MA385 – Numerical Analysis I ("NA1")

Niall Madden

October 25, 2017

Chapter 0

MA385: Preliminaries

0.1 Introduction

0.1.1 Welcome to MA385 ("NA1")

This is a Semester 1, upper level module on *numerical analysis*. It is often taken in conjunction with MA378 ("NA2"). You may be taking this course as part of your degree in

- Bachelor of Science in Mathematics, Applied Mathematics and/or Computer Science;
- Denominated B.Sc. in Mathematical Science, Financial Mathematics, or Computer Science and Information Technology;
- Graduate programme in Applied Science or Data Analytics.

The basic information for the course is as follows.

Lecturer: Dr Niall Madden, School of Maths. My office is in room ADB-1013, Arás de Brún.
Email: Niall.Madden@NUIGalway.ie

Lectures: Monday at 9 and Thursday at 3, in AC201.

Tutorial/Lab: TBA (to begin during Week 3)

Assessment:
- Two written assignments (20%)
- Three computer labs (10%)
- One in-class test in Week 6 [tbc] (10%)
- Two-hour exam at the end of semester (60%)

The main text-book is **Süli and Mayers, An Introduction to Numerical Analysis**, Cambridge University Press [1]. This is available from the library at 519.4 MAY, and there are copies in the bookshop. It is very well suited to this course: though it does not over complicate the material, it approaches topics with a reasonable amount of rigour. There is a good selection of interesting problems. The scope of the book is almost perfect for the course, especially for those students taking both semesters. *You should buy this book.*

Other useful books include

- G.W. Stewart, *Afternotes on Numerical Analysis*, SIAM [3]. In the library at 519.4 STE. Moreover, *the full text is freely available online to NUI Galway users!* This book is very readable, and suited to students who would enjoy a bit more discussion.
- Cleve Moler, *Numerical Computing with MATLAB* [2]. The emphasis is on the implementation of algorithms in MATLAB, but the techniques are well explained and there are some nice exercises. Also, it is freely available online.
- James F Epperson, *An Introduction to Numerical Methods and Analysis*,[5]. There are five copies in the library at 519.4. It is particularly good a motivating *why* we study particular numerical methods.
- Stoer and Bulirsch,*Introduction to Numerical Analysis* [6]. A very complete reference for this course.

Web site:

The on-line content for the course will be hosted at http://www.maths.nuigalway.ie/MA385. There you'll find various pieces of these notes, copies of slides, problem sets, and lab sheets,

We will also use the MA385 module on BlackBoard for announcements, emails, and the Grade Book. If you are registered for MA385, you should be automatically enrolled onto the blackboard site. If you are enrolled in MA530, please send an email to me.

These notes are a synopsis of the course material. My aim is to provide these in three main sections, and always in advance of the class. They contain most of the main remarks, statements of theorems, results and exercises. However, they will not contain proofs of theorems, examples, solutions to exercises, etc. Please let me know of the typos and mistakes that you spot.

You should try to bring these notes to class. It will make following the lecture easier, and you'll know what notes to take down.

0.1.2 What is Numerical Analysis?

Numerical analysis is the design, analysis and implementation of numerical methods that yield *exact* or *approximate* solutions to mathematical problems.

It does not involve long, tedious calculations. We won't (usually) implement Newton's Method by hand, or manually do the arithmetic of Gaussian Elimination, etc.

The *Design* of a numerical method is perhaps the most interesting; its often about finding a clever way swapping the problem for one that is easier to solve, but has the same or similar solution. If the two problems have the same solution, then the method is *exact*. If they are similar (but not the same), then it is *approximate*.

The *Analysis* is the mathematical part; its usually culminates in proving a theorem that tells us (at least) one of the following

- The method will work: that our algorithm will yield the solution we are looking for;
- how much effort is required;
- if the method is approximate, determine how close the true solution be to the real one. A description of this aspect of the course, to quote the Epperson [5], is being "*rigorously imprecise* or *approximately precise*".

We'll look at the implementation of the methods in labs.

Topics

0. We'll preface the course with a review of Taylor's theorem. It is central to the algorithms of the following sections.
1. Root-finding and solving non-linear equations.
2. Initial value ordinary differential equations.
3. Matrix Algorithms: solving systems of linear equations and estimating eigenvalues.

We also see how these methods can be applied to real-world, including Financial Mathematics.

Learning outcomes

When you have successfully completed this course, you will be able to demonstrate your factual knowledge of the core topics (root-finding, solving ODEs, solving linear systems, estimating eigenvalues), using appropriate mathematical syntax and terminology.

Moreover, you will be able to describe the fundamental principles of the concepts (e.g., Taylor's Theorem) underpinning Numerical Analysis. Then, you will apply these principles to design algorithms for solving mathematical problems, and discover the properties of these algorithms. course to solve problems.

Students will gain the ability to use a MATLAB to implement these algorithms, and adapt the codes for more general problems, and for new techniques.

Mathematical Preliminaries

Anyone who can remember their first and second years of analysis and algebra should be well prepared for this module. Students who know a little about differential equations (initial value and boundary value) will find a certain sections (particularly in Semester II) somewhat easier than those who haven't.

If its been a while since you covered basic calculus, you will find it very helpful to revise the following: the Intermediate Value Theorem; Rolle's Theorem; The Mean Value Theorem; **Taylor's Theorem**, and the triangle inequality: $|a+b| \leqslant |a|+|b|$. You'll find them in any good text book, e.g., Appendix 1 of Süli and Mayers.

You'll also find it helpful to recall some basic linear algebra, particularly relating to eigenvalues and eigenvectors. Consider the statement: "all the eigenvalues of a real symmetric matrix are real". If are unsure what the meaning of any of the terms used, or if you didn't know that its true, you should have a look at a book on Linear Algebra.

0.1.3 Why take this course?

Many industry and academic environments require graduates who can solve real-world problems using a mathematical model, but these models can often only be resolved using numerical methods. To quote one Financial Engineer: "We prefer approximate (numerical) solutions to exact models rather than exact solutions to simplified models".

Another expert, who leads a group in fund management with DB London, when asked "what sort of graduates would you hire", the list of specific skills included

- A programming language and a 4th-generation language such as MATLAB (or S-PLUS).
- Numerical Analysis

Graduates of our Financial Mathematics, Computing and Mathematics degrees often report to us that they were hired because that had some numerical analysis background, or were required to go and learn some before they could do some proper work. This is particularly true in the financial sector, games development, and mathematics civil services (e.g., MET office, CSO).

Bibliography

[1] E Süli and D Mayers, *An Introduction to Numerical Analysis*, 2003. 519.4 MAY.

[2] Cleve Moler, *Numerical Computing with MATLAB*, Cambridge University Press. Also available free from http://www.mathworks.com/moler

[3] G.W. Stewart, *Afternotes on Numerical Analysis*, SIAM, 1996. 519.4 STE.

[4] G.W. Stewart, *Afternotes goes to Graduate School*, SIAM, 1998. 519.4 STE.

[5] James F Epperson, *An introduction to numerical methods and analysis*. 519.4EPP

[6] Stoer and Bulirsch, *An Introduction to Numerical Analysis*, Springer.

[7] Michelle Schatzman, *Numerical Analysis: a mathematical introduction*, 515 SCH.

0.2 Taylor's Theorem

Taylor's theorem is perhaps the most important mathematical tool in Numerical Analysis. Providing we can evaluate the derivatives of a given function at some point, it gives us a way of approximating the function by a polynomial.

Working with polynomials, particularly ones of degree 3 or less, is much easier than working with arbitrary functions. For example, polynomials are easy to differentiate and integrate. Most importantly for the next section of this course, their zeros are easy to find.

Our study of Taylor's theorem starts with one of the first theoretical results you learned in university mathematics: *the mean value theorem*.

. . . . . .

Theorem 0.2.1 (Mean Value Theorem)**.** *If* f *is function that is continuous and differentiable for all* $a \leqslant x \leqslant b$*, then there is a point* $c \in [a, b]$ *such that*

$$\frac{f(b) - f(a)}{b - a} = f'(c).$$

This is just a consequence of Rolle's Theorem, and has few different interpretations. One is that the slope of the line that intersects f at the points a and b is equal to the slope of the tangent to f at some point between a and b.

Take notes:

There are many important consequences of the MVT, some of which we'll return to later. Right now, we interested in the fact that the MVT tells us that we can approximate the value of a function by a near-by value, with accuracy that depends on f':

Take notes:

Or we can think of it as approximating f by a line:
Take notes:

.....

But what if we want a better approximation? We could replace our function with, say, a quadratic polynomial. Let $p_2(x) = b_0 + b_1(x-a) + b_2(x-a)^2$ and solve for the coefficients b_0, b_1 and b_2 so that

$$p_2(a) = f(a), \quad p_2'(a) = f'(a), \quad p_2''(a) = f''(a).$$

Take notes:

This gives that

$$p_2(x) = f(a) + f'(a)(x-a) + (x-a)^2 \frac{f''(a)}{2}.$$

Next, if we try to construct an approximating cubic of the form

$$\begin{aligned} p_3(x) &= b_0 + b_1(x-a) + b_2(x-a)^2 + b_3(x-a)^3, \\ &= \sum_{k=0}^{3} b_k (x-a)^k, \end{aligned}$$

with the property that

$$\begin{aligned} p_3(a) &= f(a), & p_3'(a) &= f'(a), \\ p_3''(a) &= f''(a), & p_3'''(a) &= f'''(a). \end{aligned} \tag{0.2.1}$$

Note: we can write (0.2.1) in a more succinct way, using the mathematical short-hand:

$$p_3^{(k)}(a) = f^{(k)}(a) \quad \text{for} \quad k = 0, 1, 2, 3.$$

Again we find that

$$b_k = \frac{f^{(k)}}{k!} \quad \text{for} \quad k = 0, 1, 2, 3.$$

As you can probably guess, this formula can be easily extended for arbitrary k, giving us the *Taylor Polynomial*.

Definition 0.2.2 (Taylor Polynomial)**.** The *Taylor*[1] *Polynomial* of degree k (also called the *Truncated Taylor Series*) that approximates the function f about the point $x = a$ is

$$\begin{aligned} p_k(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2} f''(a) + \\ \frac{(x-a)^3}{3!} f'''(a) + \cdots + \frac{(x-a)^k}{k!} f^{(k)}(a). \end{aligned}$$

We'll return to this topic later, with a particular emphasis on quantifying the "error" in the Taylor Polynomial.

Example 0.2.3. Write down the Taylor polynomial of degree k that approximates $f(x) = e^x$ about the point $x = 0$.

Take notes:

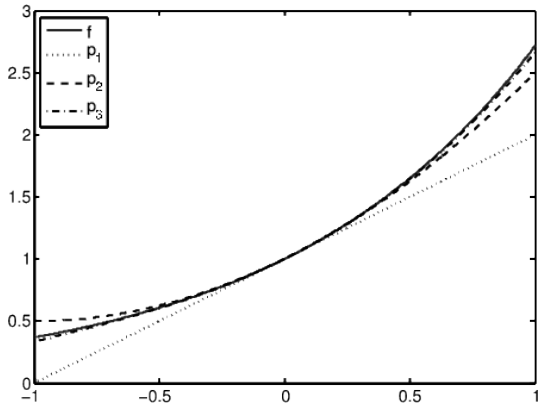


Fig. 0.1: Taylor polynomials for $f(x) = e^x$ about $x = 0$

As Figure 0.1 suggests, in this case $p_3(x)$ is a more accurate estimation of e^x than $p_2(x)$, which is more accurate than $p_1(x)$. This is demonstrated in Figure 0.2 were it is shown the difference between f and p_k.

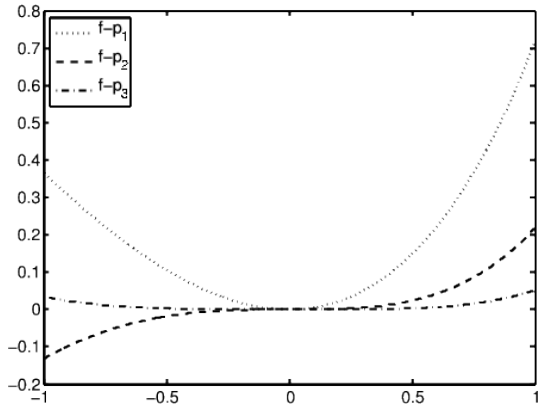


Fig. 0.2: Error in Taylor polys for $f(x) = e^x$ about $x = 0$

[1] Brook Taylor, 1865 – 1731, England. He (re)discovered this polynomial approximation in 1712, though it's importance was not realised for another 50 years.

0.2.1 The Remainder

We now want to examine the *accuracy* of the Taylor polynomial as an approximation. In particular, we would like to find a formula for the *remainder* or *error*:

$$R_k(x) := f(x) - p_k(x).$$

With a little bit of effort one can prove that:

$$R_k(x) := \frac{(x-a)^{k+1}}{(k+1)!} f^{(k+1)}(\sigma), \text{ for some } \sigma \in [x, a].$$

We won't prove this in class, since it is quite standard and features in other courses you have taken. But for the sake of completeness, a proof is included below in Section 0.2.4 below.

Example 0.2.4. With $f(x) = e^x$ and $a = 0$, we get that

$$R_k(x) = \frac{x^{k+1}}{(k+1)!} e^{\sigma}, \text{ some } \sigma \in [0, x].$$

Example 0.2.5. How many terms are required in the Taylor Polynomial for e^x about $x = 0$ to ensure that the error at $x = 1$ is

- no more than 10^{-1}?
- no more than 10^{-2}?
- no more than 10^{-6}?
- no more than 10^{-10}?

Take notes:

There are other ways of representing the remainder, including the *Integral Representation of the Remainder*:

$$R_n(x) = \int_a^x \frac{f^{(n+1)}(t)}{n!}(x-t)^n dt. \qquad (0.2.2)$$

0.2.2 An application of Taylor's Theorem

The reasons for emphasising Taylor's theorem so early in this course are that

- It introduces us to the concept of approximation, and error estimation, but in a very simple setting;
- It is the basis for deriving methods for solving both nonlinear equations, and initial value ordinary differential equations.

With the last point in mind, we'll now outline how to derive Newton's method for nonlinear equations. (This is just a *taster*: we'll return to this topic in the next section).

Take notes:

0.2.3 Exercises

Exercise 0.1. Write down the formula for the Taylor Polynomial for

(i) $f(x) = \sqrt{1+x}$ about the point $x = 0$,

(ii) $f(x) = \sin(x)$ about the point $x = 0$,

(iii) $f(x) = \log(x)$ about the point $x = 1$.

Exercise 0.2. Prove the *Integral Mean Value Theorem*: there exists a point $c \in [a, b]$ such that

$$f(x) = \frac{1}{b-a}\int_a^b f(x)dx.$$

Exercise 0.3. The *Fundamental Theorem of Calculus* tells us that $\int_a^x f'(t)dt = f(x) - f(a)$. This can be rearranged to get $f(x) = f(a) + \int_a^x f'(t)dt$. Use this and integration by parts to deduce (0.2.2) for the case $n = 1$. (Hint: Check Wikipedia!)

0.2.4 A proof of Taylor's Theorem

Here is a proof of Taylor's theorem. It wasn't covered in class. One of the ingredients need is *Generalised Mean Value Theorem*: if the functions F and G are continuous and differentiable, etc, then, for some point $c \in [a, b]$,

$$\frac{F(b)-F(a)}{G(b)-G(a)} = \frac{F'(c)}{G'(c)}. \tag{0.2.3}$$

Theorem 0.2.6 (Taylor's Theorem)**.** *Suppose we have a function* f *that is sufficiently differentiable on the interval* $[a, x]$*, and a Taylor polynomial for* f *about the point* $x = a$

$$p_n(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \frac{(x-a)^3}{3!}f''(a) + \ldots \frac{(x-a)^k}{k!}f^{(n)}(a). \tag{0.2.4}$$

If the remainder *is written as* $R_n(x) := f(x) - p_n(x)$*, then*

$$R_n(x) := \frac{(x-a)^{n+1}}{(n+1)!}f^{(n+1)}(\sigma), \tag{0.2.5}$$

for some point $\sigma \in [x, a]$.

Proof. We want to prove that, for any $n = 0, 1, 2, \ldots$, there is a point $\sigma \in [a, x]$ such that

$$f(x) = p_n(x) + R_n(x).$$

If $x = a$ then this is clearly the case because $f(a) = p_n(a)$ and $R_n(a) = 0$.

For the case $x \neq a$, we will use a *proof by induction*. The Mean Value Theorem tells us that there is some point $\sigma \in [a, x]$ such that

$$\frac{f(x)-f(a)}{x-a} = f'(\sigma).$$

Using that $p_0(a) = f(a)$ and that $R_0(x) = (x-a)f'(\sigma)$ we can rearrange to get

$$f(x) = p_0(a) + R_0(x),$$

as required.

Now we will *assume* that (0.2.4)–(0.2.5) are true for the case $n = k-1$; and use this to show that they are true for $n = k$. From the Generalised Mean Value Theorem (0.2.3), there is some point c such that

$$\frac{R_k(x)}{(x-a)^{k+1}} = \frac{R_k(x) - R_k(a)}{(x-a)^{k+1} - (a-a)^{k+1}} = \frac{R_k'(c)}{(k+1)(c-a)^k},$$

where here we have used that $R_k(a) = 0$. Rearranging we see that we can write R_k in terms of it's own derivative:

$$R_k(x) = R_k'(c)\frac{(x-a)^{k+1}}{(k+1)(c-a)^k}. \tag{0.2.6}$$

So now we need an expression for R_k'. This is done by noting that it also happens to be the remainder for the Taylor polynomial of degree $k-1$ for the function f'.

$$R_k'(x) = f'(x) - \frac{d}{dx}\left(f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \cdots + \frac{(x-a)^k}{k!}f^{(k)}(a)\right).$$

$$R_k'(x) = f'(x) - \left(f'(a) + (x-a)f''(a) + \cdots + \frac{(x-a)^{k-1}}{(k-1)!}f^{(k)}(a)\right).$$

But the expression on the last line of the above equation is the formula for the Taylor Polynomial of degree -1 for f'. By our inductive hypothesis:

$$R_k'(c) := \frac{(c-a)^k}{k!}f^{(k+1)}(\sigma),$$

for some σ. Substitute into (0.2.6) above and we are done. □

Chapter 1

Solving nonlinear equations

1.1 Bisection

1.1.1 Introduction

Linear equations are of the form:

$$\textit{find } x \textit{ such that } ax + b = 0$$

and are easy to solve. Some nonlinear problems are also easy to solve, e.g.,

$$\textit{find } x \textit{ such that } ax^2 + bx + c = 0.$$

Cubic and quartic equations also have solutions for which we can obtain a formula. But most equations to not have simple formulae for this soltuions, so numerical methods are needed.

References

- Süli and Mayers [1, Chapter 1]. We'll follow this pretty closely in lectures.
- Stewart (*Afternotes ...*), [3, Lectures 1–5]. A well-presented introduction, with lots of diagrams to give an intuitive introduction.
- Moler (Numerical Computing with MATLAB) [2, Chap. 4]. Gives a brief introduction to the methods we study, and a description of MATLAB functions for solving these problems.
- The proof of the convergence of Newton's Method is based on the presentation in [5, Thm 3.2].

Our generic problem is:

Let f *be a continuous function on the interval* $[a, b]$.
Find $\tau = [a, b]$ *such that* $f(\tau) = 0$.

Here f is some specified function, and τ is the *solution* to $f(x) = 0$.

This leads to two natural questions:

(1) How do we know there is a solution?

(2) How do we find it?

The following gives *sufficient* conditions for the existence of a solution:

Proposition 1.1.1. *Let* f *be a real-valued function that is defined and continuous on a bounded closed interval* $[a, b] \subset \mathbb{R}$. *Suppose that* $f(a)f(b) \leqslant 0$. *Then there exists* $\tau \in [a, b]$ *such that* $f(\tau) = 0$.

Take notes:

OK – now we know there is a solution τ to $f(x) = 0$, but how to we actually solve it? Usually we don't! Instead we construct a sequence of estimates $\{x_0, x_1, x_2, x_3, \dots\}$ that *converge* to the true solution. So now we have to answer these questions:

(1) How can we construct the sequence $x_0, x_1, \dots$?

(2) How do we show that $\lim_{k\to\infty} x_k = \tau$?

There are some subtleties here, particularly with part (2). What we would like to say is that at each step the error is getting smaller. That is

$$|\tau - x_k| < |\tau - x_{k-1}| \qquad \text{for } k = 1, 2, 3, \dots.$$

But we can't. Usually all we can say is that the *bounds* on the error is getting smaller. That is: *let* ε_k *be a bound on the error at step* k

$$|\tau - x_k| < \varepsilon_k,$$

then $\varepsilon_{k+1} < \mu\varepsilon_k$ for some number $\mu \in (0, 1)$. It is easiest to explain this in terms of an example, so we'll study the simplest method: *Bisection*.

1.1.2 Bisection

The most elementary algorithm is the "*Bisection Method*" (also known as "Interval Bisection"). Suppose that we know that f changes sign on the interval $[a, b] = [x_0, x_1]$ and, thus, $f(x) = 0$ has a solution, τ, in $[a, b]$. Proceed as follows

1. Set x_2 to be the midpoint of the interval $[x_0, x_1]$.

2. Choose one of the sub-intervals $[x_0, x_2]$ and $[x_2, x_1]$ where f change sign;

3. Repeat Steps 1–2 on that sub-interval, until f sufficiently small at the end points of the interval.

This may be expressed more precisely using some *pseudocode*.

Method 1.1.2 (Bisection)**.**
Set eps to be the stopping criterion.
If $|f(a)| \leqslant eps$, return a. Exit.
If $|f(b)| \leqslant eps$, return b. Exit.
Set $x_0 = a$ and $x_1 = b$.
Set $x_L = x_0$ and $x_R = x_1$.
Set $k = 1$
while($|f(x_k)| > eps$)
    $x_{k+1} = (x_L + x_R)/2$;
    if $(f(x_L)f(x_{k+1}) < eps)$
        $x_R = x_{k+1}$;
    else
        $x_L = x_{k+1}$
    end if;
    $k = k + 1$
end while;

Example 1.1.3. *Find an estimate for $\sqrt{2}$ that is correct to 6 decimal places.*
Solution: Try to solve the equation $f(x) := x^2 - 2 = 0$ on the interval $[0, 2]$. Then proceed as shown in Figure 1.1 and Table 1.1.

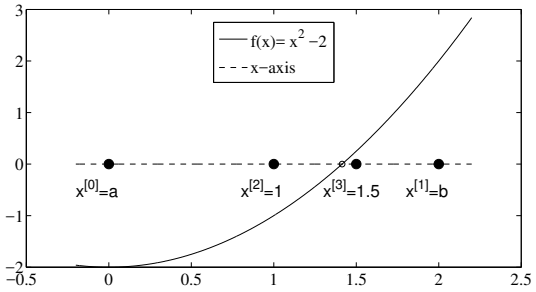


Fig. 1.1: Solving $x^2 - 2 = 0$ with the Bisection Method

Note that at steps 4 and 10 in Table 1.1 the error actually *increases*, although the bound on the error is decreasing.

1.1.3 The bisection method works

The main advantages of the Bisection method are

- It will always work.
- After k steps we know that

Theorem 1.1.4.

$$|\tau - x_k| \leqslant (\frac{1}{2})^{k-1}|b - a|, \quad \textit{for } k = 2, 3, 4, ...$$

k	x_k	$\lvert x_k - \tau \rvert$	$\lvert x_k - x_{k-1} \rvert$
0	0.000000	1.41	
1	2.000000	5.86e-01	
2	1.000000	4.14e-01	1.00
3	1.500000	8.58e-02	5.00e-01
4	1.250000	1.64e-01	2.50e-01
5	1.375000	3.92e-02	1.25e-01
6	1.437500	2.33e-02	6.25e-02
7	1.406250	7.96e-03	3.12e-02
8	1.421875	7.66e-03	1.56e-02
9	1.414062	1.51e-04	7.81e-03
10	1.417969	3.76e-03	3.91e-03
⋮	⋮	⋮	⋮
22	1.414214	5.72e-07	9.54e-07

Table 1.1: Solving $x^2 - 2 = 0$ with the Bisection Method

Take notes:

A disadvantage of bisection is that it is not as efficient as some other methods we'll investigate later.

1.1.4 Improving upon bisection

The bisection method is not very efficient. Our next goals will be to derive better methods, particularly the *Secant Method* and *Newton's method*. We also have to come up with some way of expressing what we mean by "better"; and we'll have to use Taylor's theorem in our analyses.

1.1.5 Exercises

Exercise 1.1. Does Proposition 1.1.1 mean that, if there is a solution to $f(x) = 0$ in $[a, b]$ then $f(a)f(b) \leqslant 0$? That is, is $f(a)f(b) \leqslant 0$ a *necessary* condition for their being a solution to $f(x) = 0$? Give an example that supports your answer.

Exercise 1.2. Suppose we want to find $\tau \in [a, b]$ such that $f(\tau) = 0$ for some given f, a and b. Write down an estimate for the number of iterations K required by the bisection method to ensure that, for a given ε, we know $|x_k - \tau| \leqslant \varepsilon$ for all $k \geqslant K$. In particular, how does this estimate depend on f, a and b?

Exercise 1.3. How many (decimal) digits of accuracy are gained at each step of the bisection method? (If you prefer, how many steps are need to gain a single (decimal) digit of accuracy?)

Exercise 1.4. Let $f(x) = e^x - 2x - 2$. Show that there is a solution to the problem: *find* $\tau \in [0, 2]$ *such that* $f(\tau) = 0$.
Taking $x_0 = 0$ and $x_1 = 2$, use 6 steps of the bisection method to estimate τ. Give an upper bound for the error $|\tau - x_6|$. (You may use a computer program to do this).

1.2 The Secant Method

1.2.1 Motivation

Looking back at Table 1.1 we notice that, at step 4 the error *increases* rather *decreases*. You could argue that this is because we didn't take into account how close x_3 is to the true solution. We could improve upon the bisection method as described below. The idea is, given x_{k-1} and x_k, take x_{k+1} to be the zero of the line intersects the points $\big(x_{k-1}, f(x_{k-1})\big)$ and $\big(x_k, f(x_k)\big)$. See Figure 1.2.

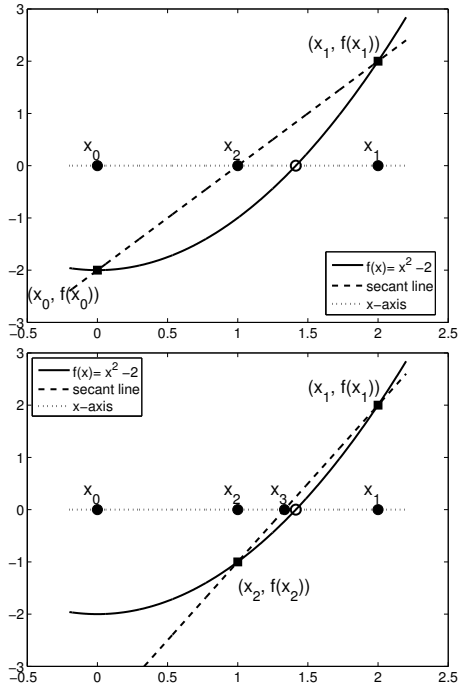


Fig. 1.2: The Secant Method for Example 1.2.2

Method 1.2.1 (Secant). [1] *Choose* x_0 *and* x_1 *so that there is a solution in* $[x_0, x_1]$. *Then define*

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \tag{1.2.1}$$

Example 1.2.2. *Use the Secant Method to solve the nonlinear problem* $x^2 - 2 = 0$ *in* $[0, 2]$. The results are shown in Table 1.2. By comparing Tables 1.1 and 1.2, we see that for this example, the Secant method is *much* more efficient than Bisection. We'll return to why this is later.

The Method of Bisection could be written as the weighted average

$$x_{k+1} = (1 - \sigma_k)x_k + \sigma_k x_{k-1}, \quad \text{with } \sigma_k = 1/2.$$

[1] The name comes from the name of the line that intersects a curve at two points. There is a related method called "false position" which was known in India in the 3rd century BC, and China in the 2nd century BC.

k	x_k	$\lvert x_k - \tau\rvert$
0	0.000000	1.41e
1	2.000000	5.86e-01
2	1.000000	4.14e-01
3	1.333333	8.09e-02
4	1.428571	1.44e-02
5	1.413793	4.20e-04
6	1.414211	2.12e-06
7	1.414214	3.16e-10
8	1.414214	4.44e-16

Table 1.2: Solving $x^2 - 2 = 0$ using the Secant Method

We can also think of the Secant method as being a *weighted average*, but with σ_k chosen to obtain faster convergence to the true solution. Looking at Figure 1.2 above, you could say that we should choose σ_k depending on which is smaller: $f(x_{k-1})$ or $f(x_k)$. If (for example) $|f(x_{k-1})| < |f(x_k)|$, then probably $|\tau - x_{k-1}| < |\tau - x_k|$. This gives another formulation of the Secant Method.

$$x_{k+1} = (1 - \sigma_k)x_k + \sigma_k x_{k-1}, \tag{1.2.2}$$

where

$$\sigma_k = \frac{f(x_k)}{f(x_k) - f(x_{k-1})}.$$

When its written in this form it is sometimes called a *relaxation method*.

1.2.2 Order of Convergence

To compare different methods, we need the following concept:

Definition 1.2.3 (Linear Convergence). *Suppose that* $\tau = \lim_{k\to\infty} x_k$. *Then we say that the sequence* $\{x_k\}_{k=0}^{\infty}$ *converges to* τ **at least linearly** *if there is a sequence of positive numbers* $\{\varepsilon_k\}_{k=0}^{\infty}$, *and* $\mu \in (0, 1)$, *such that*

$$\lim_{k\to\infty} \varepsilon_k = 0, \tag{1.2.3a}$$

and

$$|\tau - x_k| \leqslant \varepsilon_k \quad \textit{for} \quad k = 0, 1, 2, \ldots. \tag{1.2.3b}$$

and

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k} = \mu. \tag{1.2.3c}$$

So, for example, the bisection method converges at least linearly.

The reason for the expression "*at least*" is because we usually can only show that a set of upper bounds for the errors converges linearly. If (1.2.3b) can be strengthened to the equality $|\tau - x_k| = \varepsilon_k$, then the $\{x_k\}_{k=0}^{\infty}$ converges linearly, (not just "at least" linearly).

As we have seen, there are methods that converge more quickly than bisection. We state this more precisely:

Definition 1.2.4 (Order of Convergence). *Let* $\tau = \lim_{k\to\infty} x_k$. *Suppose there exists* $\mu > 0$ *and a sequence of positive numbers* $\{\varepsilon_k\}_{k=0}^{\infty}$ *such that (1.2.3a) and and (1.2.3b) both hold. Then we say that the sequence* $\{x_k\}_{k=0}^{\infty}$ *converges* with at least order q *if*

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{(\varepsilon_k)^q} = \mu.$$

Two particular values of q are important to us:

(i) If $q = 1$, and we further have that $0 < \mu < 1$, then the rate is *linear*.

(ii) If $q = 2$, the rate is *quadratic* for any $\mu > 0$.

1.2.3 Analysis of the Secant Method

Our next goal is to prove that the *Secant Method* converges. We'll be a little lazy, and only prove a suboptimal linear convergence rate. Then, in our MATLAB class, we'll investigate exactly how rapidly it really converges.

One simple mathematical tool that we use is the *Mean Value Theorem* Theorem 0.2.1 . See also [1, p420].

Theorem 1.2.5. *Suppose that* f *and* f' *are real-valued functions, continuous and defined in an interval* $I = [\tau - h, \tau + h]$ *for some* $h > 0$. *If* $f(\tau) = 0$ *and* $f'(\tau) \neq 0$, *then the sequence* (1.2.1) *converges at least linearly to* τ.

Before we prove this, we note the following

- We wish to show that $|\tau - x_{k+1}| < |\tau - x_k|$.
- From Theorem 0.2.1, there is a point $w_k \in [x_{k-1}, x_k]$ such that
$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} = f'(w_k). \qquad (1.2.4)$$
- Also by the MVT, there is a point $z_k \in [x_k, \tau]$ such that
$$\frac{f(x_k) - f(\tau)}{x_k - \tau} = \frac{f(x_k)}{x_k - \tau} = f'(z_k). \qquad (1.2.5)$$
Therefore $f(x_k) = (x_k - \tau)f'(z_k)$.
- Using (1.2.4) and (1.2.5), we can show that
$$\tau - x_{k+1} = (\tau - x_k)\left(1 - f'(z_k)/f'(w_k)\right).$$
Therefore
$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \left|1 - \frac{f'(z_k)}{f'(w_k)}\right|.$$
- Suppose that $f'(\tau) > 0$. (If $f'(\tau) < 0$ just tweak the arguments accordingly). Saying that f' *is continuous in the region* $[\tau - h, \tau + h]$ means that, for any $\varepsilon > 0$ there is a $\delta > 0$ such that
$$|f'(x) - f'(\tau)| < \varepsilon \text{ for any } x \in [\tau - \delta, \tau + \delta].$$
Take $\varepsilon = f'(\tau)/4$. Then $|f'(x) - f'(\tau)| < f'(\tau)/4$. Thus
$$\frac{3}{4}f'(\tau) \leqslant f'(x) \leqslant \frac{5}{4}f'(\tau) \quad \text{for any } x \in [\tau-\delta, \tau+\delta].$$
Then, so long as w_k and z_k are both in $[\tau-\delta, \tau+\delta]$
$$\frac{f'(z_k)}{f'(w_k)} \leqslant \frac{5}{3}.$$

Take notes:

(See also details in Section 1.2.5).

Given enough time and effort we *could* show that the Secant Method converges faster that linearly. In particular, that the order of convergence is $q = (1 + \sqrt{5})/2 \approx 1.618$. This number arises as the only positive root of $q^2 - q - 1$. It is called the *Golden Mean*, and arises in many areas of Mathematics, including finding an explicit expression for the Fibonacci Sequence: $f_0 = 1$, $f_1 = 1$, $f_{k+1} = f_k + f_{k-1}$ for $k = 2, 3, \ldots$. That gives, $f_0 = 1$, $f_1 = 1$, $f_2 = 2$, $f_3 = 3$, $f_4 = 5$, $f_5 = 8$, $f_6 = 13$, ...

A rigorous proof depends on, among other things, and error bound for polynomial interpolation, which is the first topic in MA378. With that, one can show that $\varepsilon_{k+1} \leqslant C\varepsilon_k\varepsilon_{k-1}$. Repeatedly using this we get:

- Let $r = |x_1 - x_0|$ so that $\varepsilon_0 \leqslant r$ and $\varepsilon_1 \leqslant r$,
- Then $\varepsilon_2 \leqslant C\varepsilon_1\varepsilon_0 \leqslant Cr^2$
- Then $\varepsilon_3 \leqslant C\varepsilon_2\varepsilon_1 \leqslant C(Cr^2)r = C^2r^3$.
- Then $\varepsilon_4 \leqslant C\varepsilon_3\varepsilon_2 \leqslant C(C^2r^3)(Cr^2) = C^4r^5$.
- Then $\varepsilon_5 \leqslant C\varepsilon_4\varepsilon_3 \leqslant C(C^4r^5)(C^2r^3) = C^7r^8$.
- And in general, $\varepsilon_k = C^{f_k-1}r^{f_k}$.

1.2.4 Exercises

Exercise 1.5. ⋆ Suppose we define the Secant Method as follows.

> *Choose any two points* x_0 *and* x_1.
> *For* $k = 1, 2, \ldots$, *set* x_{k+1} *to be the point where the line through* $\big(x_{k-1}, f(x_{k-1})\big)$ *and* $\big(x_k, f(x_k)\big)$ *that intersects the* x*-axis.*

Show how to derive the formula for the secant method.

Exercise 1.6. ⋆

(i) Is it possible to construct a problem for which the bisection method will work, but the secant method will fail? If so, give an example.

(ii) Is it possible to construct a problem for which the secant method will work, but bisection will fail? If so, give an example.

1.2.5 Appendix (Proof of convergence of the secant method)

Here are the full details on the proof of the fact that the Secant Method converges at least linearly (Theorem 1.2.5). Before you read it, take care to review the notes from that section, particularly (1.2.4) and (1.2.5).

Proof. The method is

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}.$$

We'll use this to derive an expression of the error at step $k+1$ in terms of the error at step k. In particular,

$$\begin{aligned}
\tau - x_{k+1} &= \tau - x_k + f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \\
&= \tau - x_k + f(x_k)/f'(w_k) \\
&= \tau - x_k + (x_k - \tau)f'(z_k)/f'(w_k) \\
&= (\tau - x_k)\Big(1 - f'(z_k)/f'(w_k)\Big).
\end{aligned}$$

Therefore

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \big|1 - \frac{f'(z_k)}{f'(w_k)}\big|.$$

So it remains to be shown that

$$\big|1 - \frac{f'(z_k)}{f'(w_k)}\big| < 1.$$

Lets first assume that $f'(\tau) = \alpha > 0$. (If $f'(\tau) = \alpha < 0$ the following arguments still hold, just with a few small changes). Because f' is continuous in the region $[\tau - h, \tau + h]$, for any given $\varepsilon > 0$ there is a $\delta > 0$ such that $|f'(x) - \alpha| < \varepsilon$ for and $x \in [\tau - \delta, \tau + \delta]$. Take $\varepsilon = \alpha/4$. Then $|f'(x) - \alpha| < \alpha/4$. Thus

$$\alpha\frac{3}{4} \leqslant f'(x) \leqslant \alpha\frac{5}{4} \quad \text{for any } x \in [\tau - \delta, \tau + \delta].$$

Then, so long as w_k and z_k are both in $[\tau - \delta, \tau + \delta]$

$$\frac{f'(z_k)}{f'(w_k)} \leqslant \frac{5}{3}.$$

This gives

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \frac{2}{3},$$

which is what we needed. □

1.3 Newton's Method

1.3.1 Motivation

These notes are loosely based on Section 1.4 of [1] (i.e., Süli and Mayers, Introduction to Numerical Analysis). See also, [3, Lecture 2], and [5, §3.5] The Secant method is often written as

$$x_{k+1} = x_k - f(x_k)\phi(x_k, x_{k-1}),$$

where the function ϕ is chosen so that x_{k+1} is the root of the secant line joining the points $\big(x_{k-1}, f(x_{k-1})\big)$ and $\big(x_k, f(x_k)\big)$. A related idea is to construct a method $x_{k+1} = x_k - f(x_k)\lambda(x_k)$, where we choose λ so that x_{k+1} is the point where the tangent line to f at $(x_k, f(x_k))$ cuts the x-axis. This is shown in Figure 1.3. We attempt to solve $x^2 - 2 = 0$, taking $x_0 = 2$. Taking the x_1 to be zero of the tangent to $f(x)$ at $x = 2$, we get $x_1 = 1.5$. Taking the x_2 to be zero of the tangent to $f(x)$ at $x = 1.5$, we get $x_2 = 1.4167$, which is very close to the true solution of $\tau = 1.4142$.

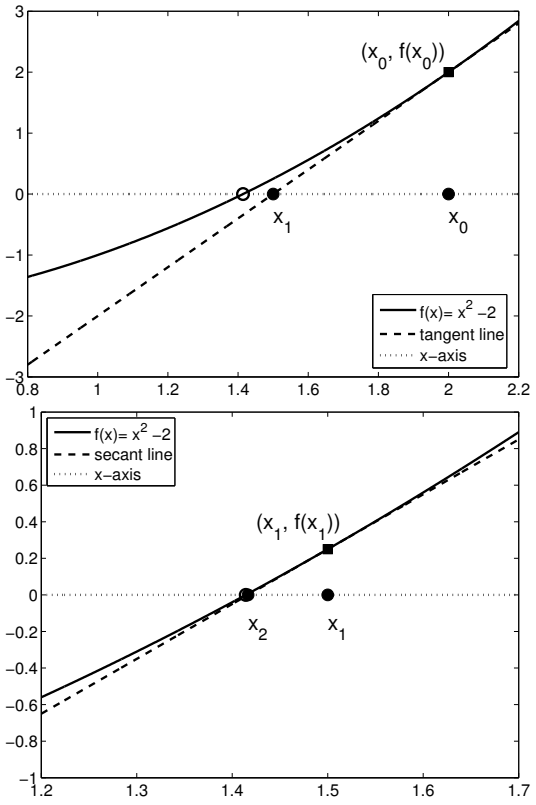


Fig. 1.3: Estimating $\sqrt{2}$ by solving $x^2 - 2 = 0$ using Newton's Method

Method 1.3.1 (Newton's Method[2]).

1. Choose any x_0 in $[a, b]$,
2. For $i = 0, 1, \ldots,$ set x_{k+1} to the root of the line through x_k with slope $f'(x_k)$.

By writing down the equation for the line at $\big(x_k, f(x_k)\big)$ with slope $f'(x_k)$, one can show (see Exercise 1.7-(i)) that the formula for the iteration is

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{1.3.6}$$

Example 1.3.2. *Use Newton's Method to solve the nonlinear problem* $x^2 - 2 = 0$ *in* $[0, 2]$. The results are shown in Table 1.3. For this example, the method becomes

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 2}{2x_k} = \frac{1}{2}x_k + \frac{1}{x_k}.$$

k	x_k	$\lvert x_k - \tau\rvert$	$\lvert x_k - x_{k-1}\rvert$
0	2.000000	5.86e-01	
1	1.500000	8.58e-02	5.00e-01
2	1.416667	2.45e-03	8.33e-02
3	1.414216	2.12e-06	2.45e-03
4	1.414214	1.59e-12	2.12e-06
5	1.414214	2.34e-16	1.59e-12

Table 1.3: Solving $x^2 - 2 = 0$ using Newton's Method

By comparing Table 1.2 and Table 1.3, we see that for this example, the Newton's method is more efficient again than the Secant method.

Deriving Newton's method geometrically certainly has an intuitive appeal. However, to analyse the method, we need a more abstract derivation based on a **Truncated Taylor Series**.

Take notes:

[2] 

Sir Isaac Newton, 1643 - 1727, England. Easily one of the greatest scientist of all time. The method we are studying appeared in his celebrated *Principia Mathematica* in 1687, but it is believed he had used it as early as 1669.

1.3.2 Newton Error Formula

We saw in Table 1.3 that Newton's method can be much more efficient than, say, Bisection: it yields estimates that converge far more quickly to τ. Bisection converges

(at least) linearly, whereas Newton's converges *quadratically*, that is, with *at least order* $q = 2$.

In order to prove that this is so, we need to

1. write down a recursive formula for the error;
2. show that it converges;
3. then find the limit of $|\tau - x_{k+1}|/|\tau - x_k|^2$.

Step 2 is usually the crucial part.

There are two parts to the proof. The first involves deriving the so-called "Newton Error formula". Then we'll apply this to prove (quadratic) convergence. In all cases we'll assume that the functions f, f' and f'' are defined and continuous on the an interval $I_\delta = [\tau-\delta, \tau+\delta]$ around the root τ. The proof we'll do in class comes directly from the above derivation (see also Epperson [5, Thm 3.2]).

Theorem 1.3.3 (Newton Error Formula)**.** *If* $f(\tau) = 0$ *and*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

then there is a point η_k *between* τ *and* x_k *such that*

$$\tau - x_{k+1} = -\frac{(\tau - x_k)^2}{2}\frac{f''(\eta_k)}{f'(x_k)},$$

Take notes:

Example 1.3.4. As an application of Newton's error formula, we'll show that the number of correct decimal digits in the approximation doubles at each step.

Take notes:

1.3.3 Convergence of Newton's Method

We'll now complete our analysis of this section by proving the convergence of Newton's method.

Theorem 1.3.5. *Let us suppose that* f *is a function such that*

- f *is continuous and real-valued, with continuous* f''*, defined on some close interval* $I_\delta = [\tau-\delta, \tau+\delta]$,
- $f(\tau) = 0$ *and* $f''(\tau) \neq 0$,
- *there is some positive constant* A *such that*

$$\frac{|f''(x)|}{|f'(y)|} \leqslant A \quad \textit{for all} \quad x, y \in I_\delta.$$

Let $h = \min\{\delta, 1/A\}$. *If* $|\tau - x_0| \leqslant h$ *then Newton's Method converges quadratically.*

Take notes:

1.3.4 Exercises

Exercise 1.7. ⋆ Write down the equation of the line that is tangential to the function f at the point x_k. Give an expression for its zero. Hence show how to derive Newton's method.

Exercise 1.8. (i) It is possible to construct a problem for which the bisection method will work, but Newton's method will fail? If so, give an example.

(ii) It is possible to construct a problem for which Newton's method will work, but bisection will fail? If so, give an example.

Exercise 1.9. (i) Write down Newton's Method as applied to the function $f(x) = x^3 - 2$. Simplify the computation as much as possible. What is achieved if we find the root of this function?

(ii) Do three iterations by hand of Newton's Method applied to $f(x) = x^3 - 2$ with $x_0 = 1$.

Exercise 1.10. (This is taken from Exercise 3.5.1 of Epperson). If f is such that $|f''(x)| \leqslant 3$ and $|f'(x)| \geqslant 1$ for all x, and if the initial error in Newton's Method is less than $1/2$, give an upper bound for the error at each of the first 3 steps.

Exercise 1.11. Here is (yet) another scheme called *Steffenson's Method*: Choose $x_0 \in [a, b]$ and set

$$x_{k+1} = x_k - \frac{\big(f(x_k)\big)^2}{f\big(x_k + f(x_k)\big) - f(x_k)} \text{ for } k = 0, 1, 2, \ldots.$$

(a) ⋆ Explain how this method relates to Newton's Method.

(b) [Optional] Write a program, in MATLAB, or your language of choice, to implement this method. Verify it works by using it to estimate the solution to $e^x = (2 - x)^3$ with $x_0 = 0$. Submit your code and test harness as Blackboard assignment. *No credit is available for this part, but feedback will be given on your code. Also, it will help you prepare for the final exam.*

Exercise 1.12. ⋆ (This is Exercise 1.6 from Süli and Mayers) The proof of the convergence of Newton's method given in Theorem 1.3.5 uses that $f'(\tau) \neq 0$. Suppose that it is the case that $f'(\tau) = 0$.

(i) What can we say about the root, τ?

(ii) Starting from the Newton Error formula, show that

$$\tau - x_{k+1} = \frac{(\tau - x_k)}{2} \frac{f''(\eta_k)}{f''(\mu_k)},$$

for some μ_k between τ and x_k. (*Hint: try using the MVT*).

(iii) What does the above error formula tell us about the convergence of Newton's method in this case?

1.4 Fixed Point Iteration

1.4.1 Introduction

Newton's method can be considered to be a particular instance of a very general approach called *Fixed Point Iteration* or *Simple Iteration*.

The basic idea is:

> *If we want to solve* $f(x) = 0$ *in* $[a, b]$*, find a function* $g(x)$ *such that, if* τ *is such that* $f(\tau) = 0$*, then* $g(\tau) = \tau$*.*
> *Next, choose* x_0 *and set* $x_{k+1} = g(x_k)$ *for* $k = 0, 1, 2 \ldots$.

Example 1.4.1. Suppose that $f(x) = e^x - 2x - 1$ and we are trying to find a solution to $f(x) = 0$ in $[1, 2]$. We can reformulate this problem as

> *For* $g(x) = \ln(2x + 1)$*, find* $\tau \in [1, 2]$ *such that* $g(\tau) = \tau$.

If we take the initial estimate $x_0 = 1$, then Simple Iteration gives the following sequence of estimates.

k	x_k	$\lvert\tau - x_k\rvert$
0	1.0000	2.564e-1
1	1.0986	1.578e-1
2	1.1623	9.415e-2
3	1.2013	5.509e-2
4	1.2246	3.187e-2
5	1.2381	1.831e-2
⋮	⋮	⋮
10	1.2558	6.310e-4

To make this table, I used a numerical scheme to solve the problem quite accurately to get $\tau = 1.256431$. (In general we don't know τ in advance–otherwise we wouldn't need such a scheme). I've given the quantities $|\tau - x_k|$ here so we can observe that the method is converging, and get an idea of how quickly it is converging.

We have to be quite careful with this method: **not every choice is g is suitable**.

Suppose we want the solution to $f(x) = x^2 - 2 = 0$ in $[1, 2]$. We could choose $g(x) = x^2 + x - 2$. Taking $x_0 = 1$ we get the iterations shown opposite.

k	x_k
0	1
1	0
2	-2
3	0
4	-2
5	0
⋮	⋮

This sequence doesn't converge!

We need to refine the method that ensure that it will converge. Before we do that in a formal way, consider the following...

Example 1.4.2. Use the Mean Value Theorem to show that the fixed point method $x_{k+1} = g(x_k)$ converges if $|g'(x)| < 1$ for all x near the fixed point.

Take notes:

This is an important example, mostly because it introduces the "tricks" of using that $g(\tau) = \tau$ and $g(x_k) = x_{k+1}$. But it is not a rigorous theory. That requires some ideas such as the *contraction mapping theorem*.

1.4.2 A short tour of fixed points and contractions

A variant of the famous Fixed Point Theorem[3] is :

> *Suppose that* $g(x)$ *is defined and continuous on* $[a, b]$*, and that* $g(x) \in [a, b]$ *for all* $x \in [a, b]$*. Then there exists a point* $\tau \in [a, b]$ *such that* $g(\tau) = \tau$*. That is,* $g(x)$ *has a* fixed point *in the interval* $[a, b]$.

Try to convince yourself that it is true, by sketching the graphs of a few functions that send all points in the interval, say, $[1, 2]$ to that interval, as in Figure 1.4.

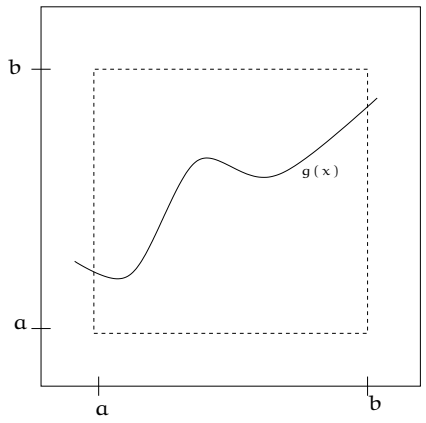


Fig. 1.4: Sketch of a function $g(x)$ such that, if $a \leqslant x \leqslant b$ then $a \leqslant g(x) \leqslant b$

The next ingredient we need is to observe that g is a *contraction*. That is, $g(x)$ is continuous and defined on $[a, b]$ and there is a number $L \in (0, 1)$ such that

$$|g(\alpha) - g(\beta)| \leqslant L|\alpha - \beta| \text{ for all } \alpha, \beta \in [a, b]. \quad (1.4.7)$$

[3] LEJ Brouwer, 1881–1966, Netherlands

Theorem 1.4.3 (Contraction Mapping Theorem)**.**
Suppose that the function g *is a real-valued, defined, continuous, and*

(a) it maps every point in $[a, b]$ *to some point in* $[a, b]$*;*

(b) and it is a contraction on $[a, b]$*,*

then

(i) g *has a fixed point* $\tau \in [a, b]$*,*

(ii) the fixed point is unique,

(iii) the sequence $\{x_k\}_{k=0}^{\infty}$ *defined by* $x_0 \in [a, b]$ *and* $x_k = g(x_{k-1})$ *for* $k = 1, 2, \ldots$ *converges to* τ*.*

Proof:
Take notes:

1.4.3 Convergence of Fixed Point Iteration

We now know how to apply to Fixed-Point Method and to check if it will converge. Of course we can't perform an infinite number of iterations, and so the method will yield only an approximate solution. Suppose we want the solution to be accurate to say 10^{-6}, how many steps are needed? That is, how large must k be so that

$$|x_k - \tau| \leqslant 10^{-6}?$$

The answer is obtained by first showing that

$$|\tau - x_k| \leqslant \frac{L^k}{1 - L}|x_1 - x_0|. \tag{1.4.8}$$

Take notes:

Example 1.4.4. If $g(x) = \ln(2x + 1)$ and $x_0 = 1$, and we want $|x_k - \tau| \leqslant 10^{-6}$, then we can use (1.4.8) to determine the number of iterations required.

Take notes:

This calculation only gives an upper bound for the number of iterations. It is correct, but not necessarily *sharp*. In practice, one finds that 23 iterations is sufficient to ensure that the error is less than 10^{-6}. Even so, 23 iterations a quite a lot for such a simple problem. So can conclude that this method is not as fast as, say, Newton's Method. However, it is perhaps the most generalizable.

1.4.4 Knowing When to Stop

Suppose you wish to program one of the above methods. You will get your computer to repeat one of the iterative methods until your solution is sufficiently close to the true solution:

```
x[0] = 0
tol = 1e-6
i=0
while (abs(tau - x[i]) > tol)  // This is the
                               // stopping criterion
  x[i+1] = g(x[i])   // Fixed point iteration
  i = i+1
end
```

All very well, except you don't know τ. If you did, you wouldn't need a numerical method. Instead, we could choose the stopping criterion based on how close successive estimates are:

```
while (abs(x[i-1] - x[i]) > tol)
```

This is fine if the solution is not close to zero. E.g., if its about 1, would should get roughly 6 accurate figures. But is $\tau = 10^{-7}$ then it is quite useless: x_k could be ten times larger than τ. The problem is that we are estimating the *absolute* error.

Instead, we usually work with *relative error*:

```
while (abs (x[i-1]-x[i] / x[i] ) > tol)
```

1.4.5 Exercises

Exercise 1.13. Is it possible for g to be a contraction on $[a, b]$ but not have a fixed point in $[a, b]$? Give an example to support your answer.

Exercise 1.14. Show that $g(x) = \ln(2x + 1)$ is a contraction on $[1, 2]$. Give an estimate for L. (Hint: Use the Mean Value Theorem).

Exercise 1.15. Consider the function $g(x) = x^2/4 + 5x/4 - 1/2$.

(i) It has two fixed points – what are they?

(ii) For each of these, find the largest region around them such that g is a contraction on that region.

Exercise 1.16. Although we didn't prove it in class, it turns out that, if $g(\tau) = \tau$, and the fixed point method given by

$$x_{k+1} = g(x_k),$$

converges to the point τ (where $g(\tau) = \tau$), and

$$g'(\tau) = g''(\tau) = \cdots = g^{(p-1)}(\tau) = 0,$$

then it converges with order p.

(i) Use a Taylor Series expansion to prove this.

(ii) We can think of Newton's Method for the problem $f(x) = 0$ as fixed point iteration with $g(x) = x - f(x)/f'(x)$. Use this, and Part (i), to show that, if Newton's method converges, it does so with order 2, providing that $f'(\tau) \neq 0$.

1.5 LAB 1: the bisection and secant methods

The goal of this section is to help you gain familiarity with the fundamental tasks that can be accomplished with MATLAB: defining vectors, computing functions, and plotting. We'll then see how to implement and analyse the Bisection and Secant schemes in MATLAB.

You'll find many good MATLAB references online. I particularly recommend:

- Cleve Moler, *Numerical Computing with MATLAB*, which you can access at `http://uk.mathworks.com/moler/chapters`
- Tobin Driscoll, *Learning MATLAB*, which you can access through the NUI Galway library portal.

MATLAB is an interactive environment for mathematical and scientific computing. It the standard tool for numerical computing in industry and research.

MATLAB stands for Matrix Laboratory. It specialises in matrix and vector computations, but includes functions for graphics, numerical integration and differentiation, solving differential equations, etc.

MATLAB differs from most significantly from, say, Maple, by not having a facility for abstract computation.

1.5.1 The Basics

MATLAB an *interpretive* environment – you type a command and it will execute it immediately.

The default data-type is a matrix of double precision floating-point numbers. A scalar variable is an instance of a 1×1 matrix. To check this set,
`>> t=10` and use `>> size(t)`
to find the numbers of rows and columns of `t`.

A vector may be declared as follows:
`>> x = [1 2 3 4 5 6 7]`
This generates a vector, x, with $x_1 = 1$, $x_2 = 2$, etc. However, this could also be done with `x=1:7`

More generally, if we want to define a vector $x = (a, a+h, a+2h, \ldots, b)$, we could use `x = a:h:b;`
For example
`>> x=10:-2:0` gives $x = (10, 8, 6, 4, 2, 0)$.
If h is omitted, it is assumed to be 1.

The i^{th} element of a vector is access by typing `x(i)`. The element of in row `i` and column `j` of a matrix is given by `A(i,j)`

Most "scalar" functions return a matrix when given a matrix as an argument. For example, if x is a vector of length n, then `y = sin(x)` sets y to be a vector, also of length n, with $y_i = \sin(x_i)$.

MATLAB has most of the standard mathematical functions: `sin`, `cos`, `exp`, `log`, etc.
In each case, write the function name followed by the argument in round brackets, e.g.,
`>> exp(x)` for e^x.

The `*` operator performs matrix multiplication. For element-by-element multiplication use `.*`
For example,
`y = x.*x` sets $y_i = (x_i)^2$.
So does `y = x.^2`. Similarly, `y=1./x` set $y_i = 1/x_i$.

If you put a semicolon at the end of a line of MATLAB, the line is executed, but the output is not shown. (This is useful if you are dealing with large vectors). If no semicolon is used, the output is shown in the command window.

1.5.2 Plotting functions

Define a vector
`>> x=[0 1 2 3]` and then set `>> f = x.^2 -2`
To plot these vectors use:
`>> plot(x, f)`
If the picture isn't particularly impressive, then this might be because Matlab is actually only printing the 4 points that you defined. To make this more clear, use
`>> plot(x, f, '-o')`
This means to plot the vector `f` as a function of the vector x, placing a circle at each point, and joining adjacent points with a straight line.

Try instead: `>> x=0:0.1:3` and `f = x.^2 -2` and plot them again.

To define function in terms of *any* variable, type:
```
>> F = @(x)(x.^2 -2);
```
Now you can use this *function* as follows:
```
>> plot(x, F(x));
```
Take care to note that MATLAB is *case sensitive*.

In this last case, it might be helpful to also observe where the function cuts the x-axis. That can be done by also plotting the line joining, for example, the points $(0, 0)$, and $(3, 0)$:
```
>> plot(x,F(x), [0,3], [0,0]);
```
Tip: *Use the* `>> help` *menu to find out what the* `ezplot` *function is, and how to use it.*

1.5.3 Programming the Bisection Method

Revise the lecture notes on the *Bisection Method*.
Suppose we want to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$ using Bisection.

- Define the function `f` as:
  ```
  >> f = @(x)(exp(x) - (2-x).^3);
  ```
- Taking $x_1 = 0$ and $x_2 = 5$, do 8 iterations of the Bisection method.
- Complete the table below. You may use that the solution is (approximately)
  $\tau = 0.7261444658054950$.

k	x_k	$\lvert\tau - x_k\rvert$
1		
2		
3		
4		
5		
6		
7		
8		

Implementing the Bisection method by hand is very tedious. Here is a program that will do it for you. You don't need to type it all in; you can download it from www.maths.nuigalway.ie/MA385/lab1/Bisection.m

```
3  clear; % Erase all stored variables
4  fprintf('\n\n---------\n Using Bisection\n');
5  % The function is
6  f = @(x)(exp(x) - (2-x).^3);
7  fprintf('Solving f=0 with the function\n');
8  disp(f);
9
10
11 tau = 0.72614446580549503614; % true solution
12 fprintf('The true solution is %12.8f\n', tau);
13
14 %% Our initial guesses are x_1=0 and x_2 =2;
15 x(1)=0;
16 fprintf('%2d | %14.8e | %9.3e \n', ...
17    1, x(1), abs(tau - x(1)));
18 x(2)=5;
19 fprintf('%2d | %14.8e | %9.3e \n', ...
20    2, x(2), abs(tau - x(2)));
21 for k=2:8
22    x(k+1) = (x(k-1)+x(k))/2;
23    if ( f(x(k+1))*f(x(k-1)) < 0)
24       x(k)=x(k-1);
25    end
26    fprintf('%2d | %14.8e | %9.3e\n', ...
27       k+1, x(k+1), abs(tau - x(k+1)));
28 end
```

Read the code carefully. If there is a line you do not understand, then ask a tutor, or look up the on-line help. For example, find out what that `clear` on Line 3 does by typing `>> doc clear`

Q1. Suppose we wanted an estimate x_k for τ so that $|\tau - x_k| \leqslant 10^{-10}$.

(i) In §1.1 we saw that $|\tau - x_k| \leqslant (\frac{1}{2})^{k-1}|b - a|$. Use this to estimate how many iterations are required in theory.

(ii) Use the program above to find how many iterations are required in practice.

1.5.4 The Secant method

Recall the the Secant Method in (1.2.1).

Q2 (a) Adapt the program above to implement the secant method.

(b) Use it to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$.

(c) How many iterations are required to ensure that the error is less than 10^{-10}?

. . . . . .

Q3 Recall from Definition 1.2.4 the *order of convergence* of a sequence $\{\varepsilon_0, \varepsilon_1, \varepsilon_2, \ldots\}$ is q if

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^q} = \mu,$$

for some constant μ.

We would like to verify that $q = (1+\sqrt{5})/2 \approx 1.618$. This is difficult to do computationally because, after a relatively small number of iterations, the round-off error becomes significant. But we can still try!

Adapt the program above so that at each iteration it displays

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^{1.618}}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^2},$$

and so deduce that the order of converges is greater than 1 (so better than bisection), less than 2, and roughly $(1+\sqrt{5})/2$.

1.5.5 To Finish

Before you leave the class upload your MATLAB code for the Q3 (only) to "**Lab 1**" in the "Assignments and Labs" section Blackboard. This file must include your name and ID number as comments. Ideally, it should incorporate your name or ID into the file name (e.g., `Lab1_Donal_Duck.m`). Include your answers to Q1 and Q2 as comments in that file.

1.5.6 Extra

The bisection method is popular because it is robust: it will always work subject to minimal constraints. However, it is slow: if the Secant works, then it converges much more quickly. How can we combine these two algorithms to get a fast, robust method? Consider the following problem:

$$\text{Solve } 1 - \frac{2}{x^2 - 2x + 2} = 0 \quad \text{on } [-10, 1].$$

You should find that the bisection method works (slowly) for this problem, but the Secant method will fail. So write a hybrid algorithm that switches between the bisection method and the secant method as appropriate.

Take care to document your code carefully, to show which algorithm is used when.

How many iterations are required?

1.6 What has Newton's method ever done for me?

When studying a numerical method (or indeed any piece of Mathematics) it is important to ask: "*why are we are doing this?*". Sometimes it is so that you can understand other topics later. Sometimes because it is interesting/beautiful in its own right; Most commonly it is because it is useful. Here are some instances of each of these:

1. The analyses we have used in this section allowed us to consider some important ideas in a simple setting.

Examples include

- **Convergence**, including *rates of convergence*.
- **Fixed-point theory**, and contractions. We'll be seeing analogous ideas in the next section (Lipschitz conditions).
- **The approximation of functions by polynomials** (Taylor's Theorem). This point will reoccur in the next section, and all through-out next semester.

2. Applications come from lots of areas of science and engineering. Less obvious might be applications to financial mathematics.

The celebrated Black-Scholes equation for pricing a put option can be written as

$$\frac{\partial V}{\partial t} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rS\frac{\partial V}{\partial S} + rV = 0.$$

where

- $V(S,t)$ is the current value of the right (but not the obligation) to buy or sell ("put" or "call") an asset at a future time T;
- S is the current value of the underlying asset;
- r is the current interest rate (because the value of the option has to be compared with what we would have gained by investing the money we paid for it);
- σ is the volatility of the asset's price.

Often one knows S, T and r, but not σ. The method of *implied volatility* is when we take data from the market and then find the value of σ which would give the data as the solution to the Black-Scholes equation. This is a nonlinear problem and so Newton's method can be used. See Chapters 13 and 14 of Higham's "An Introduction to Financial Option Valuation" for more details.

(We will return to the Black-Scholes problem again at the end of the next section).

3. Some of these ideas are interesting and beautiful. Consider Newton's method. Suppose that we want to find the complex n^{th} roots of unity: the set of numbers $\{z_0, z_1, z_2, \ldots, z_{n-1}\}$ who's n^{th} roots are 1. For example, the 4th roots of unity are 1, i, -1 and $-i$.

The n^{th} roots of unity have a simple expression:

$$z_k = e^{i\theta} \quad \text{where } \theta = \frac{2k\pi}{n}$$

for $k \in= \{0, 1, 2 \ldots, n-1\}$ and $i = \sqrt{-1}$. Plotted in the Argand Plane, these point form a regular polygon.

But pretend that we don't have this formula, and want to use Newton's method to find a given root. We could try to solve $f(z) = 0$ with $f(z) = z^n - 1$. The iteration is:

$$z_{k+1} = z_k - \frac{(z_k)^n - 1}{n(z_k)^{n-1}}.$$

However, there are n possible solutions; given a particular starting point, which root with the method converge to? If we take a number of points in a region of space, iterate on each of them, and then colour the points to indicate the ones that converge to the same root, we get the famous Julia [4] set, an example of a fractal. One such Julia set, generated by the MATLAB script Julia.m, which you can download from the course website, is shown below in Figure 1.5.

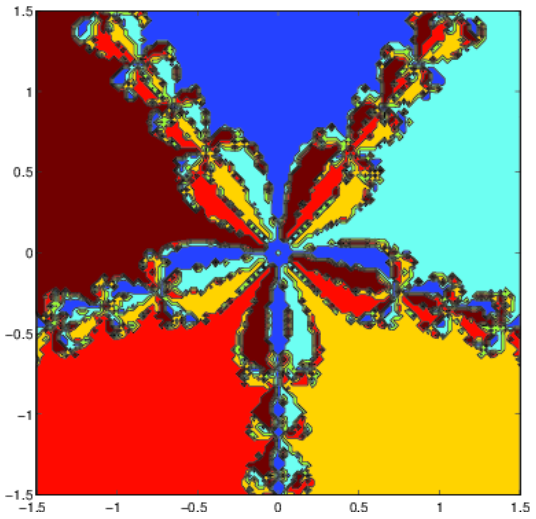


Fig. 1.5: A contour plot of a Julia set with $n = 5$

[4] Gaston Julia, French mathematician 1893–1978. The famous paper which introduced these ideas was published in 1918 when he was just 25. Interest later waned until the 1970s when Mandelbrot's computer experiments reinvigorated interest.

Chapter 2

Initial Value Problems

2.1 Introduction

The first part of this introduction is based on [5, Chap. 6]. The rest of the notes mostly follow [1, Chap. 12].
The growth of some tumours can be modelled as

$$R'(t) = -\frac{1}{3}S_i R(t) + \frac{2\lambda\sigma}{\mu R + \sqrt{\mu^2 R^2 + 4\sigma}}, \qquad (2.1.1)$$

subject to the initial condition $R(t_0) = a$, where R is the radius of the tumour at time t. Clearly, it would be useful to know the value of R at certain times in the future. Though it's essentially impossible to solve for R exactly, we can accurately estimate it. The equation in (2.1.1) is an example of an *initial value differential equation* or, simply, and *initial value problem*: we are given the solution at some initial time, and must solve a differential equation to get the solution at later times. In this section, we'll study techniques approximating solutions to such problems.

Initial Value Problems (IVPs) are differential equations of the form: *Find* $y(t)$ *such that*

$$\frac{dy}{dt} = f(t,y) \quad \text{for } t > t_0, \qquad \text{with } y(t_0) = y_0. \quad (2.1.2)$$

Here $y' = f(t,y)$ is the *differential equation* and $y(t_0) = y_0$ is the *initial value*.

Some IVPs are easy to solve. For example:

$$y' = t^2 \quad \text{with } y(1) = 1.$$

Just integrate the differential equation to get that

$$y(t) = t^3/3 + C,$$

and use the initial value to find the constant of integration. This gives the solution $y'(t) = (t^3+2)/3$. However, most problems are much harder, and some don't have solutions at all.

In many cases, it is possible to determine that a given problem does indeed have a solution, even if we can't write it down. The idea is that the function f should be "Lipschitz", a notion closely related to that of a *contraction* (1.4.7).

Definition 2.1.1. A function f satisfies a *Lipschitz* [1] *Condition* (with respect to its second argument) in the rectangular region D if there is a positive real number L such that

$$|f(t,u) - f(t,v)| \leqslant L|u - v|, \qquad (2.1.3)$$

for all $(t,u) \in D$ and $(t,v) \in D$.

Example 2.1.2. For each of the following functions f, show that is satisfies a *Lipschitz condition*, and give an upper bound on the Lipschitz constant L.

(i) $f(t,y) = y/(1+t)^2$ for $0 \leqslant t \leqslant \infty$.

(ii) $f(t,y) = 4y - e^{-t}$ for all t.

(iii) $f(t,y) = -(1+t^2)y + \sin(t)$ for $1 \leqslant t \leqslant 2$.

Take notes:

The reason we are interested in functions satisfying Lipschitz conditions is as follows:

[1] Rudolf Otto Sigismund Lipschitz, Germany, 1832–1903. Made many important contributions to science in areas that include differential equations, number theory, Fourier Series, celestial mechanics, and analytic mechanics.

Proposition 2.1.3 (Picard's[2]). *Suppose that the real-valued function* $f(t,y)$ *is continuous for* $t \in [t_0, t_M]$ *and* $y \in [y_0 - C, y_0 + C]$*; that* $|f(t,y_0)| \leqslant K$ *for* $t_0 \leqslant t \leqslant t_M$*; and that* f *satisfies the* Lipschitz condition *(2.1.3). If*

$$C \geqslant \frac{K}{L}\left(e^{L(t_M - t_0)} - 1\right),$$

then (2.1.2) us a unique solution on $[t_0, t_M]$*. Further:*

$$|y(t) - y(t_0)| \leqslant C \qquad t_0 \leqslant t \leqslant t_M.$$

You are not required to know this theorem for this course. However, it's important to be able to determine when a given f satisfies a Lipschitz condition.

2.1.1 Exercises

Exercise 2.1. For the following functions show that they satisfy a Lipschitz condition on the corresponding domain, and give an upper-bound for L:

(i) $f(t,y) = 2yt^{-4}$ for $t \in [1, \infty)$,

(ii) $f(t,y) = 1 + t\sin(ty)$ for $0 \leqslant t \leqslant 2$.

Exercise 2.2. Many text books, instead of giving the version of the Lipschitz condition we use, give the following: *There is a finite, positive, real number* L *such that*

$$\left|\frac{\partial}{\partial y} f(t,y)\right| \leqslant L \qquad \text{for all } (t,y) \in D.$$

Is this statement *stronger than* (i.e., more restrictive then), *equivalent to* or *weaker than* (i.e., less restrictive than) the usual Lipschitz condition? Justify your answer.

[2] Charles Emile Picard, France, 1856–1941. He made important discoveries in the fields of analysis, function theory, differential equations and geometry. He supervised the Ph.D. of Padraig de Brún, Professor of Mathematics (and President) of University College Galway/NUI Galway.

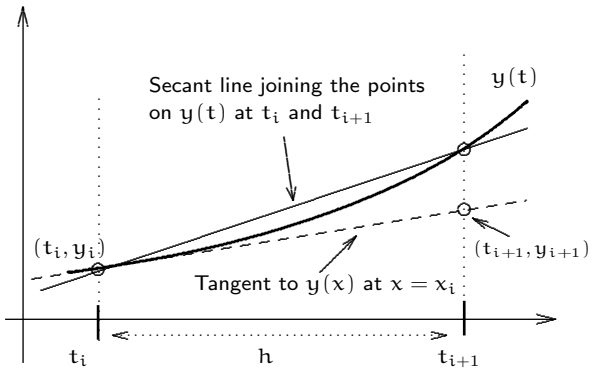
2.2 Euler's method

2.2.1 The idea

Classical numerical methods for IVPs attempt to generate approximate solutions at a finite set of discrete points $t_0 < t_1 < t_2 < \cdots < t_n$. The simplest is *Euler's Method*[3] and may be motivated as follows.

Suppose we know $y(t_i)$, and want to find $y(t_{i+1})$. From the differential equation we know the slope of the tangent to y at t_i. So, if this is similar to the slope of the line joining $(t_i, y(t_i))$ and $(t_{i+1}, y(t_{i+1}))$:

$$y'(t_i) = f(t_i, y(t_i)) \approx \frac{y_{i+1} - y_i}{t_{i+1} - t_i}.$$



2.2.2 The formula

Method 2.2.1 (Euler's Method)**.** Choose equally spaced points $t_0, t_1, \ldots, t_n$ so that

$$t_i - t_{i-1} = h = (t_n - t_0)/n \quad \text{for } i = 0, \ldots, n-1.$$

We call h the "time step". Let y_i denote the approximation for $y(t)$ at $t = t_i$. Set

$$y_{i+1} = y_i + hf(t_i, y_i), \quad i = 0, 1, \ldots, n-1. \quad (2.2.1)$$

2.2.3 An example

Example 2.2.2. Taking $h = 1$, estimate $y(4)$ where

$$y'(t) = y/(1 + t^2), \qquad y(0) = 1. \quad (2.2.2)$$

The true solution to this is $y(t) = e^{\arctan(t)}$.

Take notes:

If we had chosen $h = 4$ we would have only required one step: $y_n = y_0 + 4f(t_0, y_0) = 5$. However, this would not be very accurate. With a little work one can show that the solution to this problem is $y(t) = e^{\tan^{-1}(t)}$ and so $y(4) = 3.7652$. Hence the computed solution with $h = 1$ is much more accurate than the computed solution when $h = 4$. This is also demonstrated in Figure 2.1 below, and in Table 2.1 where we see that the error seems to be proportional to h.

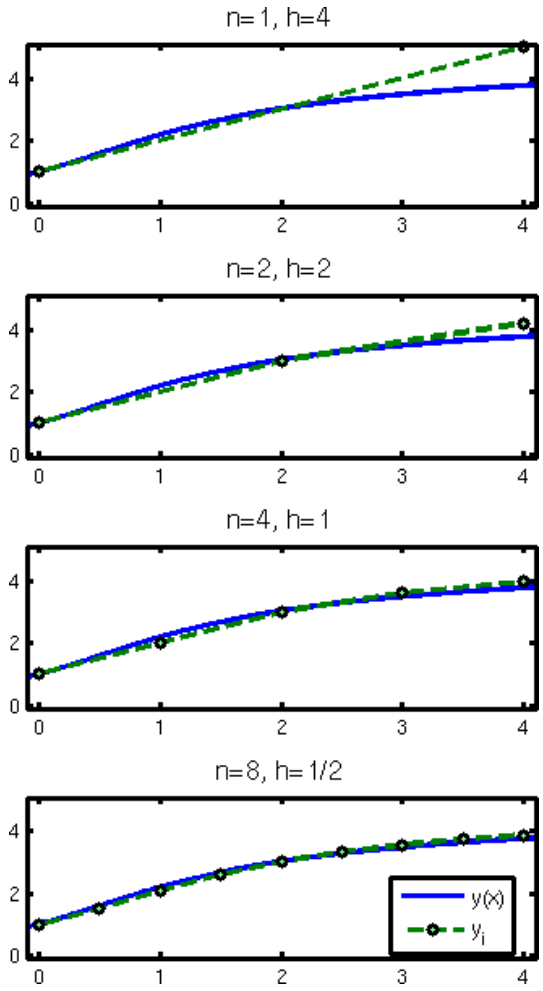


Fig. 2.1: Euler's method for Example 2.2.2 with $h = 4$, $h = 2$, $h = 1$ and $h = 1/2$

n	h	y_n	$\lvert y(t_n) - y_n \rvert$
1	4	5.0	1.235
2	2	4.2	0.435
4	1	3.960	0.195
8	1/2	3.881	0.115
16	1/4	3.831	0.065
32	1/8	3.800	0.035

Table 2.1: Error in Euler's method for Example 2.2.2

[3] Leonhard Euler, 1707–1783, Switzerland. One of the greatest Mathematicians of all time, he made vital contributions to geometry, calculus and number theory. finite differences, special functions, differential equations, continuum mechanics, astronomy, elasticity, acoustics, light, hydraulics, music, cartography, and much more.

2.2.4 Exercises

Exercise 2.3. As a special case in which the error of Euler's method can be analysed directly, consider Euler's method applied to

$$y'(t) = y(t), \quad y(0) = 1.$$

The true solution is $y(t) = e^t$.

(i) Show that the solution to Euler's method can be written as

$$y_i = (1+h)^{t_i/h}, \quad i \geqslant 0.$$

(ii) Show that

$$\lim_{h\to 0} (1+h)^{1/h} = e.$$

This then shows that, if we denote by $y_n(T)$ the approximation for $y(T)$ obtained using Euler's method with n intervals between t_0 and T, then

$$\lim_{n\to\infty} y_n(T) = e^T.$$

Hint: Let $w = (1+h)^{1/h}$, so that

$$\log w = (1/h)\log(1+h).$$

Now use l'Hospital's rule to find $\lim_{h\to 0} w$.

2.3 Error Analysis

2.3.1 General one-step methods

Euler's method is an example of a *one-step methods*, which have the general form:

$$y_{i+1} = y_i + h\Phi(t_i, y_i; h). \tag{2.3.1}$$

To get Euler's method, just take $\Phi(t_i, y_i; h) = f(t_i, y_i)$.

In the introduction, we motivated Euler's method with a geometrical argument. An alternative, more mathematical way of deriving Euler's Method is to use a *Truncated Taylor Series*.
Take notes:

This again motivates formula (2.2.1), and also suggests that at each step the method introduces a (local) error of $h^2y''(\eta)/2$. (More of this later).

2.3.2 Two types of error

We'll now give an error analysis for general one-step methods, and then look at Euler's Method as a specific example. First, some definitions.

Definition 2.3.1. *Global Error*: $\mathcal{E}_i = y(t_i) - y_i$.

Definition 2.3.2. *Truncation Error*:

$$T_i := \frac{y(t_{i+1}) - y(t_i)}{h} - \Phi(t_i, y(t_i); h). \tag{2.3.2}$$

It can be helpful to think of T_i as representing how much the difference equation (2.2.1) differs from the differential equation. We can also determine the truncation error for Euler's method directly from a Taylor Series.
Take notes:

The relationship between the global error and truncation errors is explained in the following (important!) result, which in turn is closely related to Theorem 2.1.3:

Theorem 2.3.3 (Thm 12.2 in Süli & Mayers)**.** *Let* $\Phi()$ *be* Lipschitz *with constant* L. *Then*

$$|\mathcal{E}_n| \leqslant T\left(\frac{e^{L(t_n - t_0)} - 1}{L}\right), \tag{2.3.3}$$

where $T = \max_{i=0,1,\ldots n} |T_i|$.

(Part of the following proof uses the fact that, if $|\mathcal{E}_{i+1}| \leqslant |\mathcal{E}_i|(1 + hL) + h|T_i|$, then

$$|\mathcal{E}_i| \leqslant \frac{T}{L}\left[(1 + hL)^i - 1\right] \qquad i = 0, 1, \ldots, N.$$

Show that this is indeed the case is an exercise).
Take notes:

2.3.3 Analysis of Euler's method

For Euler's method, we get

$$T = \max_{0 \leqslant j \leqslant n} |T_j| \leqslant \frac{h}{2} \max_{t_0 \leqslant t \leqslant t_n} |y''(t)|.$$

Example 2.3.4. Given the problem:

$$y' = 1 + t + \frac{y}{t} \quad \textit{for } t > 1; \quad y(1) = 1,$$

find an approximation for $y(2)$.

(i) Give an upper bound for the global error taking $n = 4$ (i.e., $h = 1/4$)

(ii) What n should you take to ensure that the global error is no more that 0.1?

To answer these questions we need to use (2.3.3), which requires that we find L and an upper bound for T. In this instance, L is easy:

Take notes:

(This is a particularly easy example. Often we need to employ the mean value theorem. See [1, Eg 12.2].)

To find T we need an upper bound for $|y''(t)|$ on $[1,2]$, even though we don't know $y(t)$. However, we do know $y'(t)$....

Take notes:

With these values of L and T, using (2.3.3) we find $\mathcal{E}_n \leqslant 0.644$. In fact, the true answer is 0.43, so we see that (2.3.3) is somewhat pessimistic.

To answer (ii): *What n should you take to ensure that the global error is no more that 0.1?* (We should get $n = 26$. This is not that sharp: $n = 19$ will do).

Take notes:

2.3.4 Convergence and Consistency

We are often interested in the *convergence* of a method. That is, is it true that

$$\lim_{h\to 0} y_n = y(t_n)?$$

Or equivalently that,

$$\lim_{h\to 0} \mathcal{E}_n = 0?$$

Given that the global error for Euler's method can be bounded:

$$|\mathcal{E}_n| \leqslant h\frac{\max|y''(t)|}{2L}\left(e^{L(t_n-t_0)}-1\right) = hK, \quad (2.3.4)$$

we can say it converges.

Definition 2.3.5. The **order of accuracy** of a numerical method is p if there is a constant K so that

$$|\mathcal{E}_n| \leqslant Kh^p.$$

(The term **order of convergence** is often use instead of **order of accuracy**).

In light of this definition, we read from (2.3.4) that Euler's method is first-order.

One of the requirements for convergence is *Consistency*:

Definition 2.3.6. A one-step method $y_{n+1} = y_n + h\Phi(t_n, y_n; h)$ is *consistent* with the differential equation $y'(t) = f\big(t, y(t)\big)$ if $f(t,y) \equiv \Phi(t,y;0)$.

It is quite trivial to see that Euler's method is consistent. In the next section, we'll try to develop methods that are of higher order than Euler's method. That is, we will study methods for which one can show

$$|\mathcal{E}_n| \leqslant Kh^p \quad \text{for some} \quad p > 1.$$

For these, showing consistency is a little (but only a little) more work.

2.3.5 Exercises

Exercise 2.4. An important step in the proof of Theorem 2.3.3, but which we didn't do in class, requires the observation that if $|\mathcal{E}_{i+1}| \leqslant |\mathcal{E}_i|(1+hL) + h|T_i|$, then

$$|\mathcal{E}_i| \leqslant \frac{T}{L}\big[(1+hL)^i - 1\big] \qquad i = 0, 1, \ldots, N.$$

Use induction to show that is indeed the case.

Exercise 2.5. Suppose we use Euler's method to find an approximation for $y(2)$, where y solves

$$y(1) = 1, \qquad y' = (t-1)\sin(y).$$

(i) Give an upper bound for the global error taking $n = 4$ (i.e., $h = 1/4$)

(ii) What n should you take to ensure that the global error is no more that 10^{-3}?

2.4 Runge-Kutta 2 (RK2)

The goal of this section is to develop some techniques to help us derive our own methods for accurately solving Initial Value Problems. Rather than using formal theory, the approach will be based on carefully chosen examples.

As a motivation, suppose we numerically solve some differential equation and estimate the error. If we think this error is too large, we could re-do the calculation with a smaller value of h. Or we could use a better method, where the error is proportional to h^2, or h^3, etc. These are high-order "Runge-Kutta" methods rely on evaluating $f(t, y)$ a number of times at each step in order to improve accuracy.

First, in Section 2.4.1, we'll motivate one such method. Then, in 2.4.2, we'll look at the general framework.

2.4.1 Modified Euler Method

Recall the motivation for Euler's method from §2.2. We can do something similar to derive what's often called the *Modified Euler's method*, or, less commonly, the *Midpoint Euler's method*.

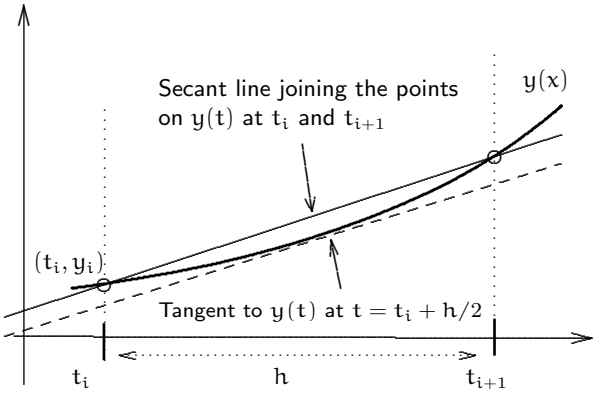
In Euler's method, we use the slope of the tangent to y at t_i as an approximation for the slope of the secant line joining the points $(t_i, y(t_i))$ and $(t_{i+1}, y(t_{i+1}))$.

One could argue, given the diagram below, that the slope of the tangent to y at $t = (t_i + t_{i+1})/2 = t_i + h/2$ would be a better approximation. This would give

$$y(t_{i+1}) \approx y_i + hf\big(t_i + \frac{h}{2}, y(t_i + \frac{h}{2})\big). \qquad (2.4.1)$$

However, we don't know $y(t_i + h/2)$, but can approximate it using Euler's Method: $y(t_i + h/2) \approx y_i + (h/2)f(t_i, y_i)$. Substituting this into (2.4.1) gives

$$y_{i+1} = y_i + hf\big(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)\big). \qquad (2.4.2)$$



Example 2.4.1. *Use the Modified Euler Method to approximate* $y(1)$ *where*

$$y(0) = 1, \qquad y'(t) = y\log(1 + t^2).$$

This has the solution $y(t) = (1+t^2)^t \exp(-2t+2\tan^{-1} t)$. In Table 2.2, and also Table 2.4.1, we compare the error in the solution to this problem using Euler's Method (left) and the Modified Euler's Method (right) for various values of n.

Table 2.2: Errors in solutions to Example 2.4.1 using Euler's and Modified Euler's Methods

	Euler		**Modified**	
n	$\mathcal{E}_n$	$\mathcal{E}_n/\mathcal{E}_{n-1}$	$\mathcal{E}_n$	$\mathcal{E}_n/\mathcal{E}_{n-1}$
1	3.02e-01		7.89e-02	
2	1.90e-01	1.59	2.90e-02	2.72
4	1.11e-01	1.72	8.20e-03	3.54
8	6.02e-02	1.84	2.16e-03	3.79
16	3.14e-02	1.91	5.55e-04	3.90
32	1.61e-02	1.95	1.40e-04	3.95
64	8.13e-03	1.98	3.53e-05	3.98
128	4.09e-03	1.99	8.84e-06	3.99

Clearly we get a much more accurate result using the Modified Euler Method. Even more importantly, we get a higher *order of accuracy*: if we reduce h by a factor of two, the error in the Modified method is reduced by a factor of four.

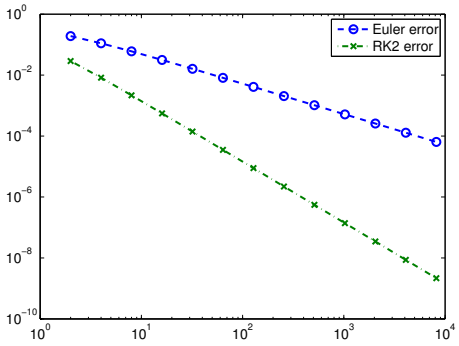


Fig. 2.2: Log-log plot of the errors when Euler's and Modified Euler's methods are used to solve Example 2.4.1

2.4.2 General RK2

The "*Modified Euler Method*" we have just studied is an example of one of the (large) family of 2^{nd}*-order Runge-Kutta* (RK2) methods. Recalling that one-step methods are written as

$$y_{i+1} = y_i + h\Phi(t_i, y_i; h),$$

then the general RK2 method is

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_i + \alpha h, y_i + \beta h k_1). \qquad (2.4.3) \\ \Phi(t_i, y_i; h) &= (ak_1 + bk_2) \end{aligned}$$

If we choose a, b, α and β in the right way, then the error for the method will be bounded by Kh^2, for some constant K.

An (uninteresting) example of such a method is if we take $a = 1$ and $b = 0$, it reduces to Euler's Method. If we can choose $\alpha = \beta = 1/2, a = 0, b = 1$ and get the "Modified" method above.

Our aim now is to deduce general rules for choosing a, b, α and β. We'll see that if we pick any one of these four parameters, then the requirement that the method be consistent and second-order determines the other three.

2.4.3 Using consistency

By demanding that RK2 be *consistent* we get that $a + b = 1$.

Take notes:

2.4.4 Ensuring that RK2 is 2^{nd}-order

Next we need to know how to choose α and β. The formal way is to use a two-dimensional Taylor series expansion. It is quite technical, and not suitable for doing in class. Detailed notes on it are given in Section 2.4.6 below. Instead we'll take a less rigorous, heuristic approach.

Because we expect that, for a second order accurate method, $|\mathcal{E}_n| \leqslant Kh^2$ where K depends on $y'''(t)$, if we choose a problem for which $y'''(t) \equiv 0$, we expect no error...

Take notes:

In the above example, the right-hand side of the differential equation, $f(t, y)$, depended only on t. Now we'll try the same trick: using a problem with a simple known solution (and zero error), but for which f depends explicitly on y.

Consider the DE $y(1) = 1, y'(t) = y(t)/t$. It has a simple solution: $y(t) = t$. We now use that any RK2 method should be exact for this problem to deduce that $\alpha = \beta$.

Take notes:

Now we collect the above results all together and show that the second-order Runge-Kutta (RK2) methods are:

$$y_{i+1} = y_i + h(ak_1 + bk_2)$$

$$k_1 = f(t_i, y_i), \qquad k_2 = f(t_i + \alpha h, y_i + \beta h k_1),$$

where we choose any $b \neq 0$ and then set

$$a = 1 - b, \qquad \alpha = \frac{1}{2b}, \qquad \beta = \alpha.$$

It is easy to verify that the Modified method satisfies these criteria.

2.4.5 Exercises

Exercise 2.6. A popular RK2 method, called the *Improved Euler Method*, is obtained by choosing $\alpha = 1$.

(i) Use the Improved Euler Method to find an approximation for $y(4)$ when

$$y(0) = 1, \qquad y' = y/(1 + t^2),$$

taking $n = 2$. (If you wish, use Matlab.)

(ii) Using a diagram similar to the one above for the Modified Euler Method, justify the assertion that the Improved Euler Method is more accurate than the basic Euler Method.

(iii) Show that the method is consistent.

(iv) Write out what this method would be for the problem: $y'(t) = \lambda y$ for a constant λ. How does this relate to the Taylor series expansion for $y(t_{i+1})$ about the point t_i?

Exercise 2.7. In his seminal paper of 1901, Carl Runge gave the following example of what we now call a *Runge-Kutta 2 method*:

$$y_{i+1} = y_i + \frac{h}{4}\Big[f(t_i, y_i) + 3f\big(t_i + \frac{2}{3}h, y_i + \frac{2}{3}hf(t_i, y_i)\big)\Big].$$

(i) Show that it is consistent.

(ii) Show how this method fits into the general framework of RK2 methods. That is, what are a, b, α, and β? Do they satisfy the following conditions?

$$\beta = \alpha, \qquad b = \frac{1}{2\alpha}, \qquad a = 1 - b. \tag{2.4.4}$$

(iii) Use it to estimate the solution at the point $t = 2$ to $y(1) = 1$, $y' = 1 + t + y/t$ taking $n = 2$ time steps.

2.4.6 Formal Dirivation of RK2

This section is based on [1, p422]. We won't actually cover this in class, instead opting to to deduce the same result in an easier, but unrigorous way in Section 2.4.4. If were to do it properly, this how we would do it.

We will require that the Truncation Error (2.3.2) being second-order. More precisely, we want to be able to say that

$$|T_n| = \Big|\frac{y(t_{n+1}) - y(t_n)}{h} - \Phi(t_n, y(t_n); h)\Big| \leqslant Ch^2$$

where C is some constant that does not depend on n or h.

So the problem is to find expressions (using Taylor series) for both $\big(y(t_{n+1}) - y(t_n)\big)/h$ and $\Phi(t_n, y(t_n); h)$ that only have $\mathcal{O}(h^2)$ remainders. To do this we need to recall two ideas from 2nd year calculus:

- To differentiate a function $f\big(a(t), b(t)\big)$ with respect to t:
$$\frac{df}{dt} = \frac{\partial f}{\partial a}\frac{da}{dt} + \frac{\partial f}{\partial b}\frac{db}{dt};$$
- The Taylor Series for a function of 2 variables, truncated at the 2nd term is:
$$f(x + \eta_1, y + \eta_2) = f(x, y) + \eta_1\frac{\partial f}{\partial x}(x, y) + \eta_2\frac{\partial f}{\partial y}(x, y) + C(\max\{\eta_1, \eta_2\})^2.$$
for some constant C. See [1, p422] for details.

To get an expression for $|y(t_{n+1}) - y(t_n)|$, use a Taylor Series:

$$\begin{aligned}
y(t_{n+1}) &= y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \mathcal{O}(h^3)\\
&= y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}\Big(f\big(t_n, y(t_n)\big)\Big)' + \mathcal{O}(h^3)\\
&= y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}\Big(\frac{\partial}{\partial t}f\big(t_n, y(t_n)\big) + y'(t_n)\frac{\partial}{\partial y}f\big(t_n, y(t_n)\big)\Big) + \mathcal{O}(h^3),\\
&= y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}\Big[\frac{\partial}{\partial t}f + f\frac{\partial}{\partial y}f\Big]\big(t_n, y(t_n)\big) + \mathcal{O}(h^3),\\
&= y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}\Big(\{_t + \{_y\mathcal{F}\Big) + \mathcal{O}(h^3).
\end{aligned}$$

This gives

$$\frac{|y(t_{n+1}) - y(t_n)|}{h} = f\big(t_n, y(t_n)\big) + \frac{h}{2}\Big[\frac{\partial}{\partial t}f + f\frac{\partial}{\partial y}f\Big]\big(t_n, y(t_n)\big) + \mathcal{O}(h^2). \tag{2.4.5}$$

Next we expand the expression $f(t_i + \alpha h, y_i + \beta h f(t_i, y_i)$ using a (two dimensional) Taylor Series:

$$\begin{aligned}
f\big(t_n + \alpha h, y_n + \beta h f(t_n, y_n)\big) = {}& f(t_n, y_n) + h\alpha\frac{\partial}{\partial t}f(t_n, y_n) +\\
& h\beta f(t_n, y_n)\frac{\partial}{\partial y}f(t_n, y_n)\big) + \mathcal{O}(h^2).
\end{aligned}$$

This leads to the following expansion for $\Phi(t_n, y(t_n))$:

$$\Phi\big(t_n, y(t_n); k\big) = (a + b)f\big(t_n, y(t_n)\big) + h\Big[b\alpha\frac{\partial}{\partial t}f + b\beta f\frac{\partial}{\partial y}f\Big]\big(t_n, y(t_n)\big) + \mathcal{O}(h^2). \tag{2.4.6}$$

So now, if we are to subtract (2.4.6) from (2.4.5) and leave only terms of $\mathcal{O}(h^2)$ we have to choose $a + b = 1$, $b\alpha = 1/2 = b\beta = 1/2$. That is, choose α and let

$$\beta = \alpha, \qquad b = \frac{1}{2\alpha}, \qquad a = 1 - b. \tag{2.4.7}$$

(For a more detailed exposition, see [1, Chap 12]).

2.5 LAB 2: Euler's Method

In this session you'll develop your knowledge of MATLAB by using it to implement Euler's Method for IVPs, and study its their order of accuracy.

You should be able to complete at least Section 2.5.8 today. At the end of the class, **verify your participation by uploading your results to Blackboard** (go to "Assignments and Labs" and then "Lab 2"). In Lab 3, you'll implement higher-order schemes.

2.5.1 Four ways to define a vector

We know that the most fundamental object in MATLAB is a matrix. The the simplest (nontriveal) example of a matrix is a vector. So we need to know how to define vectors. Here are several ways we could define the vector

$$x = (0, .2, .4, \ldots, 1.8, 2.0). \tag{2.5.1}$$

```
x = 0:0.2:2 % From 0 to  2 in steps of 0.2
x = linspace(0, 2, 11); % 11 equally spaced
              % points with x(1)=0, x(11)=2.
x = [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, ...
  1.6, 1.8, 2.0]; % Define points individually
```

The last way is rather tedious, but this one is worse:

```
x(1)=0.0; x(2)=0.2, x(3)=0.4; x(4)=0.6; ...
```

We'll see a less tedious way of doing this last approach in Section 2.5.3 below.

2.5.2 Script files

MATLAB is an *interpretative* environment: if you type a (correct) line of MATLAB code, and hit return, it will execute it immediately. For example: try `>> exp(1)` to get a decimal approximation of e.

However, we usually want to string together a collection of MATLAB operations and run the repeatedly. To do that, it is best to store these commands in a *script* file. This is done be making a file called, for example, *Lab2.m* and placing in it a series of MATLAB commands. A script file is run from the MATLAB command window by typing the name of the script, e.g., `>> Lab2`

Try putting some of the above commands for defining a vector into a script file and run it.

2.5.3 `for`–loops

When we want to run a particular set of operations a fixed number of times we use a `for`–loop.

It works by iterating over a vector; at each iteration the *iterand* takes each value stored in the vector in turn.

For example, here is another way to define the vector in (2.5.1):

```
for i=0:10 % 0:10 is the vector [0,1,...,10]
  x(i+1) = i*0.2;
end
```

2.5.4 Functions

In MATLAB we can define a function in a way that is quite similar to the mathematical definition of a function. The syntax is `>> Name = @(Var)(Formula);`
Examples:

```
f = @(x)(exp(x) - 2*x -1);
g = @(x)(log(2*x +1));
```

Now, for example, if we call $g(1)$, it evaluates as $\log(3) = 1.098612288668110$. Furthermore, if `x` is a vector, so too is `g(x)`.

A more interesting example would be to try

```
>> xk = 1;
```

and the repeat the line

```
>> xk = g(xk)
```

Try this and observe that the values of `xk` seem to be converging. This is because we are using *Fixed Point Iteration*.

Later we'll need to know how to define functions of two variables. This can be done as:

```
F = @(y,t)(y./(1 + t.^2));
```

2.5.5 Plotting functions

MATLAB has two ways to plot functions. The easiest way is to use a function called `ezplot`:

```
ezplot(f, [0, 2]);
```

plots the function $f(x)$ for $0 \leqslant x \leqslant 2$. A more flexible approach is to use the `plot` function, which can plot one vector as a function of another. Try these examples below, making sure you first have defined the vector x and the functions f and g:

```
figure(1);
plot(x,f(x));
figure(2);
plot(x,f(x), x, g(x),  '--', x,x, '-.');
```

Can you work out what the syntax `'--'` and `'-.'` does? If not, ask a tutor. Also try

```
plot(x,f(x), 'g-o', x, g(x),  'r--x', ...
   x,x, '-.');
```

2.5.6 How to learn more

These notes are not an encyclopedic guide to MATLAB – they have just enough information to get started. There are many good references online.

Exercise: access Learning MATLAB by Tobin Driscoll through the NUI Galway library portal. Read Section 1.6: "Things about MATLAB that are very nice to know, but which often do not come to the attention of beginners".

2.5.7 Initial Value Problems

The particular example of an IVP that we'll look at in this lab is: *estimate* $y(4)$ *given that* is one that we had earlier in (2.2.2):

$$y'(t) = y/(1+t^2), \text{ for } t > 0, \quad \text{and } y(0) = 1.$$

The true solution to this is $y(t) = e^{\arctan(t)}$. If you don't want to solve this problem by hand, you could use Maple. The Maple command is:

```
dsolve({D(y)(t)=y(t)/(1+t^2),y(0)=1},y(t));
```

2.5.8 Euler's Method

Euler's Method is

- Choose n, the number of points at which you will estimate $y(t)$. Let $h = (t_n - t_0)/n$, and $t_i = t_0 + ih$.
- For $i = 0, 1, 2, \ldots, n-1$ set $y_{i+1} = y_i + hf(t_i, y_i)$.

Then $y(t_n) \approx y_n$. As shown in Section 2.3.3, the global error for Euler's method can be bounded:

$$|\mathcal{E}_n| := |y(T) - y_n| \leqslant Kh,$$

for some constant K that does not depend on h (or n). That is, if h is halved (i.e., n is doubled), the error is halved as well.

Download the MATLAB script file `Euler.m`. It can be run in MATLAB simply by typing `>> Euler`
It implements Euler's method for $n = 1$. Read the file carefully and make sure you understand it.

The program computes a vector `y` that contains the estimates for y at the time-values specified in the vector `t`. However, MATLAB indexes all vectors from 1, and not 0. So $t(1) = t_0$, $t(2) = t_1$, ... $t(n+1) = t_n$.

By changing the value of n, complete the table.

We want to use this table to verify that Euler's Method is 1^{st}-order accurate. That is:

$$|\mathcal{E}_n| \leqslant Kh^\rho \qquad \text{with} \qquad \rho = 1.$$

A computational technique that verifies the order of the method is to estimate ρ by

$$\rho \approx \log_2 \left(\frac{|\mathcal{E}_n|}{|\mathcal{E}_{2n}|} \right). \tag{2.5.2}$$

Table 2.3: Complete this table showing the convergence of Euler's method

n	y_n	$\mathcal{E}_n$	ρ
2	4.2	4.347×10^{-1}	
4	3.96	1.947×10^{-1}	
8			
16			
32			
64			
128			
256			
512			

Use the data in the table verify that $\rho \approx 1$ for Euler's Method. **Upload these results to the** `Assignments -- Lab 2` **section of Blackboard**. For example, take a photo of the table and upload that. However, it would be better to upload a modified version of the `Euler.m` script that computes ρ for each n.

2.5.9 More MATLAB: formatted output

When you run the `Euler.m` script file, you'll see that the output is not very pretty. In particular, the data are not nicely tabulated. The script uses the `fprintf` function to display messages and values. Its basic syntax is:
`fprintf('Here is a message\n');`
where the `\n` indicates a new line.

Using `fprintf` to display the contents of a variable is a little more involved, and depends on *how* we want the data displayed. For example:

- To display an integer:
  `fprintf('Using n=%d steps\n', n);`
- To display an integer, padded to a maximum of 8 spaces:
  `fprintf('Using n=%8d steps\n', n);`
- To display a floating point number:
  `fprintf('y(n)=%f\n', Y(n+1));`
- To display in exponential notation:
  `fprintf('Error=%e\n', Error);`
- To display 3 decimal places:
  `fprintf('y(n)=%.3f\n', Y(n+1));`
  `fprintf('Error=%.3e\n', Error);`

Use these to improve the formatting of the output from your `Euler.m` script so that the output is nicely tabulated. To get more information on `fprintf`, type
`>> doc fprintf`
or ask one of the tutors.

2.6 Runge-Kutta 4

2.6.1 RK 4

It is possible to construct methods that have rates of convergence that are higher than RK2 methods, such as the *RK4* methods, for which

$$|y(t_n) - y_n| \leqslant Ch^4.$$

However, writing down the general form of the RK4 method, and then deriving conditions on the parameters is rather complicated. Therefore, we'll simply state the most popular RK4 method, without proving that it is 4th-order.

4th-Order Runge Kutta Method (RK4):

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1), \\ k_3 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2), \\ k_4 &= f(t_i + h, y_i + hk_3), \\ y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

It can be interpreted as

k_1 is the slope of $y(t)$ at t_i.

k_2 is an approximation for the slope of $y(t)$ at $t_i + h/2$ (using Euler's Method).

k_3 is an improved approximation for the slope of $y(t)$ at $t_i + h/2$.

k_4 is an approximation for the slope of $y(t)$ at t_{i+1} computed using the slope at $y(t_i + h/2)$.

Finally The slope of the secant line joining $y(t)$ at the points t_i and t_{i+1} is approximated using a weighted average of the of the above values.

Example 2.6.1. Recall the test problem from Example 2.4.1 Table 2.4 and Table 2.6.1 give the errors in the solutions computed using various methods and values of n.

2.6.2 RK4: consistency and convergence

Although we won't do a detailed analysis of RK4, we can do a little. In particular, we would like to show it is

(i) consistent,

(ii) convergent and fourth-order, at least for some examples.

Example 2.6.2. It is easy to see that RK4 is consistent:

Table 2.4: Errors in solutions to Example 2.4.1 using Euler's, Modified, and RK4

	$\lvert y(t_n) - y_n \rvert$		
n	Euler	Modified	RK4
1	3.02e-01	7.89e-02	8.14e-04
2	1.90e-01	2.90e-02	1.08e-04
4	1.11e-01	8.20e-03	5.07e-06
8	6.02e-02	2.16e-03	2.44e-07
16	3.14e-02	5.55e-04	1.27e-08
32	1.61e-02	1.40e-04	7.11e-10
64	8.13e-03	3.53e-05	4.18e-11
128	4.09e-03	8.84e-06	2.53e-12
256	2.05e-03	2.21e-06	1.54e-13
512	1.03e-03	5.54e-07	7.33e-15

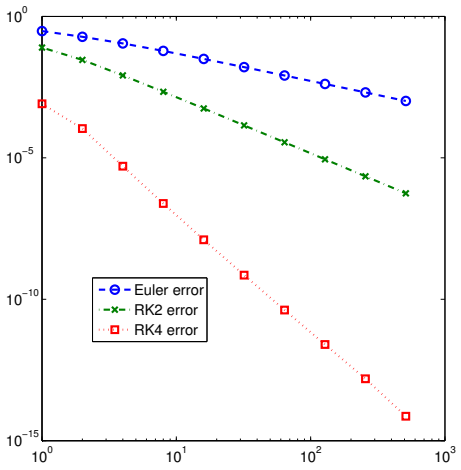


Fig. 2.3: Log-log plot of the errors when Euler's, Modified Euler's, and RK-4 methods are used to solve Example 2.4.1

Take notes:

Example 2.6.3. In general, showing the rate of convergence is tricky. Instead, we'll demonstrate how the method relates to a Taylor Series expansion for the problem $y' = \lambda y$ where λ is a constant.

Take notes:

2.6.3 The (Butcher) Tableau

A great number of RK methods have been proposed and used through the years. A unified approach of representing and studying them was developed by John Butcher (Auckland, New Zealand). In his notation, we write an s-stage method as

$$\Phi(t_i, y_i; h) = \sum_{j=1}^{s} b_j k_j, \quad \text{where}$$

$$\begin{aligned}
k_1 &= f(t_i + \alpha_1 h, y_i),\\
k_2 &= f(t_i + \alpha_2 h, y_i + \beta_{21} h k_1),\\
k_3 &= f(t_i + \alpha_3 h, y_i + \beta_{31} h k_1 + \beta_{32} h k_2),\\
&\vdots\\
k_s &= f(t_i + \alpha_s h, y_i + \beta_{s1} h k_1 + \ldots \beta_{s,s-1} h k_{s-1}),
\end{aligned}$$

The most convenient way to represent the coefficients is in a tableau:

$$\begin{array}{c|cccc}
\alpha_1 & & & & \\
\alpha_2 & \beta_{21} & & & \\
\alpha_3 & \beta_{31} & \beta_{32} & & \\
\vdots & & & & \\
\alpha_s & \beta_{s1} & \beta_{s2} & \cdots & \beta_{s,s-1} \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} \quad b_s
\end{array}$$

The tableau for the basic Euler method is trivial:

$$\begin{array}{c|c}
0 & \\
\hline
 & 1
\end{array}$$

The two best-known RK2 methods are probably the "Modified Euler method" and the "Improved Euler Method". Their tableaux are:

$$\begin{array}{c|cc}
0 & & \\
1/2 & 1/2 & \\
\hline
 & 0 & 1
\end{array} \qquad \text{and} \qquad \begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & 1/2 & 1/2
\end{array}$$

A three-stage method, some times called "RK3-1" has the tableau

$$\begin{array}{c|ccc}
0 & & & \\
2/3 & 2/3 & & \\
2/3 & 1/3 & 1/3 & \\
\hline
 & 1/4 & 0 & 3/4
\end{array}$$

The tableau for the RK4 method above is:

$$\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 2/6 & 2/6 & 1/6
\end{array}$$

You should now convince yourself that these tableaux do indeed correspond to the methods we did in class.

2.6.4 Even higher-order methods?

A Runge Kutta method has s stages if it involves s evaluations of the function f. (That it, its formula features $k_1, k_2, \ldots, k_s$). We've seen a one-stage method that is 1^{st}-order, a two-stage method that is 2^{nd}-order, ..., and a four-stage method that is 4^{th}-order. It is tempting to think that for any s we can get a method of order s using s stages. However, it can be shown that, for example, to get a 5^{th}-order method, you need at least 6 stages; for a 7^{th}-order method, you need at least 9 stages. The theory involved is both intricate and intriguing, and involves aspects of group theory, graph theory, and differential equations. Students in third year might consider this as a topic for their final year project.

2.6.5 Exercises

Exercise 2.8. We claim that, for RK4:

$$|\mathcal{E}_N| = |y(t_N) - y_N| \leqslant K h^4.$$

for some constant K. How could you verify that the statement is true using the data of Table 2.3, at least for test problem in Example 2.4.2? Give an estimate for K.

Exercise 2.9. Recall the problem in Example 2.2.2: *Estimate* $y(2)$ *given that*

$$y(1) = 1, \qquad y' = f(t, y) := 1 + t + \frac{y}{t},$$

(i) Show that $f(t, y)$ satisfies a Lipschitz condition and give an upper bound for L.

(ii) Use Euler's method with $h = 1/4$ to estimate $y(2)$. Using the true solution, calculate the error.

(iii) Repeat this for the RK2 method of your choice (with $a \neq 0$) taking $h = 1/2$.

(iv) Use RK4 with $h = 1$ to estimate $y(2)$.

Exercise 2.10. Here is the tableau for a three stage Runge-Kutta method:

$$\begin{array}{c|ccc}
0 & & & \\
\alpha_2 & 1/2 & & \\
1 & \beta_{31} & 2 & \\
\hline
 & 1/6 & b_2 & 1/6
\end{array}$$

1. Use that the method is consistent to determine b_2.
2. The method is exact when used to compute the solution to

   $$y(0) = 0, \quad y'(t) = 2t, \ t > 0.$$

   Use this to determine α_2.
3. The method should agree with an appropriate Taylor series for the solution to $y'(t) = \lambda y(t)$, up to terms that are $\mathcal{O}(h^3)$. Use this to determine β_{31}.

2.7 From IVPs to Linear Systems

In this final theoretical section, we highlight some of the many important aspects of the numerical solution of IVPs that are *not* covered in detail in this course:

- Systems of ODEs;
- Higher-order equations;
- Implicit methods; and
- Problems in two dimensions.

We have the additional goal of seeing how these methods related to the earlier section of the course (nonlinear problems) and next section (linear equation solving).

2.7.1 Systems of ODEs

So far we have solved only single IVPs. However, must interesting problems are coupled systems: find functions y and z such that

$$y'(t) = f_1(t, y, z),$$
$$z'(t) = f_2(t, y, z).$$

This does not present much of a problem to us. For example the Euler Method is extended to

$$y_{i+1} = y_i + hf_1(t, y_i, z_i),$$
$$z_{i+1} = z_i + hf_2(t, y_i, z_i).$$

Example 2.7.1. In pharmokinetics, the flow of drugs between the blood and major organs can be modelled

$$\frac{dy}{dt}(t) = k_{21}z(t) - (k_{12} + k_{elim})y(t).$$
$$\frac{dz}{dt}(t) = k_{12}y(t) - k_{21}z(t).$$
$$y(0) = d, \quad z(0) = 0.$$

where y is the concentration of a given drug in the bloodstream and z is its concentration in another organ. The parameters k_{21}, k_{12} and k_{elim} are determined from physical experiments.

Euler's method for this is:

$$y_{i+1} = y_i + h\big(-(k_{12} + k_{elim})y_i + k_{21}z_i\big),$$
$$z_{i+1} = z_i + h\big(k_{12}y_i + k_{21}z_i\big).$$

2.7.2 Higher-Order problems

So far we've only considered first-order initial value problems:

$$y'(t) = f(t, y); \qquad y(t_0) = y_0.$$

However, the methods can easily be extended to high-order problems:

$$y''(t) + a(t)y'(t) = f(t, y); \quad y(t_0) = y_0, y'(t_0) = y_1.$$

We do this by converting the problem to a system: set $z(t) = y'(t)$. Then:

$$z'(t) = -a(t)z(t) + f(t, y), \qquad z(t_0) = y_1,$$
$$y'(t) = z(t), \qquad y(t_0) = y_0.$$

Example 2.7.2. Consider the following 2nd-order IVP

$$y''(t) - 3y'(t) + 2y(t) + e^t = 0,$$
$$y(1) = e,\ y'(1) = 2e.$$

Let $z = y'$, then

$$z'(t) = 3z(t) - 2y(t) + e^t, \quad z(0) = 2e$$
$$y'(t) = z(t), \qquad y(0) = e.$$

Euler's Method is

$$z_{i+1} = z_i + h\big(3z_i - 2y_i + e^{t_i}\big),$$
$$y_{i+1} = y_i + hz_i.$$

2.7.3 Implicit methods

Although we won't dwell on the point, there are many problems for which the one-step methods we have seen will give a useful solution only when the step size, h, is small enough. For larger h, the solution can be very unstable. Such problems are called "stiff" problems. They can be solved, but are best done with so-called "implicit methods", the simplest of which is the Implicit Euler Method:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}).$$

Note that y_{i+1} appears on both sizes of the equation. To implement this method, we need to be able to solve this non-linear problem. The most common method for doing this is Newton's method.

2.7.4 Towards Partial Differential Equations

So far we've only considered *ordinary* differential equations: these are DEs which involve functions of just one variable. In our examples above, this variable was time.

But of course many physical phenomena vary in space and time, and so the solutions to the differential equations the model them depend on two or more variables. The derivatives expressed in the equations are *partial derivatives* and so they are called *partial differential equations* (PDEs).

We will take a brief look at how to solve these (and how not to solve them). This will motivate the following section, on solving systems of linear equations.

Students of financial mathematics will be familiar with the Black-Scholes equations for pricing an option, which we mentioned in Section 1.6:

$$\frac{\partial V}{\partial t} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rS\frac{\partial V}{\partial S} + rV = 0.$$

With a little effort, (see, e.g., Chapter 5 of "*The Mathematics of Financial Derivatives: a student introduction*", by Wilmott, Howison, and Dewynne) this can be transformed to the simpler-looking *heat equation*:

$$\frac{\partial u}{\partial t}(t,x) = \frac{\partial^2 u}{\partial x^2}(t,x), \quad \text{for} \quad (x,t) \in [0,L] \times [0,T],$$

and with the initial and boundary conditions

$$u(0,x) = g(x) \qquad \text{and } u(t,0) = a(t), u(t,L) = b(t).$$

Example 2.7.3. If $L = \pi$, $g(x) = \sin(x)$, $a(t) = b(t) \equiv 0$ then $u(t,x) = e^{-t}\sin(x)$ (see Figure 2.4).

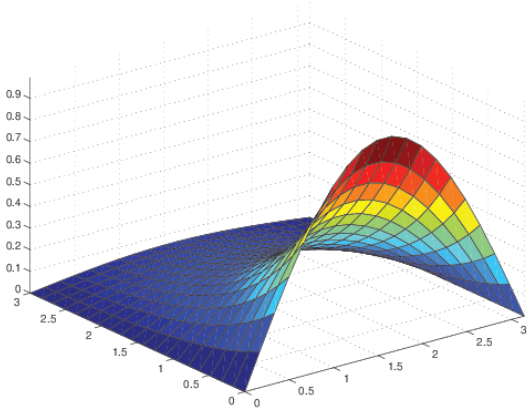


Fig. 2.4: The true solution to the heat equation

In general, however, for arbitrary g, a, and b, a explicit solution to this problem is not available, so a numerical scheme must be used. Suppose we somehow know $\partial^2 u/\partial x^2$, then we could just use Euler's method:

$$u(t_{i+1},x) = u(t_i,x) + h\frac{\partial^2 u}{\partial x^2}(t_i,x).$$

Although we don't know $\frac{\partial^2 u}{\partial x^2}(t_i,x)$ we can *approximate* it. The algorithm is as follows:

1. Divide $[0,T]$ into N intervals of width h, giving the grid $\{0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T\}$, with $t_i = t_0 + ih$.
2. Divide $[0,L]$ into M intervals of width H, giving the grid $\{0 = x_0 < x_1 < \cdots < x_M = L\}$ with $x_j = x_0 + jH$.
3. Denote by $u_{i,j}$ the approximation for $u(t,x)$ at (t_i, x_j).
4. For each $i = 0, 1, \ldots, N-1$, use the following approximation for $\frac{\partial^2 u}{\partial x^2}(t_i, x_j)$

   $$\delta_x^2 u_{i,j} = \frac{1}{H^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}),$$

   for $k = 1, 2, \ldots, M-1$, and then take

   $$u_{i+1,j} := u_{i,j} - h[\delta_x^2 u_{i,j}].$$

This scheme is called an *explicit method*: if we know $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ then we can explicitly calculate $u_{i+1,j}$. See Figure 2.5.

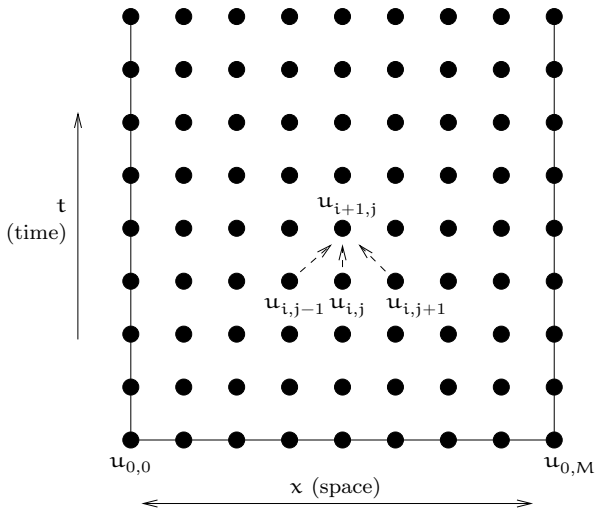


Fig. 2.5: A finite difference grid: if we know $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ we can calculate $u_{i+1,j}$.

Unfortunately, as we will see in class, this method is not very stable. Unless we are very careful choosing h and H, huge errors occur in the approximation for larger i (time steps).

Instead one might use an *implicit method*: if we know $u_{i-1,j}$, we compute $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ simultaneously. More precisely: solve $u_{i,j} - h[\delta_x^2 u_{i,j}] = u_{i-1,j}$ for $i = 1$, then $i = 2$, $i = 3$, etc. Expanding the δ_x^2 term we get, for each $i = 1, 2, 3, \ldots$, the set of simultaneous equations

$$\begin{gathered} u_{i,0} = a(t_i), \\ \alpha u_{i,j-1} + \beta u_{i,j} + \alpha u_{i,j+1} = u_{i-1,k}, \quad k = 1, 2, \ldots, M-1 \\ u_{i,M} = b(t_i), \end{gathered}$$

where $\alpha = -\frac{h}{H^2}$ and $\beta = \frac{2h}{H^2} + 1$. This could be expressed more clearly as the matrix-vector equation:

$$Ax = f,$$

where

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ \alpha & \beta & \alpha & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & \alpha & \ldots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \ldots & \alpha & \beta & \alpha & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & \alpha & \beta & \alpha \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 \end{pmatrix},$$

and

$$x = \begin{pmatrix} u_{i,0} \\ u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,n-2} \\ u_{i,n-1} \\ u_{i,n} \end{pmatrix}, y = \begin{pmatrix} a(0) \\ u_{i-1,1} \\ u_{i-1,2} \\ \vdots \\ u_{i-1,n-2} \\ u_{i-1,n-1} \\ b(T) \end{pmatrix}.$$

So "all" we have to do now is solve this system of equations. That is what the next section of the course is about.

Exercise 2.11. Write down the Euler Method for the following 3rd-order IVP

$$y''' - y'' + 2y' + 2y = x^2 - 1,$$
$$y(0) = 1, y'(0) = 0, y''(0) = -1.$$

Exercise 2.12. Use a Taylor series to provide a derivation for the formula

$$\frac{\partial^2 u}{\partial x^2}(t_i, x_j) \approx \frac{1}{H^2}\big(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\big).$$

Exercise 2.13. $\star$ (Your own RK3 method). Here are some entries for 3-stage Runge-Kutta method tableaux.

Method 0: $\alpha_2 = 2/3$, $\alpha_3 = 0$, $b_1 = 1/12$, $b_2 = 3/4$, $\beta_{32} = 3/2$

Method 1: $\alpha_2 = 1/4$, $\alpha_3 = 1$, $b_1 = -1/6$, $b_2 = 8/9$, $\beta_{32} = 12/5$

Method 2: $\alpha_2 = 1/4$, $\alpha_3 = 1/2$, $b_1 = 2/3$, $b_2 = -4/3$, $\beta_{32} = 2/5$

Method 3: $\alpha_2 = 1/4$, $\alpha_3 = 1/3$, $b_1 = 3/2$, $b_2 = -8$, $\beta_{32} = 4/45$

Method 4: $\alpha_2 = 1$, $\alpha_3 = 1/4$, $b_1 = -1/6$, $b_2 = 5/18$, $\beta_{32} = 3/16$

Method 5: $\alpha_2 = 1$, $\alpha_3 = 1/5$, $b_1 = -1/3$, $b_2 = 7/24$, $\beta_{32} = 4/25$

Method 6: $\alpha_2 = 1$, $\alpha_3 = 1/6$, $b_1 = -1/2$, $b_2 = 3/10$, $\beta_{32} = 5/36$

Method 7: $\alpha_2 = 1/2$, $\alpha_3 = 1/7$, $b_1 = 7/6$, $b_2 = 22/15$, $\beta_{32} = -10/49$

Method 8: $\alpha_2 = 1/2$, $\alpha_3 = 1/8$, $b_1 = 4/3$, $b_2 = 13/9$, $\beta_{32} = -3/16$

Method 9: $\alpha_2 = 1/3$, $\alpha_3 = 1/9$, $b_1 = 4$, $b_2 = 15/4$, $\beta_{32} = -2/27$

Answer the following questions for Method K, where K is the last digit of your ID number. For example, if your ID number is 01234567, use Method 7.

(i) (a) Assuming that the method is *consistent*, determine the value of b_3.

(b) Consider the initial value problem:

$$y(0) = 1,\ y'(t) = \lambda y(t).$$

Using that the solution is $y(t) = e^{\lambda t}$, write out a Taylor series for $y(t_{i+1})$ about $y(t_i)$ up to terms of order h^4 (use that $h = t_{i+1} - t_i$). Using that your method should agree with the Taylor Series expansion up to terms of order h^3, determine β_{21} and β_{31}.

Exercise 2.14. (Attempt this exercises after completing Lab 3). Write a MATLAB program that implements your method from Exercise 2.13.

Use this program to check the order of convergence of the method. Have it compute the error for $n = 2$, $n = 4$, ..., $n = 1024$. Then produce a log-log plot of the errors as a function of n.

2.8 LAB 3: RK2 and RK3 methods

In this lab, you will extend the code for Euler's method from Lab 2 to implement higher-order methods to solve IVPs of the form

$$y(t_0) = y_0, \quad y'(t) = f(t, y) \text{ for } t > t_0.$$

In particular, you will write programs to implement certain RK2 and RK3 methods.

2.8.1 RK2

A generic one-step method is written as

$$y_{i+1} = y_i + h\Phi(t_i, y_i; h) \text{ for } i = 1, 2, \ldots, n.$$

To get a Runge-Kutta 2 ("**RK2**") method, set

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \alpha h, y_i + \beta h k_1), \\ \Phi(t_i, y_i; h) &= a k_1 + b k_2. \end{aligned}$$

In Section 2.4, we saw that if we pick any $b \neq 0$, and let

$$a = 1 - b, \qquad \alpha = \frac{1}{2b}, \qquad \beta = \alpha, \tag{2.8.1}$$

then we get a second-order method: $|\mathcal{E}_n| \leqslant Kh^2$.

For example, if we choose $b = 1$, we get the so-called *Modified* or *mid-point* Euler Method from Section 2.4. However, *any* value of b, other than $b = 0$ should give a second-order method.

Download the MATLAB script `Euler_Solution.m` and run it. Make sure you understand how it works. Next, adapt its to preform the following tasks.

1. Choose the value of b to be the last digit of your ID number, unless that is 0, in which case take $b = -1$. (For example, if your ID number is 01234567, take $b = 7$. If your ID number is 76543210, take $b = -1$).
2. Compute the values of a, α and β according to (2.8.1).
3. Choose an initial value problem to solve, and for which you know the exact solution. To avoid having a problem that is too simple,
   - your solution should involve trigonometric, logarithmic or nth-root functions.
   - f should depend explicitly on both t and y.

   (Hint: decide on the solution first, and then differentiate that to get f). You also need to choose an initial time, t_0, and a final time for the simulation, t_n.
4. The MATLAB program should approximate the solution to this IVP using your RK2 method for $n = 2$, $n = 4$, $n = 8$, ..., $n = 512$ (at least). For each n it should output the estimate for $y(t_n)$ and the error $|\mathcal{E}_n| = |y(t_n) - y_n|$.
5. The program should produce a figure displaying a log-log plot of these errors against the corresponding values of n, as well as n^{-2} against n. If your method is second-order, then these two lines should be parallel.

2.8.2 RK3

Next write a program that implements the RK3-1 method given in Section 2.6.3:

$$\begin{array}{c|ccc} \alpha_1 & & & \\ \alpha_2 & \beta_{21} & & \\ \alpha_3 & \beta_{31} & \beta_{32} & \\ \hline & b_1 & b_2 & b_3 \end{array} = \begin{array}{c|ccc} 0 & & & \\ 2/3 & 2/3 & & \\ 2/3 & 1/3 & 1/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array}$$

The method is then

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \alpha_2 h, y_i + \beta_{21} h k_1), \\ k_3 &= f(t_i + \alpha_3 h, y_i + \beta_{31} h k_1 + \beta_{32} h k_2), \end{aligned}$$

and

$$\Phi(t_i, y_i; h) = b_1 k_1 + b_2 k_2 + b_3 k_3.$$

As with the RK2 method, the MATLAB program should approximate the solution to this IVP using this RK3 method for $n = 2$, $n = 4$, $n = 8$, ... For each n it should output the estimate for $y(t_n)$ and the error $|\mathcal{E}_n| = |y(t_n) - y_n|$.

The program should also produce a figure displaying a log-log plot of these errors against the corresponding values of n, as well as n^{-3} against n.

2.8.3 What to upload

When your RK2 and RK3 programs are complete, upload them to the "Lab 3", under the "Assignments and Labs" tab on Blackboard. This can be done in two separate files, but it is okay to combine them to a single file if you prefer. (After all, every RK2 method is an example of an RK3 method).

Add appropriate comments to the top of your file(s) indicating who wrote it, when they wrote it, what it does, and how it does it (*"Who/When/What/How?"*). Include your ID number, and give the program a sensible name, which includes something distinctive like you name and ID number (so that I don't end up with 50 programmes all called `RK2.m`).

Make sure your programmes run as-is before uploading. If you don't, you might have given it an invalid name, such as one containing spaces or mathematical symbols.

The deadline for uploading your code is **TBD**.