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.

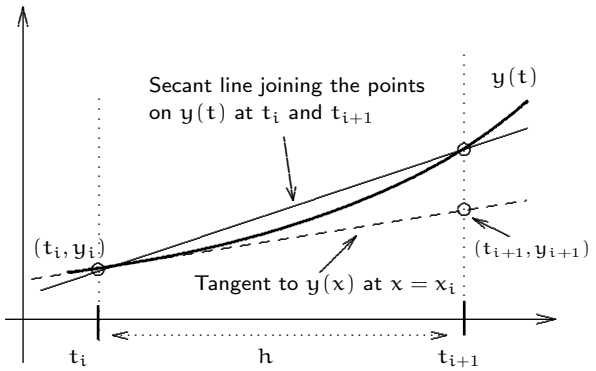
2.2 Euler's method

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.

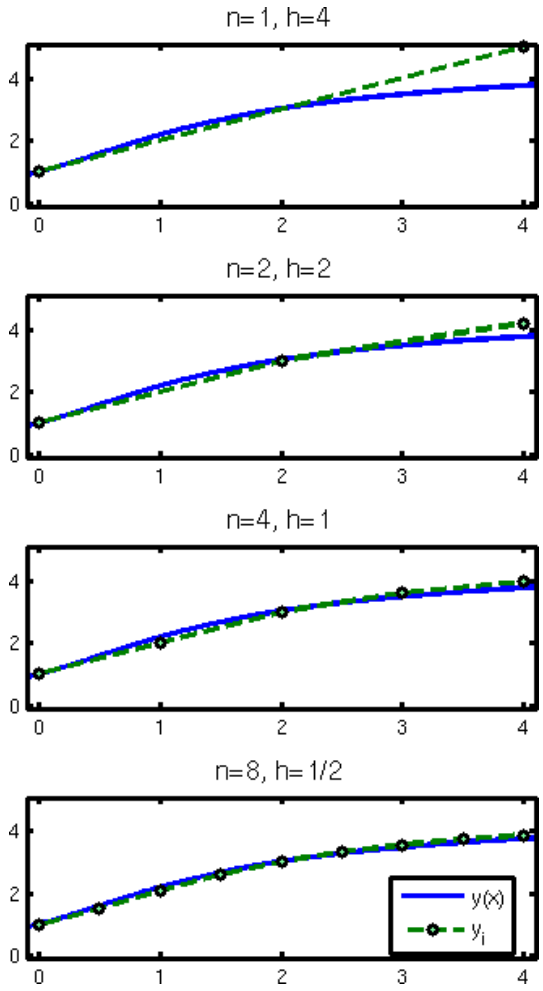


Fig. 2.1: Euler's method for Example 2.2.2 with $h = 4$, $h = 2$, $h = 1$ and $h = 1/2$

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*.

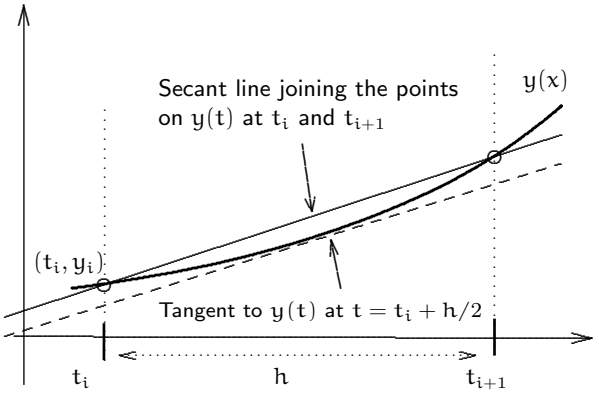
In Euler's method, we use the slope of the tangent to y at t_i as an approximation for the slope of the secant line joining the points $(t_i, y(t_i))$ and $(t_{i+1}, y(t_{i+1}))$.

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.

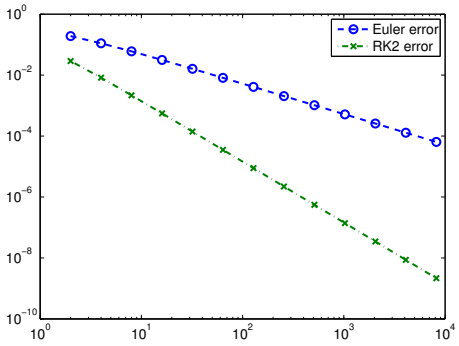


Fig. 2.2: Log-log plot of the errors when Euler's and Modified Euler's methods are used to solve Example 2.4.1

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

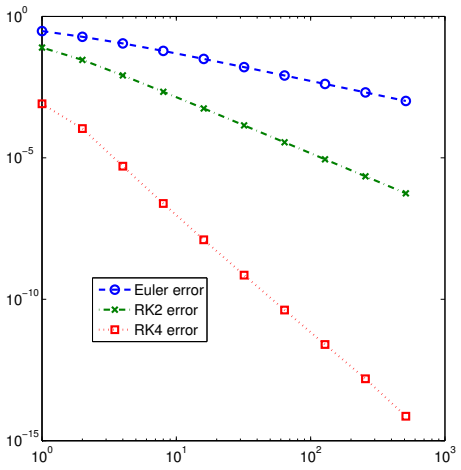


Fig. 2.3: Log-log plot of the errors when Euler's, Modified Euler's, and RK-4 methods are used to solve Example 2.4.1

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).

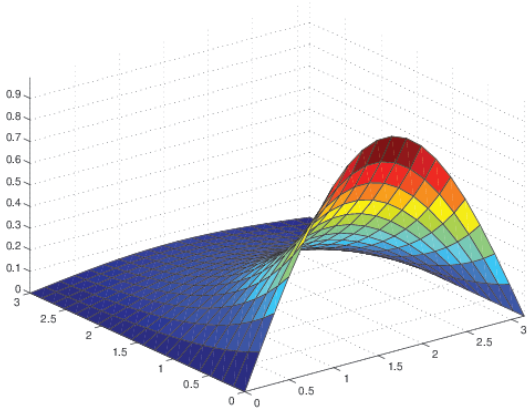


Fig. 2.4: The true solution to the heat equation

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.

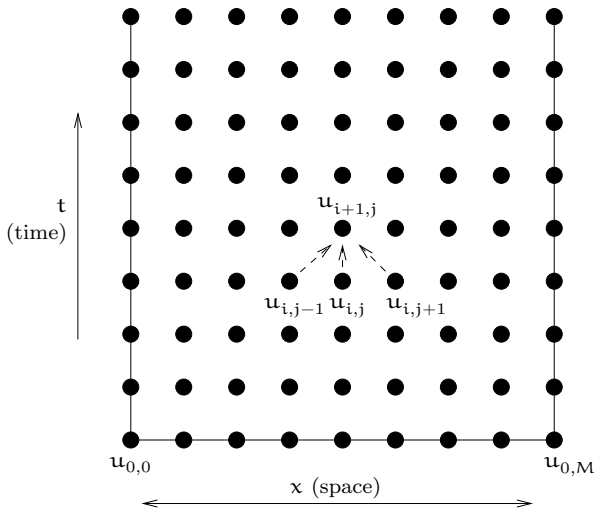


Fig. 2.5: A finite difference grid: if we know $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ we can calculate $u_{i+1,j}$.

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**.