Chapter 1

Introduction to Discretization

We begin the journey to understand numerical methods for differential equations by concentrating on a specific type of ordinary differential equation (ODE) which describes how some function evolves in time given its initial configuration. This type of problem is called an **initial value problem** (IVP) for obvious reasons. An example of this type of problem is modeling the growth of a population where we know the initial population size, we have information on how the population grows with time, (i.e., we have information on the first derivative) and we want to determine the population at subsequent times. Another example is the motion of a simple mechanical system such as a mass suspended from a spring where we want to know the position (e.g., the displacement from the rest position of the spring) at subsequent times. This system is modeled using Newton's second law of motion and thus an equation containing a second derivative of the displacement is obtained; here we typically specify the initial displacement of the spring due to the mass and the initial velocity.

Initial value problems can be either ordinary or partial differential equations; however, here we focus on ODEs. In addition to IVPs being important in their own right, our understanding of this type of problem will form a foundation for our study of time dependent partial differential equations (PDEs).

We begin this chapter by looking at the prototype IVP that we consider initially. The differential equation in this IVP is first order and gives information on the rate of change of our unknown; in addition, an initial value for the unknown is specified. Later, in Chapter 3, we consider higher order IVPs and we will see that higher order IVPs can be written as a system of first order IVPs. In fact, most ODE software for IVPs is written for first order problems. Before proceeding to approximating the solution of this prototype IVP we investigate conditions which guarantee that the analytic solution exists, is unique and depends continuously on the data. In the sequel we will only be interested in approximating the solution to such problems.

Once we have specified our prototype IVP we introduce the idea of approximating its solution using a difference equation. In general, we have to give up the notion

of finding an analytic solution which gives an expression for the solution at any time and instead find a discrete solution which is an approximation to the exact solution at a set of finite times. The basic idea is that we discretize our domain, in this case a time interval, and then derive a difference equation which approximates the differential equation in some sense. The difference equation is in terms of a discrete function and only involves differences in the function values; that is, it does not contain any derivatives. Our hope is that as the difference equation is imposed at more and more points (which much be chosen in a uniform manner) then its solution will approach the exact solution to the IVP.

The simplest method for approximating the solution to our prototype IVP is the Euler method which we derive by approximating the derivative in the differential equation by the slope of a secant line. In § 1.2.2 we demonstrate the linear convergence of the method by introducing the concepts of local truncation error and global error. The important difference in explicit and implicit methods is illustrated by comparing the forward and backward Euler methods. In § 1.3 we present two models of growth/decay which fit into our prototype IVP and give results of numerical simulations for specific problems. In addition, we demonstrate that our numerical rate of convergence matches our theoretical rate.

We conclude this chapter by demonstrating several approaches to deriving the Euler method. The reason for doing this is because the Euler method converges rather slowly and so we need to derive methods which converge faster. In addition, we will see that the forward Euler approximation may grow in an unbounded manner for some problems and so clearly other methods are needed.

1.1 Prototype Initial Value Problem

One of the problems encountered in modeling is an initial value problem where we seek a function whose value is known at some initial time and whose derivative is specified for subsequent times. The following problems are examples of first order IVPs for $y(t)$:

$$\begin{array}{rcl} y'(t) &=& \sin \pi t;\ 0 < t \le 4 \\ y(0) &=& 0 \end{array} \qquad \begin{array}{rcl} y'(t) + y(t) &=& t^2;\ 2 < t \le 10 \\ y(2) &=& 1\,. \end{array} \tag{1.1}$$

Clearly, these examples are special cases of the following general IVP.

GENERAL IVP: find $y(t)$ satisfying

$$\frac{dy}{dt} = f(t,y) \quad t_0 < t \le T \tag{1.2a}$$

$$y(t_0) = y_0\,. \tag{1.2b}$$

Here $f(t,y)$ is the given derivative of $y(t)$ and y_0 is the known value at the initial time t_0. For example, for the second IVP in (1.1) we have $f(t,y) = t^2 - y$, $t_0 = 2$,

$T = 10$ and $y_0 = 1$. Note that both linear and nonlinear differential equations are included in the general equation (1.2a).

One of the simplest situations is when $f = f(t)$, i.e., f is a function of t and not both t and y. In this case we can find the solution if the antiderivative[1] of $f(t)$ can be found. We should expect that the solution to the differential equation (1.2a) is not unique; actually there is an infinite family of solutions which satisfy the differential equation. This should be apparent because the differential equation only specifies the slope $y'(t)$ at every point and not the value of $y(t)$ itself. To be able to determine a unique solution we have to specify $y(t)$ at some point such as its initial value. The following examples illustrate two standard techniques for solving a differential equation of the form (1.2a). When we write a code to approximate the solution of the IVP (1.2) we always want to test the code on a problem where the exact solution is known so it is useful to review some standard approaches.

Example 1. METHOD OF SEPARATION OF VARIABLES FOR FINDING THE ANALYTIC SOLUTION OF (1.2).

Consider the differential equation $y'(t) = -ty(t)$ and find its general solution. Then impose the initial condition $y(0) = 2$ to determine a unique solution to the IVP.

Because $f(t, y)$ is a function of both y and t we can not directly integrate the differential equation with respect to t to obtain the solution. However, if we rewrite the equation as $\frac{dy}{y} = -tdt$ then we can integrate both sides of the equation to get

$$\ln y + C_1 = -\frac{t^2}{2} + C_2 \Rightarrow e^{\ln y + C_1} = e^{-\frac{t^2}{2} + C_2} \Rightarrow e^{C_1} y(t) = e^{-\frac{t^2}{2}} e^{C_2} \Rightarrow y(t) = Ce^{-\frac{t^2}{2}} .$$

This technique can be applied whenever we can "separate variables", i.e., bring all the terms involving y to one side of the equation and those involving t to the other side. Note that the general solution to this differential equation involves an arbitrary constant C and thus there is an infinite family of solutions which satisfy the differential equation. Figure 1.1 illustrates some of these solutions; note that as $t \to \pm\infty$ the solution approaches zero.

We can always verify that no errors have been made in computing the solution by demonstrating that it satisfies the differential equation. Here we have

$$y(t) = Ce^{-\frac{t^2}{2}} \Rightarrow y'(t) = C\frac{-2t}{2}e^{-\frac{t^2}{2}} = -t\left(Ce^{-\frac{t^2}{2}}\right) = -ty(t)$$

so the equation is satisfied.

To determine a unique solution we impose the value of $y(t)$ at some point; here we set $y(0) = 2$ to get the particular solution $y(t) = 2e^{-t^2/2}$ because

$$y(0) = 2, \quad y(t) = Ce^{-\frac{t^2}{2}} \quad \Rightarrow \quad 2 = Ce^0 \quad \Rightarrow \quad C = 2 .$$

Example 2. METHOD OF USING AN INTEGRATING FACTOR FOR FINDING THE ANALYTIC SOLUTION TO (1.2) .

Find the solution to the IVP

$$y'(t) + \sin(t)y(t) = t^2 e^{\cos(t)} \quad y(\frac{\pi}{2}) = 0$$

[1] The antiderivative of $f(t)$ is a function $y(t)$ whose derivative is $f(t)$.

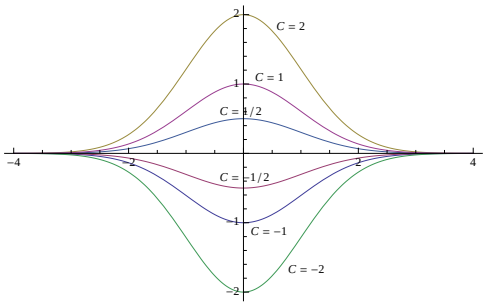


Figure 1.1: Family of solutions of $y'(t) = -ty(t)$.

Here $f(t, y) = t^2 e^{\cos(t)} - \sin(t)y(t)$ and we can not separate variables for this differential equation; however, we can find a term to multiply the differential equation by so that it is integrable. Consider the factor e^{μ} where

$$\mu = \int \sin t)\, dt = -\cos(t)$$

which is just the integral of the coefficient of $y(t)$ when the coefficient of $y'(t)$ is one. Multiplying through the equation by e^{μ} always makes the left hand side of the equation integrable. Here we have

$$e^{-\cos(t)}\Big(y'(t) + \sin(t)\, y(t)\Big) = t^2$$

where the left hand side can be simplified as

$$\frac{d}{dt}\Big(e^{-\cos(t)} y(t)\Big).$$

Now we separate variables and integrate to get

$$\int d\Big(e^{-\cos(t)} y(t)\Big) = \int t^2\, dt \Rightarrow e^{-\cos(t)} y(t) = \frac{t^3}{3} + C \Rightarrow y(t) = e^{\cos(t)}\Big(\frac{t^3}{3} + C\Big).$$

Of course this method relies on being able to integrate exactly the right-hand side times the integrating factor. To determine C we impose the initial condition

$$0 = y(\frac{\pi}{2}) = e^0\Big(\frac{\pi^3}{24} + C\Big) \Rightarrow C = -\frac{\pi^3}{24}.$$

Even if we are unable to determine the analytic solution to (1.2), we can still gain some understanding of the behavior of the solution. This is done by the visualization technique of plotting the tangent line to the solution at numerous points (t, y); recall that the slope of the tangent line to the solution curve is given and is just $f(t, y)$. Many graphic packages provide commands for automatically drawing a direction field with arrows which are scaled to indicate the magnitude of the slope; typically software packages also offer the option of drawing some solutions or streamlines. Using direction fields to determine the behavior of the solution is illustrated in the following example.

Example 3. DIRECTION FIELDS FOR (1.2)

Draw the direction fields for the ODE

$$y'(t) = t^2 + y(t) \quad 0 < t < 4$$

and indicate the specific solution which satisfies $y(0) = 1$.

At each point (t, y) we draw the line with slope $t^2 + y$; this is illustrated in Figure 1.2 where numerous streamlines have been sketched. To thread a solution through the direction field start at a point and follow the solution, remembering that solutions don't cross and that nearby tangent lines should be nearly the same.

To see which streamline corresponds to the solution with $y(0) = 1$ we locate the point $(0, 1)$ and follow the tangents; this solution is indicated by a thick line in the direction field plot. If a different initial condition is imposed, then we get a different streamline. The exact solution to this IVP can be found by using an integrating factor to get $y(t) = -2 + 3e^t - 2t + t^2$.

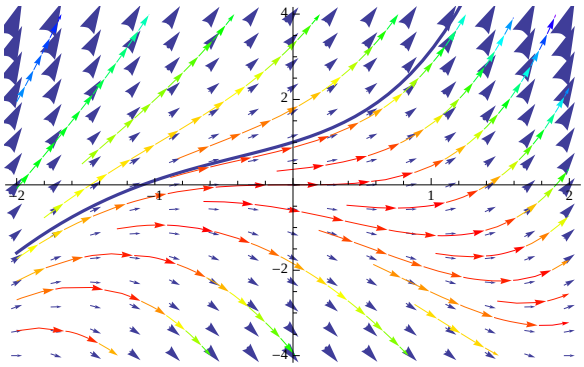


Figure 1.2: Direction field and streamlines for $y'(t) = t^2 + y(t)$. The exact solution which satisfies $y(0) = 1$ is drawn with a thick line.

For some choices of $f(t, y)$ we are able to find the *analytic solution* to the IVP; that is, an explicit function which gives the solution for any time t. However, even for the simplified case of $f(t, y) = f(t)$ this is not always possible. For example, consider $f(t) = \sin(t^2)$ which has no explicit formula for its antiderivative; in fact, a symbolic algebra software package like Mathematica gives the antiderivative in terms of the Fresnel Integral which can be represented by an infinite power series near the origin; consequently there is no closed form solution to the problem. Although there are numerous techniques for finding the analytic solution of first order differential equations, we are unable to easily obtain closed form analytic solutions for many equations. When this is the case, we must turn to a *numerical approximation* to the solution where we give up finding a formula for the solution at all times and instead find an approximation at a set of distinct times.

Before we discuss the concept of discretizing the IVP (1.2) we first need to ask ourselves if our prototype IVP actually has an analytic solution, even if we are unable to find it. We are only interested in approximating the solution to IVPs which have a unique solution. However, even if we know that a unique solution exists, we may still have unreliable numerical results if the solution of the IVP does not depend continuously on the data. If this is the case, then small changes in the data can cause large changes in the solution and thus roundoff errors in our

calculations can produce meaningless results. In this situation we say the IVP is *ill-posed* or *ill-conditioned*, a situation we would like to avoid. Most differential equations that arise from modeling real-world phenomena are well-posed so it is reasonable to assume that we only want to approximate solutions of well-posed problems.

The conditions that guarantee well-posedness of a solution to (1.2) are well known and are presented in Theorem 1.1. Basically the theorem requires that the derivative of $y(t)$ (given by $f(t, y)$) is continuous and, moreover, this derivative is not allowed to change too quickly as y changes. The precise requirement on $f(t, y)$ is that it is Lipschitz[2] continuous in y. To understand this concept first think of a function $g(x)$ of one variable defined on an interval I. Lipschitz continuity requires that the magnitude of the slope of the line joining any two points x_1 and x_2 in I must be bounded by a real number. Formally, a function $g(x)$ defined on a domain $D \subset \mathbb{R}^1$ is Lipschitz continuous on D if for any $x_1, x_2 \in D$ there is a constant L such that

$$|g(x_1) - g(x_2)| \leq L|x_1 - x_2|\,,$$

or equivalently,

$$\frac{|g(x_1) - g(x_2)|}{|x_1 - x_2|} \leq L\,.$$

Here L is called the Lipschitz constant. This condition says that we must find one constant L which works for all points in the domain. Clearly the Lipschitz constant is not unique; for example, if $L = 5$, then $L = 5.1, 6, 10, 100$, etc. also satisfy the condition. Lipschitz continuity is a stronger condition than merely saying the function is continuous so a Lipschitz continuous function is always continuous but the converse is not true. For example, the function $g(x) = \sqrt{x}$ is continuous on $D = [0, 1]$ but is not Lipschitz continuous on D. If $g(x)$ is differentiable then an easy way to determine the Lipschitz constant is to find a constant such that $|g'(x)| \leq L$ for all $x \in D$. There are functions which are Lipschitz continuous but not differentiable. For example, consider the continuous function $g(x) = |x|$ on $D = [-1, 1]$. Clearly it is not differentiable on D because it is not differentiable at $x = 0$. However, it is Lipschitz continuous with $L = 1$ because the magnitude of the slope of the secant line between any two points is always less than or equal to one.

For the existence and uniqueness result for (1.2), we need $f(t, y)$ to be Lipschitz continuous in y so we need to extend the above definition because f is now a function of two variables. Formally, we have that a function $g(t, y)$ defined on a domain $D \subset \mathbb{R}^2$ is Lipschitz continuous in D for the variable y if for any $(t, y_1), (t, y_2) \in D$ there is a constant L such that

$$|g(t, y_1) - gt, y_2)| \leq L|y_1 - y_2|\,. \tag{1.3}$$

We are now ready to state the theorem which guarantees existence and uniqueness of a solution to (1.2) as well as guaranteeing that the solution depends continuously on the data; i.e., the problem is well-posed. Note that $y(t)$ is defined on $[t_0, T]$

[2]Named after the German mathematician Rudolf Lipschitz (1832-1903)

whereas $f(t, y)$ must be defined on a domain in $\mathbb{R}^2$. Specifically the first argument t is in $[t_0, T]$ but y can be any real number so that $D = \{(t, y) \mid t \in [t_0, T], y \in \mathbb{R}^1\}$; a shorter notation for expressing D is $D = [t_0, T] \times \mathbb{R}^1$ which we will employ in the sequel.

Theorem 1.1. *Let $D = [t_0, T] \times \mathbb{R}^1$ and assume that $f(t, y)$ is continuous on D and is Lipschitz continuous in y on D; i.e., it satisfies (1.3). Then the IVP (1.2) has a unique solution in D and moreover, the problem is well-posed.*

In the sequel we will only consider IVPs which are well-posed, that is, which have a unique solution that depends continuously on the data.

1.2 Discretization

One approach to discretizing a differential equation is to approximate the derivatives in the equation by difference quotients; i.e., derive a *difference equation* which involves only differences in function values. The solution to the difference equation will not be a continuous function but rather a *discrete function* which is defined over a set of points. If the solution is needed at some other point, interpolation is often used. In this chapter we concentrate on two of the simplest difference equations for (1.2a).

Because the difference equation is defined at a finite set of points we first discretize the time domain $[t_0, T]$; alternately, if our solution depended on the spatial domain x instead of t we would discretize the given spatial interval. For now we use $N + 1$ evenly spaced points t_i, $i = 0, 1, 2, \ldots, N$

$$t_1 = t_0 + \Delta t, \quad t_2 = t_0 + 2\Delta t, \quad \cdots, \quad t_N = t_0 + N\Delta t = T\,,$$

where $\Delta t = (T - t_0)/N$ and is called the *step size* or *time step*.

Our task is to find a means for approximating the solution at each of these discrete values and our hope is that as we perform more calculations with N getting large, or equivalently $\Delta t \to 0$, our approximate solution will approach the exact solution in some sense. In the left plot in Figure 1.3 we plot an exact solution (the continuous curve) to a specific IVP and a discrete approximation for $\Delta t = 0.5$. The approximate solutions are plotted only at the points where the difference equation is enforced. From this plot we are unable to say if our discrete solution appears to approach the continuous solution. However, in the figure on the right we plot this discrete solution plus three additional discrete solutions where we cut Δt in halve for each approximation. By observing the plot and using the "eyeball norm" we can convince ourselves that as $\Delta t \to 0$ our discrete solution is approaching the analytic solution. One of our goals is to make this statement precise and to determine the rate at which our approximate solution converges to the exact solution.

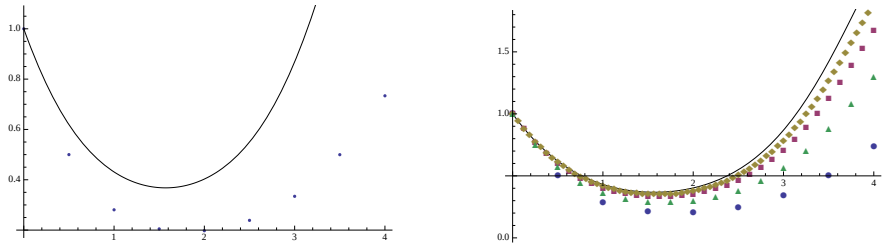


Figure 1.3: The exact solution to an IVP is shown as a solid curve. In the figure on the left a discrete solution using $\Delta t = 0.5$ is plotted. From this plot, it is not possible to say that the discrete solution is approaching the exact solution. However, in the figure on the right the discrete solutions for $\Delta t = 0.5, 0.25, 0.125$, and 0.625 are plotted. From this figure, the discrete approximations appear to be approaching the exact solution as Δt decreases.

We will see that there are many approaches to deriving discrete methods for our IVP (1.2). The most straightforward approach to obtain a difference equation is to approximate the derivative appearing in the equation; we discuss how to do this next to obtain the Euler Method. Another goal of this section is to discuss the discretization errors that occur when we implement a method on a computer.

1.2.1 The Euler Method

The simplest method for approximating the solution of (1.2) is called the Euler Method named after Leonhard Euler who wrote about the method in the latter half of the eighteenth century. The basic idea is to obtain a difference equation which involves differences of approximations to $y(t)$ at certain points t_i and then use the difference equation to approximate the solution in a step-by-step manner. For example, to determine an approximation to $y(t_1)$ we use the initial value $y_0 = y(t_0)$ and the known function f at (t_0, y_0). Once this is done, we use this approximation to $y(t_1)$ and the known function f at t_1 and our approximation at $y(t_1)$ to get an approximation to $y(t_2)$; we continue in this manner until we have reached the final time T.

The Euler method can be derived from several different viewpoints; initially we take the approach of replacing the derivative with an approximation but in § 1.4 we look at other approaches which will lead the way to obtaining more accurate methods. From calculus we know that the definition of the derivative of a function $y(t)$ at a point $t = a$ is

$$y'(a) = \lim_{h \to 0} \frac{y(a+h) - y(a)}{h}. \tag{1.4}$$

Graphically this just says that we take the slope of the secant line joining $\big(a, y(a)\big)$ and $\big(a+h, y(a+h)\big)$ and as $a+h$ gets closer to a the slope of the secant line gets closer to the slope of the tangent line at a which is given by $y'(a)$. If we compute

the quotient in (1.4) for a small fixed h, then we have an approximation to $y'(a)$. We know that for the limit to exist, both the limit as $h \to 0$ from the right and from the left must agree. Initially we fix $h > 0$ and in the quotient in (1.4) we set $h = \Delta t$, $a = t_0$ and let $t_1 = t_0 + \Delta t$ to get an approximation to $y'(t_0)$

$$y'(t_0) \approx \frac{y(t_1) - y(t_0)}{\Delta t}\,.$$

When we write a difference equation we need to use different notation for the exact solution $y(t)$ and its discrete approximation; to this end, we let Y_i denote our approximation to $y(t_i)$. Clearly $Y_0 = y_0$ which is the given initial condition (1.2b). To obtain Euler's method at t_1 we use the above difference approximation in the differential equation evaluated at t_0, i.e., in $y'(t_0) = f(t_0, y_0)$ to get

$$\frac{Y_1 - Y_0}{\Delta t} = f(t_0, Y_0)\,.$$

We have a starting point $Y_0 = y_0$ from our initial condition and thus we can solve for our approximation to $y(t_1)$ from

$$Y_1 = Y_0 + \Delta t f(t_0, Y_0)$$

which is a difference equation for Y_1. Once Y_1 is obtained we can repeat the process to obtain a difference equation for Y_2. This procedure is known as the forward Euler method and is defined by the following formula.

$$\text{Forward Euler:} \qquad Y_{i+1} = Y_i + \Delta t f(t_i, Y_i)\,, \quad i = 0, 1, 2, \ldots, N-1 \qquad (1.5)$$

The term "forward" is used in the name because we write the equation at the point t_i and difference forward in time to t_{i+1}; note that this implies that the given slope f is evaluated at the known point (t_i, Y_i).

A simple graphical interpretation for Euler's method is shown in Figure 1.4. To start the method, consider the slope of the tangent line at $(t_0, Y_0) = (t_0, y_0)$ which is a point which lies on the solution curve as well as on the tangent line. The tangent line has slope $y'(t_0) = f(t_0, Y_0)$; if Y_1 denotes the point on the tangent line corresponding to t_1 then the point (t_1, Y_1) satisfies the tangent line equation $Y_1 - Y_0 = f(t_0, Y_0)(t_1 - t_0)$ which is just Euler's equation for the approximation to $y(t_1)$. Now for the second step we don't have a point on the solution curve to compute the tangent line but if Δt is small, then $Y_1 \approx y(t_1)$ and $f(t_1, Y_1) \approx f(t_1, y(t_1)) = y'(t_1)$. So we write the equation passing through (t_1, Y_1) with slope $f(t_1, Y_1)$ and evaluate it at t_2 to get $Y_2 - Y_1 = \Delta t f(t_1, Y_1)$ which again is just the formula for Y_2 from (1.5).

The forward Euler method is straightforward to program and only involves a single loop and a single evaluation of $f(t, y)$ per step; in addition, only one value Y_i needs to be stored at any time because we can overwrite Y_{i-1} with Y_i. If we

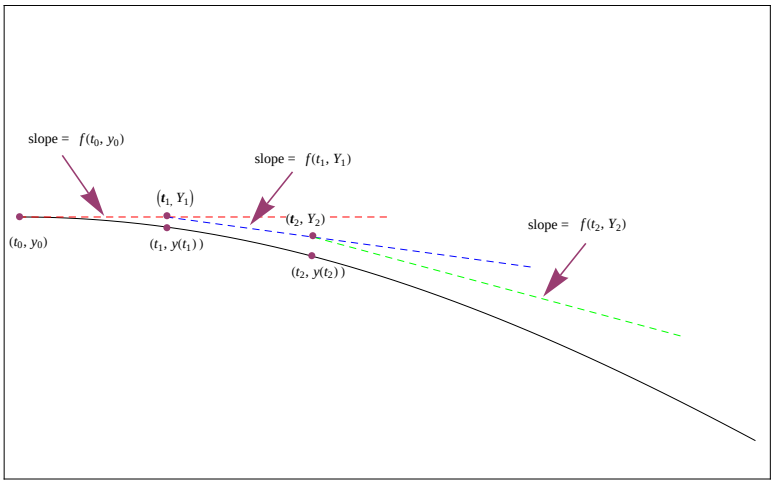


Figure 1.4: Graphical interpretation of Euler's method. At the first step Y_1 is found by writing the tangent line at (t_0, Y_0) which is on the solution curve (the solid line) and evaluating it at t_1. At the second step the slope of the tangent line to the exact solution at t_1 is not known so instead of using the exact slope $f(t_1, y(t_1))$ we use the approximation $f(t_1, Y_1)$.

need to save the values for plotting, then Y_i can simply be written to a file once it is computed. The numerical results in Figure 1.3 were obtained using the forward Euler method. In § 1.3 we will approximate the solution for several IVPs using this difference scheme.

The forward Euler scheme was derived by using the definition of the derivative at a point a where we let $h \to 0^+$, i.e., h approached zero through positive values. We now want to see if we get the same difference equation when we let $h \to 0$ through values less than zero in (1.4); in this case $a + h$ lies to the left of a, that is, we will use a secant line passing through $(a, y(a))$ and a point to its left to approximate $y'(a)$. In the quotient in (1.4) we set $h = t_0 - t_1 < 0$, $a = t_1$, and $a + h = t_0$ to get

$$y'(t_1) \approx \frac{y(t_0) - y(t_1)}{t_0 - t_1}$$

which leads to the approximation

$$\frac{Y_1 - Y_0}{\Delta t} = f(t_1, Y_1)\,,$$

where we have used the fact that $t_0 - t_1 = -\Delta t < 0$. We have the following general formula for the method.

Backward Euler: $$Y_{i+1} = Y_i + \Delta t f(t_i, Y_{i+1})\,, \quad i = 0, 1, 2, \ldots, N-1 \qquad (1.6)$$

This method is called the backward Euler method because we are writing the equation at t_{i+1} and differencing backwards in time to t_i. It is important to realize that this method is inherently different from (1.5) because we must evaluate $f(t, y)$ at the unknown point (t_i, Y_{i+1}). In general, this leads to a nonlinear equation to solve for each Y_i which can be computationally expensive. For example, if we have $f(t, y) = e^{ty}$ then the equation for Y_1 is

$$Y_1 = Y_0 + \Delta t e^{t_1 Y_1}$$

which is a nonlinear equation in the unknown Y_1.

The difference between the forward and backward Euler schemes is so important that we use this characteristic to broadly classify methods. The forward Euler scheme given in (1.5) is called an **explicit** scheme because we can write the unknown explicitly in terms of known values. The backward Euler method given in (1.6) is called an **implicit** scheme because the unknown is written implicitly in terms of known values and itself. The terms explicit/implicit are used in the same manner as when one differentiates a function which is given explicitly in terms of the independent variable (such as $y(t) = t^3 + \sec t$) or when it is given implicitly (such as $y^2 + \sin y - t^2 = 4$).

If we have an existing code for the forward Euler method and want to modify it to use the backward Euler scheme then we need to add an interior loop to solve the resulting nonlinear equation with a method such as Newton's method. If $f(t, y)$ is linear in $y(t)$ then the resulting equation will be linear and Newton's method will get the answer in one step. In § 1.3 we look at examples where we employ both the forward and backward Euler methods. However, in practice we will use implicit methods for IVPs with a different strategy because a straightforward approach leads to the necessity of solving nonlinear equations for each t_i. In the exercises you will get practice in identifying schemes as explicit or implicit.

1.2.2 Discretization errors

When we implement the forward or backward Euler method on a computer the error we make is due to both *round-off* and *discretization* error. Rounding error is due to using a machine which has finite precision. First of all, we may not be able to represent a number exactly; this is part of round-off error and is usually called representation error. Even if we use numbers which can be represented exactly on the computer, we encounter rounding errors when these numbers are manipulated such as when we divide two integers like 3 and 7. In some problems, round-off error can accumulate in such a way as to make our results meaningless; this often happens in ill-conditioned problems.

We are mainly concerned with discretization error here and when we derive error estimates we will assume that no rounding error exists. In Figure 1.5 we illustrate approximations to a known exact solution. As you can see from the plot, the approximate solution agrees with the exact solution at t_0; at t_1 there is an error in our approximation due to the fact that we have used an approximation to $y'(t)$ at t_0; i.e., we have solved a difference equation rather than the differential

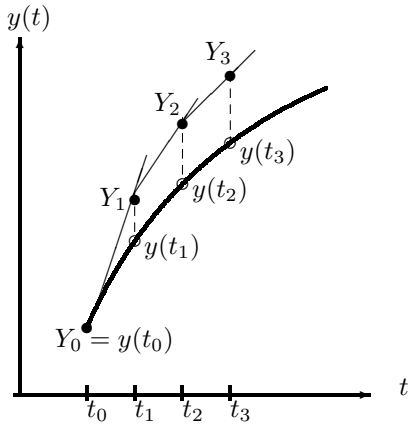


Figure 1.5: The exact solution and the discrete solution agree at t_0. At t_1 the error $|Y_1 - y(t_1)|$ is due to approximating the derivative in the ODE by a difference quotient. At t_2 the error $|Y_2 - y(t_2)|$ is due to approximating the derivative in the ODE and the fact that the starting value, Y_1, does not lie on the solution curve as Y_0 did.

equation. However at t_2 and subsequent points the discretization error comes from two sources; the first is our approximation to $y'(t)$ and the second is because we have started from the incorrect point, i.e., we did not start on the solution curve as we did in calculating Y_1. The **global discretization error** at a point t_i is the magnitude of the actual error at the point whereas the **local truncation error** or **local discretization error** in the Euler method is the error made in approximating the derivative by the difference quotient. Thus to measure the local truncation error we take one step of the discrete method assuming that the *exact* solution at t_{i-1} is used as a starting value. Ignoring round-off errors, the local truncation error is solely due to the error in approximating the differential equation.

A comparison of the global error and the truncation error for the forward Euler method is illustrated in Figure 1.6. The figure on the left demonstrates the global error at t_2 while the figure on the right illustrates the local truncation error at t_2 because the approximation uses $y(t_1)$ instead of Y_1.

To determine the local truncation error for the forward Euler method we compute the error made in taking one step where we start on the solution curve, i.e., we compute the difference in the exact solution at t_i and the result of applying the Euler method where we use $y(t_{i-1})$ instead of Y_{i-1}. Specifically we calculate

$$\tau_i = |y(t_i) - \widetilde{Y}_i| \quad \text{where } \widetilde{Y}_i = y(t_{i-1}) + \Delta t f\big(t_{i-1}, y(t_{i-1})\big)\,. \tag{1.7}$$

Our strategy is to first quantify the local truncation error τ_i in terms of Δt and

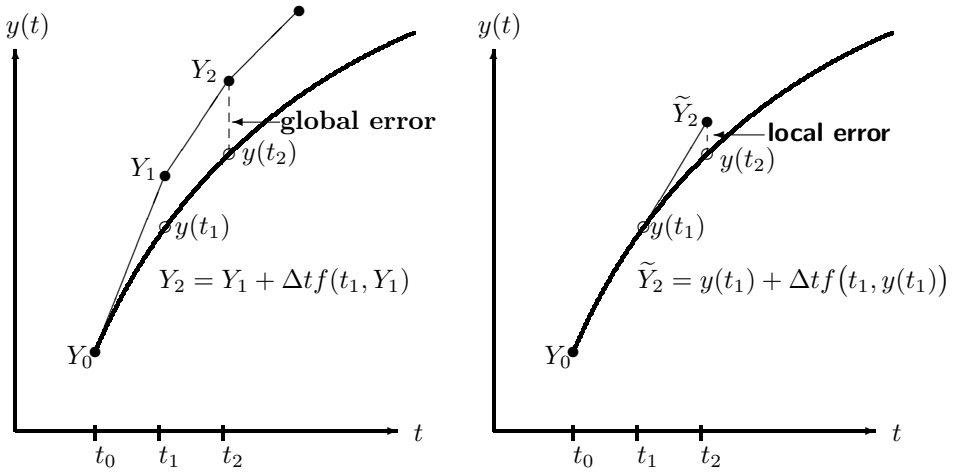


Figure 1.6: A comparison of the global error (left figure) and the local truncation error (right figure) at t_2. The global error is the total error made whereas the local truncation error is the error due to the discretization of the differential equation.

then use the result to determine the global error. The local truncation error is

$$\tau_i = \left| y(t_i) - y(t_{i-1}) - \Delta t f\big(t_{i-1}, y(t_{i-1})\big) \right| ; \tag{1.8}$$

in order to to simplify this expression we use a Taylor series with remainder for the exact solution $y(t_i) = y(t_{i-1} + \Delta t)$ because the expansion is in terms of $y(t_{i-1})$ and its derivatives at t_{i-1}. Recall that the Taylor series expansion with remainder for a differentiable function $g(x)$ in the neighborhood of $x = a$ is

$$\begin{aligned} g(x) &= g(a) + g'(a)(x-a) + \frac{g''(a)}{2!}(x-a)^2 + \frac{g'''(a)}{3!}(x-a)^3 + \cdots \\ &\quad + \frac{g^{[n]}(a)}{n!}(x-a)^n + \frac{g^{[n+1]}(\xi)}{(n+1)!}(x-a)^{n+1} \qquad \text{for } \xi \in (a,x)\,. \end{aligned}$$

Note that the result says that there *exists* a point $\xi \in (a, x)$ where the expansion holds but it does not give a means for determining the specific ξ; this makes sense because the usual Taylor series is an infinite series and if we could find the ξ that satisfies the remainder term then we would be converting an infinite series into a finite one. We will typically use Taylor series in the form

$$\begin{aligned} g(t+\Delta t) &= g(t) + \Delta t g'(t) + \frac{(\Delta t)^2}{2!} g''(t) + \frac{(\Delta t)^3}{3!} g'''(t) + \cdots \\ &\quad + \frac{(\Delta t)^n}{n!} g^{[n]}(t) + \frac{(\Delta t)^{n+1}}{(n+1)!} g^{[n+1]}(\xi) \quad \xi \in (t, t+\Delta t)\,, \end{aligned} \tag{1.9}$$

where we have set $t + \Delta t = x$ and $t = a$ so $x - a = \Delta t$. Setting $t = t_{i-1}$ and $n = 1$ in (1.9) gives the expansion for $y(t_{i-1} + \Delta t)) = y(t_i)$

$$y(t_i) = y(t_{i-1}) + \Delta t y'(t_{i-1}) + \frac{(\Delta t)^2}{2!} y''(\xi_i) \quad \xi_i \in (t_{i-1}, t_i)\,.$$

Substituting this into our expression (1.8) for the truncation error τ_i yields

$$\begin{aligned} \tau_i &= \Big[y(t_{i-1}) + \Delta t f\big(t_{i-1}, y(t_{i-1})\big) + \frac{(\Delta t)^2}{2!} y''(\xi_i) \Big] - y(t_{i-1}) - \Delta t f\big(t_{i-1}, y(t_{i-1})\big) \\ &= \frac{(\Delta t)^2}{2!} y''(\xi_i)\,, \end{aligned}$$

where we have used the differential equation at t_{i-1}, i.e., $y'(t_{i-1}) = f\big(t_{i-1}, y(t_{i-1})\big)$. If $y''(t)$ is bounded on $[0, T]$, say $|y''(t)| \le M$, then we have

$$\tau_i = \left| y(t_i) - \widetilde{Y}_i \right| = \frac{(\Delta t)^2}{2!} \left| y''(\xi_i) \right| \le (\Delta t)^2 \frac{M}{2}\,. \tag{1.10}$$

We say that the local truncation error for Euler's method is *order* $(\Delta t)^2$ which we write as $\mathcal{O}\big((\Delta t)^2\big)$. This says that the local error is proportional to the square of the step size; i.e., it is a constant times the square of the step size. Remember, however, that this is not the global error but rather the error made because we have used a finite difference quotient to approximate $y'(t)$.

We now turn to estimating the global error in the forward Euler method. We should expect to only be able to find an upper bound for the error because if we can find a formula for the exact error, then we can calculate this and add it to the approximation to get the exact solution.

Our goal is to demonstrate that the global discretization error for the forward Euler method is $\mathcal{O}(\Delta t)$ which says that the method is first order, i.e., *linear* in Δt. At each step we make a local error of $\mathcal{O}\big((\Delta t)^2\big)$ due to approximating the derivative in the differential equation; at each fixed time we have the accumulated errors of all previous steps and we want to demonstrate that this error does not exceed a constant times Δt. We can intuitively see why this should be the case. Assume that we are taking N steps of length $\Delta t = (T - t_0)/N$; at each step we make an error of order Δt^2 so for N steps we have $NC(\Delta t)^2 = [(T - t_0)/\Delta t]C\Delta t^2 = \mathcal{O}(\Delta t)$. The following result makes this argument precise. Later we will see that, in general, if the local truncation error is $\mathcal{O}((\Delta t)^r)$ then we expect the global error to be one power of Δt less, i.e., $\mathcal{O}((\Delta t)^{r-1})$.

Theorem 1.2 provides a formal statement and proof for the global error of the forward Euler method. Note that one hypothesis of Theorem 1.2 is that $f(t, y)$ must be Lipschitz continuous in y which is also the hypothesis of Theorem 1.1 which guarantees existence and uniqueness of the solution to the IVP (1.2) so it is a natural assumption. We also assume that $y(t)$ possesses a bounded second derivative; however, this condition can be relaxed but it is adequate for our needs.

Theorem 1.2. *Let $D = \{(t, y) \mid t \in [t_0, T], y \in \mathbb{R}^1\}$ and assume that $f(t, y)$ is continuous on D and is Lipschitz continuous in y on D; i.e., it satisfies (1.3) with Lipschitz constant L. Also assume that there is a constant M such that*

$$|y''(t)| \le M \quad \text{for all } t \in [t_0, T]\,.$$

Let τ_i represent the local truncation error of the forward Euler method given in (1.10). Then the global error at each point t_i satisfies

$$|y(t_i) - Y_i| \le C\Delta t \quad \text{where } C = \frac{Me^{TL}}{2L}(e^{TL} - 1)\,;$$

thus the forward Euler method converges linearly.

Proof. Let E_n represent the global discretization error at the specific time t_n, i.e., $E_n = |y(t_n) - Y_n|$. We want to demonstrate that

$$E_n \le K\, E_{n-1} + \tau \tag{1.11}$$

where K is the constant $K = 1 + \Delta t L$ and $\tau = \max_i \tau_i$, i.e., the largest value of τ_i given in (1.10) for $i = 1, 2, \ldots, N$. If we can do this, then the proof follows. To see this, first note that a common approach in error analysis is to apply a

formula recursively; in our case we obtain

$$\begin{aligned} E_n &\leq KE_{n-1} + \tau \leq K[KE_{n-2} + \tau] + \tau = K^2 E_{n-2} + (K+1)\tau \\ &\leq K^3 E_{n-3} + (K^2 + K + 1)\tau \\ &\leq \cdots \\ &\leq K^n E_0 + \tau \sum_{i=0}^{n-1} K^i \,. \end{aligned}$$

Because we assume for analysis that there are no roundoff errors, $E_0 = |y_0 - Y_0| = 0$ so we are left with $\sum_{i=0}^{n-1} \tau K^i$. To simplify the sum we note that it is a geometric series of the form $\sum_{k=0}^{n-1} ar^k$ with $a = \tau$ and $r = K$. From calculus we know that the sum is given by $a(1-r^n)/(1-r)$ so that if we use the fact that $K = 1 + \Delta tL$ we arrive at

$$E_n \leq \tau \left(\frac{K^n - 1}{K - 1} \right) = \frac{\tau}{\Delta tL} \left[(1 + \Delta tL)^n - 1 \right] .$$

Also from the Taylor series expansion of e^z near zero we have that $1 + z \leq e^z$ so that $(1+z)^n \leq e^{nz}$. If we set $z = \Delta tL$ in the last bound for E_n we have

$$E_n \leq \frac{\tau}{\Delta tL} \left[(1 + \Delta tL)^n - 1 \right] \leq \frac{\tau}{\Delta tL} (e^{n\Delta tL} - 1) \,.$$

From the hypothesis of the theorem $|y''(t)| \leq M$ so using the expression (1.10) we get

$$\tau = \max_{1 \leq i \leq N} \tau_i = \max_{1 \leq i \leq N} \frac{\Delta t^2}{2} \left| y''(\xi_i) \right| \leq M \frac{\Delta t^2}{2} \,.$$

Also n in E_n is the number of steps taken from t_0 so $n\Delta t = t_n \leq T$ where T is our final time. Combining these gives the desired result that the global error is linear in Δt

$$E_n \leq \frac{M\Delta t^2}{2\Delta tL} (e^{TL} - 1) = C\Delta t \quad \text{where } C = \frac{M}{2L} (e^{TL} - 1) \,.$$

All that is left to complete the proof is to to demonstrate that (1.11) holds with $K = 1 + \Delta tL$. To show this we substitute the expression for the local truncation error given in (1.8) and the forward Euler formula for Y_n into the expression for E_n to get

$$\begin{aligned} E_n &= \left| y(t_n) - Y_n \right| = \left| \left[y(t_{n-1}) + \Delta t y'(t_{n-1}) + \tau_n \right] \right. \\ &\quad \left. - \left[Y_{n-1} + \Delta t f(t_{n-1}, Y_{n-1}) \right] \right| \\ &\leq \left| y(t_{n-1}) - Y_{n-1} \right| + \Delta t \left| f(t_{n-1}, y(t_{n-1}) - f(t_{n-1}, Y_{n-1}) \right| + \tau \\ &\leq E_{n-1} + \Delta t \left| f(t_{n-1}, y(t_{n-1}) - f(t_{n-1}, Y_{n-1}) \right| + \tau \,, \end{aligned}$$

where we have used the differential equation $y'(t) = f(t, y)$ evaluated at the point $(t_{n-1}, y(t_{n-1}))$ in the second step. To estimate the second term on the right hand side recall that $f(t, y)$ satisfies a Lipschitz condition on y so that

$$|f\big(t_{n-1}, y(t_{n-1})\big) - f\big(t_{n-1}, Y_{n-1}\big)| \leq L\big|y(t_{n-1}) - Y_{n-1}\big|$$

and thus we have the final result

$$E_n \leq E_{n-1} + \Delta t L E_{n-1} + \tau = (1 + \Delta t L)E_{n-1} + \tau\,.$$

■

In the following section we look at some specific examples of the IVP (1.2) and use both forward and backward Euler methods; we will demonstrate that our numerical rate of convergence agrees with the theoretical rate. However, we should keep in mind that $\mathcal{O}(\Delta t)$ is a very slow rate of convergence and ultimately we need to derive methods which converge more quickly to the solution.

The backward Euler method also has a local truncation error of $\mathcal{O}\big((\Delta t)^2\big)$ and a global error of $\mathcal{O}\big(\Delta t\big)$. You are asked to demonstrate this rigorously in the exercises.

1.3 Numerical Computations

In this section we provide some numerical simulations for IVPs of the form (1.2) using both the forward and backward Euler methods. In particular, we look at two common examples which arise in modeling which we describe in § 1.3.1. Our goal is to determine if the numerical results are approaching the exact solution as Δt approaches zero and if the results are converging, determine if they are approaching the exact solution at the linear rate that we derived. In § 1.3.2 we demonstrate how to compute the numerical rate of convergence to verify this linear rate. We will see that the forward Euler method does not converge for all choices of Δt for some problems. Reasons for this failure will be discussed in Chapter 3.

When we test numerical methods for (1.2) we apply the scheme to an IVP for which we know the analytic solution. Techniques such as separation of variables or using an integrating factor can be used to do this. However, in many cases, we may not be able to solve the problem analytically; in fact, this is why we use numerical methods. An approach to finding a test problem which avoids finding the analytic solution is called the *method of manufactured solutions*. In this method we begin by choosing a function which satisfies the desired initial and/or boundary conditions and then plug it into the given differential equation to get the right hand side. For the differential equation (1.2) whose right hand side is the slope we simply differentiate the chosen function. The following examples illustrate this technique.

Example 4. METHOD OF MANUFACTURED SOLUTIONS
Use the method of manufactured solution to find an analytic solution to the IVP

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

We begin by choosing a function $y(t)$ which satisfies $y(0) = 1$; for example let $y(t) = \cos \pi t$. Then $y'(t) = -\pi \sin \pi t$ so the solution to

$$y'(t) = -\pi \sin \pi t \quad 0 \leq t \leq 1 \qquad y(0) = 1$$

is $y(t) = \cos \pi t$.

Example 5. METHOD OF MANUFACTURED SOLUTIONS
Use the method of manufactured solution to find an analytic solution to the second order IVP

$$y''(t) + 2y'(t) - y^2(t) = f(t,y), \quad 0 \le t \le 1 \qquad y(0) = 0, y'(0) = 0.5\,.$$

We begin by choosing a function $y(t)$ which satisfies the initial conditions $y(0) = 0$, $y'(0) = 0.5$; for example let $y(t) = t^3 + \frac{t}{2}$. Then $y'(t) = 3t^2 + 0.5$ and $y''(t) = 6t$ so that

$$y''(t) + 2y'(t) - y^2(t) = 6t + 2(3t^2 + 0.5) - \left(t^3 + \frac{t}{2}\right)^2 = -t^6 - t^4 + t^2(6 - \frac{1}{4}) + 6t + 1\,.$$

Thus the function $y(t) = t^3 + \frac{t}{2}$ satisfies the IVP

$$y''(t) + 2y'(t) - y^2(t) = -t^6 - t^4 + \frac{23}{4}t^2 + 6t + 1 \qquad y(0) = 0, y'(0) = 0.5\,.$$

1.3.1 Exponential and Logistic Growth Models

A common problem that arises in modeling is an IVP whose solution obeys exponential growth or decay. Exponential behavior just means that the solution can be represented by the function $Ce^{\alpha t}$ where $\alpha > 0$ for growth and $\alpha < 0$ for decay. First we look at a description of an IVP whose solution is this type of exponential growth or decay and then we look at the slightly more complicated model of logistic growth.

Exponential growth and decay. Suppose you are interested in modeling the growth of some quantity and your initial hypothesis is that the growth rate is proportional to the amount present at any time and you know the population at some initial time t_0. To write an IVP for this model we have to translate this expression into mathematical terms. We know that the derivative represents the instantaneous rate of growth and the phrase "proportional to" just means a constant times the given quantity. So if $p(t)$ represents the population at time t and p_0 represents the initial population at time $t = 0$ we express the hypothesis that the growth rate is proportional to the amount present at any time as

$$p'(t) = r_0 p(t) \quad t \in (t_0, T] \tag{1.12}$$

along with the initial condition

$$p(0) = p_0\,,$$

where r_0 is the given proportionality constant. We can solve this differential equation as we did in the first example of this chapter where we took $r_0 = -1$. We have

$$\int_0^t \frac{dp}{p} = r_0 \int_0^t dt \Rightarrow \ln p(t) - \ln p_0 = r_0(t-0) \Rightarrow e^{\ln p} = e^{r_0 t} e^{\ln p_0} \Rightarrow p(t) = p_0 e^{r_0 t}\,.$$

Thus we see that if the population at any time t is proportional to the amount present at that time, then it behaves exponentially where the initial population is a multiplicative constant and the proportionality constant r_0 is the rate of growth

if it is positive; otherwise it is the rate of decay. In the exercises you are asked to explore an exponential growth model for bread mold.

Logistic growth and decay The previous model of population growth assumes there is an endless supply of resources and no predators. Logistic growth of a population attempts to incorporate resource availability by making the assumption that the rate of population growth (i.e., the proportionality constant) is dependent on the population density. Figure 1.7 compares exponential growth and logistic growth; clearly exponential growth allows the population to grow in an unbounded manner whereas logistic growth requires the population to stay below a fixed amount K which is called the carrying capacity of the population. When the population is considerably below this threshold the two models produce similar results. The logistic model we consider restricts the growth rate in the following way

$$r = r_0 \left(1 - \frac{p}{K}\right) \tag{1.13}$$

where K is the maximum allowable population and r_0 is a given growth rate for small values of the population. As the population p increases to near the threshold value K then $\frac{p}{K}$ becomes closer to one (but less than one) and so the term $(1 - \frac{p}{K})$ gets closer to zero and the growth rate decreases because of fewer resources; the limiting value is when $p = K$ and the growth rate is zero. However when p is small compared with K, the term $(1 - \frac{p}{K})$ is near one and it behaves like exponential growth with a rate of r_0. Assuming the population at any time is proportional to the current population using the proportionality constant (1.13), our differential equation becomes

$$p'(t) = r_0 \left(1 - \frac{p(t)}{K}\right) p(t) = r_0 p(t) - \frac{r_0}{K} p^2(t)$$

along with $p(t_0) = p_0$. This equation is $nonlinear$ in the unknown $p(t)$ due to the $p^2(t)$ term and is more difficult to solve than the exponential growth equation. However, it can be shown that the solution is

$$p(t) = \frac{K p_0}{(K - p_0)e^{-r_0 t} + p_0} \tag{1.14}$$

which can be verified by substitution into the differential equation. We expect that as we take the $\lim_{t\to\infty} p(t)$ we should get the threshold value K. Clearly this is true because $\lim_{t\to\infty} e^{-r_0 t} = 0$.

1.3.2 Computing the Numerical Rate of Convergence

In Figure 1.3 we plotted several approximations to an IVP using the forward Euler method and from the plot it appeared that the discrete solution approached the exact solution as $\Delta t \to 0$. However, from Theorem 1.2 we now know that the analytic rate of convergence of the forward Euler method is first order, i.e., it is linear in Δt. Consequently, we want to verify that the numerical rate of convergence

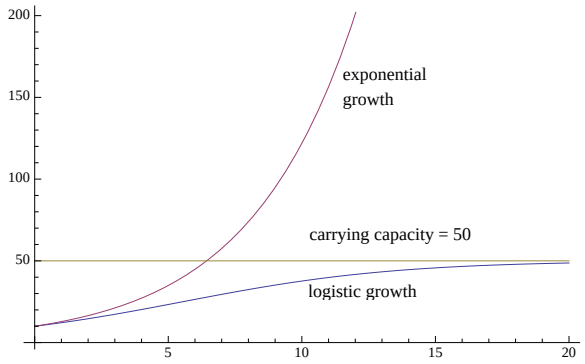


Figure 1.7: A comparison of exponential growth and logistic growth. The population in the logistic model never reaches the carrying capacity of the population which in this case is 50 while exponential growth is unbounded. For values of the population much less than the threshold value, logistic growth is very similar to exponential growth.

gets close to one as $\Delta t \to 0$. It is easy for us to tabulate the errors at a fixed time t for a sequence of time steps. If Δt is cut in half each time (for example, $\Delta t = 0.4, 0.2, 0.1, \ldots$) then we expect that the error should be approximately halved at each step because the global error is linear. To see this let E_i denote the error using time step $(\Delta t)_i$; then for linear convergence we have $E_1 \approx C(\Delta t)_1$ and $E_2 \approx C(\Delta t)_2$ and thus $E_2 \approx .5C(\Delta t)_1 \approx \frac{1}{2}E_1$ when $(\Delta t)_2 = .5(\Delta t)_1$.

To determine the actual numerical rate we assume that the numerical error using step size Δt at any point is $E = C(\Delta t)^r$ and our goal is to compute r. For the Euler method we expect to show that $r \to 1$ as $\Delta t \to 0$. Now when we use this formula at a fixed value of the step size we have two unknowns C and r. To solve for r we look at two calculations

$$E_1 = C(\Delta t)_1^r \quad \text{and} \quad E_2 = C(\Delta t)_2^r$$

and solve for r from these. We have

$$\frac{E_1}{(\Delta t)_1^r} = \frac{E_2}{(\Delta t)_2^r} \Rightarrow \frac{E_1}{E_2} = \left(\frac{(\Delta t)_1}{(\Delta t)_2}\right)^r .$$

Using properties of logarithms we get

$$r = \frac{\ln \frac{E_1}{E_2}}{\ln \frac{(\Delta t)_1}{(\Delta t)_2}} \tag{1.15}$$

and if the time step is halved each time we have

$$r = \frac{\ln \frac{E_1}{E_2}}{\ln 2} .$$

We will use this formula to calculate the numerical rate of convergence for the examples in the following section and also in subsequent chapters.

1.3.3 Numerical Examples Using Euler's Method

We now consider examples involving the exponential and logistic growth/decay models discussed in § 1.3.1. We apply both the forward and backward Euler methods for each problem and compare the results and work involved. The global error is computed at a fixed time and we expect that as Δt decreases the global error at that fixed time decreases. To confirm that the numerical approximations are valid, for each pair of successive errors we use (1.15) to calculate the numerical rate. Then we can easily see if the numerical rates approach one as $\Delta t \to 0$. In the penultimate example we compare the results from the forward and backward Euler methods for an exponential decay problem and see that in this case the forward Euler method gives numerical approximations which oscillate and grow unexpectedly whereas the backward Euler method provides reliable results. In the last example, we compare the local truncation error with the global error.

As we indicated previously, the computer implementation of the forward Euler method requires a single loop where only the current value of the solution is saved. The information which changes for each IVP is the interval $[t_0, T]$, the initial condition, the given slope and the exact solution for the error calculation. We incorporate separate functions for $f(t, y)$ and the exact solution and input the other variables as well as Δt. When we modify the code to incorporate the backward Euler scheme we must add a loop which solves the possibly nonlinear equation.

Example 6. The first example is the exponential growth problem

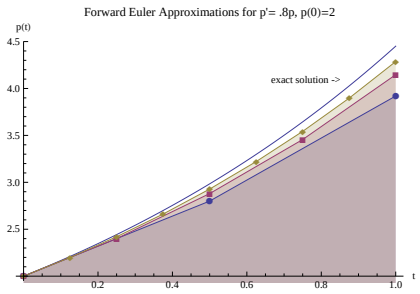
$$p'(t) = 0.8p(t) \quad 0 < t \leq 1, \quad p(0) = 2$$

whose exact solution is $p(t) = 2e^{\cdot 8t}$. The given slope in this problem is $f(t, p) = 0.8p$. The exact solution along with three Euler approximations using uniform step sizes of $\Delta t = 0.5, 0.25, 0.125$ are plotted in figure below and we can easily see that the error is decreasing as we decrease Δt. A line is drawn between points where the discrete solution is known for easier viewing but the discrete solution is only known at specific points. To verify that the global error is $\mathcal{O}(\Delta t)$ we compare the discrete solution to the exact solution at the point $t = 1$ where we know that the exact solution is $e^{\cdot 8}$=4.45108; we tabulate our approximations P_n to $p(t)$ at $t = 1$, the actual errors at this point and the computed numerical rates using (1.15) for successive errors. By looking at the errors we see that as Δt is halved the error is approximately halved so this suggests linear convergence; the calculation of the numerical rate of convergence makes this result precise because we see that the sequence $\{.825, .902, .948, .973, .986, .993\}$ is tending to one. In the table the approximations and errors at $t = 1$ are given to five digits of accuracy.

We can also demonstrate graphically that the convergence rate is linear by using a log-log plot. Recall that if we plot a polynomial $y = ax^r$ on a log-log scale then the slope is r.[3] Because we have that the error, E, is $E = C\Delta t^r$ then if we plot the error at each step size then for Euler's method we expect the slope to be one. In Figure **??** we plot the errors versus the step size from the table below.

Δt	1/2	1/4	1/8	1/16	1/32	1/64	1/128
P_n	3.9200	4.1472	4.28718	4.36575	4.40751	4.42906	4.4400
$\lvert p(1) - P_n \rvert$	0.53108	0.30388	0.16390	0.085333	0.043568	0.022017	0.011068
num. rate		0.805	0.891	0.942	0.970	0.985	0.992

[3] Using the properties of logarithms we have $\log y = \log a + r \log x$ which implies $Y = mX + b$ where $Y = \log y$, $X = \log x$, $m = r$ and $b = \log a$.



We can also demonstrate graphically that the convergence rate is linear by using a log-log plot. Recall that if we plot a polynomial $y = ax^r$ on a log-log scale then the slope is r.[4] Because we have that the error, $E = C\Delta t^r$, then if we plot the error at each step size then for Euler's method we expect the slope to be one.

If we tabulate the errors at a different time then we will get different errors but the numerical rate should still converge to one. In the table below we demonstrate this by computing the errors and rates at $t = 0.5$; note that the error is smaller at $t = 0.5$ than $t = 1$ for a given step size because we have not taken as many steps and we have less accumulated error.

Δt	$1/2$	$1/4$	$1/8$	$1/16$	$1/32$	$1/64$	$1/128$
P_n	2.8000	2.8800	2.9282	2.9549	2.9690	2.9763	2.9799
$\lvert p(1) - P_n \rvert$	0.18365	0.10365	0.055449	0.028739	0.014638	0.0073884	0.0037118
num. rate		0.825	0.902	0.948	0.973	0.986	0.993

To solve this IVP using the backward Euler method we see that for $f = 0.8p$ the equation is linear:

$$P_i = P_{i-1} + 0.8\Delta t P_i$$

where $P_i \approx p(t_i)$. Thus we do not need to use Newton's method for this particular problem but rather just solve the equation

$$P_i = \frac{1}{1 + 0.8\Delta t} P_{i-1} \, .$$

The results are tabulated below. Note that the numerical rate of convergence is also approaching one but for this method it is approaching one from above whereas using the forward Euler scheme for this problem the convergence was from below, i.e., through values smaller than one. The amount of work required for the backward Euler method is essentially the same as the forward Euler for this problem because the derivative $f(t, p)$ was linear in the unknown p.

Δt	$1/2$	$1/4$	$1/8$	$1/16$	$1/32$	$1/64$	$1/128$
P_n	5.5556	4.8828	4.6461	4.5441	4.4966	4.4736	4.4623
$\lvert p(1) - P_n \rvert$	1.1045	0.43173	0.19503	0.093065	0.045498	0.022499	0.011188
num. rate		1.355	1.146	1.067	1.032	1.015	1.008

[4] Using the properties of logarithms we have $\log y = \log a + r \log x$ which implies $Y = mX + b$ where $Y = \log y$, $X = \log x$, $m = r$ and $b = \log a$.

Example 7. We now want to approximate the solution to the logistic model

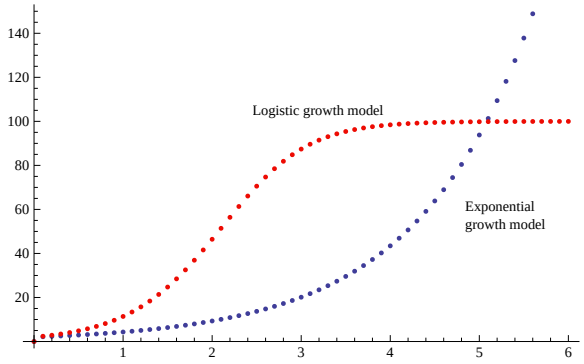
$$p'(t) = 2\left(1 - \frac{p(t)}{100}\right)p(t) \quad 0 < t \leq 6 \quad p(0) = 2$$

using both the forward and backward Euler schemes and demonstrate that we get linear convergence. Also we want to compare the results from this example with those from the previous example of exponential growth.

To obtain approximations to $p(t)$ using the forward Euler method code we used in the previous example all we have to do is modify the routines defining $f(t, p)$ and the exact solution for the error calculation; the initial condition p_0 is the same. The exact solution to this problem is given by (1.14). Before generating any simulations we should think about what we expect the behavior of this solution to be compared with the exponential growth solution in the previous example. Initially the population should grow faster because here $r_0 = 2$ and in the previous example the growth rate is 0.8. However, the solution should not grow unbounded but rather always stay below the carrying capacity $p = 100$. The approximations at $t = 1$ for a sequence of decreasing values of Δt are presented below along with the calculated numerical rates. The exact value at $t = 1$ is 13.1037059. Again we see that the numerical rate approaches one.

Δt	$1/8$	$1/16$	$1/32$	$1/64$	$1/128$	$1/256$
P_n	11.0346	11.9695	12.5084	12.7985	12.94917	13.0259
$\lvert p(1) - P_n \rvert$	2.0691	1.1343	0.59535	0.30519	0.15454	0.077620
num. rate		0.867	0.930	0.964	0.982	0.991

Below we plot the approximate solution on $[0, 6]$ for this logistic growth problem and the previous exponential growth problem. Note that the exponential growth solution increases without bound whereas the logistic growth solution never exceeds the carrying capacity of $K = 100$.



If we compare the size of the errors in these two examples at say $\Delta t = 1/64$ we see that the errors for the logistic model are about a factor of ten larger than for the exponential problem. What makes one error so much larger than the other even though they both are converging linearly? When we say that the convergence is linear we mean that it is a constant times Δt where for the Euler method the constant depends on the second derivative of the exact solution. The exact solution $p(t) = 2e^{.8t}$ to the exponential growth function has a second derivative of $1.28e^{.8t}$ so it is increasing on $[0, 1]$ and has a maximum value of 2.849 whereas the exact solution to the logistic equation has an increasing second derivative (it's a bit messy to include here) on $[0, 1]$ but its maximum value is around 34 so this accounts for the difference.

We now turn to implementing the backward Euler scheme for this problem. At each step we have the nonlinear problem

$$P_i = P_{i-1} + 2\Delta t\left(P_i - \frac{P_i^2}{100}\right)$$

for P_i. Thus to determine each P_i we have to employ a method such as Newton's method. To find the root z of the nonlinear equation $g(z) = 0$ (a function of one independent variable) each iteration of Newton's method is given by

$$z^k = z^{k-1} - \frac{g(z^{k-1})}{g'(z^{k-1})}$$

for $k = 1, 2, \ldots$ where an initial guess z^0 is prescribed. For us, we have the nonlinear equation

$$g(z) = z - P_{i-1} - 2\Delta t \left(z - \frac{z^2}{100} \right) = 0$$

where $z = P_i$ plays the role of the independent variable; our goal is to approximate the value of z which makes $g(z) = 0$ and this will be our approximation P_i. For an initial guess z^0 we simply take P_{i-1} because if Δt is small enough and the solution is smooth then the approximation at t_{i-1} will be close to the solution at t_i. To implement Newton's method we also need the derivative g' which for us is just

$$g'(z) = 1 - 2\Delta t \left(1 - \frac{z}{50} \right) .$$

The results using backward Euler are tabulated below; note that the numerical rates of convergence approach one as $\Delta t \to 0$. We have imposed the convergence criteria for Newton's method that the normalized difference in successive iterates is less than a prescribed tolerance, i.e.,

$$\frac{|z^k - z^{k-1}|}{|z^k|} \leq 10^{-8} .$$

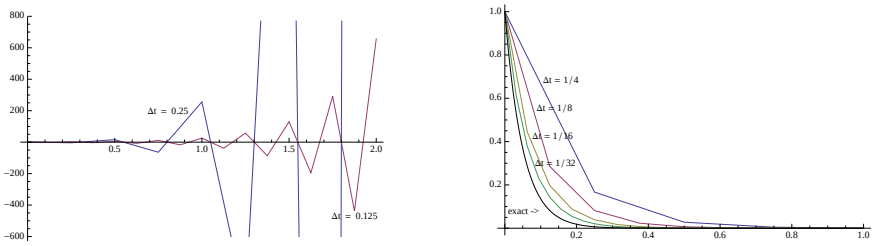
Typically, three to four Newton iterations were typically required to satisfy this convergence criteria. Recall that Newton's method typically converges quadratically (when it converges) and so the convergence is rapid.

Δt	$1/8$	$1/16$	$1/32$	$1/64$	$1/128$	$1/256$
P_n	16.2227	14.4961	13.7633	13.4249	13.2622	13.1825
$\lvert p(1) - P_n \rvert$	3.1190	1.3924	0.65961	0.32124	0.15855	0.078765
num. rate		1.164	1.0779	1.038	1.019	1.009

Example 8. In this example we consider exponential decay where the decay rate is large. Specifically, we seek $y(t)$ such that

$$y'(t) = -20y(t) \quad 0 < t \leq 2, \quad y(0) = 1$$

which has an exact solution of $y(t) = e^{-20t}$. In the plot on the left of the figure below we plot the approximate solutions on $[0, 2]$ using the forward Euler method with $\Delta t = \frac{1}{4}$ and $\frac{1}{8}$. Note that for this problem the approximate solution is oscillating and becoming unbounded. In the plot on the right in the same figure we plot approximations using the backward Euler method and the exact solution. As can be seen from the plot, it appears that the discrete solution is approaching the exact solution as $\Delta t \to 0$. Recall that the backward Euler method is an implicit scheme whereas the forward Euler method is an explicit scheme.



Why are the results for the forward Euler method not reliable for this problem whereas they were for previous examples? In this example the numerical approximations are not converging as $\Delta t \to 0$; the reason for this is a stability issue which we address in Chapter 3. When we determined the theoretical rate of convergence we tacitly assumed that the method converged; which of course in this method it does not. The implicit backward Euler method provided convergent results but remember that, in general, we have to solve a nonlinear equation at each time; in § 2.6 we will investigate efficient approaches to implementing an implicit method for an IVP.

Example 9. We consider the IVP

$$y'(t) = \cos(t)e^{\sin t} \quad 0 < t \leq \pi \qquad y(0) = 0$$

whose exact solution is $e^{\sin t}$. The goal of this example is to demonstrate that the local truncation error for the forward Euler method is second order, i.e., $\mathcal{O}(\Delta t^2)$ and to compare the local and global errors at a fixed time.

The local truncation error at t_n is computed from the formula

$$|y(t_n) - \widetilde{Y}_n| \quad \text{where } \widetilde{Y}_n = y(t_{n-1}) + \Delta t f(t_{n-1}, y(t_{n-1}))$$

that is, we use the correct value $y(t_{n-1})$ instead of Y_{n-1} and evaluate the slope at the point $(t_{n-1}, y(t_{n-1}))$ which is on the solution curve. In the table below we have tabulated the local and global errors at $t = \pi$ using decreasing values of Δt and from the numerical rates of convergence you can clearly see that the local truncation error is $\mathcal{O}(\Delta t^2)$, as we demonstrated analytically. As expected, the global error converges linearly. Except at the first step (where the local and global errors are identical) the global error is always larger than the truncation error because it includes the accumulated errors as well as the error made by approximating the derivative by a difference quotient.

Δt	$1/8$	$1/16$	$1/32$	$1/64$	$1/128$
local error	$7.713\ 10^{-3}$	$1.947\ 10^{-3}$	$4.879\ 10^{-4}$	$1.220\ 10^{-4}$	$3.052\ 10^{-5}$
num. rate		1.99	2.00	2.00	2.00
global error	$2.835\ 10^{-2}$	$1.391\ 10^{-2}$	$6.854\ 10^{-3}$	$3.366\ 10^{-3}$	$1.681\ 10^{-3}$
num. rate		1.03	1.02	1.02	1.00

One different aspect about this problem is that in previous examples the stopping point was an integer multiple of Δt whereas here the interval is $[0, \pi]$. The computer implementation of the method must check if the time the solution is being approximated at is greater than the final time. If it is, then Δt must be reset to the difference in the final time and the previous time so that the last step is taken with a smaller time step; otherwise you will be comparing the solution at different final times.

1.4 Alternate Approaches for Deriving Methods

We obtained both the forward and backward Euler methods by looking at the definition of the derivative and realizing that we could use the slope of a secant line to approximate the derivative. However, it is not obvious how to extend this idea to derive higher order methods. In this section we illustrate several alternate approaches for deriving the forward Euler method and these will be helpful in the next chapter when we look at higher order methods. The approaches we consider are not exhaustive but they represent the major ones. In particular, we consider methods derived using Taylor series, numerical quadrature formulas, and interpolating polynomials. In Chapter 2 we will investigate these approaches in more detail.

The first approach we consider is using a Taylor series expansion for $y(t_i + \Delta t)$ when we know $y(t_i)$. From (1.9) we have

$$y(t_i + \Delta t) = y(t_i) + \Delta t y'(t_i) + \frac{(\Delta t)^2}{2!} y''(t_i) + \cdots + \frac{(\Delta t)^k}{k!} y^{[k]}(t_i) + \cdots .$$

This is an infinite series so if we truncate it then we have an approximation to $y(t_i + \Delta t)$. For example, if we truncation the series at the term which is $\mathcal{O}(\Delta t)$ we have

$$y(t_i + \Delta t) \approx y(t_i) + \Delta t y'(t_i) = y(t_i) + \Delta t f(t_i, y(t_i))$$

which leads to our difference equation for the forward Euler Method $Y_{i+1} = Y_i + \Delta t f(t_i, Y_i)$. So theoretically, if we keep additional terms in the series then we get a higher order approximation to $y'(t)$.

To derive the implicit backward Euler method using Taylor series we use the expansion for $y(t_i - \Delta t)$

$$y(t_i - \Delta t) = y(t_i) - \Delta t y'(t_i) + \frac{(\Delta t)^2}{2!} y''(t_i) + \cdots + (-1)^k \frac{(\Delta t)^k}{k!} y^{[k]}(t_i) + \cdots .$$

and truncating the series yields

$$y(t_i) \approx y(t_i - \Delta t) + \Delta t y'(t_i) = y(t_{i-1}) + \Delta t f(t_i, y(t_i)$$

which gives the difference equation $Y_i = Y_{i-1} + \Delta t f(t_i, Y_i)$.

We explore using Taylor series to derive higher order methods in § 2.1. One of the drawbacks of this approach is that as we add more terms we need higher order derivatives of $y(t)$ or equivalently of $f(t, y)$. These may be tedious to compute or not smooth. For this reason this approach is rarely used although it is probably the most obvious strategy.

Another approach to derive the Euler method is to use numerical integration rules to approximate the integrals obtained when we integrate (1.2a) from t_i to t_{i+1}. Formally we have

$$\int_{t_i}^{t_{i+1}} y'(t)\, dt = \int_{t_i}^{t_{i+1}} f(t, y)\, dt$$

and we note that the left hand side can be evaluated exactly by the Fundamental Theorem of Calculus to get $y(t_{i+1}) - y(t_i)$; however, in general, we must use numerical quadrature to approximate the integral on the right hand side. Recall from calculus that one of the simplest approximations to an integral is to use either a left or right Riemann sum. If we use a left sum for the integral we approximate the integral by a rectangle whose base is Δt and whose height is determined by the function at the left endpoint of the interval; i.e., we use the formula

$$\int_a^b g(x) \approx g(a)(b - a)\,.$$

Using the left Riemann sum to approximate the integral of $f(t, y)$ gives

$$y(t_{i+1}) - y(t_i) = \int_{t_i}^{t_{i+1}} f(t, y)\, dt \approx \Delta t f(t_i, y(t_i))$$

which once again leads us to the difference equation for the forward Euler method. In the exercises you will explore the implications of using a left Riemann sum. Clearly different approximations to the integral of $f(t, y)$ yield different methods; we consider these in § 2.2.

Still another approach to derive the Euler method is to use interpolation. There are basically two choices for how one can use an interpolating polynomial to derive other schemes. One choice is to use an interpolating polynomial for $y(t)$ through grid points such as t_{i+1}, t_i, t_{i-1}, etc., then differentiate it (this is easy because it's a polynomial) and substitute the derivative for $y'(t)$. Another option is to use an interpolating polynomial for $f(t, y)$ and then integrate the differential equation as we did above; the integral of $f(t, y)$ is now trivial to integrate because f is approximated by a polynomial. Both approaches yield difference equations to approximate the solution to $y'(t) = f(t, y)$.

We first look at the approach of representing $y(t)$ by an interpolating polynomial. The Lagrange form of the unique linear polynomial that passes through the points $\big(t_i, y(t_i)\big)$ and $\big(t_{i+1}, y(t_{i+1})\big)$ is

$$p_1(t) = y(t_i)\frac{t - t_{i+1}}{-\Delta t} + y(t_{i+1})\frac{t - t_i}{\Delta t}\,.$$

If we use this linear interpolant to represent an approximation to $y(t)$ at any point between t_i and t_{i+1} then differentiating with respect to t gives

$$p_1'(t) = \frac{-1}{\Delta t} y(t_i) + \frac{1}{\Delta t} y(t_{i+1})$$

which leads to the approximation

$$y'(t) \approx \frac{y(t_{i+1}) - y(t_i)}{\Delta t}\,.$$

Using this expression for $y'(t)$ in the differential equation $y'(t) = f(t, y)$ at t_i yields the forward Euler Method and at t_{i+1} gives the implicit backward Euler method.

The second choice for deriving schemes using an interpolating polynomial is to use an interpolating polynomial for $f(t, y)$. For example, suppose we approximate $f(t, y)$ by a polynomial of degree zero, i.e., a constant in the interval $[t_i, t_{i+1}]$. If use the approximation $f(t, y) \approx f\big(t_i, y(t_i)\big)$ in $[t_i, t_{i+1}]$ then integrating the differential equation yields

$$y(t_{i+1}) - y(t_i) = \int_{t_i}^{t_{i+1}} f(t, y)\, dt \approx \int_{t_i}^{t_{i+1}} f\big(t_i, y(t_i)\big)\, dt = f\big(t_i, y(t_i)\big)\Delta t$$

which leads to the forward Euler method. If we choose to approximate $f(t, y)$ in $[t_i, t_{i+1}]$ by $f(t_{i+1}, y(t_{i+1}))$ then we get the backward Euler method. We will investigate methods derived using interpolating polynomials in § 2.3.

EXERCISES

1. Classify each difference equation as explicit or implicit. Justify your answer.

   a. $Y_{i+1} = Y_{i-1} + 2\Delta t f(t_i, Y_i)$

   b. $Y_{i+1} = Y_{i-1} + \frac{\Delta t}{3}\big[f(t_{i+1}, Y_{i+1}) + 4f(t_i, Y_i) + f(t_{i-1}, Y_{i-1})\big]$

   c. $Y_{i+1} = Y_i + \frac{\Delta t}{2}\big[f\big(t_i, Y_i + \frac{\Delta t}{2} f(t_i, Y_i) - \frac{\Delta t}{2} f(t_i, Y_{i+1})\big)\big] + \frac{\Delta t}{2}\big[f\big(t_{i+1}, Y_i + \frac{\Delta t}{2} f(t_{i+1}, Y_{i+1})\big)\big]$

   d. $Y_{i+1} = Y_i + \frac{\Delta t}{4}\big[f(t_i, Y_i) + 3f\big(t_i + \frac{2}{3}\Delta t, Y_i + \frac{2}{3}\Delta t f(t_i + \frac{\Delta t}{3}, Y_i + \frac{\Delta t}{3} f(t_i, Y_i)\big)\big]$

2. Assume that the following set of errors were obtained from three different methods for approximating the solution of an IVP of the form (1.2) at a specific time. First look at the errors and try to decide the accuracy of the method. Then use the result (1.15) to determine a sequence of approximate numerical rates for each method using successive pairs of errors. Use these results to state whether the accuracy of the method is linear, quadratic, cubic or quartic.

Δt	Errors Method I	Errors Method II	Errors Method III
$1/4$	0.23426×10^{-2}	0.27688	0.71889×10^{-5}
$1/8$	0.64406×10^{-3}	0.15249	0.49840×10^{-6}
$1/16$	0.16883×10^{-3}	0.80353×10^{-1}	0.32812×10^{-7}
$1/32$	0.43215×10^{-4}	0.41292×10^{-1}	0.21048×10^{-8}

3. Show that if we integrate the IVP (1.2a) from t_i to t_{i+1} and use a right Riemann sum to approximate the integral of $f(t, y)$ then we obtain the backward Euler method.

4. We showed that if we use a linear interpolating polynomial to approximate $y(t)$ on $[t_i, t_{i+1}]$ then we obtain the Euler method. What happens if you use a constant polynomial on $[t_i, t_{i+1}]$ which interpolates $y(t_i)$?

5. Write a code which implements the forward Euler method to solve an IVP of the form (1.2). Use your code to approximate the solution of the IVP

$$y'(t) = 1 - y^2 \quad y(0) = 0$$

   which has an exact solution $y(t) = (e^{2t} - 1)/(e^{2x} + 1)$. Compute the errors at $t = 1$ using $\Delta t = 1/4, 1/8, 1/16, 1/32, 1/64$.

a. Tabulate the *global error* at $t = 1$ for each value of Δt and demonstrate that your method converges with accuracy $\mathcal{O}(\Delta t)$; justify your answer by calculating the numerical rate of convergence for successive pairs of errors.

b. Tabulate the *local error* at $t = 1$ for each value of Δt and determine the rate of convergence of the local error; justify your answer by calculating the numerical rate of convergence for successive pairs of errors. Compare your results with those obtained in (a).

6. Suppose you are interested in modeling the growth of the Bread Mold Fungus, *Rhizopus stolonifer* and comparing your numerical results to experimental data that is taken by measuring the number of square inches of mold on a slice of bread over a period of several days. Assume that the slice of bread is a square of side 5 inches.

   a. To obtain a model describing the growth of the mold you first make the hypothesis that the growth rate of the fungus is proportional to the amount of mold present at any time with a proportionality constant of k. Assume that the initial amount of mold present is 0.25 square inches. Let $p(t)$ denote the number of square inches of mold present on day t. Write an initial value problem for the growth of the mold.

   b. Assume that the following data is collected over a period of ten days. Assuming that k is a constant, use the data at day one to determine k. Then using the forward Euler method with Δt a fourth and an eight of a day, obtain numerical estimates for each day of the ten day period; tabulate your results and compare with the experimental data. When do the results become physically unreasonable?

$t = 0$	p =0.25	$t = 1$	p =0.55
$t = 2$	p =1.1	$t = 3$	p =2.25
$t = 5$	p =7.5	$t = 7$	p =16.25
$t = 8$	p =19.5	$t = 10$	p =22.75

   c. The difficulty with the exponential growth model is that the bread model grows in an unbounded way as you saw in (b). To improve the model for the growth of bread mold, we want to incorporate the fact that the number of square inches of mold can't exceed the number of square inches in a slice of bread. Write a logistic differential equation which models this growth using the same initial condition and growth rate as before.

   d. Use the forward Euler method with Δt a fourth and an eighth of a day to obtain numerical estimates for the amount of mold present on each of the ten days using your logistic model. Tabulate your results as in (b) and compare your results to those from the exponential growth model.

Fix tables for math font ADD problem about truncation error for backward Euler.

Chapter 2

Higher order accurate methods

In the last chapter we looked at the forward and backward Euler method for approximating the solution to the first order IVP (1.2). Although it is simple to understand and program, the method converges at a linear rate which is quite slow. In addition, we saw an example in which the forward Euler failed to converge as $\Delta t \to 0$. For these reasons, it is worthwhile to investigate schemes with a higher order of accuracy and which have better convergence properties. Also, not all problems can be solved efficiently with a uniform time step so we would like to develop methods which allow us to determine a reasonable choice of Δt at each time step.

In § 1.4 we demonstrated that the Euler method can be derived from several different viewpoints. In particular we used Taylor series expansions, quadrature rules, and interpolating polynomials to obtain the forward and backward Euler methods. We investigate these approaches in more detail in this chapter. We will see that using Taylor series expansions is a straightforward approach to deriving higher order schemes but it requires the repeated differentiation of $f(t, y)$ which makes the methods impractical. Integrating the differential equation (1.2a) requires approximating the integral $\int_{t_i}^{t_{i+1}} f(t, y)\, dt$ which can be done by using a quadrature rule. This leads to families of methods called Runge-Kutta methods. Another approach to approximating this integral is to use an interpolating polynomial for $f(t, y)$ so that the resulting integral can be determined exactly. This approximation leads to families of methods called multistep methods. Still another approach using an interpolating polynomial to derive methods is to approximate $y(t)$ and then differentiate the interpolating polynomial to get a difference equation.

In Chapter 1 we saw that implicit methods were inherently more costly to implement than explicit methods due to the fact that they typically require the solution of a nonlinear equation at each time step. In this chapter we will investigate an efficient way to implement an implicit method by pairing it with an explicit method to yield the so-called Predictor-Corrector methods.

In the last chapter we demonstrated that the forward Euler method has a local truncation error of $\mathcal{O}(\Delta t^2)$ and a global error of $\mathcal{O}(\Delta t)$, i.e., the global error is one

order of Δt less than the truncation error. For difference schemes it is straightforward, but sometimes tedious, to compute the local truncation error because it just involves Taylor series expansions. However, the global error is much more difficult to determine. For the majority of convergent difference schemes for (1.2) the global error follows the same pattern as Euler's method, that is, the global error is one power of Δt less than the local truncation error. We will assume this is the case for the methods we derive in this chapter. In the next chapter we will discuss the properties of a scheme which guarantee that it is convergent.

2.1 Taylor Series Methods

Taylor Series is an extremely useful tool in numerical analysis, especially in deriving and analyzing difference methods. Previously we saw that it can be used to derive the Euler method by dropping all terms of $\mathcal{O}(\Delta t^2)$ and higher; thus a natural approach to obtaining higher order methods is to retain more terms in the expansion. To see how this approach works, we now keep three terms in the expansion and have a remainder term of $\mathcal{O}(\Delta t^3)$. From (1.9) we have

$$y(t_i + \Delta t) = y(t_i) + \Delta t y'(t_i) + \frac{(\Delta t)^2}{2!} y''(t_i) + \frac{(\Delta t)^3}{3!} y'''(\xi_i)\,,$$

so we expect a local error of $\mathcal{O}(\Delta t^3)$ which leads to an expected global error of $\mathcal{O}(\Delta t^2)$. The expression for $y'(t_i)$ is

$$y'(t_i) = \frac{y(t_i + \Delta t) - y(t_i)}{\Delta t} - \frac{\Delta t}{2!} y''(t_i) + \frac{(\Delta t)^2}{3!} y'''(\xi_i)\,.$$

Now the problem we have to address when we use this expansion is what to do with $y''(t_i)$ because we only know $y'(t) = f(t, y)$. If our function is smooth enough, we can differentiate this equation with respect to t to get $y''(t)$. To do this recall that we have to use the chain rule because f is a function of t and y where y is also a function t. Specifically, we have

$$y'(t) = f(t, y) \Rightarrow y''(t) = \frac{\partial f}{\partial t}\frac{dt}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = f_t + f_y f\,.$$

Substituting this into the expression for $y'(t_i)$ gives the approximation

$$y'(t_i) \approx \frac{y(t_i + \Delta t) - y(t_i)}{\Delta t} - \frac{\Delta t}{2!}\left[f_t\big(t_i, y(t_i)\big) + f\big(t_i, y(t_i)\big) f_y\big(t_i, y(t_i)\big)\right]$$

which leads to the difference equation

$$\frac{Y_{i+1} - Y_i}{\Delta t} - \frac{\Delta t}{2}\left[f_t(t_i, Y_i) + f(t_i, Y_i) f_y(t_i, Y_i)\right] = f(t_i, Y_i)\,.$$

Second order Taylor series method

$$Y_{i+1} = Y_i + \Delta t f(t_i, Y_i) + \frac{(\Delta t)^2}{2}\left[f_t(t_i, Y_i) + f(t_i, Y_i) f_y(t_i, Y_i)\right] \tag{2.1}$$

If we neglect the last terms on the right-hand side of this method which are $\mathcal{O}((\Delta t)^2)$ then we just have the forward Euler so we can view these terms as corrections to the first order Euler method.

To implement this method, we must provide function routines not only for $f(t, y)$ but also $f_t(t, y)$ and $f_y(t, y)$. In some cases this will be easy, but in others it can be tedious or even not possible. For these reasons, higher order Taylor series are not often used in practice; in the sequel we will discuss other higher order methods which are much more tractable. The following example applies the second order Taylor scheme to a specific IVP and in the exercises we explore a third order Taylor series method.

Although using Taylor series results in methods with higher order accuracy than the Euler method, they are not considered practical because of the requirement of repeated differentiation of $f(t, y)$. For example, the first full derivative has two terms and the second has five terms. So even if $f(t, y)$ can be differentiated, the methods become unwieldy. For this reason we look at other approaches to derive higher order schemes.

Example 1. SECOND ORDER TAYLOR METHOD

We want to approximate the solution to

$$y'(t) = 3yt^2 \qquad y(0) = \frac{1}{3}$$

whose exact solution is $\frac{1}{3}e^{t^3}$ using (2.1). Then we want to verify numerically that its global error convergences quadratically and compare the results with the first order forward Euler method.

Before writing a code for a particular method, it is helpful to first perform some calculations by hand so it is clear that the method is completely understood and also to have some results with which to compare the numerical simulations. To this end, we first calculate Y_1 and Y_2 using $\Delta t = 0.1$. Then we provide numerical results at $t = 1$ for several choices of Δt and compare with a first order Taylor series method, i.e., with the forward Euler method.

From the discussion in this section, we know that we need f_t and f_y so

$$f(t, y) = 3yt^2 \Rightarrow f_t = 6ty \quad \text{and} \quad f_y = 3t^2\,.$$

Substitution into the difference equation (2.1) gives the expression

$$Y_{i+1} = Y_i + 3\Delta t Y_i t_i^2 + \frac{(\Delta t)^2}{2}\left(6t_i Y_i + 9t_i^4 Y_i\right)\,. \tag{2.2}$$

For $Y_0 = 1/3$ we have

$$Y_1 = \frac{1}{3} + 0.1(3)\left(\frac{1}{3}\right)0 + \frac{(.1)^2}{2}(0) = \frac{1}{3}$$

$$Y_2 = \frac{1}{3} + 0.1(3)\left(\frac{1}{3}\right)(.1)^2 + \frac{(.1)^2}{2}\left(6(.1)\frac{1}{3} + 9(.1)^4\frac{1}{3}\right) = 0.335335\,.$$

The exact solution at $t = 0.2$ is 0.336011 which gives an error of $0.675862\ 10^{-3}$.

To implement this method in a computer code we modify our program for the forward Euler method to include the $\mathcal{O}(\Delta t^2)$ terms in (2.1). In addition to a function for $f(t, y)$ we also need to provide function routines for its first partial derivatives f_y and f_t; note that in our program we code the general equation (2.1), not the equation (2.2) specific to our problem. We perform calculations with decreasing values of Δt and compare with results at $t = 1$ using the forward Euler method. When we compute the numerical rate of convergence we see that the rate of convergence is $\mathcal{O}(\Delta t^2)$, as expected whereas the forward Euler is only linear. At each step the error in the second order Taylor is much smaller than the corresponding error in the forward Euler method.

Δt	Error in Euler	Numerical rate	Error in second order Taylor	Numerical rate
$1/4$	$3.1689\ 10^{-1}$		$1.2328\ 10^{-1}$	
$1/8$	$2.0007\ 10^{-1}$	0.663	$4.1143\ 10^{-2}$	1.58
$1/16$	$1.1521\ 10^{-1}$	0.796	$1.1932\ 10^{-2}$	1.79
$1/32$	$6.2350\ 10^{-2}$	0.886	$3.2091\ 10^{-3}$	1.89
$1/64$	$3.2516\ 10^{-2}$	0.939	$8.3150\ 10^{-4}$	1.95
$1/128$	$1.6615\ 10^{-2}$	0.969	$2.1157\ 10^{-4}$	1.97

2.2 Methods from Integration Formulas

Another approach we used to obtain the Euler method in § 1.4 was to integrate the differential equation with respect to t from t_i to t_{i+1} and use the Fundamental Theorem of Calculus to evaluate the left-hand side of the equation and a numerical quadrature rule to evaluate the right hand side. We saw that a choice of the left Riemann sum resulted in the forward Euler method and a choice of the right Riemann sum resulted in the backward Euler method. Clearly there are many other choices for quadrature rules. Recall that numerical quadrature rules for single integrals have the general form

$$\int_a^b g(t)\ dt \approx \sum_{i=1}^{Q} w_i g(q_i)$$

where the scalars w_i are called the quadrature weights and the points q_i are the quadrature points in $[a, b]$.

One common numerical integration rule is the midpoint rule where, as the name indicates, we evaluate the integrand at the midpoint of the interval; specifically the midpoint quadrature rule is

$$\int_a^b g(t)\ dt \approx (b-a) g\left(\frac{a+b}{2}\right).$$

Integrating the differential equation (1.2a) from t_i to t_{i+1} and using the midpoint quadrature rule to integrate $f(t, y)$ over the domain gives

$$y(t_{i+1}) - y(t_i) \approx \Delta t f\big(t_i + \frac{\Delta t}{2}, y(t_i + \frac{\Delta t}{2})\big).$$

The problem with this approximation is that we don't know y evaluated at the midpoint so our only recourse is to use an approximation. If we use forward Euler

starting at t_i and take a step of length $\Delta t/2$ then this produces an approximation to y at the midpoint i.e.,

$$y(t_i + \frac{\Delta t}{2}) \approx y(t_i) + \frac{\Delta t}{2} f\big(t_i, y(t_i)\big)\,.$$

Thus we can view our method as having two parts; first we approximate y at the midpoint using Euler's method and then use it to approximate $y(t_{i+1})$. Combining these into one equation allows the scheme to be written as

$$Y_{i+1} = Y_i + \Delta t f\big(t_i + \frac{\Delta t}{2}, Y_i + \frac{1}{2}\Delta t f(t_i, Y_i)\big)\,.$$

However, the method is usually written in the following way for simplicity and to emphasize the fact that there are two function evaluations required.

Midpoint Rule

$$\begin{aligned} k_1 &= \Delta t f(t_i, Y_i) \\ k_2 &= \Delta t f(t_i + \tfrac{\Delta t}{2}, Y_i + \tfrac{1}{2}k_1) \\ Y_{i+1} &= Y_i + k_2 \end{aligned} \tag{2.3}$$

Computationally, we see that we have to do extra work compared with the Euler method because we have to approximate the intermediate value $y(t_i + \Delta t/2)$; the extra work required is an additional function evaluation $f(t_i + \frac{\Delta t}{2}, Y_i + \frac{1}{2}k_1)$. Because we are doing more work than the Euler method, we would like to think that the scheme converges faster.

We now demonstrate that the local truncation error of the Midpoint method is $\mathcal{O}(\Delta t^3)$ so that we expect the method to converge with a global error of $\mathcal{O}(\Delta t^2)$. The steps in estimating the local truncation error for the Midpoint method are analogous to the ones we performed for determining the local truncation error for the Euler Method except now we will need to use a Taylor series expansion in two independent variables for $f(t, y)$ because of the term $f(t_i + \frac{\Delta t}{2}, Y_i + \frac{1}{2}k_1)$. One way to arrive at a Taylor series expansion for a function for two independent variables is to first hold one variable fixed in (1.9) and expand in the other and then repeat the procedure for all terms. For completeness we give Taylor series in two independent variables in the following proposition. Note that in the result we assume that the function is continuously differentiable so that the order of differentiation does not matter; e.g., $f_{xy} = f_{yx}$.

Proposition 2.1. *Let $f(x,y)$ be continuously differentiable. Then*

$$\begin{aligned} f(x+h,y+k) &= f(x,y) + hf_x(x,y) + kf_y(x,y) \\ &\quad + \frac{h^2}{2!}f_{xx}(x,y) + \frac{k^2}{2!}f_{yy}(x,y) + 2\frac{kh}{2!}f_{xy}(x,y) \\ &\quad + \frac{h^3}{3!}f_{xxx}(x,y) + \frac{k^3}{3!}f_{yyy}(x,y) + 2\frac{k^2h}{3!}f_{xyy}(x,y) \\ &\quad + 2\frac{h^2k}{3!}f_{xxy}(x,y) + \cdots \end{aligned} \tag{2.4}$$

To estimate the local error recall that we apply one step of the difference formula starting from the exact solution and compare the result with the actual solution. For the Midpoint rule the local truncation error τ_{i+1} at t_{i+1} is

$$\tau_{i+1} = y(t_{i+1}) - \left[y(t_i) + \Delta t f\big(t_i + \frac{\Delta t}{2}, y(t_i) + \frac{\Delta t}{2} f(t_i, y(t_i))\big) \right] .$$

As before, we expand $y(t_{i+1})$ with a Taylor series but this time we keep the actual terms through $(\Delta t)^3$ and have a remainder because we want to demonstrate that terms in the expression for the truncation error through $(\Delta t)^2$ cancel but terms involving $(\Delta t)^3$ do not; this way we will demonstrate that the local truncation is *exactly* $\mathcal{O}(\Delta t^3)$ rather than it is *at least* $\mathcal{O}(\Delta t^3)$. We have

$$y(t_{i+1}) = y(t_i) + \Delta t y'(t_i) + \frac{(\Delta t)^2}{2} y''(t_i) + \frac{(\Delta t)^3}{3!} y'''(t_i) + \mathcal{O}(\Delta t^4) . \tag{2.5}$$

We know that $y'(t) = f(t,y)$ but the above expression also includes $y''(t)$ and $y'''(t)$. To express $y''(t)$ in terms of $f(t,y)$ and its partial derivatives we have to differentiate $f(t,y)$ with respect to t which requires the use of the chain rule. We have

$$y''(t) = \frac{\partial f}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} = f_t + f_y y' = f_t + f_y f .$$

Similarly,

$$\begin{aligned} y'''(t) &= \frac{\partial (f_t + f_y f)}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial (f_t + f_y f)}{\partial y}\frac{\partial y}{\partial t} \\ &= f_{tt} + f_{yt} f + f_y f_t + \big(f_{ty} + f_{yy} f + f_y^2\big) f \\ &= f_{tt} + 2 f_{yt} f + f_y f_t + f_{yy} f^2 + f f_y^2 . \end{aligned}$$

In the expression for τ_{i+1} we substitute the expansion (2.5) but we still have to deal with the term $f(t_i + \frac{\Delta t}{2}, y(t_i) + \frac{\Delta t}{2} f(t_i, y(t_i)))$; because this term is a function of two variables instead of one we need to use Proposition 2.4 for its expansion. To

use this proposition we note that the change in the first variable t is $h = \Delta t/2$ and the change in the second variable y is $k = (\Delta t/2) f\big(t_i, y(t_i)\big)$. We have

$$\Delta t f\big(t_i + \frac{\Delta t}{2}, y(t_i) + \frac{\Delta t}{2} f(t_i, y(t_i))\big) = \Delta t\Big[f + \frac{\Delta t}{2} f_t + \frac{\Delta t}{2} f f_y + \frac{(\Delta t)^2}{4 \cdot 2!} f_{tt} + \frac{(\Delta t)^2}{4 \cdot 2!} f^2 f_{yy} + 2\frac{(\Delta t)^2}{4 \cdot 2!} f f_{ty}\Big] + \mathcal{O}(\Delta t^4)\,.$$

All terms involving f or its derivatives on the right-hand side of this equation are evaluated at $(t_i, y(t_i))$ and we have omitted this explicit dependence for brevity. Substituting this expansion and (2.5) into the expression for τ_{i+1} and collecting terms involving each power of Δt yields

$$\begin{aligned} \tau_{i+1} &= \Delta t(y' - f) + \frac{\Delta t^2}{2}\big(y'' - (f_t + f f_y)\big) \\ &\quad + \Delta t^3\big(\frac{1}{3!} y''' - \frac{1}{8}\big(f_{tt} + f^2 f_{yy} + 2 f f_{ty}\big) + \mathcal{O}(\Delta t^4)\,. \end{aligned}$$

To cancel terms recall that using the differential equation and the chain rule allow us to write $y''(t) = f_t + f f_y$ and $y''' = f_{tt} + 2 f f_{ty} + f^2 f_{yy} + f_t f_y + f f_y^2$. Thus the terms involving Δt and $(\Delta t)^2$ cancel but the terms involving $(\Delta t)^3$ do not. Consequently the local truncation error converges cubically and we expect the global convergence rate to be quadratic in Δt. The following example demonstrates the numerical accuracy of the Midpoint method.

Example 2. MIDPOINT METHOD (2.3)

Consider the IVP

$$y'(t) = 3yt^2 \qquad y(0) = \frac{1}{3}$$

that we approximated by a second order Taylor series method in the previous example. To implement the method we require two function evaluations as compared with the Euler method. The following table provides errors at $t = 1$ for $\Delta t = 1/4, 1/8, \ldots, 1/128$ and the numerical rates. As can be seen from the table, the numerical rate is approaching two as $\Delta t \to 0$.

Δt	Error	Numerical rate
$1/4$	$6.9664\ 10^{-2}$	
$1/8$	$2.2345\ 10^{-2}$	1.64
$1/16$	$6.3312\ 10^{-3}$	1.82
$1/32$	$1.6827\ 10^{-3}$	1.91
$1/64$	$4.3346\ 10^{-4}$	1.96
$1/128$	$1.0998\ 10^{-4}$	1.98

If we use a Riemann sum or the Midpoint rule to approximate an integral $\int_a^b g(t)dt$ where $g(t) \geq 0$ on $[a, b]$ then we are using a rectangle to approximate the area. An alternate approach is to use a trapezoid to approximate this area. The trapezoidal integration rule is found by calculating the area of the trapezoid with base $(b - a)$ and height determined by the line passing through $(a, g(a))$ and $(b, g(b))$; specifically the rule is

$$\int_a^b g(t)\, dt \approx \frac{(b-a)}{2}\Big(g(a) + g(b)\Big)\,.$$

Integrating the differential equation (1.2a) from t_i to t_{i+1} and using this quadrature rule gives

$$y(t_{i+1}) - y(t_i) \approx \frac{\Delta t}{2}\Big[f\big(t_i, y(t_i)\big) + f\big(t_{i+1}, y(t_{i+1})\big)\Big].$$

Trapezoidal Rule $$Y_{i+1} = Y_i + \frac{\Delta t}{2}\big[f(t_i, Y_i) + f(t_{i+1}, Y_{i+1})\big] \tag{2.6}$$

However, like the backward Euler method this is an implicit scheme and thus for each t_i we need to solve a nonlinear equation for most choices of $f(t, y)$. This can be done, but there are better approaches for using implicit schemes in the context of ODEs as we will see in § 2.6.

Other numerical quadrature rules lead to additional explicit and implicit methods. The Euler method, the Midpoint Rule and the Trapezoidal rule all belong to a family of methods called **Runge-Kutta methods**. There is actually an easier way to derive these methods which we discuss in § 2.4.

2.3 Methods from Interpolation

Another approach to deriving methods is to use an interpolating polynomial to approximate either $y(t)$ or $f(t, y)$. If we choose to use an interpolating polynomial for $y(t)$ over some interval then we differentiate it and use this as an approximation to $y'(t)$ in the equation $y'(t) = f(t, y)$. On the other hand, if we choose to use an interpolating polynomial for $f(t, y)$ then we integrate. Recall that in § 2.2 we integrated our differential equation from t_i to t_{i+1} and used a quadrature formula for the integral involving $f(t, y)$. However, if we approximate $f(t, y)$ by an interpolating polynomial then this can be integrated exactly. Both approaches lead to families of methods called **multistep methods** discussed in detail in § 2.5. These methods use previous approximations such as Y_i, Y_{i-1}, Y_{i-2}, etc. and corresponding slopes to extrapolate the solution at t_{i+1}. This methods are contrasted with methods such as the Midpoint or Trapezoidal method in § 2.3.3. We first look at methods obtained by approximating $y(t)$ by an interpolating polynomial and differentiating and concentrate on the family of implicit methods called **Backward Difference Formulas (BDF)** . Then we look at an example of a method derived by using an interpolating polynomial for $f(t, y)$ and integrating the equation.

2.3.1 Using an interpolation polynomial for $\mathbf{y(t)}$

In § 1.4 we showed that if we use a linear interpolating polynomial $p_1(t)$ using the points t_i and t_{i+1} to approximate $y(t)$ for $t_i \le t \le t_{i+1}$ then we can differentiate it to get an approximation to $y'(t)$. We can derive an explicit method such as the forward Euler method by using $p_1'(t_i)$ to approximate $y'(t_i)$ where we use the

equation $y'(t_i) = f\big(t_i, y(t_i)\big)$. On the other hand, if we use $p_1'(t_{i+1})$ to approximate $y'(t_{i+1})$ then we use the equation $y'(t_{i+1}) = f\big(t_{i+1}, y(t_{i+1})\big)$ and obtain the implicit backward Euler scheme.

Backward difference formulas are a family of implicit multistep methods and the backward Euler method is considered a first order BDF. Backward difference formulas are especially useful when the IVP is difficult to solve in the sense that smaller and smaller time steps are required. This property is discussed in Chapter 3.

If we want a higher order scheme than first order then an obvious approach is to use a higher order interpolating polynomial to approximate y. For example, we can use a quadratic polynomial; however we know that fitting a quadratic requires three points. We have the points t_i) and t_{i+1} but need to choose a third. In multistep formulas information at previously calculated points are used; this has the advantage that no additional function evaluations are required. Thus for a quadratic interpolating polynomial the point t_{i-1} is chosen as the third point. The Lagrange form of the interpolating polynomial $p_2(t)$ for the points $(t_{i-1}, y(t_{i-1}))$, $(t_i, y(t_i))$, and $(t_{i+1}, y(t_{i+1}))$ is

$$\begin{aligned} p_2(t) &= y(t_{i-1})\frac{(t-t_i)(t-t_{i+1})}{(t_{i-1}-t_i)(t_{i-1}-t_{i+1})} + y(t_i)\frac{(t-t_{i-1})(t-t_{i+1})}{(t_i-t_{i-1})(t_i-t_{i+1})} \\ &\quad + y(t_{i+1})\frac{(t-t_{i-1})(t-t_i)}{(t_{i+1}-t_{i-1})(t_{i+1}-t_i)} \end{aligned}$$

and differentiating with respect to t and assuming a constant Δt gives

$$p_2'(t) = \frac{y(t_{i-1})}{2(\Delta t)^2}\big[2t - t_i - t_{i+1}\big] - \frac{y(t_i)}{(\Delta t)^2}\big[2t - t_{i-1} - t_{i+1}\big] + \frac{y(t_{i+1})}{2(\Delta t)^2}\big[2t - t_{i-1} - t_i\big].$$

We will concentrate on BDF schemes here so we want an implicit scheme; in the exercises we look at an explicit formula derived using this interpolating polynomial. Consequently, we use $p_2'(t_{i+1})$ as an approximation to $y'(t_{i+1})$ in the equation $y'(t_{i+1}) = f\big(t_{i+1}, y(t_{i+1})\big)$; we have

$$p_2'(t_{i+1}) = \frac{y(t_{i-1})}{2(\Delta t)^2}\Delta t - \frac{y(t_i)}{(\Delta t)^2}2\Delta t + \frac{y(t_{i+1})}{2(\Delta t)^2}3\Delta t = f\big(t_{i+1}, y(t_{i+1})\big).$$

This suggest the BDF

$$\frac{3}{2}Y_{i+1} - 2Y_i + \frac{1}{2}Y_{i-1} = \Delta t f(t_{i+1}, Y_{i+1}).$$

Second order BDF
$$Y_{i+1} = \frac{4}{3}Y_i - \frac{1}{3}Y_{i-1} + \frac{2}{3}\Delta t f(t_{i+1}, Y_{i+1}) \tag{2.7}$$

Some references give the BDF formulas so that the coefficient of Y_{i+1} is one and others will not normalize by the coefficient of Y_{i+1}. In general BDF formulas

order s	a_{s1}	a_{s2}	a_{s3}	a_{s4}	a_{s5}	β
1	1					1
2	4/3	-1/3				2/3
3	18/11	-9/11	2/11			6/11
4	48/25	-36/25	16/25	-3/25		12/25
5	300/137	-300/137	200/137	-75/137	12/137	60/137

Table 2.1: Coefficients for implicit BDF formulas of the form (2.8) where the coefficient of Y_{i+1} is normalized to one.

using approximations at t_{i+1}, $t_i, \cdots, t_{i+1-s}$ have the general normalized form

$$Y_{i+1} = \sum_{j=1}^{s} a_{sj} Y_{(i+1)-j} + \beta \Delta t f(t_{i+1}, Y_{i+1}) \,. \tag{2.8}$$

For our scheme (2.7) we have $s = 2$, $a_{21} = 4/3$, $a_{22} = 1/3$ and $\beta = 2/3$. Table 2.1 gives coefficients for other uniform BDF formulas using the terminology of (2.8). Note that we have included the order of accuracy of each method in Table 2.1. However, we have not rigorously proved that the method (2.7) is second order but it is what we should expect from interpolation theory. Recall that the backward Euler method can be derived by using a linear polynomial to interpolate $y(t)$ and for (2.7) we used a quadratic interpolating polynomial so we should expect to gain one power of Δt. This can be rigorously demonstrated.

It is also possible to derive BDFs for nonuniform time steps. The formulas are derived in an analogous manner but are a bit more complicated because for the interpolating polynomial we must keep track of each Δt_i; in the case of a uniform Δt there are some cancellations which simplify the resulting formulas. In the exercises a BDF formula corresponding to (2.7) is explored for nonuniform time steps.

2.3.2 Using an interpolating polynomial for $\mathbf{f}(\mathbf{t}, \mathbf{y})$

Another way to derive schemes using interpolation is to use an interpolation polynomial to approximate $f(t, y)$. If we do this, then when we must integrate the equation over the interval $[t_i, t_{i+1}]$ where the integral of the interpolating polynomial for $f(t, y)$ can be integrated exactly. The interpolating polynomial can include the point t_{i+1} or not; if it includes that point then the resulting scheme will be implicit because it will involve $f(t_{i+1}, Y_{i+1})$; otherwise it will be explicit.

To see how to derive a scheme, suppose we want an explicit method where we use the previous information at t_i and t_{i-1}. We write the linear interpolating polynomial for $f(t, y)$ through the two points and integrate the equation from t_i to t_{i+1}. As before we use the Fundamental Theorem of Calculus to integrate $\int_{t_i}^{t_{i+1}} y'(t)\ dt$.

Using uniform step sizes, we have

$$\begin{aligned} y(t_{i+1}) - y(t_i) &\approx \int_{t_i}^{t_{i+1}} \left[f\big(t_{i-1}, y(t_{i-1})\big) \frac{t - t_i}{-\Delta t} + f\big(t_i, y(t_i)\big) \frac{t - t_{i-1}}{\Delta t} \right] dt \\ &= -\frac{1}{\Delta t} f\big(t_{i-1}, y(t_{i-1})\big) \frac{(t - t_i)^2}{2}\Big|_{t_i}^{t_{i+1}} + \frac{1}{\Delta t} f\big(t_i, y(t_i)\big) \frac{(t - t_{i-1})^2}{2}\Big|_{t_i}^{t_{i+1}} \\ &= -\frac{1}{\Delta t} f\big(t_{i-1}, y(t_{i-1})\big) \left(\frac{(\Delta t)^2}{2} \right) + \frac{1}{\Delta t} f\big(t_i, y(t_i)\big) \frac{3\Delta t^2}{2} \end{aligned}$$

which suggests the scheme

$$Y_{i+1} = Y_i + \frac{3}{2} \Delta t f\big(t_i, y(t_i)\big) - \frac{\Delta t}{2} f\big(t_{i-1}, y(t_{i-1})\big) . \tag{2.9}$$

This is an example of a multistep method; these types of methods will be discussed in detail in § 2.5.

2.3.3 Single step versus multistep methods

Note that the BDF scheme (2.8) and the method (2.9) differ from the other schemes we derived because they use the history of approximations to y and f to extrapolate the solution at t_{i+1}. For example, (2.7) specifically uses approximations to $y(t)$ at t_i and t_{i-1} and (2.9) using approximations to the slope at t_i and t_{i-1}. These types of methods are called multistep methods because they use the approximate solution and the slope at previously calculated points other than at t_i to approximate the solution at the next point t_{i+1}. This is in contrast to a method such as the Midpoint method which uses only one previously calculated approximation, Y_i, to approximate Y_{i+1}; of course it also uses an approximation at $t_i + \frac{\Delta t}{2}$. Such a method is called a **single step method**.

Single step methods perform approximations to $y(t)$ in the interval $[t_i, t_{i+1}]$ as a means to bootstrap an approximation to $y(t_{i+1})$. Multistep methods combine information that was previously calculated at points such as $t_i, t_{i-1}, t_{i-2} \ldots$ to extrapolate the solution at t_{i+1}. A method is called an m-*step* method if it uses information from m grid points (including t_i) to calculate Y_{i+1}; this is why a single step method is also called a one-step method since it only uses t_i.

There are advantages and disadvantages to both single step and multistep methods. Because multistep methods use previously calculated information, we must store these values; this is not an issue when we are solving a single IVP but if we have a system then our solution and the slope are vectors and so this requires more storage. However multistep methods have the advantage that $f(t, y)$ has already been evaluated at prior points so this information can be stored. Consequently multistep methods require fewer function evaluations per step than single step methods and should be used where it is costly to evaluate $f(t, y)$.

If we look at the second order BDF (2.7) that we derived then we realize another shortcoming of multistep methods. Initially we set $Y_0 = y(t_0)$ and use this to start a single step method such as the Midpoint method. However, in (2.7) we need both

Y_0 and Y_1 to implement the scheme. How can we get an approximation to $y(t_1)$? The obvious approach is to use a single step method. So if we use m previous values (including t_i) then we must take $m-1$ steps of a single step method to start the simulations; it is $m-1$ steps because we have the value Y_0. Of course care must be taken in the choice of which single step method to use. For example, if our multistep method is $\mathcal{O}(\Delta t^r)$ then we should choose a single step method of the same accuracy; a lower order accurate scheme could contaminate the error.

2.4 Runge-Kutta Methods

Runge-Kutta (RK) methods are a family of single step methods which include both explicit and implicit methods. The forward Euler and the Midpoint method are examples of explicit RK methods. The backward Euler and the Trapezoidal method are examples of implicit RK methods. When we derived the Midpoint and Trapezoidal methods we used a numerical quadrature rule to approximate $\int_{t_i}^{t_{i+1}} f(t,y)dt$. To derive other single step methods we can use other numerical quadrature rules such as Gauss quadrature. However, there is an easier approach to deriving families of single step methods which have a desired accuracy which we will introduce here.

Runge was a German mathematician who first pointed out that it was possible to get higher order accurate methods without having to perform the successive differentiation of $f(t,y)$ that is required in Taylor series methods. Instead of approximating the integral by the unsymmetrical and somewhat inaccurate left or right Riemann sum, Runge used more accurate formulas such as the Midpoint or Trapezoidal method. In 1895 he published a seminal paper putting forth these ideas and demonstrating the improved accuracy of the methods over the Euler method. The family of RK methods were developed primarily from 1895 to 1925 and involved work by Heun, Kutta and Nystrom. Interested readers should refer to the paper by J.C. Butcher entitled "A history of Runge-Kutta methods."

The standard approach to deriving families of RK methods is to form a problem with undetermined parameters and approximate $y(t)$ and its slope $f(t,y)$ at a fixed number of unknown points in $[t_i, t_{i+1}]$; we then determine the parameters so that the accuracy is as high as possible. We assume a total of s unknown points (including the approximation at t_i) in $[t_i, t_{i+1}]$ and write the most general formula for such a method which will involve unknown parameters. Then we use Taylor series to determine the parameters governing the points in $[t_i, t_{i+1}]$ which guarantee the highest local truncation error possible. In the following examples we illustrate how this approach works.

Example 3. Derivation of a first order RK method

When $s=1$ we only use y and its slope at t_i so the most general difference equation is

$$Y_{i+1} = \beta Y_i + b_1 f(t_i, Y_i)\,,$$

where we have two unknown parameters β and b_1. Now we determine the parameters so that the scheme has as high a local truncation error as possible. We proceed as before when we determined a local truncation error and we expand $y(t_{i+1})$ in a Taylor series to get

$$\tau_{i+1} = \left[y(t_i) + \Delta t y'(t_i) + \frac{\Delta t^2}{2} y''(\xi_i)\right] - \left[\beta y(t_i) + b_1 f(t_i, y(t_i))\right].$$

We want to choose the parameters β, b_1 so that all terms $(\Delta t)^r$ for $r = 0, 1, \ldots R$ vanish for the largest possible value of R. If we force the terms involving $(\Delta t)^0$ and $(\Delta t)^1$ to be zero we get

$$(\Delta t)^0\big[y(t_i) - \beta y(t_i)\big] = 0 \quad \text{and} \quad (\Delta t)^1\big[y'(t_i) - b_1 y'(t_i)\big] = 0$$

where we have used the differential equation $y' = f(t, y)$. Clearly we have $\beta = 1$ and $b_1 = 1$ but are unable to make the term which is $\mathcal{O}(\Delta t^2)$ vanish. This method is just the forward Euler method so it is the simplest explicit RK method and the local truncation error is second order. In the sequel we will dispense with the coefficient β because it must always be equal to one.

Example 4. Derivation of a second order RK method

When $s = 2$ we use the slope at one intermediate point in $(t_i, t_{i+1}]$ in addition to the point t_i. Because we are doing an additional function evaluation, we expect that we should be able to make the truncation error smaller if we choose the parameters correctly; i.e., we choose an appropriate point in $(t_i, t_{i+1}]$. We must leave the choice of the location of the point as a variable so our general difference equation is

$$Y_{i+1} = Y_i + b_1 \Delta t f(t_i, Y_i) + b_2 \Delta t f\big(t_i + c_2\Delta t, Y_i + a_{21}\Delta t f(t_i, Y_i)\big)$$

where our new point in $(t_i, t_{i+1}]$ is $\big(t_i + c_2\Delta t, Y_i + a_{21}\Delta t f(t_i, Y_i)\big)$. To determine constraints on the parameters b_1, b_2, c_2 and a_{21} which result in the highest order for the truncation error, we compute the local truncation error and use Taylor series to expand the terms. For simplicity, in the following expansion we have omitted the explicit evaluation of f and its derivatives at the point $(t_i, y(t_i))$; however, if f is evaluated at some other point we have explicitly noted this. We use Proposition 2.4 for a Taylor series expansion in two variables to get

$$\begin{aligned}
\tau_{i+1} &= \Big[y + \Delta t y' + \frac{\Delta t^2}{2!}y'' + \frac{\Delta t^3}{3!}y''' + \mathcal{O}(\Delta t^4)\Big] \\
&\quad -\Big[y + b_1\Delta t f + b_2\Delta t f(t_i + c_2\Delta t, y + a_{21}\Delta t f)\Big] \\
&= \Big[\Delta t f + \frac{\Delta t^2}{2}\big(f_t + f f_y\big) + \frac{\Delta t^3}{6}\big(f_{tt} + 2f f_{ty} + f^2 f_{yy} + f_t f_y + f f_y^2\big) + \mathcal{O}(\Delta t)^4)\Big] \\
&\quad -b_1\Delta t f - b_2\Delta t\Big[f + c_2\Delta t f_t + a_{21}\Delta t f f_y \\
&\quad + \frac{c_2^2(\Delta t)^2}{2}f_{tt} + \frac{a_{21}^2(\Delta t)^2 f^2}{2}f_{yy} + c_2 a_{21}(\Delta t)^2 f f_{ty} + \mathcal{O}(\Delta t^3)\Big].
\end{aligned}$$

We first see if we can determine the parameters so that the scheme has a local truncation error of $\mathcal{O}(\Delta t^3)$; to this end we must determine the equations that the unknowns coefficients must satisfy in order for the terms involving $(\Delta t)^1$ and $(\Delta t)^2$ to vanish:

$$\begin{aligned}
\Delta t\,[f(1 - b_1 - b_2)] &= 0 \\
\Delta t^2\left[f_t\big(\frac{1}{2} - b_2 c_2\big) + f f_y\big(\frac{1}{2} - b_2 a_{21}\big)\right] &= 0
\end{aligned}$$

where once again we have dropped the explicit evaluation of y and f at $(t_i, y(t_i))$. Thus we have the conditions

$$b_1 + b_2 = 1, \quad b_2 c_2 = \frac{1}{2} \quad \text{and} \quad b_2 a_{21} = \frac{1}{2}. \tag{2.10}$$

Note that the Midpoint method given in (2.3) satisfies these equations with $b_1 = 0, b_2 = 1$, $c_2 = a_{21} = 1/2$.

Because we have four parameters and only three constraints we might ask ourselves if it is possible to choose the parameters so that the local truncation error is one order higher, i.e., $\mathcal{O}(\Delta t^4)$. To see that this is impossible note that in the expansion of $y(t_{i+1})$ the term y''' involves terms such as $f_t f_y$ for which there are no corresponding terms in the expansion of $f\big(t_i + c_2\Delta t, Y_i + a_{21}\Delta t f(t_i, Y_i)\big)$ so these $\mathcal{O}(\Delta t^3)$ terms will remain.

There are many other schemes which satisfy the conditions (2.10) because we only have three constraints and four degrees of freedom or parameters. A commonly used choice is the Heun method where the intermediate point is $\big(t_i + \frac{2}{3}\Delta t, Y_i + \frac{2}{3}\Delta t f(t_i, Y_i)\big)$; note that $y(t_i + \frac{2}{3}\Delta t)$ is approximated by taking an Euler step of length $\frac{2}{3}\Delta t$. Specifically the Heun method is

$$Y_{i+1} = Y_i + \frac{1}{4}\Delta t f(t_i, Y_i) + \frac{3}{4}\Delta t f\big(t_i + \frac{2}{3}\Delta t, Y_i + \frac{2}{3}\Delta t f(t_i, Y_i)\big)$$

where $b_1 = 1/4$, $b_2 = 3/4$, $c_2 = 2/3$ and $a_{21} = 2/3$. RK methods are usually written in a slightly different form to make clear how many points in $[t_i, t_{i+1}]$ are used to approximate $y(t_{i+1})$ and thus how many function evaluations are needed.

Heun Method

$$\begin{aligned} k_1 &= \Delta t f(t_i, Y_i) \\ k_2 &= \Delta t f(t_i + \tfrac{2}{3}\Delta t, Y_i + \tfrac{2}{3}k_1) \\ Y_{i+1} &= Y_i + \tfrac{1}{4}k_1 + \tfrac{3}{4}k_2 \end{aligned} \tag{2.11}$$

So any choice of coefficients which satisfy (2.10) leads to a RK scheme which has a local truncation error of $\mathcal{O}(\Delta t^3)$ and thus a global error of $\mathcal{O}(\Delta t^2)$.

To obtain a RK scheme which has a local truncation error of $\mathcal{O}(\Delta t^4)$ we need to use approximations at two intermediate points in the interval (t_i, t_{i+1}). In general, we have a scheme of the form

$$\begin{aligned} k_1 &= \Delta t f(t_i, Y_i) \\ k_2 &= \Delta t f(t_i + c_2\Delta t, Y_i + a_{21}k_1) \\ k_3 &= \Delta t f(t_i + c_3\Delta t, Y_i + a_{31}k_1 + a_{32}k_2) \\ Y_{i+1} &= Y_i + b_1k_1 + b_2k_2 + b_3k_3\,. \end{aligned}$$

To obtain conditions on the eight coefficients we would proceed as before by writing the local truncation error and using Taylor expansions; the calculation is straightforward but tedious. The calculations demonstrate that we can find a family of methods which have a local truncation of $\mathcal{O}(\Delta t^4)$ but not higher using t_i and two additional points in $(t_i, t_{i+1}]$.

There is a general form for explicit RK methods and we identify the methods by the number of **stages** s and the coefficients.

General s-stage explicit RK method

$$\begin{aligned} k_1 &= \Delta t f(t_i, Y_i) \\ k_2 &= \Delta t f(t_i + c_2\Delta t, Y_i + a_{21}k_1) \\ k_3 &= \Delta t f(t_i + c_3\Delta t, Y_i + a_{31}k_1 + a_{32}k_2) \\ &\vdots \\ k_s &= \Delta t f(t_i + c_s\Delta t, Y_i + a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{ss-1}k_{s-1} \\ Y_{i+1} &= Y_i + \textstyle\sum_{j=1}^{s} b_j k_j \end{aligned} \tag{2.12}$$

For example, the forward Euler method is a one-stage ($s = 1$) RK method and the Midpoint method and the Heun method are two-stage ($s = 2$) methods. To carry out a single step of an s stage RK method we need to evaluate s slopes; i.e., we must evaluate $f(t, y)$ at s points. In addition, we have $(s - 1)$ values to compute, $Y_i + a_{21}k_1$, $Y_i + a_{31}k_1 + a_{32}k_2$, $\cdots$, $Y_i + a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{ss-1}k_{s-1}$.

Once the stage s is set and the coefficients are determined, the method is completely specified; for this reason, the RK explicit methods are often described by a *Butcher*[1] *tableau* of the form

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{2.13}$$

for an s-stage RK method. Note that $c_1 = 0$ because we always use the point (t_i, Y_i). As an example, a commonly used 4-stage RK method is described by the tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{2.14}$$

which uses an approximation at the point t_i, two approximations at the point $t_i + \Delta t/2$, and the fourth approximation at t_{i+1}. In the examples of RK methods provided, it is important to note that c_i in the term $t_i + c_i\Delta t$ satisfy the property that $c_i = \sum_{j=1}^{i-1} a_{ij}$ so that along a row of the tableau the a_{ij} must sum to the c_i. In addition, the weights b_i satisfy $\sum_{i=1}^{s} b_i = 1$. This is true in general and can be used as a check in a computer code to confirm the coefficients have been entered correctly.

Many RK methods were derived in the early part of the 1900's; initially, the impetus was to find higher order explicit methods. We have seen examples where a one-stage RK method produced a global error of $\mathcal{O}(\Delta t)$, a two-stage RK method produced a global error of $\mathcal{O}(\Delta t^2)$ and a three-stage method produced a $\mathcal{O}(\Delta t^3)$ accuracy. One might be tempted to generalize that an s-stage method always produces a method with global error $\mathcal{O}(\Delta t^s)$, however, this is *not* the case. In the table below we give the minimum stage number required to gain a specific accuracy. As you can see from the table, a five-stage RK method does not produce a fifth

[1] Named after John C. Butcher, a mathematician from New Zealand.

order scheme; we need a six-stage method to produce that accuracy. Consequently higher stage RK methods are not as efficient as RK methods with ≤ 4 stages. Once it was realized that the stage number of a RK method did not correspond to the accuracy, the effort to derive additional RK methods moved to finding methods which optimize the local truncation error and to investigating implicit RK methods.

Order	1	2	3	4	5	6	7	8	9
Min. stage	1	2	3	4	6	7	9	11	11

Analogous to the general explicit s-stage RK scheme (2.12) we can write a general form of an implicit s-stage RK method. The difference in implicit methods is that in the calculation of k_i the approximation to $y(t_i + c_i\Delta t)$ can be over all values of s whereas in explicit methods the sum only goes through the previous k_j, $j = 1, \cdots, j-1$ terms.

General s-stage implicit RK method

$$\begin{aligned} k_1 &= \Delta t f(t_i, Y_i + a_{11}k_1 + +a_{12}k_2 + \cdots a_{1s}k_s) \\ k_2 &= \Delta t f(t_i, Y_i + a_{21}k_1 + +a_{22}k_2 + \cdots a_{2s}k_s) \\ &\vdots \\ k_s &= \Delta t f(t_i + c_s\Delta t, Y_i + a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{ss1}k_s) \\ Y_{i+1} &= Y_i + \textstyle\sum_{j=1}^{s} b_j k_j \end{aligned} \tag{2.15}$$

and its tableau is no longer upper triangular

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{2.16}$$

Unlike explicit RK methods, implicit s-stage RK methods can have an accuracy higher than s; in fact, it can be shown that the maximum possible accuracy of an s-stage implicit RK method is $2s$. Interest in deriving methods which can be used for error control blossomed in the 1960's; we will look at error control in the next section. Interest in implicit methods also rose when solving more difficult stiff problems became important; this will be discussed in § 3.5.

2.4.1 Step size control in Runge Kutta methods

So far we have assumed that the step size Δt is uniform; however, in many problems this is not practical when the solution varies much more rapidly at one time than another. At instances when the solution varies quickly, i.e., the slope is large, we

need to take a small step size and at times when the solution hardly changes using a large step size makes the scheme more efficient. The question is how to determine the appropriate step size at any instance. It turns out that there is an way to do this with RK methods.

To use RK methods for step size control we use two different methods to approximate the solution at t_{i+1} and compare the approximations. If the results are close, then we are confident that a correct step size was chosen; if they vary considerably then we assume that too large a step size was chosen and if they are extremely close then this suggests a larger step size can be used. Of course, to efficiently implement this approach we should choose the methods so that they have function evaluations in common to reduce the work.

A commonly used pair for error control is a combination of a fourth and fifth order explicit method; it is called the Runge-Kutta-Fehlberg method (RKF45) and was developed by the mathematician Erwin Fehlberg in the late 1960's. Recall that to get an accuracy of $(\Delta t)^5$ at least six function evaluations are required. Specifically we have

$$\begin{aligned}
k_1 &= f(t_i, Y_i) \\
k_2 &= f(t_i + \frac{1}{4}\Delta t, Y_i + \frac{1}{4}k_1) \\
k_3 &= f(t_i + \frac{3}{8}\Delta t, Y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2) \\
k_4 &= f(t_i + \frac{12}{13}\Delta t, Y_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3) \\
k_5 &= f(t_i + \Delta t, Y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4 \\
k_6 &= f(t_i + \frac{1}{2}\Delta t, Y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5)\,.
\end{aligned}$$

and the fourth order RK method

$$Y_{i+1} = Y_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \tag{2.17}$$

is used to approximate $y(t_{i+1})$ and the fifth order RK method

$$Y_{i+1} = Y_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \tag{2.18}$$

is used for comparison. Note that the fifth order method uses all of the coefficients of the fourth order method so it is efficient to implement because it only requires a single additional function evaluation. Typically the Butcher tableau is written for the fifth order method and then two lines are appended at the bottom for the

coefficients b_i in each method. For example, for RKF45 the tableau is

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{2856}{56430} & -\frac{9}{50} & \frac{2}{55}
\end{array}
$$

To implement the RKF45 scheme we find two approximations, $Y_{i+1}^{(4)}$ using the fourth order scheme (2.17) and $Y_{i+1}^{(5)}$ using the fifth order scheme (2.18). We then determine the difference $|Y_{i+1}^{(5)} - Y_{i+1}^{(4)}|$ which should be $\mathcal{O}(\Delta t)$. This error is used to make the decision whether to accept the step or not; if we accept the step then the decision must be made whether or not to increase the step size for the next calculation or keep it the same. One must choose a priori a minimum and maximum acceptable value for the normalized difference between $Y_{i+1}^{(4)}$ and $Y_{i+1}^{(5)}$ and use these values for deciding whether a step is acceptable or not.

2.5 Multistep Methods

Recall that single step methods such as RK methods use information at additional points in the interval $(t_i, t_{i+1}]$ to obtain an approximation at t_{i+1} whereas multistep methods take the viewpoint that the history of the solution should affect the approximation at the next time level. The BDF formula (2.7) derived in § 2.3.1 is an example of an implicit multistep method. Specifically, explicit multistep methods use information at t_i plus additional computed approximations at previous times such as t_{i-1}, t_{i-2} to extrapolate the solution at t_{i+1}, i.e., it uses information at multiple grid points. Implicit multistep methods include the point t_{i+1} and interpolate the solution there. This is illustrated in Figure **??**. A method is called an m-*step* method if it uses information from m grid points (including t_i) to calculate Y_{i+1}. An advantage of using a multistep method over a single step method is that it requires fewer function evaluations. A disadvantage is that it requires storing previous values which is only an issue when we are solving systems of equations. In addition, multistep methods require the use of a single step method to obtain additional starting values.

General m-step multistep method

$$\begin{aligned} Y_{i+1} &= a_{m-1}Y_i + a_{m-2}Y_{i-1} + a_{m-3}Y_{i-2} + \cdots + a_0 Y_{i+1-m} \\ &\quad + \Delta t\Big[b_m f(t_{i+1}, Y_{i+1}) + b_{m-1} f(t_i, Y_i) + b_{m-2} f(t_{i-1}, Y_{i-1}) \\ &\quad + \cdots + b_0 f(t_{i+1-m}, Y_{i+1-m})\Big]. \end{aligned} \tag{2.19}$$

If $b_m = 0$ then the method is explicit; otherwise it is implicit.

A commonly used family of explicit multistep methods are called *Adams-Bashforth* which use the derivative f evaluated at m prior points (including t_i) but only use the approximation to $y(t)$ at t_i; i.e., $a_0 = \cdots = a_{m-2} = 0$. The one step Adams-Bashforth method is the forward Euler method. In § 2.3.2 we used an interpolation polynomial for $f(t, y)$ to derive the 2-step scheme

$$Y_{i+1} = Y_i + \frac{3}{2}\Delta t f\big(t_i, y(t_i)\big) - \frac{\Delta t}{2} f\big(t_{i-1}, y(t_{i-1})\big)$$

which belongs to the Adams-Bashforth family with $b_2 = 0$, $b_1 = 3/2$ and $b_0 = -1/2$. In the exercises, you are asked to rigorously demonstrate that the local truncation error for (2.9) is third order and thus the scheme is second order accurate. The methods up to five steps are listed here for completeness.

Adams-Bashforth 2-step, 3-step, 4-step and 5-step methods

$$\begin{aligned} Y_{i+1} &= Y_i + \Delta t\left(\frac{3}{2}f(t_i, Y_i) - \frac{1}{2}f(t_{i-1}, Y_{i-1})\right) \\ Y_{i+1} &= Y_i + \Delta t\left(\frac{23}{12}f(t_i, Y_i) - \frac{4}{3}f(t_{i-1}, Y_{i-1}) + \frac{5}{12}f(t_{i-2}, Y_{i-2})\right) \\ Y_{i+1} &= Y_i + \Delta t\left(\frac{55}{24}f(t_i, Y_i) - \frac{59}{24}f(t_{i-1}, Y_{i-1}) + \frac{37}{24}f(t_{i-2}, Y_{i-2})\right. \\ &\qquad \left. - \frac{3}{8}f(t_{i-3}, Y_{i-3})\right) \\ Y_{i+1} &= Y_i + \Delta t\left(\frac{1901}{720}f(t_i, Y_i) - \frac{1387}{360}f(t_{i-1}, Y_{i-1}) + \frac{109}{30}f(t_{i-2}, Y_{i-2})\right. \\ &\qquad \left. - \frac{637}{360}f(t_{i-3}, Y_{i-3}) + \frac{251}{720}f(t_{i-4}, Y_{i-4})\right) \end{aligned} \tag{2.20}$$

Schemes in the *Adams-Moulton* family are commonly used implicit multistep method which use the derivative f evaluated at t_{i+1} plus m prior points but only use Y_i. The first step Adams-Moulton method is the backward Euler scheme and the 2-step method is the Trapezoidal rule; several methods are listed here for completeness.

Adams-Moulton 2-step, 3-step, 4-step and 5-step methods

$$
\begin{aligned}
Y_{i+1} &= Y_i + \frac{\Delta t}{2}\big(f(t_{i+1},Y_{i+1}) + f(t_i,Y_i)\big) \\
Y_{i+1} &= Y_i + \Delta t\left(\frac{5}{12}f(t_{i+1},Y_{i+1}) + \frac{2}{3}f(t_i,Y_i) - \frac{1}{12}f(t_{i-1},Y_{i-1})\right) \\
Y_{i+1} &= Y_i + \Delta t\left(\frac{3}{8}f(t_{i+1},Y_{i+1}) + \frac{19}{24}f(t_i,Y_i) - \frac{5}{24}f(t_{i-1},Y_{i-1})\right. \\
&\qquad \left. + \frac{1}{24}f(t_{i-2},Y_{i-2})\right) \\
Y_{i+1} &= Y_i + \Delta t\left(\frac{251}{720}f(t_{i+1},Y_{i+1}) + \frac{646}{720}f(t_i,Y_i) - \frac{264}{720}f(t_{i-1},Y_{i-1})\right. \\
&\qquad \left. + \frac{106}{720}f(t_{i-2},Y_{i-2}) - \frac{19}{720}f(t_{i-3},Y_{i-3})\right)
\end{aligned}
\tag{2.21}
$$

One drawback of an m-step method is that we need m starting values Y_0, Y_1, $Y_2, \ldots, Y_{m-1}$ and we only have Y_0 from the initial condition. Typically one uses a single step method to start the scheme. How do we decide what method to use? A "safe" approach is to use a method which has the same accuracy as the multistep method but we will see in the following examples that you can actually use a method which has one power of Δt less because we are only taking a small number of steps with the method. For example, if we use the 2-step second order Adams-Bashforth method we need Y_1 in addition to Y_0. If we take one step with the forward Euler method it is actually second order accurate at the first step because the error there is only due to the local truncation error. However, if we use a 3-step third order Adams-Bashforth method then using the forward Euler method to get the two starting values results in a loss of accuracy. This issue is illustrated in the following examples.

Example 5. STARTING VALUES FOR MULTISTEP METHODS

In this example we implement the 3-step third order accurate Adams Bashforth method given in (2.20) to solve the IVP

$$y'(t)t^2 + y(t) \quad 2 < t < 5 \qquad y(2) = 1\,,$$

which has the exact solution

$$y(t) = 11e^{t-2} - t^2 + 2t + 2\,.$$

We compare the numerical rates of convergence when we use different methods to generate the starting values. Specifically we use RK methods of order one through four to generate the starting values which for a 4-step method are Y_1, Y_2, and Y_3 because we have $Y_0 = 1$. The results are tabulated below for the errors at $t = 3$. As you can see from the table, if a second, third or fourth order scheme is used to compute the starting values then the method is third order. There is nothing gained by using a higher order scheme (the fourth order) for the starting values. However, if a first order scheme (forward Euler) is used then the rate is degraded to second order even though we only used it to calculate two values, Y_1 and Y_2. Consequently to compute starting values we should use a scheme that has the same overall accuracy or one degree less than the method we are using.

	accuracy of starting method							
	first		second		third		fourth	
Δt	error	rate	error	rate	error	rate	error	rate
1/10	$2.425\ 10^{-1}$		$1.618\ 10^{-2}$		$8.231\ 10^{-3}$		$8.042\ 10^{-3}$	
1/20	$6.106\ 10^{-2}$	1.99	$2.241\ 10^{-3}$	2.87	$1.208\ 10^{-3}$	2.77	$1.195\ 10^{-3}$	2.75
1/40	$1.529\ 10^{-2}$	2.00	$2.946\ 10^{-4}$	2.92	$1.628\ 10^{-4}$	2.89	$1.620\ 10^{-4}$	2.88
1/80	$3.823\ 10^{-3}$	2.00	$3.777\ 10^{-5}$	2.96	$2.112\ 10^{-5}$	2.95	$2.107\ 10^{-5}$	2.94

Example 6. COMPARISON OF ADAMS BASHFORTH METHODS

In this example we solve the IVP from the previous example a by 2-step through 5-step Adams Bashforth method. In each case we use a scheme that is one degree less accurate to calculate the starting values. As can be seen from the table, all methods have the expected numerical rate of convergence.

	2-step method		3-step method		4-step method		5-step method	
Δt	error	rate	error	rate	error	rate	error	rate
1/10	$2.240\ 10^{-1}$		$1.618\ 10^{-2}$		$9.146\ 10^{-4}$		$5.567\ 10^{-5}$	
1/20	$5.896\ 10^{-2}$	1.93	$2.241\ 10^{-3}$	2.87	$6.986\ 10^{-5}$	3.71	$2.463\ 10^{-6}$	4.50
1/40	$1.509\ 10^{-2}$	1.97	$2.946\ 10^{-4}$	2.92	$4.802\ 10^{-6}$	3.86	$8.983\ 10^{-8}$	4.78
1/80	$3.816\ 10^{-3}$	1.98	$3.777\ 10^{-5}$	2.96	$3.144\ 10^{-7}$	3.93	$3.022\ 10^{-9}$	4.89

2.6 Predictor-Corrector Methods

We have considered several implicit schemes for approximating the solution of an IVP. However, when we implement these schemes the solution of a nonlinear equation is usually necessary. This requires extra work and we know that methods such as the Newton-Raphson method for nonlinear equations are not guaranteed to converge globally. For this reason, we need a more efficient way to use implicit schemes.

In predictor-corrector (PC) methods implicit schemes are used to improve (or correct) the solution that is first obtained (or predicted) by an explicit scheme. The implicit scheme is implemented as an explicit scheme because instead of computing $f(t_{i+1}, Y_{i+1})$ we use the predicted value for Y_{i+1}. In simulations where a variable step size is needed, we can also use predictor-corrector methods to estimate the appropriate step size.

For example, we first consider the Euler-Trapezoidal predictor-corrector pair where the explicit scheme is forward Euler and the implicit scheme is the Trapezoidal method (2.6). Recall that the forward Euler scheme is first order and the Trapezoidal is second order. If the result of the predicted solution at t_{i+1} is Y^p_{i+1} then we have the pair

Euler-Trapezoidal Predictor-Corrector Method

$$\begin{aligned} Y^p_{i+1} &= Y_i + \Delta t f(t_i, Y_i) \\ Y_{i+1} &= Y_i + \tfrac{\Delta t}{2}\Big[f(t_{i+1}, Y^p_{i+1}) + f(t_i, Y_i)\Big] \end{aligned} \tag{2.22}$$

It is important to realize that the implicit Trapezoidal method is now implemented like an explicit method because we evaluate $f(t_{i+1}, Y_{i+1}^p)$ instead of the unknown point $f(t_{i+1}, Y_{i+1})$. The predicted solution Y_{i+1}^p from the forward Euler method is first order but we add a correction to it using the Trapezoidal method and improve the error. We can view the predictor-corrector pair as implementing the difference scheme

$$Y_{i+1} = Y_i + \frac{\Delta t}{2}\Big[f\big(t_{i+1}, Y_i + \Delta t f(t_i, Y_i)\big) + f(t_i, Y_i)\Big]$$

which uses an average of the slope at (t_i, Y_i) and t_{i+1} and the Euler approximation there. In the exercises you are asked to show that this predictor-corrector pair is second order.

Example 7. EULER-TRAPEZOIDAL PREDICTOR-CORRECTOR PAIR

In this example we perform two steps of the Euler-Trapezoidal predictor-corrector pair by hand to demonstrate how it is implemented and then compare the numerical results with those obtained by just using the forward Euler method. Specifically we consider the IVP

$$y'(t) = -ty(t) \qquad y(0) = \frac{1}{2}$$

Using $Y_0 = 0.5$ and $\Delta t = 0.1$ we first predict the value at 0.1 using the forward Euler method with $f(t, y) = -ty$ to get

$$Y_1^p = Y_0 + .1 f(t_0, Y_0) = 0.5 + 0.1(-0 \cdot .5) = 0.5$$

and then correct to obtain the approximation at t_1

$$Y_1 = Y_0 + \frac{0.1}{2}\big[f(t_1, Y_1^P) + f(t_0, Y_0)\big] = 0.5 + 0.05\big[-.05 + 0\big] = 0.4975\,.$$

To get the approximation at $t_2 = .2$ we predict with

$$Y_2^P = Y_1 + .1 f(t_1, Y_1) = 0.4975 + .1(-.04975) = 0.492525$$

and correct to get

$$Y_2 = Y_1 + \frac{0.1}{2}\big[f(t_2, Y_2^P) + f(t_1, Y_1)\big] = 0.4975 + 0.05\big[-0.098505 - 0.04975\big] = 0.490087\,.$$

The exact values are $y(.1) = 0.497506$ and $y(.2) = 0.490099$.

The results for the approximate solutions at $t = 1$ are given in the table below using decreasing values of Δt; the corresponding results from just using the forward Euler method are also given. As can be seen from the table, the predictor-corrector pair is second order. Note that it requires one additional function evaluation, $f(t_{i+1}, Y_i^P)$, than the Euler method. The Midpoint rule requires the same number of function evaluations and has the same accuracy as this predictor-corrector pair. However, the predictor-corrector pair provides an easy way to estimate the error at each step.

	Predictor only		PC pair	
Δt	Error	rate	Error	rate
$1/10$	$1.0813\ 10^{-2}$		$9.3644\ 10^{-5}$	
$1/20$	$5.2266\ 10^{-3}$	1.05	$2.7654\ 10^{-5}$	1.76
$1/40$	$2.5698\ 10^{-3}$	1.02	$7.4148\ 10^{-6}$	1.90
$1/80$	$1.2742\ 10^{-3}$	1.01	$1.9146\ 10^{-6}$	1.95
$1/160$	$6.3444\ 10^{-4}$	1.01	$4.8616\ 10^{-7}$	1.98

Typically predictor-corrector pairs consist of an explicit multistep method such as an Adams-Bashforth method and a corresponding implicit Adams-Moulton multistep method. The pair should be chosen so that the only additional function

evaluation in the corrector equation is at the predicted point. One can demonstrate that if the order of the corrector is p, then the order of the pair is order p provided the order of the predictor is not less than $p-1$.

For example, one such pair is an explicit third order Adams-Bashforth predictor coupled with an implicit third order Adams-Moulton.

Third order Adams-Moulton predictor-corrector pair

$$\begin{aligned} Y^p_{i+1} &= Y_i + \tfrac{\Delta t}{12}\left[23f(t_i,Y_i) - 16f(t_{i-1},Y_{i-1}) + 5f(t_{i-2},Y_{i-2})\right] \\ Y_{i+1} &= Y_i + \tfrac{\Delta t}{12}\left[5f(t_{i+1},Y^p_{i+1}) + 8f(t_i,Y_i) - f(t_{i-1},Y_{i-1})\right] \end{aligned} \tag{2.23}$$

Example 8. THIRD ORDER ADAMS-MOULTON PREDICTOR CORRECTOR PAIR

In the table below we compare the errors and rates of convergence for the PC pair (2.23) and the third order Adams-Bashforth method defined in (2.20). Note that both numerical rates are approaching three but the error in the PC pair is almost an order of magnitude smaller at a fixed Δt.

	Predictor only		PC pair	
Δt	Error	rate	Error	rate
1/10	$2.0100\ 10^{-2}$		$1.5300\ 10^{-3}$	
1/20	$3.6475\ 10^{-3}$	2.47	$3.3482\ 10^{-4}$	2.19
1/40	$5.4518\ 10^{-4}$	2.74	$5.5105\ 10^{-5}$	2.60
1/80	$7.4570\ 10^{-5}$	2.87	$7.9035\ 10^{-6}$	2.80
1/160	$9.7513\ 10^{-6}$	2.93	$1.0583\ 10^{-6}$	2.90

Using predictor-corrector pairs also provides a way to estimate the error and thus determine if the current step size is appropriate. For example, for our third order predictor and corrector pair (2.23) one can specifically compute the constant in the local truncation error to get

$$|y(t_{i+1}) - Y^p_{i+1}| = \frac{9}{24}y^{[4]}(\xi)(\Delta t)^4 \qquad |y(t_{i+1}) - Y_{i+1}| = \frac{1}{24}y^{[4]}(\eta)(\Delta t)^4\,.$$

For small Δt we assume that the fourth derivative is constant over the interval and so

$$|y(t_{i+1}) - Y_{i+1}| \approx \frac{1}{9}|y(t_{i+1}) - Y^p_{i+1}|\,.$$

If the step size Δt is too large, then the assumption that the fourth derivative is constant from t_i to t_{i+1} may not hold and the above relationship is not true. Typically the exact solution $y(t_{i+1})$ is not known so instead we monitor the difference in the predicted and corrected solution $|Y_{i+1} - Y^p_{i+1}|$. If it is larger than our prescribed tolerance, then the step is rejected and Δt is decreased. Otherwise the step is accepted; if the difference is below the minimum prescribed tolerance then the step size is increased in the next calculation.

EXERCISES

1. Each of the following Runge-Kutta schemes is written in the Butcher tableau format. Identify each scheme as explicit or implicit and then write the scheme as

$$Y_{i+1} = Y_i + \sum_{i=1}^{s} b_i f(t_i + c_i, Y_i + k_i)$$

where the appropriate values are substituted for b_i, c_i, and k_i.

a.
$$\begin{array}{c|ccc} 0 & 0 & 0 & \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

b.
$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

2. Modify the derivation of the explicit second order Taylor series method in § 2.1 to derive an implicit second order Taylor series method.

3. Use a Taylor series to derive a third order accurate explicit difference equation for the IVP (1.2).

4. Gauss quadrature rules are popular for numerical integration because one gets the highest accuracy possible for a fixed number of quadrature points; however one gives up the "niceness" of the quadrature points. In addition, these rules are defined over the interval $[-1, 1]$. For example, the one-point Gauss quadrature rule is

$$\int_{-1}^{1} g(x)\, dx = \frac{1}{2} g(0)$$

and the two-point Gauss quadrature rule is

$$\int_{-1}^{1} g(x)\, dx = \frac{1}{2}\left[g(\frac{-1}{\sqrt{3}}) + g(\frac{1}{\sqrt{3}})\right.$$

Use the one-point Gauss rule to derive a Gauss-Runge-Kutta method. Is the method explicit or implicit? Does it coincide with any method we have derived?

5. Simpson's numerical integration rule is given by

$$\int_{a}^{b} g(x)\, dx = \frac{b-a}{6}\left[g(a) + 4g\left(\frac{a+b}{2}\right) + g(b)\right]$$

If $g(x) \geq 0$ on $[a, b]$ then it approximates the area under the curve $g(x)$ by the area under a parabola passing through the points $(a, g(a))$, $(b, g(b))$ and $((a+$

$b)/2, g((a+b)/2))$. Use this quadrature rule to approximate $\int_{t_i}^{t_{i+1}} f(t,y)\,dt$ to obtain an explicit 3-stage RK method. When you need to evaluate terms such as f at $t_i + \Delta t/2$ use an appropriate Euler step to obtain an approximation to the corresponding y value as we did in the Midpoint method. Write your method in the format of (2.12) and in a Butcher tableau.

6. In § 2.3 we derived a second order BDF formula for uniform grids. In an analogous manner, derive the corresponding method for a nonuniform grid.

7. Use an appropriate interpolating polynomial to derive the multistep method

$$Y_{i+1} = Y_{i-1} + 2\Delta t f(t_i, Y_i)\,.$$

Determine the accuracy of this method.

8. Determine the local truncation error for the 2-step Adams-Bashforth method (2.9).

Chapter 3

Systems and Stability Issues

When modeling phenomena where we know the initial state and how it changes with time, we often have either a higher order IVP or a system of IVPs rather than a single first order IVP. In this chapter we first demonstrate how a higher order IVP can be transformed into a system of first order IVPs. Then we extend in a straightforward manner some of the methods from Chapter 2 to systems of equations. We discuss implementation issues and examples that illustrate the use of systems of IVPs.

The final concept we investigate in our study of IVPs is that of stability and its affect on convergence. So far we have demonstrated the accuracy of certain methods, that is, as $\Delta t \to 0$ we determined the rate at which the error goes to zero. However, in these calculations we tacitly assumed convergence. Now we look at convergence in more depth because, as we saw in some examples, not all methods converge for every problem. Lastly we will look at so-called stiff systems of IVPs which have characteristics that make simulations using many explicit schemes unreliable.

3.1 Higher Order IVPs

Suppose we have the second order IVP

$$\begin{aligned} y''(t) &= 2y'(t) - \sin(\pi y) + 4t \quad 0 < t \leq 2 \\ y(0) &= 1 \\ y'(0) &= 0 \end{aligned}$$

where now the right-hand side is a function of t, y and y'. The methods we have learned only apply to first order IVPs. However, we can easily convert this second order IVP into two coupled first order IVPs. To do this, we let $w_1(t) = y(t)$, $w_2(t) = y'(t)$ and substitute into the equations and initial conditions to get a first

order system for w_1, w_2

$$\begin{aligned} w_1'(t) &= w_2(t) \qquad 0 < t \le 2 \\ w_2'(t) &= 2w_2(t) - \sin(\pi w_1) + 4t \qquad 0 < t \le 2 \\ w_1(0) &= 1 \qquad w_2(0) = 0\,. \end{aligned}$$

Note that these two differential equations are *coupled*, that is, the differential equation for w_1 depends on w_2 and the equation for w_2 depends on w_1.

In general, if we have the pth order IVP for $y(t)$

$$\begin{aligned} y^{[p]}(t) &= f(t, y, y', y'', \cdots, y^{[p-1]}) \quad t_0 < t \le T \\ y(t_0) &= \alpha_1, \quad y'(t_0) = \alpha_2, \quad y''(t_0) = \alpha_3, \quad \cdots \quad y^{[p-1]}(t_0) = \alpha_p \end{aligned}$$

then we convert it to a system of p first-order IVPs by letting $w_1(t) = y(t)$, $w_2(t) = y'(t), \cdots, w_p(t) = y^{[p-1]}(t)$ which yields the first order coupled system

$$\begin{aligned} w_1'(t) &= w_2(t) \\ w_2'(t) &= w_3(t) \\ &\vdots \\ w_{p-1}'(t) &= w_p(t) \\ w_p'(t) &= f(t, w_1, w_2, \ldots, w_p) \end{aligned} \tag{3.1}$$

along with the initial conditions $w_k = \alpha_k$, $k = 1, 2, \ldots, p$. Thus any higher order IVP that we encounter can be transformed into a coupled system of first order IVPs.

Example 1. CONVERTING A HIGH ORDER IVP INTO A SYSTEM

Write the fourth order IVP

$$y^{[4]}(t) + 2y''(t) + 4y(t) = 5 \quad y(0) = 1, y'(0) = -3, y''(0) = 0, y'''(0) = 2$$

as a system of first order equations.

We want four first order differential equations for $w_i(t)$, $i = 1, 2, 3, 4$; to this end let $w_1 = y$, $w_2 = y'$, $w_3 = y''$, and $w_4 = y'''$. Using the first two expressions we have $w_1' = w_2$, and the second and third gives $w_2' = w_3$, the third and fourth gives $w_3' = w_4$ and the original differential equation provides the last first order equation $w_4' + 2w_3 + 4w_1 = 5$. The system of equations is thus

$$\begin{aligned} w_1'(t) - w_2(t) &= 0 \\ w_2'(t) - w_3(t) &= 0 \\ w_3'(t) - w_4(t) &= 0 \\ w_4' + 2w_3 + 4w_1 &= 5 \end{aligned}$$

along with the initial conditions

$$w_1(0) = 1, \quad w_2(0) = -3, \quad w_3(0) = 0, \quad \text{and } w_4(0) = 2.$$

Oftentimes our model is already in the form of a system of first order IVPs. Our goal is to apply the methods of the previous chapter to a system of first order IVPs. The notation we use for a general system of N first order IVPs is

$$\begin{aligned} w_1'(t) &= f_1(t, w_1, w_2, \ldots, w_N) \quad t_0 < t \le T \\ w_2'(t) &= f_2(t, w_1, w_2, \ldots, w_N) \quad t_0 < t \le T \\ &\vdots \\ w_N'(t) &= f_N(t, w_1, w_2, \ldots, w_N) \quad t_0 < t \le T \end{aligned} \tag{3.2}$$

along with the initial conditions $w_k(t_0) = \alpha_k$, $k = 1, 2, \ldots, N$. For example, using this notation the pth order IVP written as the system (3.1) has $f_1 = w_2$, $f_2 = w_3$, etc.

Existence, uniqueness and continuous dependence of the solution to the system (3.2) can be established. Analogous to the case of a single IVP each function f_i must satisfy a Lipschitz condition. Details of this analysis can be found in standards texts in ODEs. For the sequel, we will assume that each system has a unique solution which depends continuously on the data.

In the next two sections we demonstrate how the single step and multistep methods from Chapter 2 are easily extended to the system of N equations (3.2). We use the notation $W_{k,i}$ as the approximation to $w_k(t_i)$ where the first subscript of W refers to the unknown number and the second to the point t_i.

3.2 Single Step Methods for Systems

We now want to extend single step methods to the system (3.2). For simplicity we first extend the forward Euler method for a system and then with the intuition gained from that method we extend a general explicit Runge-Kutta method to a system. Implicit RK methods can be extended in an analogous way.

Suppose we have the first order system (3.2) with the initial conditions $w_k(t_0) = \alpha_k$ for $k = 1, 2, \ldots, N$. The forward Euler method for each equation is

$$W_{k,i+1} = W_{k,i} + \Delta t f_k\big(t_i, W_{1,i}, W_{2,i}, \cdots, W_{N,i}\big)\,.$$

We write the Euler method as a vector equation so we can solve for all N unknowns simultaneously at each t_i; this is not necessary but results in an efficient implementation of the method. We set $\mathbf{W}_i = \big(W_{1,i}, W_{2,i}, \cdots, W_{N,i}\big)^T$, $\mathbf{W}_0 = (\alpha_1, \alpha_2, \ldots, \alpha_N)^T$, and $\mathbf{F}_i = \big(f_1(t_i, \mathbf{W}_i), f_2(t_i, \mathbf{W}_i), \cdots, f_N(t_i, \mathbf{W}_i)\big)^T$. For $i = 0, 1, 2, \ldots$ we have the following vector equation for the forward Euler method for a system

$$\mathbf{W}_{i+1} = \mathbf{W}_i + \Delta t \mathbf{F}_i\,. \tag{3.3}$$

To implement the scheme at each point t_i we have N function evaluations to form the vector $\mathbf{F}_i$, then we perform the scalar multiplication to get $\Delta t \mathbf{F}_i$ and then a vector addition to obtain the final result $\mathbf{W}_{i+1}$.

Example 2. FORWARD EULER FOR A SYSTEM

Consider the system of three IVPs

$$\begin{array}{rcll} w_1'(t) & = & 2w_2(t) - 4t & 0 < t < 10 \\ w_2'(t) & = & -w_1(t) + w_3(t) - e^t + 2 & 0 < t < 10 \\ w_3'(t) & = & w_1(t) - 2w_2(t) + w_3(t) + 4t & 0 < t < 10 \\ w_1(0) & = & -1, \quad w_2(0) = 0, & w_3(0) = 2 \end{array}$$

for the unknown $(w_1, w_2, w_3)^T$. The exact solution is $(-\cos(2t), \sin(2t) + 2t, \cos(2t) + e^t)^T$. We want to compute an approximation at $t = 0.2$ using $\Delta t = 0.1$ and the forward Euler method. We

set $\mathbf{W}_0 = (-1, 0, 2)^T$ and because $\mathbf{F}_i = \big(2W_{2,i} - 4t_i, -W_{1,i} + W_{3,i} - e^{t_i} + 2, W_{1,i} - 2W_{2,i} + W_{3,i} + 4t_i\big)^T$ we have $\mathbf{F}_0 = (0, 4, 1)^T$. With $\Delta t = 0.1$ we form $\mathbf{W}_1$ from

$$\mathbf{W}_1 = \mathbf{W}_0 + \Delta t \mathbf{F}_0 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} + 0.1 \begin{pmatrix} 0 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} -1.0 \\ 0.4 \\ 2.1 \end{pmatrix} .$$

Now to determine $\mathbf{W}_2$ we need $\mathbf{F}_1$ which is given by

$$\mathbf{F}_1 = \begin{pmatrix} 2(0.4) - 4(.1) \\ 1 + 2.1 - e^{.1} + 2 \\ -1 - 2(.4) + 2.1 + 4(.1) \end{pmatrix} = \begin{pmatrix} 0.400 \\ 3.995 \\ 0.7000 \end{pmatrix}$$

so that

$$\mathbf{W}_2 = \mathbf{W}_1 + \Delta t \mathbf{F}_1 = \begin{pmatrix} -1.0 \\ 0.4 \\ 2.1 \end{pmatrix} + 0.1 \begin{pmatrix} 0.400 \\ 3.995 \\ 0.700 \end{pmatrix} = \begin{pmatrix} -0.9600 \\ 0.7995 \\ 2.1700 \end{pmatrix} .$$

The exact solution at $t = 0.2$ is $(-0.921061, 0.789418, 2.14246)^T$. Unlike the case of a single IVP we now have an error *vector* instead of a single number; at $t = 0.2$ the error vector in our calculations is $(0.038939, .010082, .02754)^T$. To obtain a single number from this vector to use in the calculation of a numerical rate, we must use a vector norm. A common vector norm is the standard Euclidean norm which is often called ℓ_2 norm or the "little l2 norm". For this calculation at $t = 0.2$ the Euclidean norm of the error is $1.98\ 10^{-2}$.

In the table below we tabulate the results using the forward Euler method for this system at $t = 1$ where both the normalized ℓ_2-norm and ℓ_∞-norm (i.e., the maximum norm) of the error normalized by the corresponding norm of the solution is reported. Clearly we have linear convergence as we did in the case of a single equation.

Δt	ℓ_2 Error	rate	ℓ_∞ Error	rate
1/10	$6.630\ 10^{-2}$		$6.019\ 10^{-2}$	
1/20	$3.336\ 10^{-2}$	0.99	$3.156\ 10^{-2}$	0.93
1/40	$1.670\ 10^{-2}$	1.00	$1.631\ 10^{-2}$	0.95
1/80	$8.350\ 10^{-3}$	1.00	$8.277\ 10^{-3}$	0.98

Suppose now that we have an s-stage RK method; recall that for a single first order equation we have s function evaluations for each t_i. If we have N first order IVPs, then we need sN function evaluations at each t_i. For example, if we use a 4-stage RK with 10,000 equations then at each time we need 40,000 function evaluations; if we do 100 time steps then we have 4 million function evaluations. If function evaluations are expensive, multistep methods may be more efficient.

In an s-stage RK method for a single equation we must compute each k_i, $i = 1, 2, \ldots, s$ as defined in (2.12). For a system, we have a vector of slopes so each k_i is a vector. Thus for a system an s-stage RK method is written as

$$\begin{aligned}
\mathbf{k}_1 &= \Delta t \mathbf{F}\big(t_i, \mathbf{W}_i\big) \\
\mathbf{k}_2 &= \Delta t \mathbf{F}\big(t_i + c_2 \Delta t, \mathbf{W}_i + a_{21}\mathbf{k}_1\big) \\
\mathbf{k}_3 &= \Delta t \mathbf{F}\big(t_i + c_3 \Delta t, \mathbf{W}_i + a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2\big) \\
&\vdots \\
\mathbf{k}_s &= \Delta t \mathbf{F}\big(t_i + c_s \Delta t, \mathbf{W}_i + a_{s1}\mathbf{k}_1 + a_{s2}\mathbf{k}_2 + \cdots + a_{ss-1}\mathbf{k}_{s-1}\big) \\
\mathbf{W}_{i+1} &= \mathbf{W}_i + \sum_{j=1}^{s} b_j \mathbf{k}_j \, .
\end{aligned}$$

The following example uses the Heun method, a 2-stage RK scheme given in (2.11), to approximate the solution to the IVP in the previous example.

Example 3. HEUN METHOD FOR A SYSTEM

We want to approximate the solution to the system given in the previous example using the Heun method. Recall for the Heun method the coefficients are $c_2 = 2/3$, $a_{21} = 2/3$, $b_1 = 1/4$ and $b_2 = 3/4$ so for a system we have

$$\begin{aligned} \mathbf{k}_1 &= \Delta t \mathbf{F}(t_i, \mathbf{W}_i) \\ \mathbf{k}_2 &= \Delta t \mathbf{F}(t_i + \tfrac{2}{3}\Delta t, \mathbf{W}_i + \tfrac{2}{3}\mathbf{k}_1) \\ \mathbf{W}_{i+1} &= \mathbf{W}_i + \tfrac{1}{4}\mathbf{k}_1 + \tfrac{3}{4}\mathbf{k}_2 \,. \end{aligned}$$

As in the previous example, $\mathbf{W}_0 = (-1, 0, 2)^T$ and $\mathbf{F}_i = \big(2W_{2,i} - 4t_i, -W_{1,i} + W_{3,i} - e^{t_i} + 2, W_{1,i} - 2W_{2,i} + W_{3,i} + 4t_i\big)^T$. For the first step of length $\Delta t = 0.1$ we have $\mathbf{k}_1 = 0.1(0, 4, 1)^T$ and to determine $\mathbf{k}_2$ we need to evaluate $\mathbf{F}$ at $(\frac{2}{3}(.1), \mathbf{W}_0 + \frac{2}{3}\mathbf{k}_1)$; performing this calculation gives $\mathbf{k}_2 = (.026667, .399773, .08)^T$ so that

$$\mathbf{W}_1 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 0.0 \\ 0.4 \\ 0.1 \end{pmatrix} + \frac{3}{4}\begin{pmatrix} .026667 \\ .399773 \\ .080000 \end{pmatrix} = \begin{pmatrix} -0.98000 \\ 0.39983 \\ 2.08500 \end{pmatrix}.$$

Similarly for the approximation at 0.2 we have

$$\mathbf{W}_2 = \begin{pmatrix} -0.98000 \\ 0.39983 \\ 2.08500 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 0.039966 \\ 0.395983 \\ -0.070534 \end{pmatrix} + \frac{3}{4}\begin{pmatrix} 0.066097 \\ 0.390402 \\ 0.051767 \end{pmatrix} = \begin{pmatrix} -0.92040 \\ 0.79163 \\ 2.14150 \end{pmatrix}.$$

The exact solution at $t = 0.2$ is $(-0.921061, 0.789418, 2.14246)^T$ giving an error vector of $(0.000661, .002215, .000096)^T$; calculating the standard Euclidean norm of the error and normalizing by the Euclidean norm of the solution gives 1.0166×10^{-3} which is considerably smaller than we obtained for the forward Euler. The following table provides results at $t = 1$ with the error measured in both the normalized ℓ_2 and ℓ_∞ norms. The rates of convergence clearly demonstrate quadratic convergence.

Δt	ℓ_2 Error	rate	ℓ_∞ Error	rate
1/10	$5.176\ 10^{-3}$		$5.074\ 10^{-3}$	
1/20	$1.285\ 10^{-3}$	2.01	$1.242\ 10^{-3}$	2.03
1/40	$3.198\ 10^{-4}$	2.01	$3.067\ 10^{-4}$	2.02
1/80	$7.975\ 10^{-5}$	2.00	$7.614\ 10^{-5}$	2.01

3.3 Multistep methods for systems

Recall that explicit multistep methods use values from previously calculated times to extrapolate the solution to the new point. The m-step explicit method from § 2.5 for a single IVP is

$$\begin{aligned} Y_{i+1} = \; & a_{m-1}Y_i + a_{m-2}Y_{i-1} + a_{m-3}Y_{i-2} + \cdots + a_0 Y_{i+1-m} \\ & + \Delta t \Big[b_{m-1} f(t_i, Y_i) + b_{m-2} f(t_{i-1}, Y_{i-1}) \\ & + \cdots + b_0 f(t_{i+1-m}, Y_{i+1-m}) \Big] . \end{aligned}$$

For a system of N equations the function f is now a vector $\mathbf{F}$ so we must store its value at the previous m steps. In the Adams-Bashforth or Adams Moulton methods

$a_0, a_1, \ldots, a_{m-2} = 0$ so only the solution at t_i is used. This saves additional storage because we only have to store m slope vectors and a single vector approximation to the solution. So for the system of N equations using an m-step Adams-Bashforth method we must store $(m+1)$ vectors of length N.

As a concrete example, consider the 2-step Adams-Bashforth method

$$Y_{i+1} = Y_i + \Delta t\Big[\frac{3}{2}f(t_i, Y_i) - \frac{1}{2}f(t_{i-1}, Y_{i-1})\Big]$$

for a single IVP. Using the notation of the previous section we extend the method for the system of N equations as

$$\mathbf{W}_{i+1} = \mathbf{W}_i + \Delta t\Big[\frac{3}{2}\mathbf{F}(t_i, \mathbf{W}_i) - \frac{1}{2}\mathbf{F}(t_{i-1}, \mathbf{W}_{i-1})\Big]. \tag{3.4}$$

At each step we must store three vectors $\mathbf{W}_i$, $\mathbf{F}(t_i, \mathbf{W}_i)$, and $\mathbf{F}(t_{i-1}, \mathbf{W}_{i-1})$. In the next example we apply this 2-step method to the system of the previous examples.

Example 4. ADAMS-BASHFORTH METHOD FOR A SYSTEM

To apply the 2-step Adams-Bashforth method (3.4) to the system of the previous examples we need values for $\mathbf{W}_1$ because we set $\mathbf{W}_0$ from the initial conditions. Because this method is second order we can use either a first or second order scheme to generate an approximation to $\mathbf{W}_1$. Here we use the results from the Heun method from the previous example for $\mathbf{W}_1$ and use $\Delta t = 0.1$ as before. Consequently we have

$$\mathbf{W}_0 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \quad \text{and} \quad \mathbf{W}_1 = \begin{pmatrix} -0.98000 \\ 0.39982 \\ 2.08500 \end{pmatrix}.$$

From the previous example we have $\mathbf{F}(0, \mathbf{W}_0) = (0.0, 4.0, 1.0)^T$ and $\mathbf{F}(0.1, \mathbf{W}_1)$=$(.39966, 3.95982, -.704659)^T$. Then $\mathbf{W}_2$ is given by

$$\mathbf{W}_2 = \begin{pmatrix} -0.98000 \\ 0.39982 \\ 2.08500 \end{pmatrix} + 0.1\left[\frac{3}{2}\begin{pmatrix} 0.39966 \\ 3.95983 \\ -0.70466 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 0.0 \\ 4.0 \\ 1.0 \end{pmatrix}\right] = \begin{pmatrix} -0.92005 \\ 0.79380 \\ 1.92930 \end{pmatrix}.$$

The table below tabulates the errors at $t = 1$. Of course we can only use the starting value $\mathbf{W}_1 = (. - 0.98000, 0.39982, 2.08500)^T$ as starting values for the computations at $\Delta t = 0.1$; for the other step sizes we must generate starting values at the different value of t_1. From the results we see that the rate is two, as expected.

Δt	ℓ_2 Error	rate	ℓ_∞ Error	rate
1/10	$1.346\ 10^{-2}$		$1.340\ 10^{-2}$	
1/20	$3.392\ 10^{-3}$	1.99	$3.364\ 10^{-3}$	1.99
1/40	$8.550\ 10^{-4}$	1.99	$8.456\ 10^{-4}$	1.99
1/80	$2.149\ 10^{-5}$	1.99	$2.121\ 10^{-5}$	1.99

3.4 Consistency, Stability and Convergence

For a difference scheme to give meaningful results we require the discrete solution to be close to the analytic solution in the sense that the global error at each point goes to zero as Δt approaches zero. In the previous chapters we investigated the accuracy of many of the methods we derived; specifically we obtained results of

the form $\mathcal{O}(\Delta t^r)$ for the rate of convergence. Of course in doing this, we tacitly assumed that the methods converged but this may not always be the case. Recall that for the IVP $y'(t) = -20y(t)$ the forward Euler method exhibited unbounded behavior for some choices of Δt whereas the backward Euler appeared to converge. Why does one numerical method produce reasonable results and the other produces unbounded results even though their global error is theoretically the same? The answer lies in the stability properties of the numerical scheme. It is important to realize that when we implement methods small errors are introduced due to round off and so we want to make sure that the solutions we compute remain close to the ones we would get if exact arithmetic was used. The local truncation error τ_i measures the error after one step in the exact solution of the differential equation and the exact solution of the difference method assuming both start at the same point so it only addresses the error due to the choice of discretization method for the derivative. We know that we want the global error $\frac{1}{\Delta t}\tau_i$ to approach zero as $\Delta t \to 0$ but the requirement that $\tau_i = \mathcal{O}(\Delta t^r)$ for $r \geq 2$ does not guarantee this. The additional condition needed is stability. In this section we introduce the concepts of consistency and stability which are necessary for a convergent method and investigate the numerical stability for single step and multistep methods. The literature on stability of single step and multistep methods is quite extensive; we will only touch on some of the results here but interested readers should consult standard texts in numerical ODEs for a well-developed analysis.

Any numerical scheme we use must be **consistent** with the differential equation we are approximating; the requirement of consistency is that the local truncation error is small enough that the accumulated errors must go to zero as the step size approaches zero. For example, we demonstrated that the forward Euler method for (1.2) has a local truncation error of $\mathcal{O}(\Delta t^2)$ and a global error of $\mathcal{O}(\Delta t)$ so it is consistent. If we have a difference method that has a local truncation error of $\mathcal{O}(\Delta t)$ we would expect a global error of a constant so it would not be consistent. Unfortunately, we have seen in our numerical simulations that a consistent method does not always converge.

We now want to determine how to make a consistent scheme convergent. Intuitively we know that for a scheme to be convergent the discrete solution must get closer to the exact solution as the step size reduces. To write this precisely, we form two sequences; the first is a sequence of values of Δt which approach zero monotonically such as $0.1, 0.05, 0.025, 0.0125, \ldots$ and the second is a sequence where the kth term is the maximum global error in $[t_0, T]$ where the Δt used is the value in the kth term of the first sequence. Then the method is convergent if the limit of the sequence of errors goes to zero. Formally we write that a method is **convergent** if

$$\lim_{\Delta t \to 0} \max_{1 \leq i \leq m} |y(t_i) - Y_i| = 0$$

where $m = (T - t_0)/\Delta t$; for a system we have

$$\lim_{\Delta t \to 0} \max_{1 \leq i \leq m} \|\mathbf{w}(t_i) - \mathbf{W}_i\| = 0$$

where $\|\cdot\|$ is a norm on $\mathbb{R}^N$. The reason the consistency requirement is not sufficient for convergence is that it requires the $exact$ solution to the difference equation to be

close to the exact solution of the differential equation. It does not take into account the fact that we are not computing the exact solution of the difference equation due to round off. It turns out that the additional condition that is needed is **stability** which requires the difference in the computed solution to the difference equation and its exact solution to be small. This requirement combined with consistency gives convergence.

Consistency + **Stability** = **Convergence**

Let $\widetilde{Y}_i$ represent the computed solution to the difference equation which has an actual solution of Y_i at time t_i. We want to show that the difference between $y(t_i)$ and $\widetilde{Y}_i$ is sufficiently small. At a specific t_i we have

$$|y(t_i) - \widetilde{Y}_i| = |y(t_i) - Y_i + Y_i - \widetilde{Y}_i| \leq |y(t_i) - Y_i| + |Y_i - \widetilde{Y}_i|\,,$$

where we have used the triangle inequality. Now the first term $|y(t_i) - Y_i|$ is governed by making the local truncation error sufficiently small and the second term is controlled by the stability requirement. So if each of these two terms can be made sufficiently small then when we take the maximum over all points t_i and take the limit as Δt approaches zero we get convergence. In the remainder of this section we investigate the stability of both single step and multistep methods.

3.4.1 Stability of Single Step Methods

For stability we want to know that the computed solution to the difference equation remains close to the actual solution of the difference equation and so does not grow in an unbounded manner. We first look at stability of the differential equation for the model problem

$$y'(t) = \lambda y \quad 0 < t \leq T, \lambda \in \mathbb{C}, \tag{3.5}$$

with the initial condition $y(0) = y_0$ and solution $y(t) = y_0 e^{\lambda t}$. Note that in general λ is a complex number but to understand why we look at this particular problem first consider the case when λ is real. If $\lambda > 0$ then small changes in the initial condition can result in the solutions becoming far apart. For example, if we have two initial conditions say $y_1(0) = \alpha$ and $y_2(0) = \beta$ which differ by $\delta = |\beta - \alpha|$ then the solutions $y_1 = \alpha e^{\lambda t}$ and $y_2 = \beta e^{\lambda t}$ differ by $\delta e^{\lambda t}$. Consequently, for large $\lambda > 0$ these solutions can differ dramatically as illustrated in the table below for various choices of δ and λ; however, if $\lambda < 0$ the term $\delta e^{\lambda t}$ approaches zero as $t \to 0$. Therefore for stability of this model IVP when λ is real we require $\lambda < 0$.

λ	δ	$\lvert y_1(0.5) - y_2(0.5)\rvert$	$\lvert y_1(1) - y_2(1)\rvert$	$\lvert y_1(10) - y_2(10)\rvert$
1	0.01	0.0165	0.0272	220
1	0.1	0.165	0.272	2203
10	0.01	1.48	220	10^{41}
10	0.1	14.8	2203	10^{42}
-1	0.1	$6.07\ 10^{-2}$	$3.68\ 10^{-2}$	$4.54\ 10^{-6}$
-10	0.1	$6.73\ 10^{-4}$	$4.54\ 10^{-6}$	10^{-45}

In general, λ is complex so it can be written as $\lambda = \alpha + i\beta$ where α, β are real numbers and $i = \sqrt{-1}$. The exact solution is

$$y(t) = y_0 e^{\lambda t} = y_0 e^{\alpha t + i\beta t} = y_0 e^{\alpha t} e^{i\beta t}\,.$$

Now $e^{i\beta t} = \cos(\beta t) + i\sin(\beta t)$ so this term does not grow in time; however the term $e^{\alpha t}$ will grow in an unbounded manner if $\alpha > 0$. Consequently we say that the differential equation $y' = \lambda y$ is stable when the real part of λ is less than or equal to zero, i.e., $\mathrm{Re}(\lambda) \le 0$ or λ is in the left half of the complex plane.

When we approximate our model IVP (3.5) we want to know that small changes, such as those due to round off, do not cause large changes in the solution. Here we are going look at stability of a difference equation of the form

$$Y_{i+1} = \zeta(\lambda\Delta t) Y_i \tag{3.6}$$

applied to the model problem (3.5). Our single step methods fit into this framework. For example, for the forward Euler method applied to the differential equation $y' = \lambda y$ we have $Y_{i+1} = Y_i + \Delta t \lambda Y_i$ so $\zeta(\lambda\Delta t) = 1 + \Delta t\lambda$. For the backward Euler method we have $Y_{i+1} = Y_i + \Delta t \lambda Y_{i+1}$ so $\zeta(\lambda\Delta t) = 1/(1 - \lambda\Delta t)$. For explicit RK methods $\zeta(z)$ will be a polynomial in z and for implicit RK methods it will be a rational function. We will address stability of multistep methods later.

We apply the difference equation (3.6) recursively to get

$$Y_i = \zeta(\lambda\Delta t) Y_{i-1} = \zeta^2(\lambda\Delta t) Y_{i-2} = \cdots = \zeta^i(\lambda\Delta t) Y_0$$

so we can view ζ as an *amplification factor* because the solution at time t_{i-1} is amplified by a factor of ζ to get the solution at t_i, the solution at time t_{i-2} is amplified by a factor of ζ^2 to get the solution at t_i, etc. We know that the magnitude of ζ must be less than or equal to one or else Y_i will become unbounded. This condition is known as **absolute stability**. There are many other definitions of different types of stability; some of these are explored in the exercises.

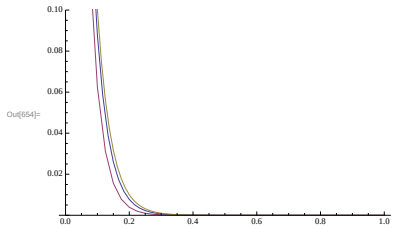
Definition 1. The region of absolute stability for the difference equation (3.6) is $\{\lambda\Delta t \in C \mid |\zeta(\lambda\Delta t)| \le 1\}$. A method is called A**-stable** if $|\zeta(\lambda\Delta t)| \le 1$ for the entire left half plane.

Example 5. We want to determine if the forward Euler method and the backward Euler method are A-stable; if not, we want to determine the region of absolute stability. We then discuss our previous numerical results for $y'(t) = -20y(t)$ in light of these results.

For the forward Euler method $\zeta(\lambda\Delta t) = 1+\lambda\Delta t$ so the condition for A-stability is that $|1+\lambda\Delta t| \leq 1$ for the entire left plane. Now λ is, in general, complex which we can write as $\lambda = \alpha + i\beta$ but let's first look at the real case, i.e., $\beta = 0$. Then we have

$$-1 \leq 1 + \lambda\Delta t \leq 1 \Rightarrow -2 \leq \lambda\Delta t \leq 0$$

so on the real axis we have the interval $[-2, 0]$. This says that for a fixed real $\lambda < 0$, Δt must satisfy $\Delta t \leq 2/|\lambda|$ and thus the method is not A-stable but has a region $[-2, 0]$ of absolute stability if λ is real. If $\beta \neq 0$ then we have the region of stability as a circle in the complex plane of radius one centered at -1. For example, when $\lambda = -20$ Δt must satisfy $\Delta t \leq 0.1$. In Figure **??** we plotted results for $\Delta t = 1/4$ and $1/8$ which do not satisfy the stability criteria. In the figure below we plot approximations to the same problem using $\Delta t = 1/20$, $1/40$ and $1/60$. As you can see from the graph, the solution appears to be converging.



For the backward Euler method $\zeta(\lambda\Delta t) = 1/(1 - \lambda\Delta t)$. To determine if it is A-stable we see if it satisfies the stability criteria for the entire left plane. As before, we first find the region when λ is real. For $\lambda \leq 0$ have $1 - \lambda\Delta t \geq 1$ so that $\zeta(\lambda\Delta t) \leq 1$ for all Δt and we have the entire left plane. The backward Euler method is A-stable so any choice of Δt provides stable results for $\lambda < 0$.

To be precise, the region of absolute stability for the backward Euler method is actually the region outside the circle in the complex plane centered at one with radius one. Clearly, this includes the left half plane. To see this, note that when $\lambda\Delta t \geq 2$ then $|1/(1 - \lambda\Delta t)| \leq 1$. However, we are mainly interested in the case when $Re(\lambda) < 0$ because the differential equation $y'(t) = \lambda y$ is stable.

Example 6. In this example we investigate the regions of absolute stability for the explicit 2-stage Heun method

$$Y_{i+1} = Y_i + \frac{\Delta t}{4}\left[f(t_i, Y_i) + 3f\big(t_i + \frac{2}{3}\Delta t, Y_i + \frac{2}{3}\Delta t f(t_i, Y_i)\big)\right].$$

We have written the scheme as a single equation rather than the standard way of specifying k_i because this makes it easier to determine the amplification factor.

We apply the difference scheme to the model problem $y' = \lambda y$ where $f(t, y) = \lambda y(t)$ to get

$$Y_{i+1} = Y_i + \frac{\Delta t}{4}\left[\lambda Y_i + 3\lambda(Y_i + \frac{2}{3}\Delta t\lambda Y_i)\right] = \left[1 + \frac{1}{4}(\lambda\Delta t) + \frac{3}{4}(\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2\right]Y_i$$

so $\zeta(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2$. The region of absolute stability is all points z in the complex plane where $|\zeta(z)| \leq 1$. If λ is real and non-positive we have

$$-1 \leq 1 + z + \frac{z^2}{2} \leq 1 \Rightarrow -2 \leq z(1 + \frac{z}{2}) \leq 0\,.$$

For $\lambda \leq 0$ so that $z = \lambda\Delta t \leq 0$ we must have $1 + \frac{1}{2}\lambda\Delta t \geq 0$ which says $\Delta t\lambda \geq -2$. Thus the region of stability is $[-2, 0]$ when λ is real and when it is complex we have a circle of radius one centered at -1. This is the same region as the one computed for the forward Euler method.
In Figure 3.1 numerical results are presented for the case when $\lambda = -20$. For this choice of λ the stability criteria becomes $\Delta t \leq 0.1$.

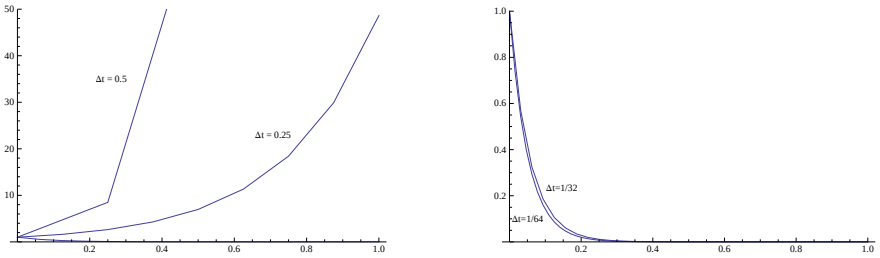


Figure 3.1: Approximations for the IVP $y'(t) = -20y$, $y(0) = 1$ using the Heun method. For the plot on the left the step size Δt is too large and numerical instability is occurring. When we reduce the step size the method converges as the graph on the right demonstrates.

It can be shown that there is no *explicit* RK method that has an unbounded region of absolute stability such as the left half plane region of stability that we got for the backward Euler method. In general, implicit methods do not have stability restrictions so this is one reason that we need implicit methods. Implicit methods will be especially important when we study initial boundary value problems.

Oftentimes we have a system of first order IVPs or we have a higher order IVP which we first write as a system of first order IVPs. We want to extend our definition of absolute stability to a system but we first look at stability for the differential equations themselves. Analogous to the problem $y'(t) = \lambda y$ we consider the linear model problem

$$\mathbf{w}'(t) = A\mathbf{w}$$

for an $N \times N$ system of IVPs where A is an $N \times N$ matrix. Consider first the simple case where A is a diagonal matrix and the equations are uncoupled so basically we have the same situation as a single equation. Thus the stability criteria is that the real part of each diagonal entry must be less than or equal to zero. But the diagonal entries of a diagonal matrix are just its N eigenvalues λ_i[1] counted according to multiplicity. So an equivalent statement of stability when A is diagonal is that $\text{Re}(\lambda_i) < 0$, $i = 1, 2, \ldots, N$; it turns out that this is the stability criteria for a general matrix A. Recall that even if the entries of A are real the eigenvalues may be complex. If A is symmetric we are guaranteed that the eigenvalues are real. If we have the general system (3.2) where $f_i(t, \mathbf{w})$ is not linear in $\mathbf{w}$, then the condition becomes one on the eigenvalues of the Jacobian matrix for f; the (i, j) entry of the Jacobian is $\partial f_i / \partial w_j$.

[1]The eigenvalues of an $N \times N$ matrix A are scalars λ such that $A\mathbf{x} = \lambda\mathbf{x}$; the vector $\mathbf{x}$ is called the eigenvector corresponding to the eigenvalue λ.

Now if we apply the forward Euler method to the system $\mathbf{w}'(t) = A\mathbf{w}$ where the entries of A are a_{ij} then we have the system

$$\mathbf{W}_{i+1} = \begin{pmatrix} 1+\Delta t a_{11} & \Delta t a_{12} & \Delta t a_{13} & \cdots & \Delta t a_{1N} \\ \Delta t a_{21} & 1+\Delta t a_{22} & \Delta t a_{23} & \cdots & \Delta t a_{2N} \\ & \ddots & \ddots & & \\ & \cdots & \cdots & \Delta t a_{N,N-1} & 1+\Delta t a_{N,N} \end{pmatrix} \mathbf{W}_i$$

The condition on Δt is determined by choosing it so that all the eigenvalues of the matrix have real parts less than zero. If the system is not linear, then the condition is on the eigenvalues of the Jacobian matrix.

3.4.2 Stability of Multistep Methods

The numerical stability of a single step method depends on the initial condition y_0 but in a m-step multistep method there are $m-1$ other starting values $Y_1, Y_2, \ldots, Y_{m-1}$ which are obtained by another method such as a RK method. In 1956 Dahlquist[2] published a seminal work formulating criteria for the stability of linear multistep methods. We will give an overview of the results here.

We first rewrite the m-step multistep method (2.19) by shifting the indices to get

$$\begin{aligned} Y_{i+m} &= a_{m-1}Y_{i+m-1} + a_{m-2}Y_{i+m-2} + a_{m-3}Y_{i+m-3} + \cdots + a_0 Y_i \\ &\quad + \Delta t \Big[b_m f(t_{i+m}, Y_{i+m}) + b_{m-1} f(t_{i+m-1}, Y_{i+m-1}) \\ &\quad + b_{m-2} f(t_{i+m-2}, Y_{i+m-2}) + \cdots + b_0 f(t_i, Y_i) \Big] \end{aligned}$$

or equivalently

$$Y_{i+m} - \sum_{j=0}^{m-1} a_j Y_{i+j} = \Delta t \sum_{j=0}^{m} b_j f(t_{i+j}, Y_{i+j}) \,.$$

As before, we apply it to the model IVP $y' = \lambda y$, $y(0) = y_0$ for $Re(\lambda) < 0$ which guarantees the IVP itself is stable. Substituting $f = \lambda y$ into the difference equation gives

$$Y_{i+m} - \sum_{j=0}^{m-1} a_j Y_{i+j} = \Delta t \sum_{j=0}^{m} b_j \lambda Y_{i+j} \,.$$

Recall that a technique for solving a linear homogeneous ODE such as $y''(t) + 2y'(t) - y(t) = 0$ is to look for solutions of the form $y = e^{rt}$ and get a polynomial equation for r such as $e^{rt}(r^2 + 2r - 1) = 0$ and then determine the roots of the equation. We take the analogous approach for the difference equation and seek a solution of the form $Y_i = z^i$. Substitution into the difference equation yields

$$z^{i+m} - \sum_{j=0}^{m-1} a_j z^{i+j} = \Delta t \sum_{j=0}^{m} b_j \lambda z^{i+j} \,.$$

[2]Germund Dahlquist (1925-2005) was a Swedish mathematician.

Canceling the lowest order term z^i gives a polynomial equation in z which is a function of λ and Δt resulting in the stability equation

$$Q(\lambda\Delta t) = z^m - \sum_{j=0}^{m-1} a_j z^j - \Delta t \sum_{j=0}^{m} b_j \lambda z^j = \rho(z) - \Delta t \lambda \sigma(z)\,,$$

where

$$\rho(z) = z^m - \sum_{j=0}^{m-1} a_j z^j \quad \text{and} \quad \sigma(z) = \sum_{j=0}^{m} b_j z^j\,. \tag{3.7}$$

For stability, we need the roots of $\rho(z)$ to have magnitude ≤ 1 and if a root is identically one then it must be a simple root. If this root condition is violated, then the method is unstable so a simple check is to first see if the root condition is satisfied; if the root condition is satisfied then we need to find the region of stability. To do this, we find the roots of $\mathcal{Q}(\lambda\Delta t)$ and require that each root has magnitude less than or equal to one. To simplify the calculations we rewrite $\mathcal{Q}(\lambda\Delta t)$ as

$$\begin{aligned}\mathcal{Q}(\lambda\Delta t) &= z^m(1-\lambda\Delta t b_m) - z^{m-1}(a_{m-1} + b_{m-1}\lambda\Delta t) \\ &\quad -z^{m-2}(a_{m-2} + b_{m-2}\lambda\Delta t) - \cdots - (a_0 + b_0\lambda\Delta t)\,.\end{aligned}$$

The following two examples determine the region of stability using this approach.

Example 7. In this example we investigate the stability of the forward and backward Euler methods by first demonstrating that the root condition for $\rho(z)$ is satisfied and then finding the region of absolute stability; we confirm that we get the same results as before.

The forward Euler method is written as $Y_{i+1} = Y_i + \Delta t f(t_i, Y_i)$ so in the form of a multistep method with $m = 1$ we have $a_0 = 1$, $b_0 = 1$, $b_1 = 0$ and thus $\rho(z) = z - 1$ whose root is $z = 1$ so the root condition is satisfied. To find the region of absolute stability we have $\mathcal{Q}(\lambda\Delta t) = z - (1 + \lambda\Delta t)$ which has a single root $1 + \lambda\Delta t$; thus the region of absolute stability is $|1 + \lambda\Delta t| \leq 1$ which is the condition we got before by analyzing the method as a single step method.

For the backward Euler method $a_0 = 1$, $b_0 = 0$, $b_1 = 1$ and so $\rho(z) = z - 1$ which has the root $z = 1$ and so the root condition is satisfied. To find the region of absolute stability we have $\mathcal{Q}(\lambda\Delta t) = z(1 - \lambda\Delta t) - 1$ which has a single root $1/(1 - \lambda\Delta t)$ and we get the same restriction that we got before by analyzing the method as a single step method.

Example 8. In this example we want to show that the 2-step Adams-Bashforth method

$$Y_{i+1} = Y_i + \frac{\Delta t}{2}\left[3f(t_i, Y_i) - f(t_{i-1}, Y_{i-1})\right]$$

is stable.

For this Adams-Bashforth method we have $m = 2$, $a_0 = 0$, $a_1 = 1$, $b_0 = -1/2$, $b_1 = 3/2$, and $b_2 = 0$. The characteristic polynomial is $\rho(z) = z^2 - z = z(z - 1)$ whose two roots are $z = 0, 1$ and the root condition is satisfied.

In summary, we have seen that some methods can be unstable if the step size Δt is too large (such as the forward Euler method) while others are stable even

for a large choice of Δt (such as the backward Euler method). In general, explicit methods have stability restrictions whereas implicit methods are stable for all step sizes. Of course, one must have a small enough step size for accuracy. We have just touched on the ideas of stability of numerical methods for IVPs; the interested reader is referred to standard texts in numerical analysis for a thorough treatment of stability. The important concept is that we need a consistent and stable method to guarantee convergence of our results.

3.5 Stiff Systems

Some differential equations are more difficult to solve than others. We know that for problems where the solution curve varies a lot, we should take a small step size and where it changes very little a larger step size should be used for efficiency. If the change in the solution curve is relatively small everywhere then a uniform step size is the most efficient approach. This all seems very heuristic. However, there are problems which require a very small step size even when the solution curve is very smooth. There is no universally accepted definition of stiff differential equations but typically the solution curve changes rapidly and then tends towards a slowly-varying solution. Because the stability region for implicit methods is typically much larger than explicit methods, most stiff equations are approximated using an implicit method.

To illustrate the concept of stiffness we look at a single IVP which is considered stiff. The example is from a combustion model and is due to Shampine (2003) who is one of the authors of the Matlab ODE suite. The idea is to model flame propagation as when you light a match. We know that the flame grows rapidly initially until it reaches a critical size which is dependent on the amount of oxygen. We assume that the flame is a ball and $y(t)$ represents its radius; in addition we assume that the problem is normalized so that the maximum radius is one. We have the IVP

$$y'(t) = y^2(1-y) \quad 0 < t \leq \frac{2}{\delta}; \qquad y(0) = \delta \tag{3.8}$$

where $\delta << 1$ is the small given initial radius. At ignition the solution y increases rapidly to a limiting value of one; this happens quickly on the interval $[0, 1/\delta]$ but on the interval $[1/\delta, 2/\delta]$ the solution is approximately equal to one. Knowing the behavior of the problem suggests that we should take a small step size initially and then on $[1/\delta, 2/\delta]$ where the solution is almost constant we should be able to take a large step size. However, if we use the RKF45 method with an automatic step size selector, then we can capture the solution on $[0, 1/\delta]$ but on $[1/\delta, 2/\delta]$ the step size is reduced by so much that the minimum allowable step size is surpassed and the method often fails if the minimum step size is set too large. Initially the problem is not stiff but it becomes stiff as its approaches the value one, its steady state solution. The term "stiff" was used to described this phenomena because it was felt the steady state solution is so "rigid".

When one has a system of equations like (3.2) the stiffness of the problem depends upon the eigenvalues of the Jacobian matrix. Recall that we said we need

all eigenvalues to have real part less than zero for stability. If the Jacobi matrix has eigenvalues which have a very large negative real part and eigenvalues with a very small negative real part, then the system is stiff and special care must be used to solve it. You probably don't know a priori if a system is stiff but if you encounter behavior where the solution curve is not changing much but you find that your step size needs to be smaller and smaller, then your system is probably stiff. In that case, an implicit method is typically used.

EXERCISES

1. Convert each IVP into a system of first order IVPs.

   a. $y''(t) + 6y'(t) - \frac{1}{2}y(t) = 4 \qquad y(0) = 1, y'(0) = -3$

   b. $y'''(t) - 2y'(t) + y^2(t) = 0 \qquad y(0) = 0, y'(0) = 1, y''(0) = 4$

2. Determine the amplification factor for the Midpoint method (2.3). Then determine the region of absolute stability.

3. Show that all members of Adams multistep methods (both explicit and implicit) are stable.