between x_j and x_{j+1} to determine the area $A_{j,j+1}$:

$$A_{j,j+1} = \int_{x=x_j}^{x_{j+1}} f_L(x)\,\mathrm{d}x = \int_{x=x_j}^{x_{j+1}} \frac{f_{j+1} - f_j}{h}(x - x_j) + f_j\,\mathrm{d}x.$$

We introduce the linear transformation $X = x - x_j$, where $\mathrm{d}X = \mathrm{d}x$ and when $x = x_j$, $X = 0$ and $x = x_{j+1}$ correspond to $X = h$. The integral becomes

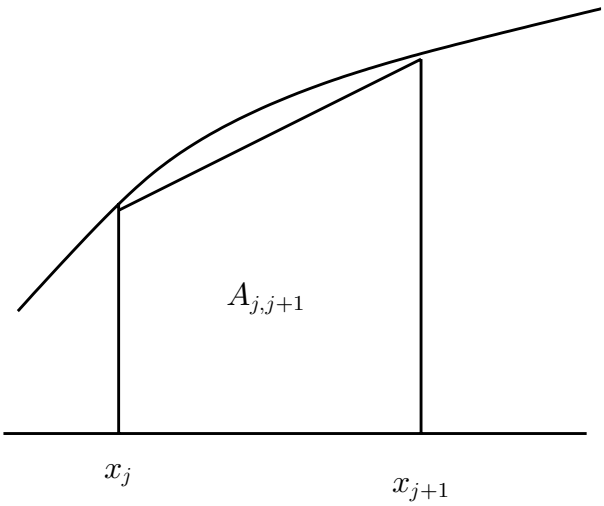
$$A_{j,j+1} = \int_{X=0}^{h} \left[\frac{f_{j+1} - f_j}{h} X + f_j\right] \mathrm{d}X,$$

which can be integrated to yield

$$A_{j,j+1} = \frac{h}{2}(f_j + f_{j+1}).$$

This is the area of a trapezium with vertices at $(x_j, 0)$, (x_j, f_j), (x_{j+1}, f_{j+1}) and $(x_{j+1}, 0)$[1].

[1] The area of a trapezium is the mean of the length of the two parallel sides times the perpendicular distance between them; that is $(f_j + f_{j+1})/2$ times $(x_{j+1} - x_j)$.



This method is unsurprisingly called the trapezium rule. In order to determine the total area from $x = a$ to $x = b$ we sum all the parts

$$\text{Area } = \sum_{j=1}^{j=N-1} A_{j,j+1} = \sum_{j=1}^{j=N-1} \frac{h}{2} (f_j + f_{j+1}) .$$

In order to perform this summation it is instructive to write out the series

$$\begin{aligned} \text{Area } =& \frac{h}{2} (f_1 + f_2) + \frac{h}{2} (f_2 + f_3) + \frac{h}{2} (f_3 + f_4) + \cdots \\ &+ \frac{h}{2} (f_{N-2} + f_{N-1}) + \frac{h}{2} (f_{N-1} + f_N) , \end{aligned}$$

so that

$$\text{Area } = \frac{h}{2} (f_1 + 2f_2 + 2f_3 + \cdots + 2f_{N-1} + f_N) \approx \int_{x=a}^{b} f(x) \, \mathrm{d}x.$$

Example 7.1 *We shall calculate the integral of the function* $f(x) = x^3 \sin x$ *between zero and one. We shall use* N *points*

```
N = 10;
x = linspace(0,1,N);
h = x(2)-x(1);
f = x.^3.*sin(x);
g = h*(sum(f)-f(1)/2-f(N)/2);
```

Here we have constructed a grid of points running from zero to one and then worked out the gap between successive points (that is `h`*). We now construct the function* `f` *and work out the expression for the trapezium rule, which is the sum of the values of* `f` *minus half of the end values. This gives the value of the integral as* `g`*.*

7.2.1 Errors in the Trapezium Method

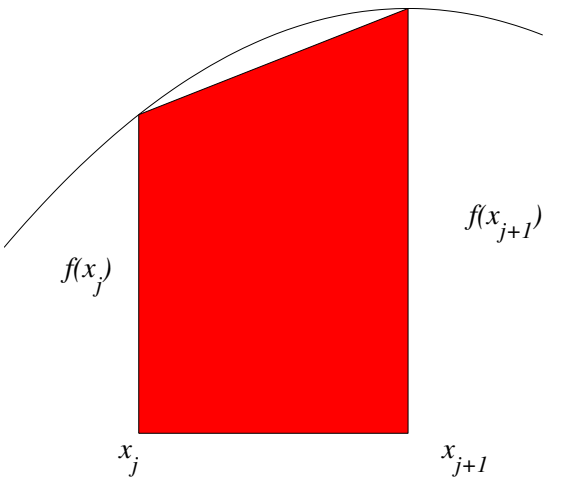
The number of points required for a calculation depends on how exactly you need to know the answer (in general). The error in this scheme is encountered because we approximate the curve between the points (x_j, f_j) and (x_{j+1}, f_{j+1}) by a straight line. This can be reduced by using a quadratic instead over the interval spanned by the requisite three points.

We could have used the fundamental code which did not make use of the combinations of the terms:

```
integral = 0;
for i = 2:N
    integral = integral+(f(i)+f(i-1))/2*h;
end
```

The advantage of this form of the code is it is far easier to convert to one in which intermediate results are available. We note that the command `/2*h` divides by two and multiplies by `h` rather than dividing by `2h`: this would be accomplished via parentheses, that is `/(2*h)`.

Before we proceed we should consider the error involved in approximating the area by a set of trapezia.



The error is the unshaded part below the curve. To estimate the extent of this we again use "Taylor Series" and note that

$$f(x) = f(x_j) + \frac{f(x_{j+1}) - f(x_j)}{h}(x - x_j) + \frac{(x - x_j)(x - x_{j+1})}{2} \left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_{x=\xi},$$

where $\xi \in [x_j, x_{j+1}]$. This can be integrated to give

$$\int_{x=x_j}^{x_{j+1}} f(x)\,\mathrm{d}x = \frac{h}{2}\left(f(x_j) + f(x_{j+1})\right) + \frac{h^3}{6} \left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_{x=\xi}.$$

Consequently the error in using just the first term is proportional to h^3 and $f''(\xi)$. Notice that if the second derivative is zero over the range the error is actually zero. Unsurprisingly this corresponds to $f(x)$ being a straight line.

7.3 Integration Using Quadratics

We need to construct a curve which passes through the points (x_{j-1}, f_{j-1}), (x_j, f_j) and (x_{j+1}, f_{j+1}). Let us consider a value of $j = 1$: this is purely to reduce the verbosity of our expressions. The quadratic passing through these three points can be written as

$$f_q(x) = f_0 + \Delta f_0 \frac{x - x_0}{h} + \Delta^2 f_0 \frac{(x - x_0)(x - x_1)}{2h^2},$$

using a truncation of Newton's forward difference formula, (5.1).

In passing we mention this series expansion is similar to a Taylor series expansion but instead of the terms being differentials they are differences. In order to derive the formula we now need to integrate the function $f_q(x)$ from $x = x_0$ to $x = x_2$. This is quite straightforward: however at this point we will exploit the symbolic capabilities of MATLAB

```
syms x f0 f1 f2 h x0
q = f0+(f1-f0)/h*(x-x0)+(f2-2*f1+f0)/(2*h^2)*(x-x0)*(x-(x0+h));
iq = int(q,x0,x0+2*h);
simplify(iq)

ans =

1/3*h*(f0+4*f1+f2)
```

We shall dissect this portion of code so you can appreciate what is happening.

– In the first line we assign the variables `x`, `f0`, `f1`, `f2`, `h` and `x0` to be symbolic. This means that MATLAB does not have to know the value of the variables and treats them as mathematical objects.

– In the second line we set up the function `q` (which is just the quadratic $f_q(x)$).

– This is integrated in the third line (between `x0` and `x2 = x0+2*h`. (This produces a verbose answer.)

– Finally we simplify our answer.

Hence we have the answer

$$\int_{x=x_0}^{x=x_2} f_q(x)\,\mathrm{d}x = \frac{h}{3}\left(f_0 + 4f_1 + f_2\right).$$

Although we know there is only one quadratic through a given set of points, it is instructive to re-derive this in another way. We shall start with the general quadratic

$$f_q(x) = a + b(x - x_0) + c(x - x_0)(x - x_1).$$

The requirement that the quadratic goes through the point (x_0, f_0) gives us that $a = f_0$. Using the point (x_1, f_1) gives us $b = (f_1 - f_0)(x_1 - x_0)$. Finally using (x_2, f_2) we have

$$f_2 = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x_2 - x_0) + c(x_2 - x_0)(x_2 - x_1),$$

which can be manipulated to give:

$$c = \frac{1}{x_2 - x_1}\left(\frac{f_2 - f_0}{x_2 - x_0} - \frac{f_1 - f_0}{x_1 - x_0}\right).$$

This gives us the general form of the quadratic through three points (which can be used for Task 7.10). We now return to the form for regularly spaced points so $\Delta x_j = h$ and integrating from $x = x_0$ to $x = x_2 = x_0 + 2h$:

$$\int_{x=x_0}^{x=x_0+2h} f_q(x)\,\mathrm{d}x = \int_{x=x_0}^{x=x_0+2h} f_0 + \frac{f_1 - f_0}{h}(x - x_0) + \frac{1}{h}\left(\frac{f_2 - f_0}{2h} - \frac{f_1 - f_0}{h}\right)(x - x_0)(x - x_1)\,\mathrm{d}x.$$

Now using the substitution $X = x - x_0$, so that $x = x_0$ corresponds to $X = 0$ and $x = x_2 = x_0 + 2h$ corresponds to $X = 2h$ the expression $x - x_1 =$

$X + x_0 - x_1 = X - h$ and $\mathrm{d}x = \mathrm{d}X$. Hence we have

$$
\begin{aligned}
&= \int_{X=0}^{2h} f_0 + \frac{f_1 - f_0}{h} X + \frac{1}{2h^2} \left(f_2 - f_0 - 2(f_1 - f_0)\right) X(X-h)\,\mathrm{d}X \\
&= \left[f_0 X + \frac{f_1 - f_0}{h} \frac{X^2}{2} + \frac{1}{2h^2} \left(f_0 - 2f_1 + f_2\right) \left(\frac{X^3}{3} - h\frac{X^2}{2} \right) \right]_0^{2h} \\
&= 2hf_0 + \frac{f_1 - f_0}{h} 2h^2 + \frac{1}{2h^2} \left(f_0 - 2f_1 + f_2\right) \left(\frac{8h^3}{3} - 2h^3 \right) \\
&= h \left(2f_0 + 2(f_1 - f_0) + \frac{1}{3} \left(f_0 - 2f_1 + f_2\right) \right) \\
&= \frac{h}{3} \left(f_0 + 4f_1 + f_2\right).
\end{aligned}
$$

We now need to divide the range of integration into the appropriate number of subintervals. Notice the number of intervals needs to be even (and hence the number of points needs to be odd). The total integral is approximated by

$$
\begin{aligned}
\int_{x=a}^{b} f(x)\,\mathrm{d}x \approx & \frac{h}{3} \left(f_0 + 4f_1 + f_2\right) + \frac{h}{3} \left(f_2 + 4f_3 + f_4\right) + \frac{h}{3} \left(f_4 + 4f_5 + f_6\right) + \cdots \\
& + \frac{h}{3} \left(f_{N-4} + 4f_{N-3} + f_{N-2}\right) + \frac{h}{3} \left(f_{N-2} + 4f_{N-1} + f_N\right).
\end{aligned}
$$

This can be simplified to give

$$
\int_{x=a}^{b} f(x)\,\mathrm{d}x \approx \frac{h}{3} \left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{N-2} + 4f_{N-1} + f_N\right).
$$

This is called Simpson's $\frac{1}{3}$ rule.

We shall now construct a code to determine the integral. At this point we could use a code which used a conventional approach, but we shall try to use a version which exploits the power of MATLAB.

```
%
% Simpson's 1/3 rule.
%
x = 0.0:0.1:1.0;
h = x(2)-x(1);
N = length(x);
if mod(N,2) == 0
         disp('Routine needs an odd number of points')
         break     % Ensure the number of points is odd
end

rodd = 1:2:N;
reven = 2:2:(N-1);
weights(rodd) = 2; weights(1) = 1; weights(N) = 1;
weights(reven) = 4;

f = sin(x.^2);
integral = h/3*sum(weights.*f);
format long e
disp([integral])
```

This calculates the value of the expression

$$\int_0^1 \sin x^2 \, \mathrm{d}x.$$

The value which MATLAB comes out with is `3.102602344332209e-01`, where we have changed the way in which these numbers are displayed by using the command `format long e`. This answer can be checked using MATLAB's symbolic capabilities:

```
syms x f
f = sin(x^2);
f1 = int(f,0,1)

f1 =

1/2*FresnelS(2^(1/2)/pi^(1/2))*2^(1/2)*pi^(1/2)
```

This value can be compared with that attained using the symbolic toolbox, using the command `vpa` (variable precision arithmetic):

```
vpa(f1)

ans =


.31026830172338110180815242316 5
```

(Note we can change the number of digits using the command `digits(10)`). We further note that this is still a symbolic object: in order to obtain a value (which can be used for plotting, for example) we use the command `double`. This is a case of MATLAB being too clever: it has solved the integral and written it as a Fresnel integral (for further details see Abramowitz and Stegun – *Handbook of Mathematical Functions*). Our simple integration using ten points does quite well: in fact the error is proportional to h^4, which can be shown using the same technique as we used for the trapezium rule on page 230.

In the above code we have introduced the term `weights` which is applied to the terms before they are added together to obtain the integral. The MATLAB command `mod` allows the user to determine the remainder when the first argument is divided by the second. In this case when considering `mod` 2 we are checking for parity (that is whether N is even or odd).

In the code for Simpson's one third rule we have re-introduced another MATLAB command, namely `break`. This stops the code, or more exactly it exits the current level: for example if it is used in a nested loop it will terminate the current level and return to the previous level. This manual entry for `break` is instructive here

```
>> help break

 BREAK Terminate execution of WHILE or FOR loop.
    BREAK terminates the execution of FOR and WHILE loops.
    In nested loops, BREAK exits from the innermost loop only.
```

We now progress to consider cubic approximations to the function in the hope that this will provide even more accurate answers.

Again we can use a technique which involves summing the separate intervals, which you may prefer:

```
integral = 0;
for j = 1:2:N-2
    integral = integral + h*(f(j)+4*f(j+1)+f(j+2))/3;
end
```

7.4 Integration Using Cubic Polynomials

As in the previous sections we need to define an approximating curve and in order to do so we need four points (x_0, f_0), (x_1, f_1), (x_2, f_2) and (x_3, f_3). Using the same form as above we can write the cubic equation as

$$f_c(x) = f_0 + \Delta f_0 \frac{x - x_0}{h} + \Delta^2 f_0 \frac{(x - x_0)(x - x_1)}{2h^2} + \Delta^3 f_0 \frac{(x - x_0)(x - x_1)(x - x_2)}{6h^3}.$$

In addition to the terms defined earlier we have introduced the third-order forward difference. This can be defined recursively using Δ^2, so that

$$\begin{aligned} \Delta^3 f_0 = \Delta^2(\Delta f_0) &= \Delta^2 f_1 - \Delta^2 f_0 \\ &= (f_3 - 2f_2 + f_1) - (f_2 - 2f_1 + f_0) \\ &= f_3 - 3f_2 + 3f_1 - f_0, \end{aligned}$$

using the Newton forward difference for $\Delta^2 f_0$. We now need to integrate from $x = x_0$ to $x = x_3$, and we again exploit the symbolic tool box for this, using the code

```
syms x f0 f1 f2 f3 h x0
x1 = x0+h; x2 = x0+2*h; x3 = x0+3*h;
t1 = (f1-f0)/h*(x-x0);
t2 = (f2-2*f1+f0)/(2*h^2)*(x-x0)*(x-x1);
t3 = (f3-3*f2+3*f1-f0)/(6*h^3)*(x-x0)*(x-x1)*(x-x2);
q = f0+t1+t2+t3;
q1 = int(q,x,x0,x3);
simplify(q1)
```

This gives the answer `3/8*h*(f0+3*f1+3*f2+f3)`. Thus we have

$$\int_{x=x_0}^{x_3} f_c(x)\,\mathrm{d}x = \frac{3h}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right).$$

You should note it is possible to do all these integrals by hand, but we wish to demonstrate the power and the utility of this particular toolbox.

We now need to combine all the subintervals, so

$$\begin{aligned} \int_{x=a}^{b} f(x)\,\mathrm{d}x \quad \approx \quad & \frac{3h}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right) + \frac{3h}{8}\left(f_3 + 3f_4 + 3f_5 + f_6\right) + \cdots \\ & + \frac{3h}{8}\left(f_{N-3} + 3f_{N-2} + 3f_{N-1} + f_N\right), \end{aligned}$$

which can be simplified to give

$$\int_{x=a}^{b} f(x)\,\mathrm{d}x \approx \frac{3h}{8}\left(f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + 3f_5 + \cdots \right.$$
$$\left. +2f_{N-3} + 3f_{N-2} + 3f_{N-1} + f_N\right).$$

This is referred to as Simpson's $\frac{3}{8}$ rule.

We shall now give a MATLAB program based on the same structure as the previous one. Notice this time the number of points needs to be divisible by three.

```
%
% simpson's 3/8 rule.
%
N = 10;
x = linspace(0,1,N);
h = x(2)-x(1);
ms = 'Number of intervals should be divisible by three';

if mod(N-1,3) ~= 0
         disp(ms)
         break
end
m = (N-1)/3;
rdiff = 3*(1:(m-1))+1;
weights = 3*ones(1,N);
weights(1) = 1; weights(N) = 1;
weights(rdiff) = 2;
f = sin(x.^2);
integral = 3*h/8*sum(weights.*f);
disp([integral])
```

The answer given by this is `0.31024037588964`. In fact this is not quite as good as the previous method. The error is again proportional to h^4 but the constant of proportionality is larger. We could also write the integral in separate regions, that is without combination, so

```
integral = 0;
for j = 1:3:N-3
    integral = integral + h*(f(j)+3*f(j+1) ...
                            +3*f(j+2)+f(j+3))*3/8;
end
```

Each of these methods has a restriction on the numbers of points one can use. However these can be circumvented by using an amalgamation of the two schemes. There are other methods available for integrating functions and some of these will be met in due course. We could continue this process, especially with the symbolic toolbox at our disposal to perform the algebra. For instance the formula obtained by using a quartic over five points is

$$\int_{x=x_0}^{x_4} f(x)\,\mathrm{d}x \approx \frac{2h}{45}\left(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4\right).$$

Although you might think the higher the order the polynomial the more accurately you would know the answer, there are problems which are intrinsic to using high-order polynomials. In fact the optimum method is to use a combination of the two Simpson methods. If the number of points supplied is even we use the one third rule for the first $N-3$ points (which is necessarily an odd number of points and consequently an even number of intervals) and then we use the three-eighths rule on the remaining points.

7.5 Integrating Using MATLAB Commands

As with many examples in this text we can also use standard MATLAB commands, for instance `quad` and `quad8` (see `help quad`). These commands use similar techniques to those above but with the advantage of automation. For instance they exploit grids which can adapt. This means that in regions which are harder to integrate (perhaps with more variation in the function) the scheme adds extra points.

The syntax is:

```
tol = [1e-4 1e-5];
a = 0; b = pi;
trace = 1;
q = quad('sin',a,b,tol,trace)
```

These quantities are respectively the extent of the domain $x \in [a, b]$, the tolerances (relative and absolute) and whether the user wants to see a trace or not (setting trace as non-zero shows the evolution of the calculation).

Example 7.2 *We show the integration of the function* $J_1(x)$ *for* $x = 0$ *to* $x = 10$. *This is actually a Bessel function which occurs as one of the solutions*

to the differential equation

$$x^2\frac{\mathrm{d}^2y}{\mathrm{d}x^2}+x\frac{\mathrm{d}y}{\mathrm{d}x}+(x^2-1)y=0,$$

and is revisited in Chapter 8.

Fortunately MATLAB has a routine which evaluates Bessel functions but we need to write our own routine to make sure that it is available for **quad***:*

```
function [value]=ourbess(x)
value = besselj(1,x);
```

and then we simply need the code:

```
q = quad('ourbess',0,10,[1e-5 1e-5],1);
```

This gives a value of 1.24593587184673*. The integral of the function* $J_1(x)$ *is* $-J_0(x)$ *and hence the value we seek is* $-J_0(10)+J_0(0)\approx 1.245935764$*; so as we can see the integration scheme does very well.*

7.6 Specific Examples of Integrals

We shall now describe how we can deal with other problems which arise when evaluating numerical integrals.

7.6.1 Infinite Integrals and Removable Singularities

Using the various methods we can evaluate integrals from a to b, but if one or more of these values is infinite we need a different treatment. This will vary depending on the form of the integral or more exactly the integrand.

Example 7.3 *For example let us consider the integral*

$$I=\int_0^\infty \mathrm{e}^{-x^2}\,\mathrm{d}x.$$

In this case the integrand e^{-x^2} *decays very quickly so we can adequately determine the value of the integral using*

$$I_X = \int_0^X \mathrm{e}^{-x^2}\,\mathrm{d}x,$$

for a suitable value of X *(in this case* $X = 10$ *is more than sufficient). We can evaluate the integral for a few values of* X *until* I_X *is constant (to within a defined tolerance). We could develop an algorithm to decide what value of* X *to use, but in general we will use the above method. We can also exploit the symmetry of problems, for instance*

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2}\,\mathrm{d}x = 2\int_0^{\infty} \mathrm{e}^{-x^2}\,\mathrm{d}x \approx 2\int_0^X \mathrm{e}^{-x^2}\,\mathrm{d}x,$$

provided $X \gg 1$*. We can also use transformations to rescale the regions of integration.*

In fact in this example we could have used the MATLAB command `erf` which gives the error function

$$E(q) = \frac{2}{\sqrt{\pi}} \int_{q=0}^{x} \mathrm{e}^{-q^2}\,\mathrm{d}q.$$

We can use `erf(Inf)` which gives us the value unity.

We now consider how we might perform an integral for which the integrand is singular at an end-point of the range. For instance:

$$\int_{q=0}^{1} \frac{1}{q^{1/2}}\,\mathrm{d}q = \left[2q^{1/2}\right]_{q=0}^{1} = 2.$$

Example 7.4 *We consider the integral of the function* $f(x) = \mathrm{e}^{-x}/\sqrt{x}$ *between* $x = 0$ *and* $x = 1$*. Firstly we separate the range into the two disjoint ranges* $[0, \epsilon) \cup [\epsilon, 1]$ *where* $\epsilon \ll 1$ *(that is it is very small). We now consider the integral:*

$$\int_0^{\epsilon} \frac{\mathrm{e}^{-x}}{\sqrt{x}}\,\mathrm{d}x$$

and note that over this range $\mathrm{e}^{-x} \approx 1 - x + x^2/2 + \cdots$. *Thus*

$$\int_0^\epsilon \frac{\mathrm{e}^{-x}}{\sqrt{x}}\,\mathrm{d}x \approx \int_0^\epsilon \frac{1}{\sqrt{x}}\left(1 - x + \frac{x^2}{2} + \cdots\right)\mathrm{d}x = \left[2x^{1/2} - \frac{2}{3}x^{3/2} - \frac{1}{5}x^{5/2} + \cdots\right]_0^\epsilon$$
$$= \left(2\epsilon^{1/2} - \frac{2}{3}\epsilon^{3/2} - \frac{1}{5}\epsilon^{5/2}\right).$$

The integral can now be evaluated using:

```
epsil = 0.01;
val = 2*sqrt(epsil)-2/3*epsil^1.5-1/5*epsil^2.5;
x = linspace(epsil,1,100);
f = exp(-x)./sqrt(x);
h = x(2)-x(1);
N = length(x);
int = val;
for j = 1:N-1
    int = int + h/2*(f(j+1)+f(j));
end
```

This gives a value `1.49764481658781` *(the value given by MATLAB is* `erf(1)*sqrt(pi)` *which is* $erf(1)\sqrt{\pi} \approx 1.493648266$*) where*

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_{t=0}^{x} \mathrm{e}^{-t^2}\,\mathrm{d}t.$$

7.6.2 Indefinite Integrals

So far we have only been interested in definite integrals, but we shall now consider how one might determine

$$I(x) = \int_{q=a}^{x} f(q)\,\mathrm{d}q.$$

Instead of using a scalar variable to store the cumulative total, we exploit a vector to store intermediate results, so that

```
integral = zeros(size(x));
step = x(2)-x(1);
N = length(integral);
for j = 2:N
    integral(j) = integral(j-1)+step*(f(j)+f(j-1))/2;
end
disp(['Value of total integral ' num2str(integral(N))]);
```

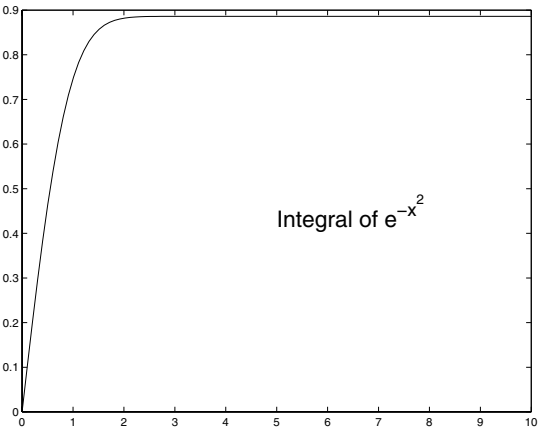
Example 7.5 *We now plot the function*

$$g(x) = \int_{q=0}^{x} \mathrm{e}^{-q^2}\,\mathrm{d}q.$$

We determine $g(x)$ *for* $x = 0$ *to* $x = 10$ *using the code*

```
xv = linspace(0,10);
f = exp(-xv.^2);
N = length(xv);
step = xv(2)-xv(1);
g(1) = 0.0;
for j = 2:N
    g(j) = g(j-1)+step*(f(j)+f(j-1))/2;
end
plot(xv,g)
text(5,max(g)/2,'Integral of e^{-x^2}','FontSize',20)
```

which gives



7.7 Tasks

Task 7.1 *Construct the sequence*

$$f(i) = \begin{cases} 1 & \textit{if} \quad \mod(i,3) = 0 \\ 2 & \textit{if} \quad \mod(i,3) = 1 \\ 3 & \textit{if} \quad \mod(i,3) = 2 \end{cases}$$

up to $N = 12$.

Task 7.2 *In Chapter 7 for Simpson's one third rule we used*

```
rodd = 1:2:N;
reven = 2:2:(N-1);
weights(rodd) = 2; weights(1) = 1; weights(N) = 1;
weights(reven) = 4;
```

and for Simpson's three eighths rule

```
m = (N-1)/3;
rdiff = 3*(1:(m-1))+1;
weights = 3*ones(1,N);
weights(1) = 1; weights(N) = 1;
weights(rdiff) = 2;
```

Write out these coefficients for $N = 9$ for the one third rule and $N = 10$ for the three eighths rule. You should construct each of the vectors by hand.

By now you should be able to read the codes `trap.m`, `simp13.m` *and* `simp38.m` *and understand what they do. You should also be able write your own program which returns the value of a function evaluated at a given point. Try this task.*

Task 7.3 *Write a routine which takes an input x and returns the value of $f(x) = \ln\left(x + \sqrt{x^2+1}\right)$.*

Task 7.4 *Using the trapezium rule calculate the integral of the quadratic $x^2 - 3x + 2$ between $x = 1$ and $x = 3$ (check your answer with the exact answer).*

Task 7.5 *Using Simpson's one third rule integrate the cubic $x^3 - x + 1$ between the limits $x = 0$ and $x = 1$ (check your answer against the exact answer).*

Task 7.6 *Using Simpson's one third rule integrate the function $f(x) = \sin x$ between the limits $x = 0$ and $x = \pi$ (check your answer against the exact answer). You might want to change the number of points you use and see what happens to the error.*

Task 7.7 *Calculate the value of the integral*

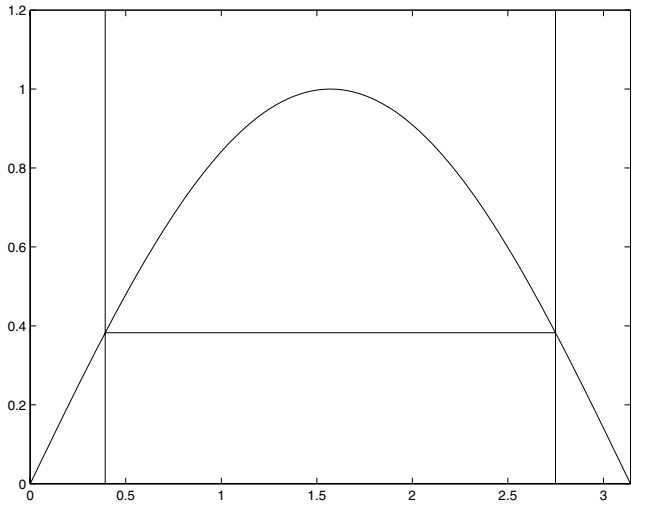
$$\int_0^\infty \frac{1}{\sqrt{x^2+1}}\,\mathrm{d}x.$$

You will need to truncate the domain, and you should investigate the effect of this truncation as well as the number of points required to accurately calculate the integral.

Task 7.8 *The length along a curve $y = y(x)$ from $x = a$ to $x = b$ is given by the expression*

$$S = \int_{x=a}^{b} \sqrt{1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2}\,\mathrm{d}x.$$

In many cases this expression can be determined analytically: however there are some very simple cases for which it can't be. Consider the problem of a sine curve truncated over the range $[\theta, \pi - \theta]$:



The expression for the length of this curve is

$$S = \int_{\phi=\theta}^{\pi-\theta} \sqrt{1+\cos^2 x}\,\mathrm{d}x.$$

Unfortunately this integral is intractable using analytic means, but determine the value numerically using the trapezium rule for a variety of values of θ*.*

Task 7.9 (*) *Determine the integral*

$$\int_0^{10} \frac{\cos x}{x^{1/2}}\,\mathrm{d}x$$

by splitting the integral into two ranges $[0,\epsilon]$ *and* $[\epsilon,10]$ *where* ϵ *is taken to be small. In this first range the function* $\cos x$ *can be approximated by its Taylor series and this form can be used to work out the contribution from the "singular part" of the integral.*

Task 7.10 (*) *Derive an expression for the integral of the quadratic passing through the points* (x_i, f_i) $i = 0, 1$ *and 2 from* $x = x_0$ *to* $x = x_2$ *(use the general quadratic on page 231). This gives you Simpson's scheme for variably spaced points this can also be extended to four points with a cubic.*

If you are feeling very brave you could also work out the errors associated with these approximations.

Task 7.11 *Integrate the function* $x \ln x$ *between the limits* 1 *and* 2 *using the MATLAB function* `quad`.