5. NUMERICAL METHODS

This section of the unit is concerned solely with the solution of algebraic equations (using ad hoc methods and the Newton-Raphson method) and numerical integration (the Trapezium rule and Simpson’s rule). As an adjunct we will also cover convergence analysis for the root-finding methods, and Richardson’s extrapolation method which is used in very many contexts to improve the accuracy of numerical integrals and the numerical solution of differential equations.

5.1. Brief introduction to root finding.

Root-finding algorithms, such as the Newton-Raphson method, are used to find the roots (solutions) of equations such as

\[ x^5 - x^2 + 1 = 0 \]

or

\[ x = e^{-x}, \]

for which analytical solutions are unavailable. They may also be used to find the solutions to simultaneous equations in more than one variable, such as the pair,

\[ y = 6x(x^2 - 1), \quad x = 6y(y^2 - 1). \]

This latter aspect, where more than one equation is to be solved, does not form examinable material in the present unit, but it is often used when solving differential equations numerically and therefore some introductory remarks will be made below — this is a preparation for ME20014.

We will cover ‘ad hoc’ methods and the Newton-Raphson method in some detail, and we shall study their rate of convergence or divergence. We will see that ad hoc methods have the advantage of being easy to write down, and therefore quick to program or to use on a pocket calculator, but they suffer from the disadvantages that they are slow, don’t find all the solutions, and may, for the case when there are multiple zeros, be appallingly slow. On the other hand, the Newton-Raphson method can find all roots, and is much quicker. There are flashier methods, such as Muller’s method, which converge even faster than the Newton-Raphson method, but they are trickier to use in practice and require much more effort; Muller’s method and its mates are not considered here.

With an equation such as, \( x^3 - 7x + 6 = 0 \), it is almost immediately obvious by the pattern of the coefficients that \( x = 1 \) is a solution, and therefore that \( (x - 1) \) is a factor. Thus \( (x - 1) \) may be factored out from the cubic to find the second factor which is a quadratic. This may, of course, be solved using the standard quadratic formula. It doesn’t take too long to find that the original cubic factorizes as follows,

\[ x^3 - 7x + 6 = (x - 1)(x - 2)(x + 3), \]

and therefore \( x = 1, 2, -3 \) are the roots/solutions. We will use this equation later to test the different iterative numerical schemes.

5.2. Iterative methods for solving algebraic equations: sketching suitable graphs.

Consider the following three equations,

\[ f(x) = x^2 - 2 = 0, \]
\[ g(x) = x \sin x - 1 = 0, \]
\[ h(x) = e^{x^2} - \sin x = 0. \]

Before any attempt is made to find the solutions of these equations it is advisable to draw suitable sketches in order to determine roughly how many possible solutions there are (if there are any!) and where they might be.
Figure 5.1. Appropriate sketches of various functions which indicate the number of roots of the equations $f(x) = x^2 - 2 = 0$, $g(x) = x \sin x - 1 = 0$ and $h(x) = e^{x^2} - \sin x = 0$.

In the first sketch in Fig. 5.1 we have the variations of both $x^2$ and 2 with $x$. These curves cross at two places, and these are the roots of $f(x) = x^2 - 2 = 0$. Clearly, then, this sketch tells us no more than we would have expected since we already know that there are two real square roots of a positive number.

For $g(x) = x \sin x - 1$ we have sketched the variations of $1/x$ and $\sin x$ since we may rearrange $g(x) = 0$ into the form, $\sin x = 1/x$. The sketch tells us that there is an infinite number of solutions because there is an infinite number of crossings of these two curves. Since $1/x$ becomes small when $x$ is large, it crosses the sine curve very close to where the sine is zero. Therefore these roots lie close to $n\pi$. This won’t be true near $x = 0$, but at least the sketch gives a rough indication of where these roots are, which is our aim prior to using the numerical method.

For $h(x)$ it is evident that the curves $e^{x^2}$ and $\sin x$ do not cross since the former is always greater than or equal to 1, whereas the latter lies between $-1$ and $+1$. Therefore the application of any method to solve $h(x) = 0$ can never converge to a real solution, and so $h(x) = 0$ has no solution.

5.3. Iterative methods for solving algebraic equations: ad hoc iteration schemes.

The latin, *ad hoc*, literally means *for this*, or perhaps one could stretch it mean something more like *for its purpose*. Effectively it also means *unplanned* or even *making it up as you go along*. We will find that when it works it works, but it sometimes doesn’t! We’ll use the above cubic equation, $x^3 - 7x + 6 = 0$, to illustrate the use of such ad hoc schemes.
We wish to solve \( x^3 - 7x + 6 = 0 \). This equation may be rearranged in either of the following ways:

\[
x = \frac{x^3 + 6}{7} \quad \text{or} \quad x = (7x - 6)^{1/3}.
\]

Therefore we may modify both of these to form the iteration schemes

\[
x_{n+1} = \frac{x_n^3 + 6}{7}, \quad \text{Scheme A}
\]

\[
x_{n+1} = (7x_n - 6)^{1/3}, \quad \text{Scheme B}
\]

Here the subscript tells us the iteration number, and we generally denote the initial iterate (or initial guess) by \( x_0 \). The general behaviours for each of these iteration schemes may be seen in the tables below which show the effect of the four different initial guesses for the roots, 3, 1.5, 0 and -4.

**Table 5.1. Successive iterations for Scheme A**

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>3</th>
<th>1.5</th>
<th>0</th>
<th>-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>4.714286</td>
<td>1.339286</td>
<td>0.857143</td>
<td>-8.285714</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>15.82466</td>
<td>1.200323</td>
<td>0.947105</td>
<td>-80.40566</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>566.9723</td>
<td>1.104199</td>
<td>0.978509</td>
<td>-74260.33</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>2.60 \times 10^7</td>
<td>1.049472</td>
<td>0.990986</td>
<td>-5.85 \times 10^{13}</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>2.52 \times 10^{21}</td>
<td>1.022268</td>
<td>0.996172</td>
<td>-2.86 \times 10^{40}</td>
</tr>
<tr>
<td>( x_6 )</td>
<td>2.29 \times 10^{63}</td>
<td>1.009758</td>
<td>0.998366</td>
<td>-3.34 \times 10^{120}</td>
</tr>
<tr>
<td>( x_\infty )</td>
<td>\infty</td>
<td>1</td>
<td>1</td>
<td>\infty</td>
</tr>
</tbody>
</table>

**Table 5.2. Successive iterations for Scheme B**

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>3</th>
<th>1.5</th>
<th>0</th>
<th>-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>2.446212</td>
<td>1.650964</td>
<td>-1.817121</td>
<td>-3.239612</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>2.241597</td>
<td>1.771224</td>
<td>-2.655221</td>
<td>-3.060878</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>2.132025</td>
<td>1.856497</td>
<td>-2.907809</td>
<td>-3.015701</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>2.074226</td>
<td>1.912519</td>
<td>-2.975906</td>
<td>-3.000405</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>2.042393</td>
<td>1.947609</td>
<td>-2.993740</td>
<td>-3.001054</td>
</tr>
<tr>
<td>( x_6 )</td>
<td>2.024430</td>
<td>1.968959</td>
<td>-2.998376</td>
<td>-3.000273</td>
</tr>
<tr>
<td>( x_\infty )</td>
<td>2</td>
<td>2</td>
<td>-3</td>
<td>-3</td>
</tr>
</tbody>
</table>

It is clear that we have managed to obtain all three roots between the two Tables, but it is also clear that some of the initial iterates seed an unstable sequence of iterates in the sense that the magnitude of successive iterates diverges rapidly.

In Table 5.1 for Scheme A, we may also notice that when iterates are close to the value, 1, then the distance to 1 decreases by a factor which is just a little less than \( \frac{1}{4} \); see \( x_5 \) and \( x_6 \) when \( x_0 = 1.5 \), for example. Likewise, for Scheme B, those iterates which are heading towards 2 do so in a manner whereby the distance to 2 decreases by a factor which is a little more than \( \frac{1}{4} \), while the corresponding factor for the root, -3, is approximately \( \frac{1}{4} \). We will explain these matters below.

Can we make sense of why these iteration schemes converge for some roots and diverge for others? Is there a general pattern? The answer is yes to both these questions and the following sketches illustrate what is happening for each scheme.
Figure 5.2. Showing a sketch of how different initial iterates converge towards and/or diverge away from the correct roots of an equation when employing Scheme A. Here iterations diverge from 2 and $-3$ but converge towards 1.

Figure 5.3. Showing a sketch of how different initial iterates converge towards and/or diverge away from the correct roots of an equation when employing Scheme B. Here iterations converge towards 2 and $-3$ but diverge away from 1.

These sketches are very inadequate in many ways, but the aim is to illustrate that each root either ‘attracts’ or ‘repels’, and that they alternate in these properties. In other words, roots are either stable or unstable in the sense that, if they are perturbed then those perturbations either decay or grow, respectively. For Scheme A only the root $x = 1$ is stable, i.e. only this one of the three may be found using iteration, while $x = 3$ and $x = 2$ are unstable and cannot be obtained. For Scheme B the roots $x = -3$ and $x = 2$ are stable,
while the root \(x = 1\) is unstable. This illustrates a general feature of ad hoc schemes that successive roots alternate in their ‘stability properties’, but different schemes may have different roots being stable.

### 5.4. Analysis of ad hoc methods

We may analyse the stability of these methods using small perturbations (and on applying either the Binomial expansion or a Taylor’s series expansion). We’ll consider the \(x = 1\) root first.

**Scheme A \((x = 1)\)**

Let \(x_n = 1 + \epsilon\) where \(\epsilon\) is assumed to be very small, and substitute this into \(x_{n+1} = \frac{1}{7}(x_n^3 + 6)\). Hence

\[
x_{n+1} = \frac{(1 + \epsilon)^3 + 6}{7} = \frac{(1 + 3\epsilon + 3\epsilon^2 + \epsilon^3) + 6}{7} = 1 + \frac{3}{7}\epsilon + \cdots.
\]

Here the dots simply mean that the \(\epsilon^2\) terms and higher powers have been omitted; if \(\epsilon\) is very very small, then these terms are negligible. But the above analysis shows that, when \(x_n\) is close to 1, then \(x_{n+1}\) is even closer to 1, and therefore \(x = 1\) is an attractor, i.e. that the root, \(x = 1\), may be obtained for this iteration scheme. Further, since \((x_{n+1} - 1)\) is proportional to \(\epsilon^1\) (rather than to \(\epsilon^2\), \(\epsilon^3\) or other powers) we may say that convergence is linear.

**Scheme B \((x = 1)\)**

Let \(x_n = 1 + \epsilon\) in \(x_{n+1} = \left(\frac{1}{7}(x_n - 6)\right)^{1/3}\). Hence

\[
x_{n+1} = \left[(7 + 7\epsilon) - 6\right]^{1/3} = (1 + 7\epsilon)^{1/3} \approx 1 + \frac{7}{3}\epsilon.
\]

Thus the small perturbation grows, and therefore we cannot achieve this root using this iteration scheme.

Now we’ll consider the \(x = 2\) root.

**Scheme A \((x = 2)\)**

Let \(x_n = 2 + \epsilon\). Hence

\[
x_{n+1} = \frac{(8 + 12\epsilon + 6\epsilon^2 + \epsilon^3) + 6}{7} = 2 + \frac{12}{7}\epsilon + \cdots.
\]

Therefore we see that the small perturbation has grown, and thus the root, \(x = 2\), is unstable for Scheme A, and cannot be achieved using the scheme.

**Scheme B \((x = 2)\)**

Let \(x_n = 2 + \epsilon\). Hence

\[
x_{n+1} = \left[7(2 + \epsilon) - 6\right]^{1/3} = \left[8 + 7\epsilon\right]^{1/3} = 2 + \frac{7}{12}\epsilon + \cdots
\]

Thus this root is stable for Scheme B, and may be found using it. Therefore we see that the small perturbation has decayed, and thus the root, \(x = 2\), is stable for Scheme B.

A similar pair of analyses for the third root, \(x = -3\), where we set \(x_n = -3 + \epsilon\), should yield \(x_{n+1} = -3 + \frac{27}{7}\epsilon\) for Scheme A and \(x_{n+1} = -3 + \frac{27}{12}\epsilon\) for Scheme B.

The following two Tables, which use \(\epsilon = \pm0.001\), illustrate the above analyses.
5.5. Analysis of ad hoc methods for repeated roots

Let us consider the solution of
\[ g(x) = x^3 - 3x + 2 = 0. \]

The roots are easily found to be \( x = 1, 1, -2 \). We will concentrate on the \( x = 1 \) root here, since it is repeated, and use \( x_n = 1 + \epsilon \). The equivalent Scheme A and Scheme B iteration schemes are now

**Scheme A**

\[ x_{n+1} = \frac{1}{3} (x_n^3 + 2), \]

**Scheme B**

\[ x_{n+1} = (3x_n - 2)^{1/3}. \]

For Scheme A we obtain,

\[ x_{n+1} = \frac{1}{3} (x_n^3 + 2) = \frac{1}{3} [(1 + \epsilon)^3 + 2] = \frac{1}{3} [3 + 3\epsilon + 3\epsilon^2 + \epsilon^3] = 1 + \epsilon + \epsilon^2 + \cdots. \]

We see that the coefficient of \( \epsilon \) hasn’t changed, but there is nevertheless a very very small \( O(\epsilon^2) \) change in the value of \( x_{n+1} \) as compared with \( x_n \). So if \( \epsilon \) is positive, then \( x_{n+1} \) has moved away from the root by an \( O(\epsilon^2) \) amount, and we have a very very slow divergence from the root. On the other hand, if \( \epsilon \) is negative, then the \( \epsilon \) term brings us very slightly closer to the root, and we have a painfully slow convergence from below the root.

For Scheme B the following happens.

\[ x_{n+1} = (3x_n - 2)^{1/3} \]

\[ = (1 + 3\epsilon)^{1/3} \]

\[ = 1 + \frac{1}{3} (3\epsilon) + \frac{(1/3)(-1/2)(3\epsilon)^2}{2} + \frac{(1/3)(-1/2)(-5/2)(3\epsilon)^3}{2 \times 3} + \cdots \text{ Binomial expansion} \]

\[ = 1 + \epsilon - \epsilon^2 + \cdots. \]

Therefore we get the opposite behaviour, namely very slow convergence when \( \epsilon > 0 \) and very slow divergence when \( \epsilon < 0 \).
In conclusion, we see that ad hoc schemes are easy to implement but perform poorly. Their usefulness lies in their ease of computation, but we are not guaranteed to be able to find all the roots for any one chosen iteration scheme, and when there are repeated roots, the methods perform exceptionally poorly. We really do need something better than this!

5.6. The Newton-Raphson method.

This is a very widely-used method for solving algebraic equations. It has the advantage of generally being very much faster than ad hoc methods because it converges quadratically. It is systematic, and always converges if the initial guess is close enough to the sought root. Although its speed of convergence deteriorates when looking for repeated roots, it nevertheless retains a reasonable speed since convergence is linear in such cases.

**Derivation of the method.**

Let \( x \) be the exact solution of \( f(x) = 0 \), and let

\[
x = x_n + \epsilon_n,
\]

where \( x_n \) is the current iterate (or even the initial guess) and \( \epsilon_n \) is the exact (though unknown) error in the current iterate. We may now use Taylor’s series to find an approximation for \( \epsilon_n \). We have

\[
0 = f(x) = f(x_n + \epsilon_n)
\]

by definition

\[
= f(x_n) + \epsilon_n f'(x_n) + \cdots
\]

we’ll ignore remaining terms

\[
\Rightarrow \quad \epsilon_n \simeq -\frac{f(x_n)}{f'(x_n)}.
\]

Now since we have ignored terms in \( \epsilon_n^2 \) and higher powers, this final expression for \( \epsilon_n \) is not exact, but it should be a good approximation. This means that \( x_n + \epsilon_n \) won’t be the exact root, but we may treat it as the next iterate, i.e. we set \( x_{n+1} = x_n + \epsilon_n \). So the Newton-Raphson method is

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \quad (5.1)
\]

It is important to note that the iteration scheme given in (5.1) yields the exact solution whenever \( f(x) \) is a linear function of \( x \). So if we were to have \( f(x) = mx + c \), and we were to choose \( x_0 \) randomly as the initial iterate, then (5.1) gives,

\[
x_1 = x_0 - \frac{mx_0 + c}{m} = -\frac{c}{m},
\]

which is clearly the root of \( mx + c = 0 \).

In fact, one may even derive (5.1) by assuming that \( f(x) \) is at least approximately linear. If we have the two values, \( x_n \) and \( x_{n+1} \), and the corresponding \( f \)-values, \( f(x_n) \) and \( f(x_{n+1}) \), then the slope of the straight line between the two points \((x_n, f(x_n))\) and \((x_{n+1}, f(x_{n+1}))\) is

\[
f'(x_n) = \frac{f(x_{n+1}) - f(x_n)}{x_{n+1} - x_n}.
\]

If we now set \( f(x_{n+1}) = 0 \) and rearrange this expression to make \( x_{n+1} \) the subject, then we obtain (5.1) immediately.
Application of the method.

Let us apply (5.1) to the cubic equation considered in §§5.3 and 5.4. On setting \( f(x) = x^3 - 7x + 6 \) we obtain

\[
x_{n+1} = x_n - \frac{x_n^3 - 7x_n + 6}{3x_n^2 - 7}
\]

(5.2)

or, in more compact form,

\[
x_{n+1} = \frac{2x_n^3 - 6}{3x_n^2 - 7}
\]

(5.3)

We obtain the following Table of results.

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_\infty )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2.400000</td>
<td>2.105837</td>
<td>2.011038</td>
<td>2.000143</td>
<td>2.000000</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0.857143</td>
<td>0.988450</td>
<td>0.999902</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1</td>
</tr>
</tbody>
</table>

Immediately we see that, not only is convergence much faster than with the ad hoc methods, but we also obtain all three roots. This is a general feature of the Newton-Raphson method.

Figure 5.4. Showing the manner in which the Newton-Raphson scheme approaches the root of a function. Here, \( f(x) = x^3 - 7x + 6 \), and the initial iterate is \( x_0 = 3 \).

Figure 5.4 shows the approach of successive iterates to the root \( x = 2 \). The method clearly finds where the tangent to the curve cuts the horizontal axis and this location gives the value of the next iterate.
5.7. Convergence of the Newton-Raphson method.

Let us consider convergence towards the root \( x = 1 \) in the above example. We begin in the same way as for the ad hoc methods, namely, by setting \( x_n = 1 + \epsilon \). Hence

\[
x_{n+1} = x_n - \frac{x_n^3 - 7x_n + 6}{3x_n^2 - 7}
\]

by definition

\[
= \frac{2x_n^3 - 6}{3x_n^2 - 7}
\]

\[
= \frac{2[1 + 3x + 3x^2 + \epsilon^3] - 6}{3[1 + 2x + x^2] - 7}
\]

note the similarity between the numerator and denominator

\[
= \frac{-4 + 6x + 6x^2 + \epsilon^3}{-4 + 6x + 3x^2} \quad \text{sneaky trick...}
\]

\[
= 1 + \frac{3x^2 + \epsilon^3}{-4 + 6x + 3x^2}
\]

\[
\simeq 1 - \frac{4}{3} \epsilon^2.
\]

Thus convergence is quadratic because the error in one iterate is proportional to the square of the error in the previous iterate, noting that this is formally true only when \( \epsilon \) is small, i.e. that one is close to the root. Thus if \( x_n \) has an error of approximately \( 10^{-3} \), then the error in the next iterate is roughly of a size comparable with \( 10^{-6} \), and the following one is comparable with \( 10^{-12} \).

Let us try the same analysis for the root, \( x = 2 \), by setting \( x_n = 2 + \epsilon \). We get

\[
x_{n+1} = \frac{2x_n^3 - 6}{3x_n^2 - 7}
\]

\[
= \frac{2[8 + 12x + 6x^2 + \epsilon^3] - 6}{3[4 + 4x + x^2] - 7}
\]

\[
= \frac{10 + 24x + 12x^2 + \epsilon^3}{5 + 12x + 3x^2}
\]

\[
= \frac{2 \left( \frac{5 + 12x + 6x^2 + \epsilon^3}{5 + 12x + 3x^2} \right)}{\frac{5 + 12x + 3x^2}{5 + 12x + 3x^2}}
\]

\[
= \frac{2 \left( 1 + \frac{3x^2 + \epsilon^3}{5 + 12x + 3x^2} \right)}{\frac{5 + 12x + 3x^2}{5 + 12x + 3x^2}} \quad \text{the same trick!}
\]

\[
\simeq 2 + \frac{6}{5} \epsilon^2.
\]

Again we obtain quadratic convergence. Don’t be put off by the fact that the coefficient of \( \epsilon^2 \) is greater than 1, because this is still a convergent scheme due to the \( \epsilon^2 \).

When we consider the third root, \( x = -3 \), then \( x_n = -3 + \epsilon \) gives \( x_{n+1} = -3 - \frac{3}{2} \epsilon^2 \). Again quadratic convergence.


These analyses explain the great speed-up which the Newton-Raphson method enjoys when compared with the ad hoc methods. But how does it fare when faced with repeated roots? Let us consider the solution of

\[
f(x) = x^3 - 3x + 2 = 0,
\]
which was also considered in §5.5. The roots are easily found to be \( x = 1, 1, -2 \). The Newton-Raphson method is

\[
x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}
\]

\[
x_n = \frac{x_n^3 - 3x_n + 2}{3x_n^2 - 3}
\]

\[
= \frac{2x_n^3 - 2}{3x_n^2 - 3} = \frac{2}{3} \frac{x_n^3 - 1}{x_n^2 - 1}
\]

\[
= \frac{2}{3} \frac{x_n^2 + x_n + 1}{x_n + 1}.
\]

\( (5.4) \)

Note that we cancelled the common factor \((x_n - 1)\) from both the numerator and denominator in the penultimate line above — in practice we wouldn’t do that because we would not know in advance that the repeated root exists. Thus we can see that there is a potential numerical problem with the formula (5.4) with both numerator and denominator becoming small as the repeated root is approached. This isn’t usually an insurmountable problem if the machine on which the computations are being done uses a large number of significant figures. If we now set \( x_n = 1 + \epsilon \) then

\[
x_{n+1} = \frac{2}{3} \left[ \frac{(1 + 2\epsilon + \epsilon^2) + (1 + \epsilon) + 1}{(1 + \epsilon) + 1} \right]
\]

\[
= \frac{2}{3} \left[ \frac{3 + 3\epsilon + \epsilon^2}{2 + \epsilon} \right]
\]

\[
= \frac{1 + \epsilon + \frac{1}{2}\epsilon^2}{1 + \frac{1}{2}\epsilon}
\]

\[
= (1 + \epsilon + \frac{1}{2}\epsilon^2)(1 - \frac{1}{2}\epsilon + \cdots)
\]

using the Binomial Theorem

\[
\simeq 1 + \frac{1}{2}\epsilon.
\]

(5.5)

Thus this example illustrates the fact that convergence is linear when applying the Newton-Raphson method to a twice-repeated root. This is also a general feature of the Newton-Raphson method but in practice there may be increasing difficulties with round-off errors because both the numerator and denominator of the iteration scheme tend to zero as one approaches the root.

Finally, for an even more highly repeated root, the Newton-Raphson method still manages to maintain linear convergence, but the speed of convergence deteriorates progressively as the number of repetitions of the root increases and problems associated with round-off error become greater.

5.9. Practicalities.

All of this seems excellent, but are there any drawbacks, or any hidden difficulties?

Well, one situation which will yield rather wild behaviour is if we happen to choose an iterate that is close to where \( f'(x) = 0 \), which is the term in the denominator in (5.1). For the present example this is where \( 3x^2 - 7 = 0 \), i.e. \( x = \pm \sqrt{7/3} = \pm 1.527525 \). Here are two cases:

\[
1.52 \rightarrow -14.8781 \rightarrow -10.0336 \rightarrow -6.8681 \rightarrow -4.8616 \rightarrow -3.6900 \rightarrow -3.1460 \rightarrow -3.0087
\]

and

\[
1.54 \rightarrow 11.3635 \rightarrow 7.6993 \rightarrow 5.3081 \rightarrow 3.7808 \rightarrow 2.8450 \rightarrow 2.3177 \rightarrow 2.0735
\]

When \( x_0 = 1.52 \) we need 9 iterations to obtain a correction which is less than \( 10^{-10} \) in magnitude, whereas it requires 10 when \( x = 1.54 \). As the initial iterate approaches 1.527525, then the number of iterations
increases. This arises because the curve, \( f(x) = 0 \), has a zero tangent there, and therefore the next iterate will be a very large distance away from \( x_0 \). We also see that there is a sudden change in the identity of the root which is obtained as \( x_0 \) passes 1.527525; below it we obtain the root, \(-3\), while above it we obtain 2.

These matters are demonstrated graphically in Fig. 5.5 where we see how the gradient of \( f(x) \) at the location of the initial iterate affects where the next iterate will be. Many locations are well-behaved in the sense that one clearly gets much closer to the nearest root. But the neighbourhoods of the extrema in \( f(x) \) will generate very large changes in the value of the iterate simply because \( f'(x) \) is close to zero.

![Figure 5.5](image.png)

Figure 5.5. Showing how the proximity of the initial iterate to a root affects the location of the next iterate. Here, \( f(x) = x^3 - 7x + 6 \), and the initial iterates \( x_0 = -3.3, -2.5, -1.55, -1.51, 0, 1.51, 1.55 \) and 3. The bullets show where the initial iterate is, and the dashed lines shown the predicted location of the next iterate.

5.10. Newton-Raphson for more than one unknown. [NOTE: this is not an examinable section of the unit.]

In many engineering contexts, which includes the numerical solutions of ordinary differential equations where more than one initial condition is unknown (and which is considered in ME20014, Modelling Techniques 1), it is necessary to solve \( N \) nonlinear algebraic equations in \( N \) unknowns. If these had been linear equations, then the equations could be arranged in matrix/vector form and Gaussian Elimination used to find the solutions. But when the equations (or even just one of them!) is nonlinear, then we need to use a multi-dimensional form of Newton-Raphson’s method.

One example of a pair of such equations was given right at the start of this set of notes. We will generalize and say that we wish to solve the pair,

\[
\begin{align*}
  f(x, y) &= 0, \\
  g(x, y) &= 0.
\end{align*}
\]

If we interpret \( f(x, y) \) as being the height above sea level and \( x \) and \( y \) to correspond to east and north, then \( f(x, y) = 0 \) will typically define a line in the \((x, y)\)-plane which we could interpret as the zero-height contour. Typically \( f \) will then be positive on one side of the line and negative on the other. The equation \( g(x, y) = 0 \) would also correspond to a line, a different one in general, and one which might intersect the one corresponding to \( f(x, y) = 0 \). The resulting intersection or intersections is/are the solution/s which we seek.
The derivation follows in roughly the same way as for the one-dimensional case. We assume that \(x\) and \(y\) are the exact solutions, and that our current iterates, \(x_n\) and \(y_n\) are in error by the quantities, \(\epsilon_n\) and \(\delta_n\). Hence,

\[
x = x_n + \epsilon_n, \quad y = y_n + \delta_n.
\]

Therefore we may say that

\[
f(x, y) = 0 \quad \Rightarrow \quad f(x_n + \epsilon_n, y_n + \delta_n) = 0.
\]

We may now expand this latter quantity as a 2D Taylor’s series, as follows:

\[
0 = f(x_n + \epsilon_n, y_n + \delta_n)
= f(x_n, y_n) + \epsilon_n \frac{\partial f}{\partial x}\bigg|_{x_n, y_n} + \delta_n \frac{\partial f}{\partial y}\bigg|_{x_n, y_n} + \cdots.
\]

This last step is equivalent to placing a tangent plane on the surface at the point, \((x_n, y_n)\). We will drop all the higher-order terms represented by the dots, and assume that the tangent plane is a good enough representation of the surface. Clearly the same may be done for \(g(x, y) = 0\) and we may write,

\[
0 = g(x_n + \epsilon_n, y_n + \delta_n)
= g(x_n, y_n) + \epsilon_n \frac{\partial g}{\partial x}\bigg|_{x_n, y_n} + \delta_n \frac{\partial g}{\partial y}\bigg|_{x_n, y_n} + \cdots.
\]

By neglecting the higher powers of \(\epsilon_n\) and \(\delta_n\), we have disentangled \(\epsilon_n\) and \(\delta\) from the original equations and have found an albeit approximate pair of linear equations which they satisfy. Therefore we may rewrite the equations for \(\epsilon_n\) and \(\delta_n\) in matrix/vector form:

\[
\begin{pmatrix}
\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\
\frac{\partial g}{\partial x} & \frac{\partial g}{\partial y}
\end{pmatrix}
\begin{pmatrix}
\epsilon_n \\
\delta_n
\end{pmatrix}
= -
\begin{pmatrix}
f \\
g
\end{pmatrix},
\]

where all quantities are evaluated at the current values of the iterates, \(x_n\) and \(y_n\). This matrix/vector equation is then solved, the next iterates defined according to

\[
x_{n+1} = x_n + \epsilon_n, \quad y_{n+1} = y_n + \delta_n,
\]

and the whole procedure is repeated until both \(|\epsilon_n|\) and \(|\delta_n|\) are sufficiently small.

We will choose to use the following as an example case. Let

\[
f(x, y) = y - 6x^3 + 6x, \quad g(x, y) = x - 6y^3 + 6y.
\]

A sketch of \(f = 0\) and \(g = 0\) could suggest that there are nine solutions. The true curves are shown below.
The Newton-Raphson iterations will be given by,

$$\begin{pmatrix} 6 - 18x_n^2 & 1 \\ 1 & 6 - 18y_n^2 \end{pmatrix} \begin{pmatrix} \epsilon_n \\ \delta_n \end{pmatrix} = - \begin{pmatrix} y_n - 6x_n^3 + 6x_n \\ x_n - 6y_n^3 + 6y_n \end{pmatrix},$$

where the values for $x_n$ and $y_n$ form the current iterate.

By this means we find the solutions:

Table 5.6. The nine solutions of Eq. (5.6).

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>1.080123</td>
<td>1.080123</td>
</tr>
<tr>
<td>-1.080123</td>
<td>-1.080123</td>
</tr>
<tr>
<td>0.912871</td>
<td>-0.912871</td>
</tr>
<tr>
<td>-0.912871</td>
<td>0.912871</td>
</tr>
<tr>
<td>0.985599</td>
<td>-0.169102</td>
</tr>
<tr>
<td>-0.985599</td>
<td>0.169102</td>
</tr>
<tr>
<td>0.169102</td>
<td>-0.985599</td>
</tr>
<tr>
<td>-0.169102</td>
<td>0.985599</td>
</tr>
</tbody>
</table>

Finally for this subsection, we generalize further to $N$ equations in $N$ unknowns. For convenience we shall alter the notation and therefore the equations which we shall be solving are,

$$f_1(x_1, x_2, \cdots x_N) = 0,$$
$$f_2(x_1, x_2, \cdots x_N) = 0,$$
$$f_3(x_1, x_2, \cdots x_N) = 0,$$
$$\vdots$$
$$f_N(x_1, x_2, \cdots x_N) = 0.$$

Here, $x_1$ to $x_N$ form the $N$ variables. And if we now denote the $n^{th}$ iterate of the $i^{th}$ variable by $x_i^{(n)}$, with
a similar notation for the corrections, $\epsilon$, then the Newton-Raphson method becomes,

$$
\begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N}
\end{pmatrix}
\begin{pmatrix}
\epsilon_1^{(n)} \\
\epsilon_2^{(n)} \\
\vdots \\
\epsilon_N^{(n)}
\end{pmatrix}
= -
\begin{pmatrix}
f_1 \\
f_2 \\
\vdots \\
f_N
\end{pmatrix}.
$$

These more complicated schemes retain quadratic convergence unless there are multiple zeros. Convergence difficulties and wild excursions of the iterates happen only when the determinant of the iteration matrix is close to zero. Practically, though, there are occasions when the Newton-Raphson scheme will converge even when it is given an appalling initial iterate, and others when it is difficult to obtain the solution one requires because the initial iterate needs to be in a small region of parameter space in order to avoid converging to an unwanted solution.

The largest system of this type which has arisen in my research involved 3591 equations in 3591 unknowns. The iteration scheme was solved using Gaussian Elimination. Larger systems (> 12000) have also arisen in my research, but the matrices were structured in such a way that block Gaussian elimination methods were used instead.
5.11. Numerical Integration: Introduction

One immediate question is, why bother? We already have tables of standard integrals, and, in any case, there are many computer algebra packages on the market, such as Mathematica, Macsyma and Maple, which can do it for us. There are three main answers: (i) sometimes the final integral is so very complicated that it is unwieldy, (ii) the indefinite integral of the function we are interested in does not exist, and (iii) the data we wish to integrate is discrete, such as data points drawn from an experiment. Of these three answers, the second and third arise very frequently, and a by-now familiar example of the second is

\[ \int_0^X e^{-x^2} \, dx. \]

When \( X \) is infinite there is a cunning way of showing that the integral is exactly \( \sqrt{\pi}/2 \) (see the Maths 1 problem sheet on volumes under surfaces), but no indefinite integral is actually known.

With regard to the third answer, discrete data will also be obtained experimentally when output is sampled. It is usually subject to some slight error which may be due to the poor resolution of the experiment or errors in reading dials or scales, for instance.

In this unit we will consider three different methods, (i) the rectangle rules, (ii) the trapezium rule, and (iii) Simpson’s rule. These rules are related to one another, as we will find out below. They involve different accuracies, a subject which we will discuss, and we will also see that it is not necessarily the best policy to use the method with the highest accuracy.

5.12. The rectangle rules

There are two of these, and they are illustrated below to find the integral of \( f(x) \) from \( x = 0 \) to \( x = X \).

![Rectangle Rule Diagram](image)

Figure 5.7 Showing the definition of the left hand rectangle rule where the areas of the strips form the numerical integral. There are \( N \) intervals of length \( h \) where, in general, \( x_n = x_0 + nh \).
Figure 5.8 Showing the definition of the right hand rectangle rule where the areas of the strips form the numerical integral. There are $N$ intervals of length $h$ where, in general, $x_n = x_0 + nh$.

Both rules are obtained by first dividing the range over which $x$ varies into $N$ equal intervals of length $h$. (Note that it is not essential to have strips of equal width, but it is easier to start this way.) Hence $X = Nh$ by definition. Rectangles are formed whose height corresponds the value of $f(x)$ on the left hand side of each strip (in the case of the left hand rule) or on the right (in the case of the right hand rule). For both rules we say that the numerical integral is the combined area of all the rectangles.

The values of $x$ which mark the boundaries of the strips are

$$x = x_0, x_1, x_2, x_3, x_4, \ldots, x_{N-2}, x_{N-1}, x_N$$

and we use $n$ as the index, which means that we let $x_n = nh$. For the left hand rectangle rule the area of the first few rectangles are $hf(x_0)$, $hf(x_1)$, $hf(x_2)$ and so on. If, for convenience, we denote $f(x_n)$ by $f_n$, then the total area is

$$I_N = h[f_0 + f_1 + f_2 + \cdots + f_{N-1}] = h \sum_{n=0}^{N-1} f_n. \quad (5.7)$$

Here we use the notation, $I_N$, to indicate that it is the numerical integral involving $N$ intervals.

The area given by the right hand rectangle rule is

$$I_N = h[f_1 + f_2 + f_3 + \cdots + f_N] = h \sum_{n=1}^{N} f_n. \quad (5.8)$$
It is clear that whatever errors are incurred by taking any particular steplength, $h$, will be reduced when $h$ is reduced. But the question now is, how quickly do these errors decrease?

In what follows I will assume the following summations.

$$\sum_{n=1}^{N} n = \frac{1}{2}N(N+1), \quad \sum_{n=1}^{N} n^2 = \frac{1}{2}N(N+1)(2N+1), \quad \sum_{n=1}^{N} n^3 = \frac{1}{4}N^2(N+1)^2.$$  

These results will be used here to illustrate aspects of numerical integration, and won’t be needed for the exam.

**Example 5.1** Consider the numerical evaluation of $\int_{0}^{1} x\,dx$.

We can, of course, use a calculator or a computer to do this, but, for now, we will use entirely analytical techniques because they will allow us to see easily how accurate the numerical integral is.

In this case we have $Nh = 1$, and $f_n = f(x_n) = x_n = nh$. Hence the right hand rectangle rule gives

$$I_N = h \sum_{n=1}^{N} f_n$$

by definition

$$= h \sum_{n=1}^{N} nh$$

since $f_n = nh$

$$= h^2 \sum_{n=1}^{N} n$$

$$= h^2 \left[ \frac{1}{2}N(N+1) \right]$$

see above

$$= \frac{1}{2}(Nh)(Nh + h)$$

$$= \frac{1}{3}(1 + h).$$

Recall that $Nh = 1$ here

This value is what we would compute on a calculator or computer. The analytical result is $I = \frac{1}{2}$ and therefore the numerical error is a term of magnitude, $h$. Other functions which are sufficiently well-behaved also have an error which is proportional to $h^2$, and therefore the method is said to be **1st order accurate**, since the lowest power of $h$ in the error is 1. Such a method is also referred to as having an $O(h)$ error (read as “order $h$”).

The left hand rectangular rule yields $I_N = \frac{1}{2}(1 - h)$; this is left as an exercise. Clearly it too is of first order accuracy since the error is of $O(h)$.

**Example 5.2** Let us try the right hand rule on $\int_{0}^{1} x^2\,dx$.

Here $f_n = (nh)^2$. Hence we have

$$I_N = h \sum_{n=1}^{N} n^2 h^2 = h^3 \sum_{n=1}^{N} n^2$$

$$= h^3 \frac{1}{6}N(N+1)(2N+1)$$

again see above

$$= \frac{1}{6}(Nh)(Nh + h)(2Nh + h)$$

$$= \frac{1}{6}(1 + h)(2 + h) = \frac{1}{3}(1 + \frac{3}{2}h + \frac{1}{2}h^2).$$

The numerical integral also has an $O(h)$ error, but the leading term, $\frac{1}{3}$, is the correct analytical solution.
5.13. **The trapezium rule.**

This rule is, in fact, the average of the left and right hand rectangle rules. It is illustrated by the following Figure.

![Trapezium Rule Diagram](image_url)

Given the trapezia shown above, the area of each one is $h$ multiplied by the mean height, and therefore the Trapezium rule is

$$I_N = \frac{1}{2}(f_0 + f_1)h + \frac{1}{2}(f_2 + f_3)h + \cdots + \frac{1}{2}(f_{N-1} + f_N) = h\left[\frac{1}{2}f_0 + \sum_{n=1}^{N-1} f_n + \frac{1}{2}f_N\right].$$

**Example 5.3** Use the Trapezium rule to integrate $f(x) = x$ between 0 and 1.

Given our earlier statement that the Trapezium rule is the mean of the rectangle rules, we would expect this numerical solution to be exact. But let us try it out....

$$\int_0^1 x \, dx \approx h \left[\frac{1}{2}(0) + \sum_{n=1}^{N-1} nh + \frac{1}{2}(1)\right] = h \left[\frac{1}{2}(N - 1)N + \frac{1}{2}\right] = \frac{1}{2}Nh = \frac{1}{2}.$$ 

Hence the answer is exact in this case, as predicted.
Example 5.4 Use the trapezium rule to integrate \( f(x) = x^2 \) between 0 and 1.

\[
\int_0^1 x^2 \, dx = h \left[ \frac{1}{2} (0) + \sum_{n=1}^{N-1} (nh)^2 + \frac{1}{2} \right]
\]

\[
= \frac{1}{2} h + h^3 \sum_{n=1}^{N-1} n^2
\]

\[
= \frac{1}{2} h + h^3 \left[ \frac{(N-1)N(2N-1)}{6} \right]
\]

\[
= \frac{1}{2} h + h^3 \left[ \frac{2N^3 - 3N^2 + N}{6} \right]
\]

\[
= \frac{1}{2} h + \left[ \frac{1}{16} - \frac{1}{2} h + \frac{1}{8} h^2 \right]
\]

\[
= \frac{1}{3} + \frac{1}{8} h^2.
\]

We now have an error which is \( O(h^2) \), and therefore this method is 2nd order accurate. Again, it is important to note that the analytical value, \( \frac{1}{3} + \frac{1}{8} h^2 \), is what we would normally compute, but I have used an analytical method of doing the numerical integration in order to show how large the numerical errors are with these integration rules.

5.14. Simpson’s rule

The formula for this appears to be a little strange, but there is a good reason for it. Simpson’s rule applies to an interval with an even number of subintervals. Its definition is

\[
I_N = \frac{1}{3} h \left[ f_0 + 4 f_1 + 2 f_2 + 4 f_3 + 2 f_4 + \cdots + 2 f_{N-2} + 4 f_{N-1} + f_N \right].
\]

An alternative form which may be easier to remember collects the even-numbered and the odd-numbered terms together:

\[
I_N = \frac{1}{3} h \left[ f_0 + f_N + 4 \left( f_1 + f_3 + f_5 + \cdots + f_{N-1} \right) + 2 \left( f_2 + f_4 + \cdots + f_{N-2} \right) \right].
\]

Example 5.5 Evaluate \( \int_0^1 x^3 \, dx \) using Simpson’s rule over 4 intervals.

We will shy away from doing this analytically (it can be done, but it uses quite a lot of space and you need to keep your wits about you....), and content ourselves with some numbers. Of course 4 intervals from 0 to 1 means that \( h = \frac{1}{4} \).

\[
I_4 = \frac{1}{3} h \left[ 0^3 + 4(0.25)^3 + 2(0.5)^3 + 4(0.75)^3 + 1^3 \right] = 0.25.
\]

This answer is exactly correct.

Example 5.6 Evaluate \( \int_0^1 x^4 \, dx \) using Simpson’s rule over 4 and over 8 intervals.

We won’t write out the formula, but the answers are

\[
I_4 = 0.200521 \quad \text{and} \quad I_8 = 0.200032.
\]

Clearly the answers are not exact this time, but it is important to notice that the error in \( I_4 \) is approximately 16 times the error in \( I_8 \). This is what we would expect for a 4th order accurate method since the error is proportional to \( h^4 \). Therefore, when the steplength is halved, the error is reduced by a factor of \( \left( \frac{1}{2} \right)^4 = \frac{1}{16} \).

This observation leads on to....
5.15. Richardson’s extrapolation.

This is a method whereby the accuracy of a large class of numerical methods may be improved. It is applied most frequently to numerical integration and the numerical solution of ordinary and partial differential equations. (Integration is, of course, a special case of the solution of ODEs.)

It is based on the fact that most numerical methods have errors which proceed as a series in integer powers of \( h \), the steplength. We have already seen that the Trapezium rule has an \( O(h^2) \) error. In practice it is usual that

\[
I_N = I_{\text{exact}} + Ah^2 + Bh^3 + Ch^6 + \cdots,
\]

i.e. the errors proceed in even powers of \( h \). The proof of this fact is outside the scope of this unit. For the Rectangle rules we have

\[
I_N = I_{\text{exact}} + Ah + Bh^2 + Ch^3 + \cdots.
\]

Let us check the actual errors in a Trapezium rule integration:

**Example 5.7** Find the errors in computing \( I = \int_0^1 (1 + x)^{1/3} \, dx \) using the Trapezium rule for 1, 2, 4 and 8 intervals.

Again \( Nh = 1 \), but the exact integral is \( I = \frac{3}{4}(2^{4/3} - 1) = 1.139882 \). We obtain the following Table of results.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( I_N )</th>
<th>( \text{Err} = I - I_N )</th>
<th>( \text{Err}/h^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.129961</td>
<td>-0.009921</td>
<td>-0.009921</td>
</tr>
<tr>
<td>2</td>
<td>1.137337</td>
<td>-0.002545</td>
<td>-0.0010180</td>
</tr>
<tr>
<td>4</td>
<td>1.139241</td>
<td>-0.000641</td>
<td>-0.000256</td>
</tr>
<tr>
<td>8</td>
<td>1.139721</td>
<td>-0.000161</td>
<td>-0.00010304</td>
</tr>
</tbody>
</table>

The third column shows that, when the steplength is halved (i.e. the number of intervals is doubled), the error is quartered. This is confirmed in the fourth column which gives the coefficient of \( h^2 \) in the error series.

Let us consider how we may remove this error term. We will consider a general 2nd order method for now, but it may be generalised easily to other orders of accuracy. Therefore, with \( N \) steps of length \( h \), the numerical solution may be written in the form,

\[
I_N = I + Ah^2 + Bh^3 + \cdots. \tag{5.9}
\]

If this is true for an essentially arbitrary steplength, then it is true when we have \( 2N \) steps of length \( \frac{1}{2}h \). Hence

\[
I_{2N} = I + A\left(\frac{1}{2}h\right)^2 + B\left(\frac{1}{2}h\right)^3 + \cdots. \tag{5.10}
\]

If we now multiply Eq. (5.10) by 4 and subtract Eq. (5.9) then the \( Ah^2 \) term will be removed. Therefore we get

\[
4I_{2N} - I_N = 3I + \frac{1}{6}h^3B + \cdots.
\]

Therefore we define the **Richardson Extrapolate**, \( T_{2N} \), as

\[
T_{2N} = \frac{4I_{2N} - I_N}{3} \quad \left( \approx I - \frac{1}{6}h^3B + \cdots \right).
\]

By design, the Richardson Extrapolate is at least 3rd order accurate, and, for the Trapezium rule it is actually 4th order accurate. Note that the application of Richardson’s extrapolation to numerical integration is called **Romberg Integration**.

Let us apply this result to the data obtained in Example 5.7.
Table 5.8. Table of results for a Richardson Extrapolation application

<table>
<thead>
<tr>
<th>$N$</th>
<th>$I_N$</th>
<th>$T_N$</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.129961</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>1.137337</td>
<td>1.139796</td>
<td>−0.000086</td>
</tr>
<tr>
<td>4</td>
<td>1.139241</td>
<td>1.139876</td>
<td>−0.000006</td>
</tr>
<tr>
<td>8</td>
<td>1.139721</td>
<td>1.139881</td>
<td>+0.000001</td>
</tr>
</tbody>
</table>

Note the huge improvement in accuracy! Even the $T_2$ value is more accurate than $I_8$ is in this case.

This leads on to a very practical reason for using Richardson’s Extrapolation. If the equivalent of $N = 2$ above is a computation which takes 1 hour, and if the CPU time of such computations increase as the cube of the number of grid points (this is fairly typical), then $N = 1$ is equivalent to $\frac{1}{3}$ hour, $N = 4$ to 8 hours and $N = 8$ to 64 hours. In such a case two runs with $N = 1$ and $N = 2$ will take 1.125 hour, and has a greater accuracy than a single $N = 8$ run of 64 hours.

For other orders of accuracy we may also find the equivalent Richardson’s Extrapolation formula. If the method is $O(\alpha)$, i.e. the error is proportional to $h^\alpha$, then

$$I_N = I + Ah^\alpha + \cdots \quad \text{and} \quad I_{2N} = I + A\left(\frac{1}{2}h\right)^\alpha + \cdots.$$  \hfill (5.11)

Hence

$$2^\alpha I_{2N} - I_N = (2^\alpha - 1)I + \cdots,$$

and therefore we may define the extrapolate as

$$T_{2N} = \frac{2^\alpha I_{2N} - I_N}{2^\alpha - 1}.$$  \hfill (5.12)

For an $O(h)$ method we have $\alpha = 1$ and so

$$T_{2N} = 2I_{2N} - I_N,$$

while for an $O(h^3)$ method it is,

$$T_{2N} = \frac{8I_{2N} - I_N}{7},$$

and for a fourth order method it is,

$$T_{2N} = \frac{16I_{2N} - I_N}{15}.$$  

Note that the value of $\alpha$ will always be a positive integer.

5.16. Nested extrapolations.

It is quite possible to use Richardson’s Extrapolation again on data which has already been obtained by using Richardson’s Extrapolation. For example, if one of the rectangle rules with, say, 10, 20 and 40 intervals, had been used to obtain three (1st order accurate) approximations to an integral, then R.E. may be applied to the $N = 10$ and $N = 20$ data (with $\alpha = 1$) to obtain a 2nd order accurate value. It may also be applied to the $N = 20$ and $N = 40$ data to obtain a second 2nd order accurate estimate. Now we may apply R.E. to these extrapolates (with $\alpha = 2$) to find a value which has at least 3rd order accuracy. Here is an example.
Example 5.8 A numerical scheme for solving the ordinary differential equation, \( y' = -y \), subject to the initial condition, \( y(0) = 1 \), in the range \( 0 \leq t \leq 1 \) using \( N \) intervals of length, \( h \), may actually be solved analytically. This analytical solution of a numerical problem takes the form,

\[
y_N = (1 - h)^N.
\]

The analytical solution is \( y = e^{-t} \). Compare the analytical solution with the exact solution at \( t = 1 \), and use Richardson Extrapolation multiple times to find a highly accurate numerical approximation to \( y \) at \( t = 1 \).

In the first instance we obtain the following data.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( y_N )</th>
<th>( y(1) - y_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.2500000000</td>
<td>0.1178794412</td>
</tr>
<tr>
<td>4</td>
<td>0.3164062500</td>
<td>0.0514731912</td>
</tr>
<tr>
<td>8</td>
<td>0.3436089158</td>
<td>0.0242705254</td>
</tr>
<tr>
<td>16</td>
<td>0.3560741305</td>
<td>0.0118053107</td>
</tr>
<tr>
<td>32</td>
<td>0.3620552893</td>
<td>0.0058241519</td>
</tr>
<tr>
<td>64</td>
<td>0.3649865242</td>
<td>0.0028929169</td>
</tr>
</tbody>
</table>

We have the exact solution to compare with, which is a luxury, but it is clear to see that the error roughly halves as the value of \( h \) halves. Therefore we shall use the Richardson Extrapolation formula (5.12) using \( \alpha = 1 \). So \( \overline{y}_N = 2y_N - y_N \). This gives us the following Table.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \overline{y}_N )</th>
<th>( y(1) - \overline{y}_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.3828125000</td>
<td>-0.0149330588</td>
</tr>
<tr>
<td>8</td>
<td>0.3708115816</td>
<td>-0.0029321404</td>
</tr>
<tr>
<td>16</td>
<td>0.3685393451</td>
<td>-0.0006599039</td>
</tr>
<tr>
<td>32</td>
<td>0.3686364481</td>
<td>-0.0001570069</td>
</tr>
<tr>
<td>64</td>
<td>0.3679177592</td>
<td>-0.0000383181</td>
</tr>
</tbody>
</table>

The errors in these extrapolates now appear to reduce by a factor of 4 when the step, \( h \), reduces by a factor of 2. Therefore we may apply Richardson Extrapolation to these extrapolates, but with \( \alpha = 2 \) in (5.12). We use \( \overline{\overline{y}}_N = (4\overline{y}_N - \overline{y}_N)/3 \). The following Table is obtained.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \overline{\overline{y}}_N )</th>
<th>( y(1) - \overline{\overline{y}}_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.3668112755</td>
<td>0.0010681657</td>
</tr>
<tr>
<td>16</td>
<td>0.3677819329</td>
<td>0.0000975082</td>
</tr>
<tr>
<td>32</td>
<td>0.3678688157</td>
<td>0.0000106255</td>
</tr>
<tr>
<td>64</td>
<td>0.3678781963</td>
<td>0.0000012449</td>
</tr>
</tbody>
</table>

These errors now reduce by a factor of 8 as \( h \) is halved, so these second extrapolates do indeed now have third order. So we may, once more, perform Richardson Extrapolation on them with \( \alpha = 3 \).
Table 5.12. The third extrapolates for the ODE solution

<table>
<thead>
<tr>
<th>$N$</th>
<th>$y_N$</th>
<th>$y(1) - y_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>0.3679205983</td>
<td>–0.0000411571</td>
</tr>
<tr>
<td>16</td>
<td>0.3678812275</td>
<td>–0.0000017864</td>
</tr>
<tr>
<td>32</td>
<td>0.3678795364</td>
<td>–0.0000000952</td>
</tr>
</tbody>
</table>

We will not proceed any further with this even though it is possible. It is interesting to compare the errors in, say, $y_{16}$, $y_{16}$ and $y_{N_{16}}$, to see how the accuracy of the raw data has been improved without having to reduce $h$ below $\frac{1}{16}$.

Cautionary note: In this last example we have performed Richardson Extrapolation three times. This is very unusual, and only works because the original raw data was virtually exact. In many cases the data could be subject to noise, and the decision for how many nestings of Richardson Extrapolation can be made will depend on how good the raw data in that regard. In some cases it ought not to be done at all.

5.17. Final comments on numerical integration and Richardson’s Extrapolation.

The following comments are for general information, and do not form part of the examinable syllabus.

The decision as to which numerical integration scheme is to be applied depends very much on the nature of the data given. If the data are exact, or at least exact to a good number of significant figures, then it is best to use Simpson’s method because it is fourth order accurate and uses almost exactly the same amount of arithmetic as the less accurate methods. If the data comes from experimental work, then it will have measurement errors. These errors may be one-sided (i.e. all positive or all negative) or may be bounded but centred around zero. It is quite possible to devise an error profile for which the Trapezium rule gives good results, whereas Simpson’s rule gives relatively poor results (consider small errors of equal magnitude but which alternate in sign from point to point). Thus, in practice, it is difficult to choose. For data which has been obtained numerically then it is pointless using a 4th order integrator when the basic data is only 2nd order accurate, as it will not convey any advantage over the use of a 2nd order integrator. Conversely, if the basic data is 4th order accurate, then useful information is lost when using a 2nd order integrator. It is best to match the orders of accuracy.

I have not yet mentioned where Simpson’s rule comes from. In one of the exercises on the problem sheet you are invited to use Richardson’s Extrapolation on the Trapezium rule. If it is done correctly, then Simpson’s rule should be obtained.

There is a second Simpson’s rule which is quoted in some textbooks. Again it is 4th order accurate, but it requires that the number of intervals to be a multiple of 3. See if you can track down this formula.

5.18. The ratio test to determine the order of accuracy.

In Example 5.8 we had access to the exact solution, and therefore it was possible to determine what the order of accuracy is from the comparison between the exact solution and the numerical approximations. Sometimes, most of the time actually, one does not have the exact solution for comparison. What should one do then to determine the order of accuracy?

Returning to Eq. (5.11), which is

$$I_N = I_{\text{exact}} + Ah^\alpha + \cdots,$$

(5.13)

we note that there are three unknown values, namely $I_{\text{exact}}$, $A$ and $\alpha$. We are only interested in $\alpha$ for now. We may remove $I_{\text{exact}}$ and $A$ in the following way. Let us apply the formula (5.13) for $2N$ steps of length, $h/2$, and for $4N$ steps of length, $h/4$. We obtain,

$$I_{2N} = I_{\text{exact}} + A(h/2)^\alpha + \cdots,$$

(5.14)
and
\[ I_{4N} = I_{\text{exact}} + A(h/4)^\alpha + \cdots. \] (5.15)

The subtraction of (5.14) from (5.13) removes \( I_{\text{exact}} \):
\[ I_N - I_{2N} = Ah^2 \left( 1 - \frac{1}{2^\alpha} \right) + \cdots. \] (5.16)

A similar subtraction of (5.15) from (5.14) gives,
\[ I_{2N} - I_{4N} = Ah^2 \left( \frac{1}{2^\alpha} - \frac{1}{4^\alpha} \right) + \cdots = \frac{Ah^2}{2^\alpha} \left( 1 - \frac{1}{2^\alpha} \right) + \cdots. \] (5.17)

Now the quotient of Eqs. (5.16) and (5.17) gives,
\[ \frac{I_N - I_{2N}}{I_{2N} - I_{4N}} = 2^\alpha + \cdots. \] (5.18)

In this analysis the dots have represented higher powers of \( h \) and it is hoped that these terms are small relative to those which have been retained. The implication of Eq. (5.18) is that, when \( h \) is sufficiently small, then the ratio on the left hand side should be close to an obvious integer power of 2. If it is isn’t, or if it is but another application of the ratio test using a finer set of grids changes the predicted value of \( 2^\alpha \) substantially, then the grid is not yet sufficiently fine.

**Example 5.9** The following set of data has been obtained using a numerical method. What is the order of accuracy of the method? When that has been found, use Richardson Extrapolation to improve the accuracy. Use the ratio test on the extrapolates to determine their order of accuracy.

The following Table gives the raw data, the results of the ratio test and the first Richardson Extrapolation based upon that test.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( I_N )</th>
<th>ratio test</th>
<th>( T_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6.141593</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>2.423949</td>
<td>–</td>
<td>1.892857</td>
</tr>
<tr>
<td>8</td>
<td>2.050064</td>
<td>9.943</td>
<td>1.996652</td>
</tr>
<tr>
<td>16</td>
<td>2.006166</td>
<td>8.517</td>
<td>1.999895</td>
</tr>
<tr>
<td>32</td>
<td>2.000768</td>
<td>8.132</td>
<td>1.999997</td>
</tr>
</tbody>
</table>

First of all, it is quite clear from the \( I_N \) column that the method for producing \( I_N \) must be of fairly high order. The value for \( I_2 \) is clearly almost complete nonsense, but as \( N \) doubles successively then there is a very rapid convergence towards 2.

The ratio test appears to be quite consistent despite how large \( I_2 \) is. The nearest power of 2 to these three ratios is the third power. Therefore the data appears to suggest quite strongly that a third order method was used to provide the raw data.

The final column gives the Richardson Extrapolates using \( \alpha = 3 \). These appear to converge even more quickly towards 2. If one were to apply the ratio test to these extrapolates, then both the values which could be obtained yield a ratio of almost exactly 32, and therefore the extrapolates themselves are of 5th order accuracy.