Chapter 4

Ordinary Differential Equations

Chapter 4

4 Ordinary Differential Equations **1**
4.9 Ordinary Differential Equations . . . 3
4.9.1 One Step Methods . . . 3
4.9.2 A-Stability of ODE methods . . . 13
4.9.3 Stepsize Control for ODE methods . . . 23
4.9.4 Numerical Solution of ODE's using Multistep Methods . . . 25

4.9 Ordinary Differential Equations

4.9.1 One Step Methods

Overview

In this chapter we will study how to solve an ODE numerically using what are called one step methods. Along the way we will introduce concepts of stability and accuracy, and derive a number of standard methods. The problem of error approximation will be studied.

There are essentially three classes of numerical methods that we will consider:

1. Simple methods which are easy to understand and analyse. Forward and backward Euler methods and low order Runge-Kutta methods fit into this class.
2. Direct extensions of the 1st class providing greater efficiency and accuracy, such as higher order Runge-Kutta methods.
3. Combinations of the methods in class (2), where issues of error control, starting procedures, output control and other software problems are addressed.

General Notation

Initial value problem for ODE: Find a differentiable function $x : [a, b] \to \mathbb{R}$ which satisfies

$$\begin{aligned} \frac{dx(t)}{dt} &= f(t, x(t)) \\ x(a) &= x_a \end{aligned}$$

where often $[a, b] = [0, 1]$.

We will first consider some simple numerical schemes and investigate the accuracy of these schemes. The integral form of the ordinary differential equation

$$x(b) = x(a) + \int_a^b f(s, x(s))\, ds \quad \text{for } a \le t \le b$$

is a useful starting point to develop numerical methods.

For the discretisation (numerical approximation) we first choose grid points $t_k = a + kh$ where $h = (b - a)/n$ and we denote the approximations of the solution at the grid points by $x_k \approx x(t_k)$.

Euler's Method

Numerical Scheme

Let us look at the most simple numerical method. We use the equation

$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(s, x(s))ds$$

that covers one section of the interval under consideration, $[a, b]$.

We need to approximate the integral term, and an obvious choice is

$$\int_{t_k}^{t_{k+1}} f(s, x(s))ds \simeq (t_{k+1} - t_k)f(t_k, x(t_k))$$

which leads to the numerical scheme, a difference equation

$$\begin{aligned} x_{k+1} &= x_k + (t_{k+1} - t_k)f(t_k, x_k) \\ &= x_k + hf(t_k, x_k). \end{aligned}$$

This is an *explicit* formula for x_{n+1} since all quantities on the right hand side are known from the previous step, and it is a *one step method* as the numerical scheme requires only the values at one previous step for calculation.

The notation: x_{k+1} is an estimate of $x(t_{k+1})$.

Discretisation Errors

Of course we cannot expect the actual solution of the differential equation to satisfy this difference equation exactly, even if no computational errors were incurred. An important question is: How well does x_k approximate $x(t_k)$?

As a first step we ask: How well does the exact solution satisfy the difference equation? In particular we consider the quantity

$$L(t, h) = \frac{x(t + h) - x(t)}{h} - f(t, x(t))$$

which is 0 if the estimate is exact.

This is known as the *local discretisation error* at time t and is a measure of how much the difference quotient $x'(t)$ differs from the actual value of $f(t, x)$ (we ignore for the moment errors due to operations).

Via a simple Taylor's series expansion, and the fact that $x(t)$ satisfies the differential equation, we see that

$$\begin{aligned} L(t, h) &= \frac{1}{h}\left[x(t) + \frac{dx(t)}{dt}h + O(h^2) - x(t)\right] - f(t, x(t)) \\ &= \frac{dx(t)}{dt} - f(t, x(t)) + O(h) \\ &= O(h). \end{aligned}$$

Clearly it is of first order. Now in terms of the local discretisation error we have that

$$x(t_{k+1}) = x(t_k) + hf(t_k, x(t_k)) + hL(t_k, h).$$

Hence the error $x_k - x(t_k)$ satisfies the relation

$$x_{k+1} - x(t_{k+1}) = x_k - x(t_k) + h\left(f(t_k, x_k) - f(t_k, x(t_k))\right) - hL(t_k, h). \tag{4.1}$$

We assume that f satisfies a Lipschitz condition with respect to the x variable. In particular, we assume there exists an M such that

$$|f(t,x) - f(t,y)| \leq M\,|x-y|\,.$$

Taking absolute values in Equation (4.1) we have

$$e_{k+1} \leq (1+hM)\,e_k + hL_k \tag{4.2}$$

where $e_k = |x_k - x(t_k)|$ and $L_k = |L(t_k,h)|$. Essentially for each time step we pickup an extra error of size hL_k while the previous error is amplified by a factor of $1+hM$.

We want to estimate the error e_n at time $T = nh$, ie. the accumulated error. To simplify the analysis we use the result that

Lemma 1. *If a sequence of values d_j satisfies*

$$d_{j+1} \leq Cd_j + D, \quad \text{for } j = 0,1,2,...$$

then

$$d_n \leq C^n d_0 + D\frac{C^n - 1}{C-1} \text{ for } n = 1,2,...$$

Now let $C = 1+hM$ and $D = h\max_j L_j$ and $e_j = d_j$. Then by the lemma we have that

$$\begin{aligned}
e_n &\leq (1+hM)^n\,e_0 + h\max_j L_j\frac{(1+hM)^n - 1}{1+hM-1} \\
&= (1+hM)^{TM/hM}\,e_0 + \max_j L_j\frac{\left[(1+hM)^{TM/hM} - 1\right]}{M} \\
&\leq \exp(TM)\,e_0 + \max_j L_j\frac{[\exp(TM) - 1]}{M}.
\end{aligned}$$

So provided $e_0 = O(h)$ (in fact it is often 0) and $\max_j L_j = O(h)$, it follows that $e_n = O(h)$, ie. Euler's Method is first order. This means that as h is decreased, so the approximation increases in accuracy, or if h is halved then the error decreases by a factor of 2. However, the number of computations increases and hence the error resulting from operations will increase.

Thus for Euler's method the local errors ($x_k - x(t_k)$ of second order) accumulate over a number of steps to produce *global error* of first order, as seen from above.

One Step Methods - General Theory

General Form of One Step Methods

A closer look at the previous proof shows that we can apply our analysis to any numerical method of the form

$$x_{k+1} = x_k + h\phi(t_k,x_k).$$

Such a method is called a one-step method, since the new value at time t_{k+1} depends only on the value of the solution at time t_k.

Stability

We say that the numerical method is *stable* if the function ϕ satisfies a Lipschitz condition with respect to the x variable.

We will encounter many other different types of stability before we are through. In this case the stability implies that a small change in initial conditions is amplified by at most an exponent term.

Consistency

The local discretisation error at t is defined as

$$L(t,h) = \frac{x(t+h)-x(t)}{h} - \phi(t,x(t)).$$

We say our method is *consistent* if

$$\lim_{h\to 0^+} \sup_{0\le t\le T} L(t,h) = 0$$

and that the method is *order* p if

$$L(t,h) = O(h^p) \text{ as } h \to 0$$

uniformly in t.

Stability and Consistency

Stability and Consistency

For a one step numerical method to converge to the solution of the associated differential equation, we need the scheme to be both consistent and stable.

Convergence

The convergence proof follows, as in the case for Euler's method:

$$x(t_{k+1}) = x(t_k) + h\phi(t_k, x(t_k)) + hL(t_k,h)$$

and so the error $x_k - x(t_k)$ satisfies the relation

$$x_{k+1} - x(t_{k+1}) = x_k - x(t_k) + h\left(\phi(t_k,x_k) - \phi(t_k,x(t_k))\right) - hL(t_k,h). \tag{4.3}$$

As ϕ satisfies a Lipschitz condition with respect to the x variable there exists an M such that

$$|\phi(t,x) - \phi(t,y)| \le M\,|x-y|\,.$$

Taking absolute values in Equation (4.3) we have

$$e_{k+1} \le (1+hM)\,e_k + hL_k \tag{4.4}$$

where again $e_k = |x_k - x(t_k)|$ and $L_k = |L(t_k,h)|$. Hence, as before, we conclude that

$$e_n \le \exp(TM)\,e_0 + \max_j L_j \frac{[\exp(TM)-1]}{M}.$$

So if the local discretisation error is $O(h^p)$ and the initial error is $O(h^p)$ then the global error is also $O(h^p)$. Such a numerical method is of order p.

Typically first order methods have slow convergence rates and thus more efficient methods are sought.

Runge-Kutta Methods

Trapezoidal Rule

We can ask the question "Are there any higher order one step methods?" The answer is yes. For the differential equation we have that

$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(s,x(s))ds.$$

A simple trapezoidal approximation of the integral term leads to the scheme

$$x_{k+1} = x_k + \frac{h}{2}\left[f(t_k,x_k) + f(t_{k+1},x_{k+1})\right].$$

This is an *implicit* method. It is called the *trapezoidal rule* (for ODEs). However it does not quite fit into the form of method we are presently considering.

Improved Euler Method

To approximate the value of x_{k+1} in the term $f\left(t_{k+1}, x_{k+1}\right)$ we can use Euler's method. This leads to the method:

$$x_{k+1} = x_k + \frac{h}{2}\left[f(t_k, x_k) + f(t_{k+1}, x_k + hf(t_k, x_k))\right]$$

known as the *Improved Euler method* or *Heun's method.*

It is a second order Runge-Kutta method. It has a local discretisation error of $O(h^2)$.

Runge-Kutta Methods

Definition 1 (Runge-Kutta)**.** *An s-stage explicit Runge-Kutta method is defined as follows. Consider an integer s defining the number of stages and a choice of real coefficients a_{21}, a_{31}, a_{32},..., a_{s1}, a_{s2}, a_{s3},..., $a_{s,s-1}$, b_1,..., b_s, c_1,...,c_s. Then the method*

$$\begin{aligned}
k_1 &= f\left(t_k, x_k\right) \\
k_2 &= f\left(t_k + c_2 h, x_k + h a_{21} k_1\right) \\
k_3 &= f\left(t_k + c_3 h, x_k + h(a_{31} k_1 + a_{32} k_2)\right) \\
&\dots \\
k_s &= f\left(t_k + c_s h, x_k + h(a_{s1} k_1 + \cdots + a_{s,s-1} k_{s-1})\right) \\
x_{k+1} &= x_k + h\left(b_1 k_1 + \cdots + b_s k_s\right)
\end{aligned}$$

is called an s-stage explicit Runge-Kutta method.

Usually $c_i = \sum_j a_{ij}$ and $\sum_i b_i = 1$.

It is straightforward to show that Heun's method is indeed a Runge-Kutta routine.

Fourth Order Runge-Kutta

Perhaps the most well known Runge-Kutta method is the classical fourth order method (*RK4*), given by

$$\begin{aligned}
k_1 &= f(t_k, x_k) \\
k_2 &= f(t_k + \frac{h}{2}, x_k + \frac{h}{2} k_1) \\
k_3 &= f(t_k + \frac{h}{2}, x_k + \frac{h}{2} k_2) \\
k_4 &= f(t_{k+1}, x_k + h k_3) \\
x_{k+1} &= x_k + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right].
\end{aligned}$$

Note that if $f(t, x)$ is just a function of t then the method reduces to *Simpson's method* for calculating integrals.

Runge-Kutta Derivation

Let us see how to derive such a scheme.

Consider a scheme of the form

$$x_{k+1} = x_k + h\left[a f(t_k, x_k) + b f(t_k + ch, x_k + dh f(t_k, x_k))\right]$$

of which Heun's method is an example.

We need to determine the parameters a, b, c, d to produce a scheme with as high as possible order of accuracy as measured by the local discretisation error.

This will entail expanding the terms of the local discretisation error via Taylor series about the point t and using the fact that $x(t)$ satisfies the differential equation.

$$L(t,h) = \frac{x(t+h)-x(t)}{h} - \left[af(t,x(t)) + bf(t+ch, x(t)+dhf(t,x(t)))\right].$$

Taking the first term we have

$$\begin{aligned}\frac{x(t+h)-x(t)}{h} &= \frac{dx(t)}{dt} + \frac{h}{2}\frac{d^2x(t)}{dt^2} + O(h^2)\\ &= f(t,x(t)) + \frac{h}{2}\left[f_t(t,x(t)) + f_x(t,x(t))f(t,x(t))\right]\\ &\quad + O(h^2).\end{aligned}$$

The second term is more complicated, but gives

$$\begin{aligned}& af(t,x(t)) + bf(t+ch, x(t)+dhf(t,x(t)))\\ =\ & af(t,x(t)) +\\ & +b\left[f(t,x(t)) + chf_t(t,x(t)) + dhf_x(t,x(t))f(t,x(t))\right] + O(h^2)\\ =\ & (a+b)f(t,x(t)) +\\ & +h\left[bcf_t(t,x(t)) + bdf_x(t,x(t))f(t,x(t))\right] + O(h^2).\end{aligned}$$

To obtain the above approximation we have used the following rule from several variable calculus

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt}.$$

Hence

$$\begin{aligned}L(t,h) &= (1-(a+b))f(t,x(t)) +\\ &\quad +h\left[(\frac{1}{2}-bc)f_t(t,x(t)) + (\frac{1}{2}-bd)f_x(t,x(t))f(t,x(t))\right]\\ &\quad + O(h^2).\end{aligned}$$

For a second order method we need to set

$$a+b=1, \quad bc = \frac{1}{2}, \quad bd = \frac{1}{2}.$$

Heun's method corresponds to

$$a = b = \frac{1}{2}, \quad c = d = 1.$$

In general, the scheme must be of the form

$$x_{k+1} = x_k + h\left[\left(1-\frac{1}{\lambda}\right)f(t_k,x_k) + \frac{1}{\lambda}f(t_k + \frac{\lambda}{2}h, x_k + \frac{\lambda h}{2}f(t_k,x_k))\right]$$

(letting $b = 1/\lambda$).

Typically we specify the type of method in a general form and then use simple Taylor series expansions to obtain a system of equations for a set of parameters to ensure a scheme with high accuracy. Other criteria can also be applied such as minimisation of the next highest term, or constraint on the numerical stability (to be discussed later).

It emerges that more accuracy means more work, or more operations and, as was discussed earlier, more operations incur a cost in terms of accumulating roundoff errors. The choice becomes a tradeoff between work required and accuracy desired.

These methods can be extended to higher order equations using their representation as systems of first order equations. These will be dealt with in later chapters.

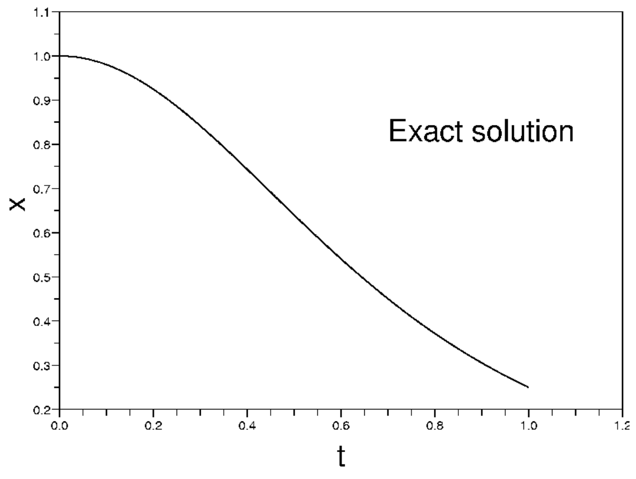

Figure 4.1: Exact solution of Example 1.

Comparison of 4 Methods

Model Problem

Example 1 (Model Problem)**.** *Consider the IVP*

$$\frac{dx}{dt} = -4t(1+t^2)x^2; \quad x(0) = 1$$

Separation of variables leads to the exact solution $x(t) = 1/(t^2+1)^2$. We can use this to test the accuracy of various numerical methods.

In particular we will look at:

- Euler method:
$$x_{k+1} = x_k + hf(t_k, x_k).$$
- (Explicit) Midpoint method:
$$x_{k+1} = x_k + hf\left(t_k + \frac{h}{2}, x_k + \frac{h}{2}f(t_k, x_k)\right).$$
- Heun's method:
$$x_{k+1} = x_k + \frac{h}{2}\left(f(t_k, x_k) + f(t_k + h, x_k + hf(t_k, x_k))\right).$$
- 4th order Runge-Kutta:
$$x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Exact Solution

The exact solution is shown in Figure 4.1.

Euler Method

The solution using Euler's method is given in Figure 4.2.

Midpoint Method

The solution using the midpoint method is given in Figure 4.3.

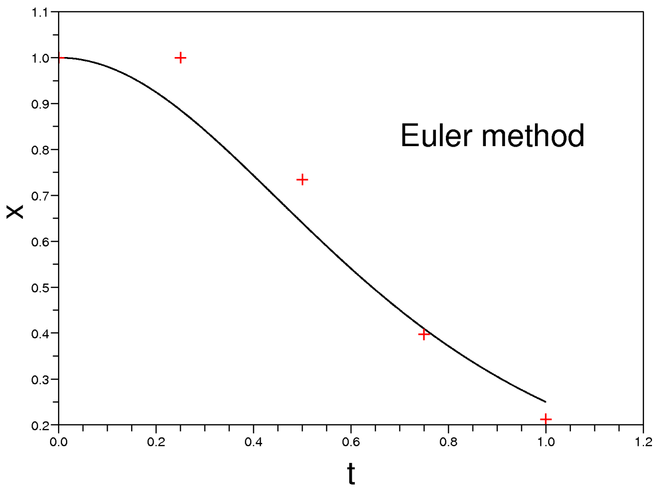

Figure 4.2: Solution of Example 1 using Euler's method.

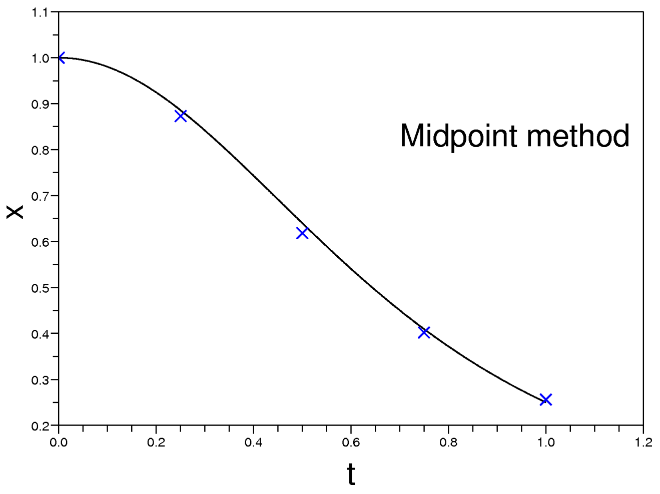

Figure 4.3: Solution of Example 1 using the midpoint method.

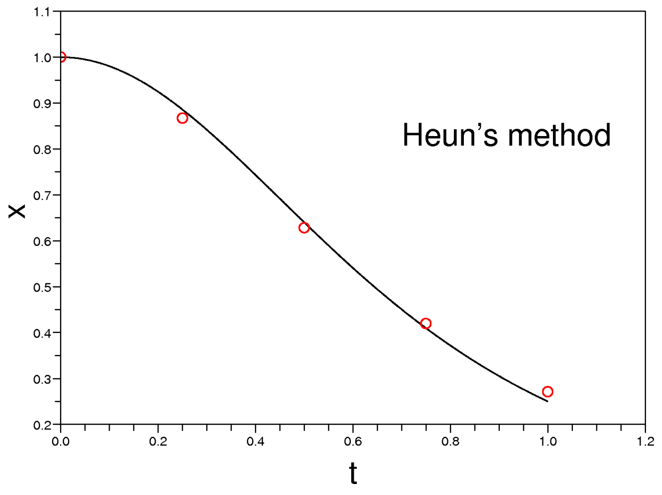

Figure 4.4: Solution of Example 1 using the Heun's method.

Derivation of the (Explicit) Midpoint Rule

We have from the central difference formula,

$$x'(t_k) \approx \frac{x(t_{k+1}) - x(t_{k-1})}{2h},$$

or

$$x'\left(t_k + \frac{h}{2}\right) \approx \frac{x(t_{k+1}) - x(t_k)}{h}.$$

Rewrite in the ODE framework to get the following iterative method

$$x(t_{k+1}) = x(t_k) + hf\left(t_k + \frac{h}{2}, x\left(t_k + \frac{h}{2}\right)\right).$$

As with Heun's method, we use Euler's method to approximate $x\left(t_k + frach/2\right)$. In which case

$$x(t_{k+1}) = x(t_k) + hf\left(t_k + \frac{h}{2}, x(t_k) + \frac{h}{2}f(t_k, x_k)\right).$$

Heun's Method

The solution using Heun's method is given in Figure 4.4.

Runge-Kutta Method

The solution using fourth order Runge-Kutta method is given in Figure 4.5.

Euler v's Runge-Kutta Method

The solution using both the fourth order Runge-Kutta method and Euler's method is given in Figure 4.6. Observe that roughly 4 times more steps are needed by Euler's method to get the same accuracy as the Runge-Kutta method.

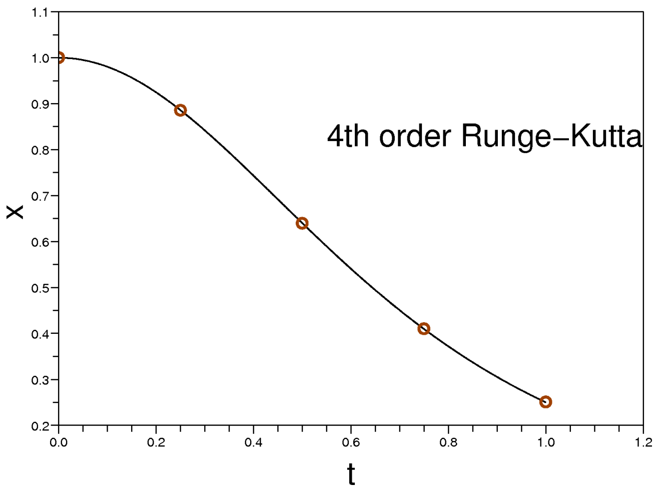

Figure 4.5: Solution of Example 1 using fourth order Runge-Kutta.

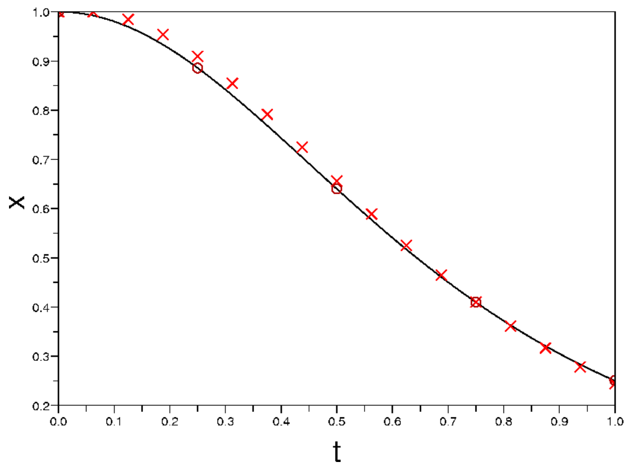

Figure 4.6: Solution of Example 1 using Euler's method (crosses) and fourth order Runge-Kutta (circles).

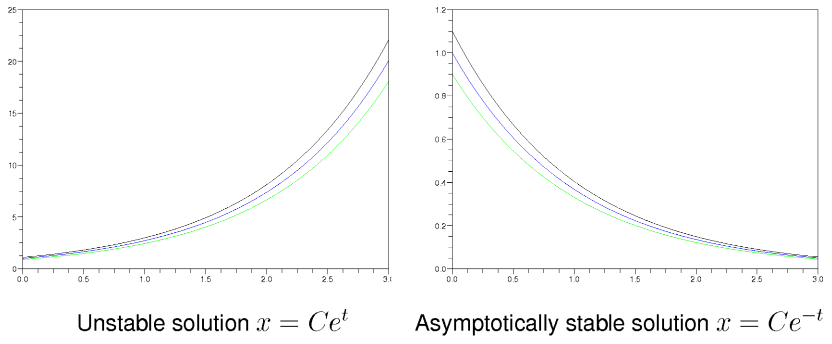

Figure 4.7: Example of a stable and unstable solution

4.9.2 A-Stability of ODE methods

Stability Revisited

In the previous section we showed that if the method was consistent and stable (in the sense that the scheme was Lipschitz) then the method will converge to the solution of the differential equation as $h \to 0$.

This seems to answer all our questions. Alas the result is an asymptotic result, so we do not know how small h needs to be before the behaviour predicted by the convergence analysis will prevail. Perhaps we will start to get predicted behaviour after $h = 1/10$, perhaps only after $h = 1/10^{16}$.

A solution to an ODE is *stable* if small perturbations in the initial data remain bounded with time. A solution to an ODE is *unstable* if small perturbations in the initial data grow without bound. A solution is *asymptotically stable* if perturbations in the initial data are damped to zero at later times. See Figure 4.7.

Model Problem

To obtain important practical information along these lines we use a somewhat heuristic approach. We study our methods on the very simple ODE

$$\frac{dx}{dt} = \lambda x$$

where λ is a complex number.

Perturbation Analysis

Suppose we are interested in perturbations of the solutions of the equation

$$\frac{dx}{dt} = f(t, x)$$

about an exact solution $x_0(t)$. We suppose our perturbed solution has the form

$$x(t) = x_0(t) + \varepsilon x_1(t).$$

Now

$$\frac{dx}{dt} = \frac{dx_0}{dt} + \varepsilon \frac{dx_1}{dt}$$

and, using a Taylor series expansion,

$$f(t, x) = f(t, x_0) + f_x(t, x_0)\varepsilon x_1 + O(\varepsilon^2).$$

Comparing terms in ε we see that

$$\frac{dx_1}{dt} \approx f_x(t, x_0)x_1.$$

So locally, perturbations of the solution of differential equations satisfy ordinary differential equations of the form

$$\frac{dx_1}{dt} = \lambda x_1.$$

Our plan is to apply our numerical methods to this simple ODE

$$\frac{dx}{dt} = \lambda x. \tag{4.5}$$

Observe that we can solve this equation exactly, namely

$$x(t) = e^{t\lambda}x(0),$$

and recall that

$$e^{t\lambda} = 1 + t\lambda + \frac{t^2\lambda^2}{2} + \dots.$$

In the case of systems of equations, the perturbation equations have the form

$$\frac{d\mathbf{x}_1}{dt} = A\mathbf{x}_1.$$

Even if A is a real matrix, the eigenvalues of A may be complex, and so a study of these system equations leads us to study problems with complex parameters.

Euler's Method

A-Stability and Explicit Euler

The explicit Euler method applied to Equation (4.5) yields the numerical method

$$\begin{aligned} x_{k+1} &= x_k + h\lambda x_k \\ &= (1 + h\lambda)x_k. \end{aligned}$$

Note that $1 + h\lambda$ is a second order approximation to $e^{t\lambda}$. This is consistent with the Euler method being a globally first order method.

Now for $\mathrm{Re}(\lambda) < 0$ the exact solution $e^{t\lambda}x(0)$ converges to 0. We would like to find the values of h that force our numerical scheme to demonstrate the same behaviour. In particular we would like to find constraints on h so that

$$|x_{k+1}| < |x_k|.$$

For this to hold we will need the amplification factor

$$\rho(h\lambda) = 1 + h\lambda$$

to satisfy

$$|\rho(h\lambda)| < 1$$

which is just

$$|1 + h\lambda| < 1.$$

This specifies the region in the $h\lambda$ complex plane, which is inside the circle, centred at -1 of radius 1. We call this region (where the amplification factor is less than one) the region of *A-Stability* for the numerical method.

If λ is real and negative, then we must choose h so that $|h\lambda| \leq 2$. That is, h must be chosen so that

$$h < \frac{2}{|\lambda|}.$$

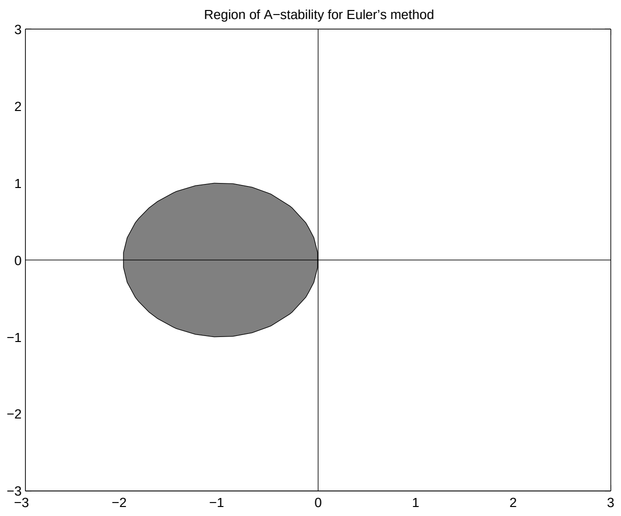

Figure 4.8: Region of A-stability for the Euler method. Region is all values $h\lambda$ inside the shaded circle.

If $|\lambda|$ is very large, then we would not expect our numerical solutions to show any similarity to our exact solution until h is chosen very small. There are many practical situations in which the corresponding λ satisfies $\text{Re}(\lambda) << -1$.

Even more importantly, there are numerous problems that contain many different time scales (range of λ's with widely differing values). For these problems, the A-Stability of Euler's method is determined by the smallest timescale, even if the exact solution has no components at that scale. Such problems are called stiff, and it is usually necessary to use implicit methods to solve these problems. Below we will see how an implicit method deals with the Equation 4.5.

Example

Example 2 (Stability Example)**.** *Solve*

$$\frac{dx}{dt} = -2x; \quad x(0) = 10$$

using the explicit Euler method with the given step size h.

Figure 4.9 shows the solution if $h = 1.1$.
Figure 4.10 shows the solution if $h = 1.0$.
Figure 4.11 shows the solution if $h = 0.75$.
Figure 4.12 shows the solution if $h = 0.50$.
Figure 4.13 shows the solution if $h = 0.25$.

The Implicit Euler Method

A-Stability and Implicit Euler

The implicit Euler method applied to Equation (4.5) yields the numerical method

$$x_{k+1} = x_k + h\lambda x_{k+1}.$$

Rearranging, leads to

$$x_{k+1} = \frac{1}{1-h\lambda}x_k.$$

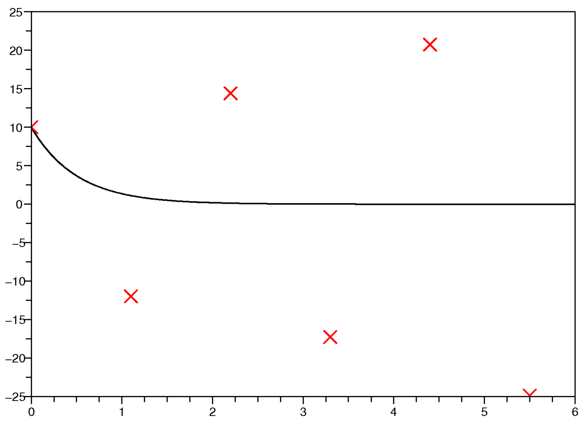

Figure 4.9: Solution of Example 2 using the explicit Euler method with $h = 1.1$.

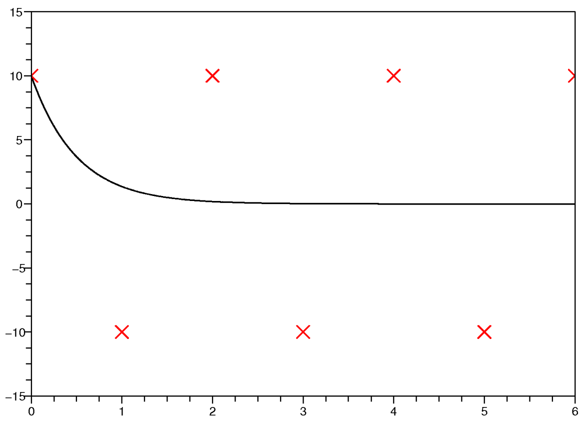

Figure 4.10: Solution of Example 2 using the explicit Euler method with $h = 1.0$.

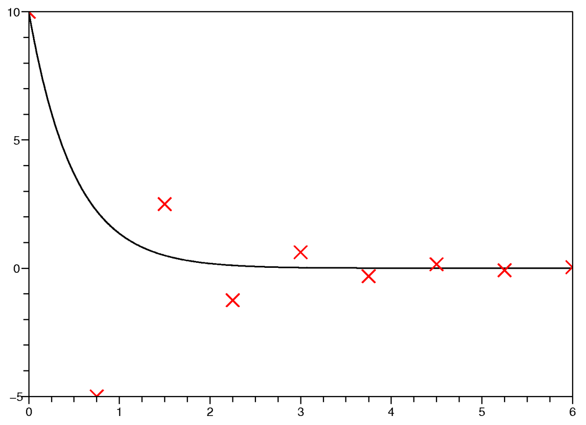

Figure 4.11: Solution of Example 2 using the explicit Euler method with $h = 0.75$.

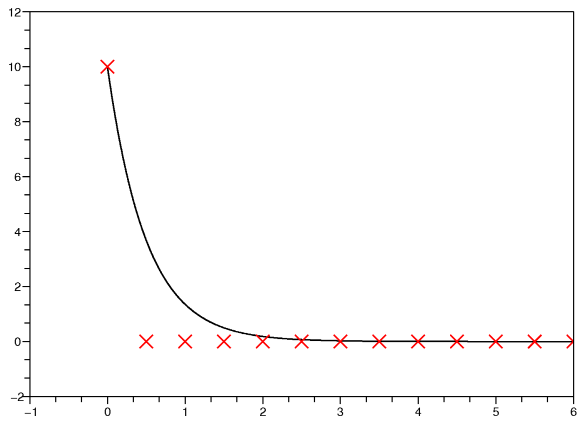

Figure 4.12: Solution of Example 2 using the explicit Euler method with $h = 0.50$.

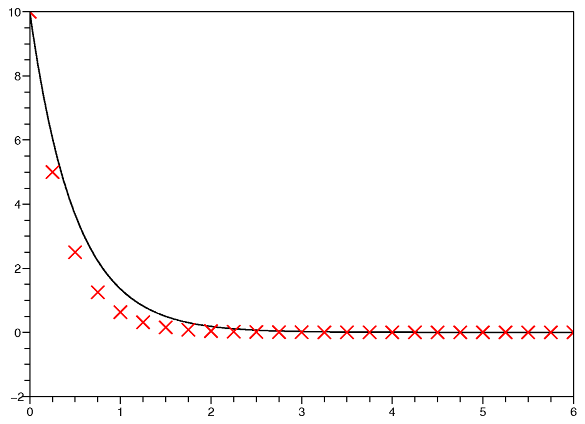

Figure 4.13: Solution of Example 2 using the explicit Euler method with $h = 0.25$.

The amplification factor for the implicit Euler's method is

$$\rho(h\lambda) = \frac{1}{1-h\lambda}.$$

To satisfy

$$|\rho(h\lambda)| < 1$$

we must have

$$1 < |1-h\lambda|\,.$$

This specifies the region in the $h\lambda$ complex plane, which is *outside* the circle, centred at 1 of radius 1. We see that this region of A-stability contains all of the negative half plane. We say that this scheme is *unconditionally stable*.

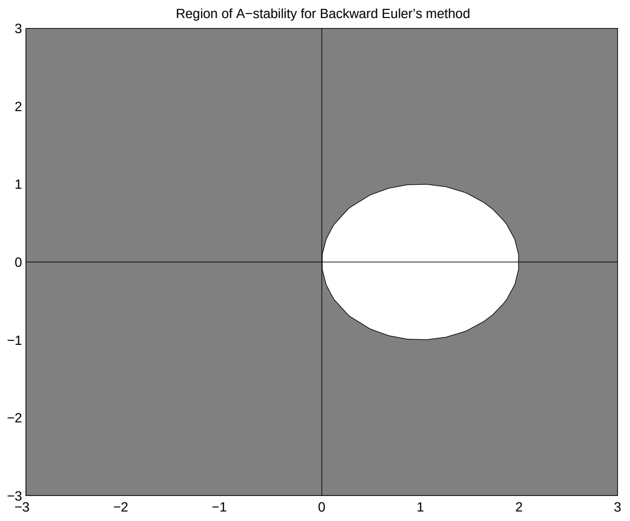

Figure 4.14: Region of A-stability for the Backward Euler method. Region is all values $h\lambda$ outside the shaded circle.

Now at least this method mimics the damping behaviour of the ODE for $\text{Re}(\lambda) \le 0$. In fact the numerical scheme is "more" stable than the ODE, as many solutions of the numerical scheme with $\text{Re}(\lambda) > 0$ will also converge to 0 (unlike the exact solutions).

An examination of the amplification factor relative to the exponential verifies that

$$\frac{1}{1-h\lambda} = 1 + h\lambda + h^2\lambda^2 + \cdots \text{ for small } |h\lambda|$$

is a good approximation of $\exp(h\lambda)$ for large (negative) $h\lambda$.

Example

Figure 4.15 shows the solution of Example 2 using the implicit Euler method with $h = 1.1$. Compare with Figure 4.9.

Classical 4th Order Runge-Kutta Method

A-Stability and 4th Order Runge-Kutta Method

Let us now consider the classical 4th order Runge-Kutta method. Applied to our simple ODE we see that the method reduces to

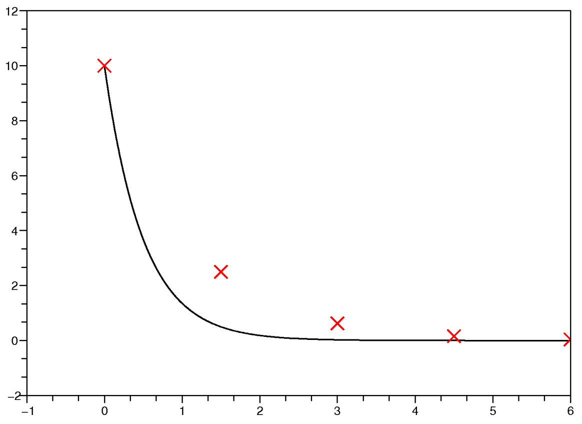

Figure 4.15: Solution of Example 2 using the implicit Euler method with $h = 1.1$.

$$\begin{aligned} x_{k+1} &= x_k + \frac{h\lambda}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= x_k + \left[h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{6} + \frac{h^4\lambda^4}{24}\right] x_k \\ &= \left[1 + h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{6} + \frac{h^4\lambda^4}{24}\right] x_k. \end{aligned}$$

So the amplification error is given by

$$\rho(h\lambda) = \left[1 + h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{6} + \frac{h^4\lambda^4}{24}\right].$$

Notice that as expected the amplification factor is an approximation of the exponential function. In Figure 49 we have plotted the contour line $|\rho(w)| = 1$ to show the somewhat complicated region of A-stability for this method.

Problems that Require Small Stepsizes

Stiff Problems

Although we can develop a general form of a numerical method there are classes of problems which seriously defy traditional methods for their solutions. One such class is that of *stiff equation* problems.

Stiff differential equations are those that are ill-conditioned (unstable) in a computational sense. It is an equation with an amplification factor such that stability and/or the error bound can only be ensured with unreasonable restrictions on the step size h (ie. excessively small).

In systems of equations, stiff problems are typically those for which the ratio of the magnitudes between the largest and the smallest eigenvalues, respectively, is large. This however is not always the case, and although often the stiffness can be related to the eigenvalues of the Jacobian matrix, definitions based on this can be restrictive.

An informal definition is given:

Definition 2. *A system of differential equations is stiff on some interval $[a, b]$ if there exists no positive constant M such that the variation of every component of every solution to every equation is bounded above by M.*

Alternatively we could use

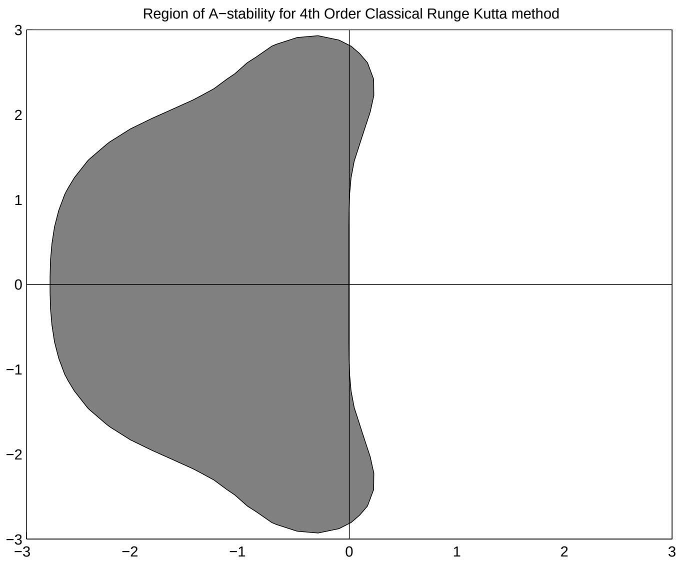

Figure 4.16: Region of A-stability for the Classical 4th order Runge Kutta method. The method is stable for those $h\lambda$ inside the shaded region.

Definition 3. *An ODE system $y' = f(x,y)$ defined on $[a,b]$ is said to be stiff in a neighbourhood of a solution y, if there exists a component of y whose variation is large compared with $(b-a)^{-1}$.*

Many applications have solutions that have components with widely differing decay rates. For example a solution may look like

$$y(x) = e^{-x} + e^{-1000x}, \qquad x \geq 0,$$

with the second component corresponding to a *much* faster decay rate (time scale) than the first. For x away from 0 the solution behaves as e^{-x} and large stepsizes could be taken for high accuracy. However, the method may be restricted to a very small timestep, and very limited region of absolute stability because of the presence of the fast time scale in the equation.

In terms of the test equation $y' = \lambda y$ with $x > 0$ the restriction of $|1 - \lambda h| \leq 1$ requires extremely small steps h to approximate a smooth curve. Severe restrictions apply not only to Euler's method, but to each one of the other more useful methods as well. Thus there is a need to seek out methods that do not have such severe absolute stability limitations.

If we do not restrict the time step to satisfy this stability constraint, then you will see numerical oscillations in your solution. For instance if we solve

$$\frac{dy}{dx} = -15y \tag{4.6}$$

using Euler's method, then we will need $\Delta x < 0.1$ for stability. Actually we need $\Delta x < 0.05$ to avoid oscillations. Figure 4.9.2 shows the typical behaviour of Euler's method for Δx ranging from greater than the stability criteria, through to $\Delta x < \frac{1}{|\lambda|}$.

As discussed above, a method is called *A-stable* if, for the test problem with Re$(\lambda < 0)$, the whole left half of the plane is within the region of absolute stability. It turns out that all the explicit numerical schemes tend to have limited regions of absolute stability, while many implicit schemes are A-stable. The downside of this is that the implicit schemes require the solution of a nonlinear problem at each step and may require much extra work. (No wonder implicit schemes to solve stiff problems remain a very active area of research!)

Currently, the most well used schemes for such problems are the backward differentiation formulae (BDF schemes), which are linear multistep schemes with $\beta_0 = \beta_1 = ... = \beta_{k-1} = 0$ and

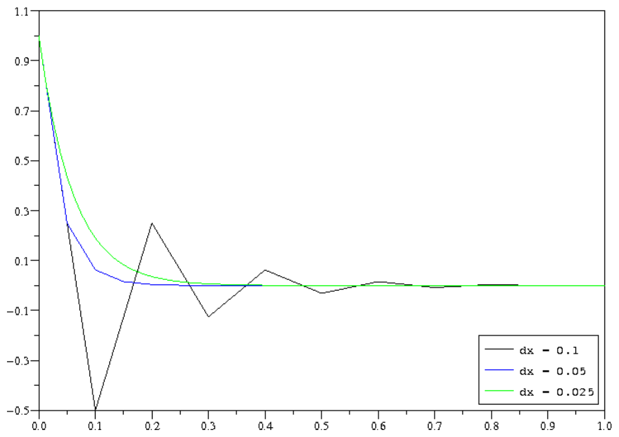

Figure 4.17: Results of Euler's method applied to Equation (4.6) for a range of Δx.

β_k and α_i determined such that the scheme is of order $k \leq 6$, where the β's and α's are as in the general numerical scheme definitions .

For $k = 1$ we get the backward Euler scheme, simply

$$x_{i+1} = x_i + hf_{i+1}.$$

For $k > 1$ the schemes are not quite A-stable, but are A-stable with sectors removed. Methods with an angle of $\frac{\pi}{2} - \alpha$ along the negative and positive imaginary axis removed, are called *A(α)-stable*.

Stiff Example

Example of a Stiff System

Example 3 (Stiff System)**.** *Let's consider the system of first order ordinary differential equations.*

$$\frac{dx}{dt} = -1001\,x - 999\,y \qquad x(0) = x_0$$
$$\frac{dy}{dt} = -999\,x - 1001\,y \qquad y(0) = y_0$$

This can be written as

$$\frac{d\boldsymbol{x}}{dt} = A\boldsymbol{x} = \begin{bmatrix} -1001 & -999 \\ -999 & -1001 \end{bmatrix} \boldsymbol{x}.$$

The eigenvalues of A: $\lambda_1 = -2000$ and $\lambda_2 = -2$.

For Euler's method this puts the constraint on h, $h < \frac{2}{|2000|}$ and $h < \frac{2}{|2|}$.

A problem with a wide range of time scales is called a *stiff* problem. We will need very small step size to avoid oscillations.

Consider

$$\frac{d}{dt}\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} -1001 & -999 \\ -999 & -1001 \end{bmatrix}\begin{bmatrix} X \\ Y \end{bmatrix}$$

Solution

$$x(t) = \frac{1}{2}x_0\,e^{-2000t} + \frac{1}{2}x_0\,e^{-2t} - \frac{1}{2}y_0\,e^{-2t} + \frac{1}{2}y_0\,e^{-2000t}$$
$$y(t) = -\frac{1}{2}x_0\,e^{-2t} + \frac{1}{2}x_0\,e^{-2000t} + \frac{1}{2}y_0\,e^{-2000t} + \frac{1}{2}y_0\,e^{-2t}$$

A phase plane of the example is given in Figure 4.18.

A plot of $x(t)$ versus time is shown in Figure 4.19. Solutions (very) quickly converge to the

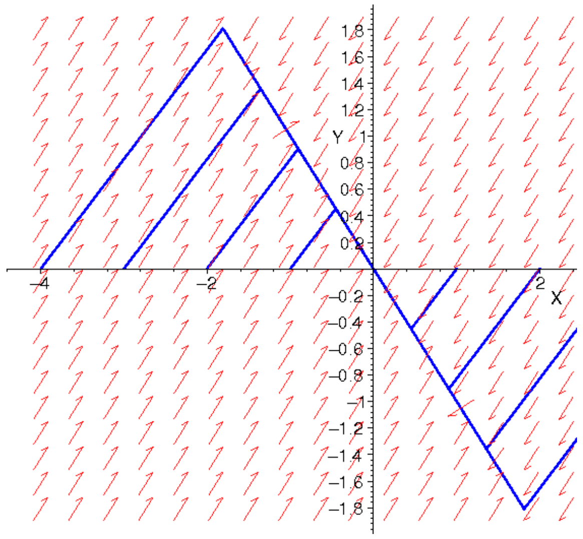

Figure 4.18: Phase plane of example stiff problem.

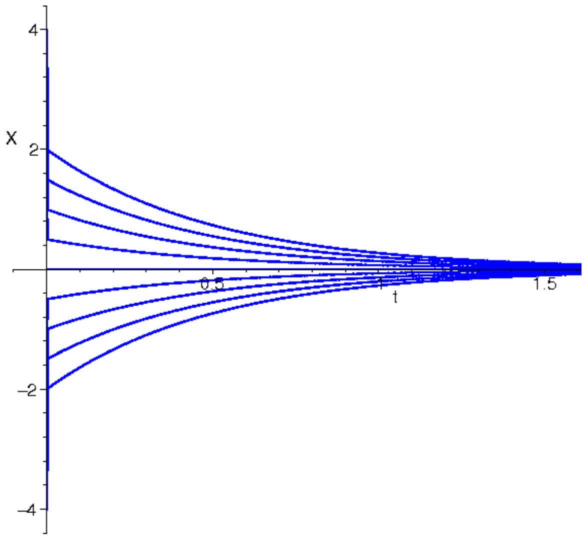

Figure 4.19: Plot of $x(t)$ versus time.

slow scale solution

$$x(t) = c_1 e^{-2t}$$
$$y(t) = -c_1 e^{-2t}$$

4.9.3 Stepsize Control for ODE methods

Introduction

Here we will discuss a way to choose the stepsize h of the numerical solvers for initial value problems. This choice is based on error estimators.

Error Control

Consider a difference scheme where the step size h_k changes over time such that

$$x_{k+1} = x_k + h_k \phi(t_k, x_k).$$

Ideally, we would like to choose the h_k as large as possible while maintaining a global error tolerance ε, i.e., $|x_k - x(t_k)| < \varepsilon$.

But we do not know the global error so we are unable to influence it directly. Instead we will control the *local discretisation error*. First we will derive an estimate for the local discretisation error $L(t_k, h_k)$.

Local Discretisation Error Estimation

Let our basic one-step solver be defined as

$$\begin{aligned} x_0 &= \alpha \\ x_{k+1} &= x_k + h_k \phi(t_k, x_k). \end{aligned}$$

The local discretisation error is then defined as

$$L(t_k, h_k) = \frac{x(t_{k+1}) - x(t_k)}{h_k} - \phi(t_k, x(t_k)) = O(h_k^p).$$

As we do not know the exact solution we cannot determine the local discretisation error by this formula. However, we would argue that the local discretisation error is approximately the same for all close-by solutions of the ODE. We will choose $\bar{x}(t)$ such that $d\bar{x}/dt = f(t, \bar{x}(t))$ and $\bar{x}(t_k) = x_k$. With this we get the approximation

$$L(t_k, h_k) \approx \frac{\bar{x}(t_{k+1}) - x_k}{h_k} - \phi(t_k, x_k).$$

We will now replace $\bar{x}(t_{k+1})$ by an approximation $\bar{x}_{k+1}$ which we get from higher order method.

$$\bar{x}_{k+1} = x_k + h_k \bar{\phi}(t_k, x_k).$$

with an associated discretisation error $\bar{L}(t_k, h_k) = O(h_k^{p+1})$.

For instance we could consider Euler's method which is $O(h_k)$ in association with the Modified Euler's method which is second order accurate, or $O(h_k^2)$.

We then get the following approximation of the local error:

$$\begin{aligned} L(t_k, h_k) &\approx \frac{\bar{x}_{k+1} - x_k}{h_k} - \phi(t_k, x_k) \\ &= \frac{\bar{x}_{k+1} - (x_k + h_k \phi(t_k, x_k))}{h_k} \\ &= \frac{\bar{x}_{k+1} - x_{k+1}}{h_k}. \end{aligned}$$

Step Size Control

Assume that we want to change the stepsize h_k at time t_k to qh_k such that the local error satisfies $L(t_k, qh_k) \approx \epsilon$.

- Furthermore, assume that we know the consistency order p of our method. We then have
$$L(t_k, qh_k) \approx q^p\, L(t_k, h_k).$$
- Recall the local error estimator for time step h_k
$$L(t_k, h_k) \approx \frac{\bar{x}_{k+1} - x_{k+1}}{h_k}.$$

Substitute these two approximations into $L(t_k, qh_k) \approx \epsilon$ to get
$$q^p\, \frac{|\bar{x}_{k+1} - x_{k+1}|}{h_k} \approx \epsilon$$
solving for q gives
$$q \approx \left(\frac{\epsilon h_k}{|\bar{x}_{k+1} - x_{k+1}|}\right)^{1/p}.$$

In summary: Here we have used *two methods* of different order to provide an estimate of the *solution*, together with an estimate of the *error*.

New Problem: We now need to make this process of calculating these two schemes as efficient as possible.

Runge-Kutta Fehlberg Method 4-5.

This is another example of the use of two schemes, one of order p and the other of order $p+1$ to estimate solutions and the error of the solution.

Furthermore, in this case where two Runge-Kutta methods of order 4 and 5 are developed, as many as possible of the function evaluations are used in both methods.

In this method the Runge-Kutta method of order 5
$$\bar{x}_{k+1} = \bar{x}_k + \tfrac{16}{135}k_1 + \tfrac{6656}{12825}k_3 + \tfrac{28561}{56430}k_4 - \tfrac{9}{50}k_5 + \tfrac{2}{55}k_6$$
is used with the Runge Kutta method of order 4
$$x_{k+1} = x_k + \tfrac{25}{216}k_1 + \tfrac{1408}{2565}k_3 + \tfrac{2197}{4104}k_4 - \tfrac{1}{5}k_5$$
where
$$\begin{aligned}
k_1 &= hf(t_k, x_k)\\
k_2 &= hf\left(t_k + \tfrac{1}{4}h, x_k + \tfrac{1}{4}k_1\right)\\
k_3 &= hf\left(t_k + \tfrac{3}{8}h, x_k + \tfrac{3}{32}k_1 + \tfrac{9}{32}k_2\right)\\
k_4 &= hf\left(t_k + \tfrac{12}{13}h, x_k + \tfrac{1932}{2197}k_1 - \tfrac{7200}{2197}k_2 + \tfrac{7296}{2197}k_3\right)\\
k_5 &= hf\left(t_k + h, x_k + \tfrac{439}{216}k_1 - 8k_2 + \tfrac{3680}{513}k_3 - \tfrac{845}{4104}k_4\right)\\
k_6 &= hf\left(t_k + \tfrac{1}{2}h, x_k - \tfrac{8}{27}k_1 + 2k_2 - \tfrac{3544}{2565}k_3 + \tfrac{1859}{4104}k_4 - \tfrac{11}{40}k_5\right).
\end{aligned}$$

Advantage: The advantage of this combined method is that there are only six function evaluations per step. Arbitrary Runge-Kutta methods of order 4 would require 4 evaluations and Runge Kutta methods of order 5 would require 6 evaluations. In total 10 function evaluations. In this case we would choose q so that
$$\begin{aligned}
q &= \left(\frac{\varepsilon h_k}{2\,|\bar{x}_{k+1} - x_{k+1}|}\right)^{1/4}\\
&= 0.84\left(\frac{\varepsilon h_k}{|\bar{x}_{k+1} - x_{k+1}|}\right)^{1/4}
\end{aligned}$$

4.9.4 Numerical Solution of ODE's using Multistep Methods

Introduction

We now discuss *multistep methods*. Their most prominent feature is that are written in terms of

$$f_k := f(t_k, x_k).$$

This allows for some computational savings as the f_k can be reused.

We will consider three different classes of multistep methods here. The first two are based on an approximation of the integral in the integral formula

$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(s, x(s))ds$$

while the third class consists of collocation methods for the ODE

$$\frac{dx}{dx}(t_{k+1}) = f(t_{k+1}, x(t_{k+1}))$$

which is obtained by approximating the derivative by a finite difference.

Adam-Bashforth(A-B) and Adam-Moulton(A-M)

- A-B approximates integral using $(t_{k-m}, f_{k-m}), \dots, (t_k, f_k)$
  - quadrature method uses polynomial extrapolation
  - method explicit, one function evaluation per step
- A-M approximates integral using $(t_{k-m}, f_{k-m}), \dots, (t_{k+1}, f_{k+1})$
  - quadrature method uses polynomial interpolation
  - method implicit as f_{k+1} depends on unknown x_{k+1}, may use multiple function and derivative evaluations per step

Typically A-M is more stable and more expensive than A-B.

The above means is that such schemes depend on the values calculated at more than one of the previous steps.

Adams Bashforth

$$x_{k+1} = x_k + h \sum_{\underset{\nwarrow \text{explicit}}{i=1}}^{n} \beta_i \, f_{k+1-i}$$

Example: fourth order A-B method

$$x_{k+1} = x_k + \frac{h}{24}(55\, f_k - 59\, f_{k-1} + 37\, f_{k-2} - 9\, f_{k-3}).$$

Adams Moulton

$$x_{k+1} = x_k + h \sum_{\underset{\nwarrow \text{implicit}}{i=0}}^{n} \beta_i \, f_{k+1-i}$$

Example: fourth order A-M method

$$x_{k+1} = x_k + \frac{h}{24}(9\, f_{k+1} + 19\, f_k - 5\, f_{k-1} + f_{k-2}).$$

Backward Differentiation Formula (BDF)

The third class of methods considered here consists of collocation methods for the differential equation rather than the integral equation. Using a finite difference approximation for the derivative dx/dt for t_{k+1} one gets

$$x_{k+1} = \sum_{i=1}^{m} \alpha_i \, x_{k+1-i} \quad + h\beta_0 \, f_{k+1}$$

Like the A-M method the BDF method implicit and requires a solution procedure for every step. Taking all terms depending on the unknown x_{k+1} one may rewrite the recursion as

$$x_{k+1} - h\beta_0 \, f(t_{k+1}, x_{k+1}) = \sum_{i=1}^{m} \alpha_i \, x_{k+1-i}$$

For the solution one often uses fixpoint (so-called predictor-corrector) methods or Newton's method.

Numerical Methods - General Theory

A general multistep scheme

Multistep methods are all of the form

$$x_{k+1} = \sum_{j=1}^{m} \alpha_j x_{k+1-j} + h \, \Phi(t_{k+1}, ..., t_{k+1-m}; x_{k+1}, ...x_{k+1-m}) \tag{4.7}$$

where α_i are known constants, $x_1, ...x_k$ are given and Φ is a known function depending on f, and m some fixed integer.

- Many methods including one and multistep methods can be cast in this form but there exist important methods which cannot. This includes Taylor-series methods which explicitely determine derivatives of the solution $x(t)$.
- Multistep methods require an extra scheme to determine $x_1, \dots, x_{m-1}$.

Linear Multistep Methods

The *linear multistep methods* are most widely used. They are of the form

$$\sum_{j=0}^{m} \alpha_j x_{k+1-j} = h \sum_{j=0}^{m} \beta_j f_{k+1-j} \tag{4.8}$$

where β_j are known constants. The A-B, A-M and BDF methods are all linear.

Note that for the case of one step methods we have $m = 1$. For example, if

$$m = 1, \quad \alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = 0, \quad \beta_1 = 1$$

then we have the explicit Euler scheme of

$$x_{k+1} - x_k = hf_k.$$

Note too that when $\beta_0 = 0$ we have an *explicit* scheme, while for $\beta_0 \neq 0$ the scheme is *implicit*.

In general, when $x_{k-m}, ..., x_{k-1}$ and their derivatives are known, then Equation (4.8) is considered as a linear equation in the unknowns x_k and x'_k:

$$\alpha_0 x_k - h\beta_0 x'_k = \text{known value}$$

and with

$$f(t_k, x_k) - x'_k = 0$$

x_k can be determined. In the case where $\beta_0 = 0$ this is particularly simple, as in Euler's method and other explicit methods. Also, when the differential equation f is linear, one only has to solve two linear equations in two unknowns (easy).

However, in the case of implicit methods ($\beta_0 \neq 0$) where f is nonlinear, this nonlinear system needs to be solved at each step (not so easy). This may provide much extra work so that an explicit method seems an obvious choice. However, implicit methods have (typically) larger regions of A-stability and are particularly useful when solving stiff equations (see below).

Derivation of $x_{k+1} = \alpha_1\, x_k + h(\beta_1\, f_k + \beta_2\, f_{k-1})$

Determine coefficients such that scheme recovers exact solutions $x(t) = 1$, $x(t) = t$ and $x(t) = t^2$.

$$\begin{aligned} 1 &= \alpha_1 \cdot 1 + h(\beta_1 \cdot 0 + \beta_2 \cdot 0) \\ t_{k+1} &= \alpha_1 t_k + h(\beta_1 \cdot 1 + \beta_2 \cdot 1) \\ t_{k+1}^2 &= \alpha_1 t_k^2 + h(\beta_1\, 2\, t_k + \beta_2\, 2\, t_{k-1}). \end{aligned}$$

We only need to require these equations to hold for $t_0 = 0$, $t_1 = h$ and $t_2 = 2h$. They will then hold for all $t_k = kh$. This leads to the system

$$\begin{aligned} 1 &= \alpha_1 + 0\,\beta_1 + 0\,\beta_2 \\ 2 &= \alpha_1 + \beta_1 + \beta_2 \\ 4 &= \alpha_1 + 2\,\beta_1 + 0\,\beta_2. \end{aligned}$$

By solving for the coefficients one gets

$$x_{k+1} = x_k + \frac{h}{2}\,(3\, f_k - f_{k-1}).$$

Some other Examples

The Secant Rule:

$$m = 2, \quad \alpha_0 = 1, \quad \alpha_1 = 0, \quad \alpha_2 = -1, \quad \beta_0 = 0, \quad \beta_1 = 2, \quad \beta_2 = 0$$

which gives

$$x_{k+1} - x_{k-1} = 2hf_k$$

Simpsons Rule:

$$m = 2, \quad \alpha_0 = 1, \quad \alpha_1 = 0, \quad \alpha_2 = -1, \quad \beta_0 = \frac{1}{3}, \quad \beta_1 = \frac{4}{3}, \quad \beta_2 = \frac{1}{3}$$

which gives

$$x_{k+1} - x_{k-1} = \frac{h}{3}(f_{k+1} + 4f_k + f_{k-1}).$$

In general, how are such formulae obtained and the α and β determined?

We apply Equation (4.8) to $x' = x$, $x(0) = 1$ and then we should get e^t (or $x_k = e^{kh}$) as a solution. We determine α and β such that we ensure $x_k = e^{hk}$, or is as close as possible.

Let $z = e^h$, then

$$\sum_{j=0}^{m} \alpha_j z^{k-j} \approx \log z \sum_{j=0}^{m} \beta_j z^{k-j}$$

for all k. The task now becomes to approximate $\log z$ which is a quotient of two polynomials

$$\log z = \frac{A(z)}{B(z)}.$$

If one is interested in a large order for the scheme then we require a good approximation in the neighbourhood of $h = 0$ or $z \approx 1$. A crude approximation of $\log z$ at 1 is given by

$$\log z \approx z - 1$$

which is

$$A(z) = z - 1, \quad B(z) = 1$$

and determines

$$\alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = 0, \quad \beta_1 = 1.$$

So this gets back to Euler's scheme. Similarly we can get the secant rule by observing

$$\log z = -\log \frac{1}{z} \approx -\left[\frac{1}{z} - 1\right] = 1 - \frac{1}{z}$$

and then averaging this with the previous approximation to approximate $\log z$ as

$$\log z \approx \frac{z^2 - 1}{2z}.$$

Further schemes may be determined by truncating the series expansion of $\log z$ after the second order term.

From

$$\log z \approx -\frac{z^2 - 4z + 3}{2}$$

we get

$$\alpha_0 = 1, \quad \alpha_1 = -4, \quad \alpha_2 = 3, \quad \beta_0 = 0, \quad \beta_1 = 0, \quad \beta_2 = -2$$

which gives

$$x_k - 4x_{k-1} + 3x_{k-2} = -2hx'_{k-2}.$$

Substituting the exact solution with $h = 0.1$ into this last equation, indicates a good match. However, using this as a recurrence formula for x_k the results diverge to ∞! Thus this method is useless, coming from a series expansion of $\log z$, while, on the other hand, methods resulting from the truncation of the series for $1/\log z$ lead to the well known *Adams-Moulton Method* an implicit method and the *Adams-Bashforth Method* an explicit scheme.

Stability for Multistep Methods

Stability of $x_{k+1} = x_{k-1} + 2hf_k$

One of the pervading concerns of scientific computing is that of stability. We have differentiated between the stability of an equation and that of a numerical scheme.

For an equation instability means loosely that for two initial conditions x_0 and $x_0 + \epsilon$, where ϵ is very small and positive, the solutions are vastly different.

For an unstable numerical scheme the notion is similar. In the following we discuss the stability of the method $x_{k+1} = x_{k-1} + 2hf_k$. This is a second order accurate multistep method. We will in particular see how spurious non-stable solutions may occur for multistep methods.

Stability – continued

As a test problem we consider here

$$x' = -2x + 1, \qquad y_0 = 1$$

which has the exact solution $\frac{1}{2}e^{-2t} + \frac{1}{2}$. If the initial condition is $x(0) = 1 + \epsilon$ then the solution becomes

$$x(t) = (\frac{1}{2} + \epsilon)e^{-2t} + \frac{1}{2}$$

and the change in the solutions is only ϵe^{-2t}.

Applying the scheme $x_{k+1} = x_{k-1} + 2hf(t, t_k)$ to this problem gives

$$x_{k+1} = x_{k-1} + 2h(-2x_k + 1) = -4hx_k + x_{k-1} + 2h, \qquad x_0 = 1.$$

Note that one needs one extra initial condition for x_1 to get the multistep method started. For example, one could choose the exact solution (or some approximation)

$$x_1 = \frac{1}{2}(e^{-2h} + 1).$$

Stability - continued

In order to understand the stability we rewrite the recursion in vector form as

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} -4h & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} + \begin{bmatrix} 2h \\ 0 \end{bmatrix}$$

We can now verify that

- the method has a constant (stationary) solution $x_k = 0.5$
- all solutions are of the form $x_k = 0.5 + \gamma_1 \lambda_1^k + \gamma_2 \lambda_2^k$
- the eigenvalues are $\lambda_1 = \sqrt{1+4h^2} - 2h \approx 1 - 2h + 2h^2 \approx \exp(-2h)$ and $\lambda_2 = -\sqrt{1+4h^2} - 2h \approx -(1 + 2h + 2h^2) \approx -\exp(2h)$ such that the solution is

$$x_k \approx 0.5 + \gamma_1 \exp(-2t_k) + \gamma_2 (-1)^k \exp(2t_k)$$

This demonstrates that the spurious solution goes to infinity and thus the method is unstable independent of the value of h

Stability Test Problem

As we have seen above, not all schemes derived from Equation (4.8) are stable. Thus it would be useful to derive criteria for the stability of such methods.

We return to our test problem $x' = \lambda x$ for λ complex. (This can be extended to systems of equations $x' = Ax$ where a solution component is associated with the eigenvalue λ.) Now (4.8) becomes

$$\sum_{j=0}^{m} \alpha_j x_{k-j} = h\lambda \sum_{j=0}^{m} \beta_j x_{k-j}$$

from which (as was the case of our previous analysis) it is clear that only the product $h\lambda$ is of importance and we will integrate with respect to this steplength $s = h\lambda$.

So we start with

$$\sum_{j=0}^{m} (\alpha_j - s\beta_j) x_{k-j} = 0$$

which is a difference formula with a solution of the form

$$x_k = \sum_{v=1}^{m} g_v z_v^k$$

which when substituted into the difference equation and rearranged yields

$$\sum_{v=1}^{m} g_v z_v^{k-m} \left\{ \sum_{j=0}^{m} (\alpha_j - s\beta_j)\, z_v^{m-j} \right\} = 0.$$

Since this must hold for all k and since the term in the curly braces is independent of k we have that

$$\sum_{j=0}^{m} (\alpha_j - s\beta_j)\, z_v^{m-j} = 0,$$

or in terms of the polynomials $A(z)$ and $B(z)$ we had before, this is

$$A(z) - sB(z) = 0 \quad \text{for} \quad z = z_v, \quad v = 1, ..., m.$$

The solutions to this equation are the values of $z_1, ..., z_m$ and hence the numerical solution can be determined. The exact solution is $x_k = c(e^s)^k$ where $t = sk$.

For large k the numerical solution is dominated by the term $g_1 z_1^k$ where z_1 is the largest of the roots $|z_1|, ..., |z_m|$. Thus *if the numerical solution is more or less to follow the exact solution, the dominant root z_1 must lie in the vicinity of e^s.*

Since α_j and β_j were determined such that

$$\frac{A(z)}{B(z)} \approx \log z$$

there *is* one root which lies near e^s, and with $s = \log z$ then $A(z) - sB(z) \approx 0$. This is not enough on its own: we still require the largest $|z_j|$ to be the root near e^s.

In order to examine this we will define the local relative error as

$$\Psi(s) = \left| \frac{\log z_1 - s}{s} \right|.$$

This defines $\Psi(s)$ as a real function of complex s, and a measure of the relative error per unit integration step.

To use this for some stability criterion we require that $\Psi(0) = 0$ and $\Psi(s)$ is small for small $|s|$. (This ensures a near exact solution when h is close to 0.) There is a root z near to e^s (resulting from the construction of the polynomials $A(z)$ and $B(z)$) and we require that for sufficiently small s this root has the largest absolute value. Thus, we require that for $s \to 0$ we have

$$\log z_1(s) - s = O(s)$$

and in particular that $\log z_1(0) = 0$ (or $z_1(0) = 1$) or

$$A(1) = 0. \tag{4.9}$$

Furthermore

$$\left.\frac{d(\log z_1)}{ds}\right|_{s=0} - 1 = 0 \qquad \text{or} \qquad \left|\frac{dz_1}{ds}\right|_{s=0} = 1.$$

And from $A(z_1) - sB(z_1) = 0$ we have

$$\frac{dA}{dz_1}\frac{dz_1}{ds} - Bz_1 - s\frac{dB}{dz1}\frac{dz_1}{ds} = 0$$

and so for $s = 0$ and $z_1 = 1$

$$A'(1) - B(1) = 0. \tag{4.10}$$

The two conditions (4.9) and (4.10) are the **consistency conditions** and are necessary for stability.

For a criterion for stability we need more. Since the roots of an algebraic equation depend continuously on the coefficients, a root of $A(z) - sB(z) = 0$ in the vicinity of e^s is, for all sufficiently small $|s|$, the largest in absolute value, if this is so for $s = 0$ (that is if $z = 1$ is the root of maximum modulus for $A(z) = 0$ and is simple). Consequently we have a condition for the stability of the general multi-step Equation (4.8):

$$\frac{A(z)}{z-1} \neq 0 \quad \text{for} \quad |z| \geq 1.$$

Automatically Choosing the Stepsize

Error Control

Any general code for automatically integrating IVP's should be sufficiently general to deal with any equation presented to it. It thus requires efficient error control and step size selection.

There are two basic premises for such error control. It is much simpler to estimate the local error than a global error, so this is what happens in practice. Also, it is assumed that the discretisation error ($L(t,h)$) dominates the roundoff error and so the latter is ignored. However, if the roundoff error does become significant because some absolute stability criterion has been violated, then the generated "noise" results in a large local discretisation error and thus the step size is decreased. In return, this helps prevent the further violation of absolute stability criteria.

As was discussed above in the section on one step methods, two schemes could be applied to control the error through a variable stepsize. For the general scheme Equation (4.8) the local discretisation error $L(t,h)$ at t is defined as

$$L(t,h) = \frac{1}{h}[x(t+h) - \sum_{j=1}^{m} \alpha_j x(t-(j-1)h)] - \sum_{j=0}^{m} \beta_j x'(t-(j-1)h).$$

Thus for any method, for any given m, α_j's and β's, one can compute the local discretisation error by the expansion of x and x' about t. In particular this can be used to compute the order of the numerical scheme.

Bibliography