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 will evolve 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 when we know the initial size and have information on how the population grows with time, i.e., we have information on the first derivative. In addition to being important in its own right, our understanding of this 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 will 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 of the unknown is specified. Later, in Chapter 3 we will consider higher order IVPs. Before proceeding to approximating the solution of this prototype IVP we will investigate conditions which guarantee the 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 our discrete solution includes more and more points then it 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 § 1.3.1 we demonstrate the linear convergence of the method by introducing the concepts of local truncation error and global error. The important concepts of explicit and implicit methods are illustrated by comparing the forward and backward Euler methods. In § 1.3.2 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 that the Euler method converges linearly and computationally we need methods which converge faster. In addition, we will see an example where the forward Euler method fails to converge at all so clearly other methods are needed.

1.1 Prototype Initial Value Problem

One of the first differential equations 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. For example, the following represent three IVPs for $y(t)$:

$$\begin{array}{rcl} \dfrac{dy}{dt} &=& \sin \pi t \\ y(0) &=& 0 \end{array} \qquad \begin{array}{rcl} \dfrac{dy}{dt} &=& -y + t^2 \\ y(2) &=& 1 \end{array} \qquad \begin{array}{rcl} \dfrac{dy}{dt} &=& e^{-.1ty} \\ y(0) &=& 4 \end{array}$$

These differential equations are called *first order* because the highest derivative appearing in the equation is the first derivative of $y(t)$. All of these problems are special cases of the following general IVP; find $y(t)$ satisfying

$$\frac{dy}{dt} = f(t, y) \quad t_0 < t \leq T \tag{1.1a}$$

$$y(t_0) = y_0 \,, \tag{1.1b}$$

where $f(t, y)$ is the given derivative of $y(t)$ and y_0 is the known value at the initial time t_0.

One of the simplest occurrences of this differential equation is when $f = f(t)$, i.e., f is a function of t and not both t and y. In this case we can often find the antiderivative[1] of $f(t)$ by integrating both sides of the equation. As an example consider the specific $f(t) = \sin(\pi t)$. Then integrating both sides of the equation with respect to t gives

$$\int \frac{dy}{dt}\, dt = \int \sin(\pi t)\, dt$$

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

and using our knowledge from calculus we have

$$y(t) + C_1 = -\frac{1}{\pi}\cos(\pi t) + C_2 \Rightarrow y(t) = -\frac{1}{\pi}\cos(\pi t) + C$$

for arbitrary constants C_1, C_2. Note that our general solution to this differential equation involves an arbitrary constant which we can determine by imposing the initial condition (1.1b). For example, if we impose $y(0) = 0$ we have $y(t) = -\frac{1}{\pi}\cos(\pi t) + \frac{1}{\pi}$ because C is determined by

$$y(0) = 0, y(t) = -\frac{1}{\pi}\cos(\pi t) + C \Rightarrow y(0) = -\frac{1}{\pi}\cos(0) + C = 0 \Rightarrow C = \frac{1}{\pi}\,.$$

For this choice of $f(t, y) = \sin(\pi t)$ 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* will give the answer 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. Also when $f(t, y)$ is a function of both y and t our technique of integrating the equation often fails to work. Although there are many other 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; that is, we have to 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 (1.1) 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 error in our calculations could produce meaningless results. In this situation we say the IVP is *ill-posed* or *ill-conditioned*, a condition 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.

Definition 1. *The IVP (1.1) is well-posed if the solution exists, is unique and depends continuously on the data y_0 and $f(t, y)$. Otherwise it is called ill-posed.*

The conditions that guarantee well-posedness of a solution to (1.1) are well known and are presented in Theorem 1. Basically the theorem requires that the derivative of $y(t)$ (given by $f(t, y)$) be continuous and that 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

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

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; this can be summarized in the following definition.

Definition 2. *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*

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

where 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 . The following example illustrates how to determine the Lipschitz constant for a differentiable function.

Example 1. Consider the function $g(x) = x^2$ where we first choose $D = [0, 4]$. We know that $g(x)$ is continuous but we want to determine if it is Lipschitz continuous on D and if it is, find the Lipschitz constant. Note that from Definition 2 we can write the condition as

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

and we note that this is just the slope of the secant line joining $(x_1, g(x_1))$ and $(x_2, g(x_2))$. An easy way to determine the Lipschitz constant if $g(x)$ is differentiable is to find a constant such that $|g'(x)| \leq L$ for all $x \in D$. This is because a bounded first derivative on D implies that the function is Lipschitz continuous there; the proof of this follows from the Mean Value Theorem of calculus. In our case $g'(x) = 2x$ which is an increasing function on D whose maximum value is attained at $x = 4$ so $L = 8$ is a Lipschitz constant for $g(x) = x^2$ on $[0, 4]$.

However, if we take $D = [0, \infty)$ then $g(x)$ is not Lipschitz continuous on D because the derivative $g'(x)$ becomes arbitrarily large. We say that $g(x) = x^2$ is locally Lipschitz but not globally Lipschitz.

We said the function $g(x) = \sqrt{x}$ was not Lipschitz continuous on $[0, 1]$. This is because the derivative $g'(x) = 1/(2\sqrt{x})$ becomes arbitrarily large as $x \to 0$ and thus we can find no L which satisfies Definition 2.

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 our existence and uniqueness result, we need $f(t, y)$ to be Lipschitz continuous in y so we need to extend our definition because f is now a function of two variables.

Definition 3. *A function $g(x, y)$ defined on a domain $D \subset \mathbb{R}^2$ is Lipschitz continuous in D for the variable y if for any $(x, y_1), (x, y_2) \in D$ there is a constant L such that*

$$|g(x, y_1) - g(x, y_2)| \leq L|y_1 - y_2| \,. \tag{1.2}$$

L is called the Lipschitz constant.

We are now ready to state the theorem which guarantees existence and uniqueness of a solution to (1.1) 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 $[t_0, T]$ whereas $f(t, y)$ must be defined on a domain in $\mathbb{R}^2$. Specifically the first argument $t \in [t_0, T]$ and 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. *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 (1.2). Then the IVP (1.1) has a unique solution in D and moreover, the problem is well-posed.*

In the sequel we will only consider IVPs which have a unique solution which depends continuously on the data.

1.2 Discretization

Our goal is to derive a *difference equation* which is an approximation to the differential equation (1.1a) that involves only differences in function values, i.e., there are no derivatives in the equation. 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.

To determine a discrete solution to the IVP (1.1) we first discretize the time domain $[t_0, T]$. For now we use $N + 1$ evenly spaced points t_i, $i = 0, 1, 2, \ldots, N$

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

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 and more calculations with N getting larger, or equivalently $\Delta t \to 0$, our approximate solution will approach the exact solution in some sense. In Figure 1.1 we plot an exact solution and

three discrete solutions using $\Delta t = 0.5, 0.25$ and $\Delta t = 0.125$. By observing the plot and using the "eyeball norm" we believe 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 is converging to the exact solution.

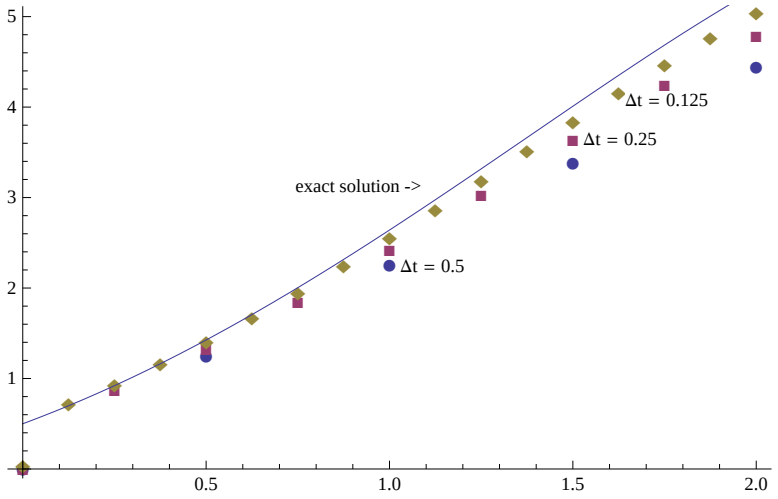

Figure 1.1: The exact solution to an IVP is shown as a solid line because it is continuous; three approximations using decreasing values of the step size Δt are plotted. Note that these approximations are discrete functions because they are only known at specific values of t

We now want to obtain discrete methods for approximating the solution to the IVP (1.1). Once we derive a method, we need to see how to analyze it to determine its theoretical accuracy; then we will look at implementation issues. For now, we assume that the methods converge to the solution of the IVP but in Chapter 3 we will address the issue of convergence. When we have a working code for the method we will demonstrate that the discrete solution converges at the predicted analytic rate for a problem whose exact solution is known. This is accomplished by approximating the solution for decreasing values of Δt. Once this is done, we will have confidence in applying our algorithm to problems for which an exact solution is not known.

We will see that there are many approaches to deriving discrete methods for our IVP. The most obvious approach is to approximate the derivative appearing in the equation. In the following section we consider such a method.

1.3 Euler Method

The simplest method for approximating the solution of (1.1) 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. For example, if our difference equation at t_1 involves our initial value $y_0 = y(t_0)$, our approximation at t_1 plus the known function f at (t_0, y_0) then we can solve for our approximation to $y(t_1)$. Once this is done, we use this value and repeat the process at t_2 to obtain an approximation there; 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 lead to ways of 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

$$y'(a) = \lim_{h \to 0} \frac{y(a+h) - y(a)}{h} \,. \tag{1.3}$$

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 (1.3) 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 (1.3) we set $h = \Delta t$, $a = t_0$ and let $t_1 = t_0 + \Delta t$. The difference quotient

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

represents an approximation to $y'(t_0)$. To write a difference equation we let Y_i denote our approximation to $y(t_i)$; clearly $Y_0 = y_0$ which is the given initial condition (1.1b). To obtain the Euler method at t_1 we use our differential equation $y'(t) = f(t, y(t))$ evaluated at t_0 along with our approximation to $y'(t_0)$ to get

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

We have a starting point 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. In general we have

$$Y_{i+1} = Y_i + \Delta t f(t_i, Y_i) \,. \tag{1.4}$$

This method is called the **forward Euler method** and can be written as the following general algorithm. 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}.

Algorithm 1. The Forward Euler Method
Given t_0, T, y_0, N, $f(t, y)$;
Initialize: set $\Delta t = (T - t_0)/N$, $t = t_0$ *and* $Y = y_0$
For $i = 1, 2, \ldots, N$
evaluate $f(t, Y)$
set $Y = Y + \Delta t f(t, Y)$
increment t: $t = t + \Delta t$

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 must be stored at any time because we can overwrite Y_i with Y_{i+1} as we indicated in Algorithm 1. If we need to save the values for plotting, then Y_{i+1} can simply be written to a file once it is computed. The numerical results in Figure 1.1 were obtained using the forward Euler method. In § 1.3.2 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 (1.3); 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 (1.3) we set $a + h = t_0$ and $a = t_1$ 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 $t_0 - t_1 < 0$. For a general point i, we have

$$Y_{i+1} = Y_i + \Delta t f(t_{i+1}, Y_{i+1})\,. \tag{1.5}$$

This method is called the **backward Euler method** because we are writing the equation at t_{i+1} and differencing backwards in time. It is important to realize that this method is inherently different from (1.4) because we must evaluate $f(t, y)$ at the unknown point (t_{i+1}, Y_{i+1}). In general, this leads to a nonlinear equation to solve for each Y_{i+1} 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. We distinguish between these two types of methods. The forward Euler scheme given in Algorithm 1 is called an **explicit** scheme because we can write the unknown explicitly in terms of known values whereas the backward Euler method in (1.5) is called an **implicit** scheme because the unknown is written implicitly in terms of known values and itself. So if we were to modify Algorithm 1 for the forward Euler method to

incorporate the backward Euler scheme then we would have to add an interior loop to solve the resulting nonlinear equation with a method such as Newton's Method. In § 1.3.2 we look at examples where we employ both the forward and backward Euler methods. However, in practice we will use implicit methods 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.

1.3.1 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 1.1 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 $y'(t)$; i.e., we have solved a difference equation rather than the differential 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.

Definition 4. *The* **global discretization error** *at a point t_i is the magnitude of the difference in the exact solution and its approximation; i.e., $|y(t_i) - Y_i|$.*

Definition 5. *The* **local truncation error** *at a point t_i is the error made it taking one step of a 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 by a difference equation.*

A comparison of the global error and the truncation error for the forward Euler method is illustrated in Figure 1.3. 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.

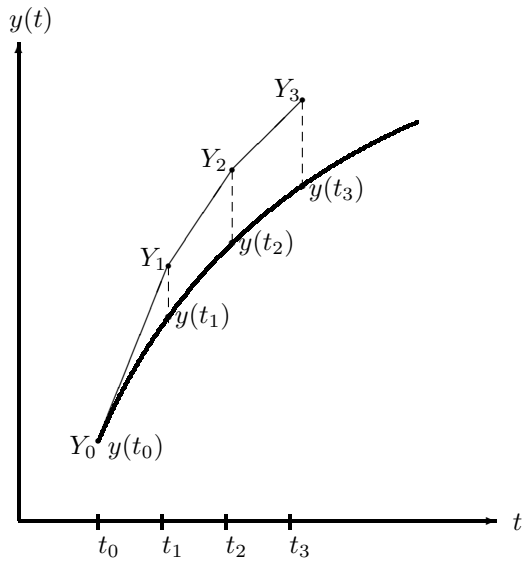

Figure 1.2: The exact solution and the discrete solution agree at t_0. At t_1 the error $|Y_1 - y(t_1)|$ is due to approximating the derivative in the ODE by a difference quotient. At t_2 the error $|Y_2 - y(t_2)|$ is due to approximating the derivative in the ODE and the fact that the starting value, Y_1 does not lie on the solution curve as Y_0 did.

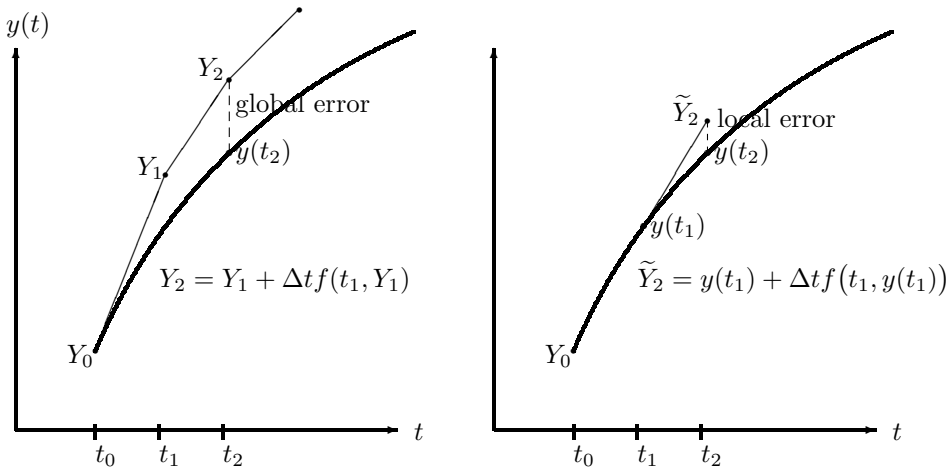

Figure 1.3: A comparison of the global error and the local truncation error at t_2. The global error is the total error made whereas the local truncation error is the error due to the discretization of the differential equation.

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+1} and the result of applying the Euler method where we use $y(t_i)$ instead of Y_i. Specifically we calculate

$$\tau_{i+1} = |y(t_{i+1}) - \widetilde{Y}_{i+1}| \quad \text{where } \widetilde{Y}_{i+1} = y(t_i) + \Delta t f\big(t_i, y(t_i)\big)\,. \tag{1.6}$$

Our strategy is to first quantify the local truncation error in terms of Δt; i.e., determine $\tau_{i+1} = |y(t_{i+1}) - \widetilde{Y}_{i+1}|$ and then use the result to determine the global error. We consider the expression for the local truncation error

$$\tau_{i+1} = \big|y(t_{i+1}) - \widetilde{Y}_{i+1}\big| = \big|y(t_{i+1}) - y(t_i) - \Delta t f\big(t_i, y(t_i)\big)\big|\,.$$

In order to to simplify this expression so we get some cancellation of terms, we use a Taylor Series with remainder for the exact solution $y(t_{i+1})$ because the expansion is in terms of $y(t_i)$ and its derivatives. Recall that the Taylor series expansion with remainder for a differentiable function $g(x)$ in the neighborhood of $x = a$ is

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

but we will typically use it in the form

$$\begin{aligned} g(t+\Delta t) &= g(t) + g'(t)\Delta t + \frac{g''(t)}{2!}(\Delta t)^2 + \frac{g'''(t)}{3!}(\Delta t)^3 + \cdots \\ &\quad + \frac{g^{[n]}(t)}{n!}(\Delta t)^n + \frac{g^{[n+1]}(\xi)}{(n+1)!}(\Delta t)^{n+1} \quad \xi \in (t, t+\Delta t) \end{aligned} \tag{1.7}$$

where we have set $t + \Delta t = x$ and $t = a$ so $x - a = \Delta t$. Setting $t = t_i$ and $n = 1$ in (1.7) gives an expansion for $y(t_i + \Delta t) = y(t_{i+1})$

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

so substitution into our expression for $y(t_{i+1}) - \widetilde{Y}_{i+1}$ yields

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

where we have used the differential equation $y'(t_i) = f(t_i, y(t_i))$. Thus we have the local truncation error τ_{i+1} at the point t_{i+1} given by

$$\tau_{i+1} = \big|y(t_{i+1}) - \widetilde{Y}_{i+1}\big| = \frac{(\Delta t)^2}{2!}\big|y''(\xi_i)\big|\,. \tag{1.8}$$

If $y''(t)$ is bounded on $[0,T]$, say $|y''(t)| \le M$, then $\tau_{i+1} \le (\Delta t)^2 \frac{M}{2}$. 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 could find a formula for the exact error, then we could determine this and add it to the approximation to get the exact solution. Our goal now 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 2 provides a formal statement and proof for the global error of the forward Euler method. Note that one hypothesis of Theorem 2 is that $f(t,y)$ is Lipschitz continuous in y which is also the hypothesis of Theorem 1 which guarantees existence and uniqueness of the solution to the IVP (1.1) 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 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 (1.2) 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 (1.8). Then the global error at each point t_i satisfies

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

so that the forward Euler method converges linearly.

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

$$E_n \le K\,E_{n-1} + \tau \tag{1.9}$$

where K is the constant $K = 1 + \Delta t L$ and $\tau = \max_i \tau_i$, i.e., the largest value that τ_i given in (1.8) takes on. If we can do this, then the proof follows easily.

To see this note that a common approach in error analysis is to apply a formula recursively; in our case we obtain

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

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

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

where we have used $K = 1 + \Delta tL$. Now n is the number of steps from t_0 so $n\Delta t = t_n \leq T$ where T is our final time. Also for each i

$$\tau_{i+1} = \frac{\Delta t^2}{2} \big| y''(\xi_i) \big| \leq M \frac{\Delta t^2}{2}$$

where we have used the bound on $y''(t)$ given in the hypothesis of the theorem. Combining these gives the final result that the global error is linear in Δt

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

All that is left for the proof is to to demonstrate that (1.9) 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

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

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

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

□

In the following section we look at some specific examples of the IVP (1.1) 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.

1.3.2 Numerical examples

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. Then we look at the slightly more complicated model of logistic growth. We will then apply our forward Euler method to approximate the solutions of each type of problem. Lastly we will look at an example of exponential growth and apply both the forward and backward Euler methods. For this example, we will see that the backward Euler performs well but the forward Euler method does not converge. Reasons for this will be discussed in Chapter 3.

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. 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

$$p'(t) = r_0 p(t) \quad t \in (t_0, T] \tag{1.10}$$

along with the initial condition

$$p(0) = p_0$$

where r_0 is the given proportionality constant. This is one of those differential equations that we can solve exactly by integration. We first separate the variables (i.e., move all terms involving p to one side of the equation and all terms involving t to the other) to obtain

$$\frac{dp}{dt} = r_0 p \Rightarrow \frac{dp}{p} = r_0\, dt \Rightarrow \int_0^t \frac{dp}{p} = r_0 \int_0^t dt$$

and perform the integration to get

$$\ln p(t)|_0^t = r_0(t-0) \Rightarrow \ln p(t) - \ln p_0 = r_0 t \Rightarrow e^{\ln p} = e^{r_0 t} e^{\ln p_0} \Rightarrow p(t) = p_0 e^{r_0 t},$$

where we have used the fact that exponentiation and the natural log are inverse functions and that $p(t) \geq 0$ for all 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 1.4 demonstrates 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 amount the two models produce similar results. The logistic model we consider restricts the growth rate in the following way

$$r = r_0 \left(1 - \frac{p}{K}\right) \tag{1.11}$$

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 (1.11); 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

$$p(t) = \frac{K p_0}{(K - p_0)e^{-r_0 t} + p_0} \tag{1.12}$$

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

We now turn to approximating the solution to both exponential and logistic growth/decay problems using both the forward and backward Euler method.

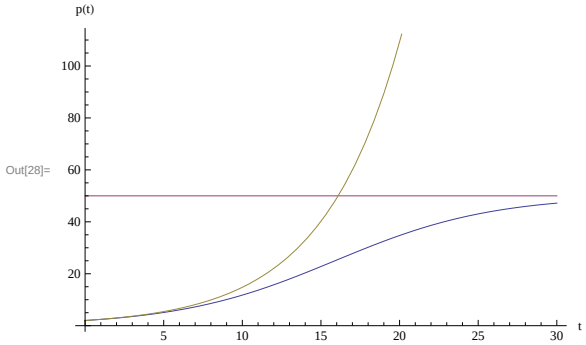

Figure 1.4: A comparison of exponential growth and logistic growth. The population in the logistic model never reaches the threshold value which in this case is 50 while exponential growth is unbounded. For values of the population much less than the threshold value, logistic growth is very similar to exponential growth.

In the examples we plot the exact solution and several approximations so we can see that as $\Delta t \to 0$ the discrete solution appears to be approaching the exact solution. Then we will compute our numerical rate of convergence and compare this with the linear rate that is predicted by Theorem **??**. If we tabulate the errors at a point for a sequence of step sizes where 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. To see this let E_i denote the error using $(\Delta t)_i$; then $E_1 \approx C(\Delta t)_1$ and $E_2 \approx C(\Delta t)_2$ where $(\Delta t)_2 = .5(\Delta t)_1$ then $E_2 \approx .5C(\Delta t)_1 \approx \frac{1}{2}E_1$.

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 we want 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

$$r = \frac{\ln \frac{E_1}{E_2}}{\ln \frac{(\Delta t)_1}{(\Delta t)_2}} . \tag{1.13}$$

Example 2. The first example we use the forward Euler method for is the specific exponential growth problem

$$p'(t) = 0.8p(t) \quad 0 < t \leq 10, \quad p(0) = 2$$

whose exact solution is $p(t) = 2e^{.8t}$. We implement Algorithm 1.4 for the forward Euler method and code a separate function for $f(t, p) = 0.8p$ because this will change for each IVP. 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 1.5. The discrete solution appears to be approaching the exact solution as the step size is reduced but we would also like 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 $e^{.8}$=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 for a sequence of decreasing values of the step size. The numerical rates were computed using (1.13) for successive errors. Note that the numerical rate is approaching one as Δt decreases so we are confident that the results are converging linearly.

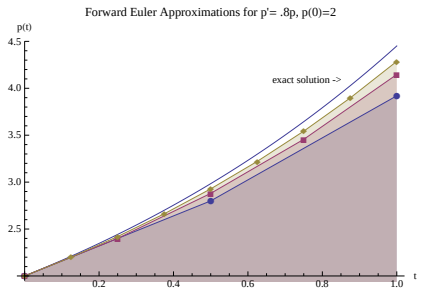

Figure 1.5: The exact solution and three approximations using Forward Euler's Method with $\Delta t = 0.5, -.25$ and 0.125 for Example 2 .

Δt	1/2	1/4	1/8	1/16	1/64	1/128
P_n	3.92	4.1472	4.28718	4.36575	4.40751	4.42906
$\lvert p(1) - P_n \rvert$	0.53108	0.30388	0.16390	0.08533	0.043568	0.022017
num. rate		0.805	0.891	0.942	0.970	0.985

Example 3. We now use the forward Euler method to approximate the solution to the logistic model

$$p'(t) = 2\left(1 - \frac{p(t)}{100}\right)p(t) \quad 0 < t \leq 5 \quad p(0) = 2\,.$$

To obtain approximations to $p(t)$ using the forward Euler method code we used in Example 2 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 (1.12). 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 Example 2. 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 $p = 100$. We first compute approximations at $t = 1.0$

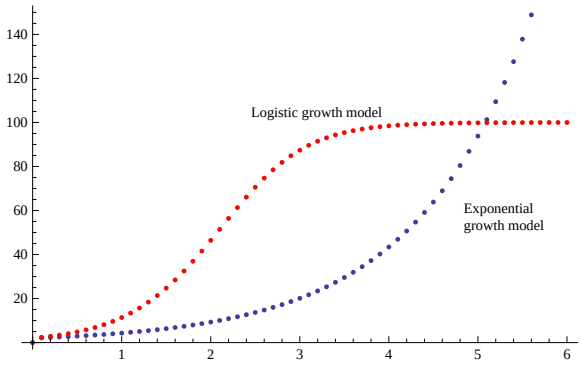

Figure 1.6: A comparison of exponential growth and logistic growth for Examples 2 and 3. The population in the logistic model never reaches the threshold value of 100 while the exponential growth is unbounded.

as we did in Example 2. The approximations at a sequence of decreasing values of Δt are presented in the table along with the calculated numerical rates. The exact value at $t = 1$ is 13.1037059. We also plot the approximate solution for a longer time for the exponential growth problem in Example 2 and this logistic 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 each example using say $\Delta t = 1/64$ we see that the errors for the logistic model are about a factor of ten larger. How do we know if this is correct and what makes one error so much larger than the other? When we calculate the numerical rate of convergence for both problems we see that the rate is one which was the theoretical rate we proved so we have confidence that the results are correct. Remember that when we say 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 an increasing second derivative on $[0, 1]$ whose maximum value is 2.849 whereas the exact solution to the logistic equation has an increasing second derivative on $[0, 1]$ but its maximum value is almost 34 so this accounts for the difference.

Δt	1/8	1/16	1/32	1/64	1/128	1/256
P_n	11.03459	11.96945	12.50836	12.79851	12.949168	13.02594
$\lvert p(1) - P_n \rvert$	2.06911	1.13426	0.59535	0.305193	0.1545388	0.07762
num. rate		0.867	0.9301	0.964	0.982	0.991

Example 4. The example we consider now is exponential decay where we use both the forward and backward Euler methods to approximate the solution. We seek $y(t)$ such that

$$y'(t) = -20y(t) \quad 0 < t \leq 0.25, \quad y(0) = 1$$

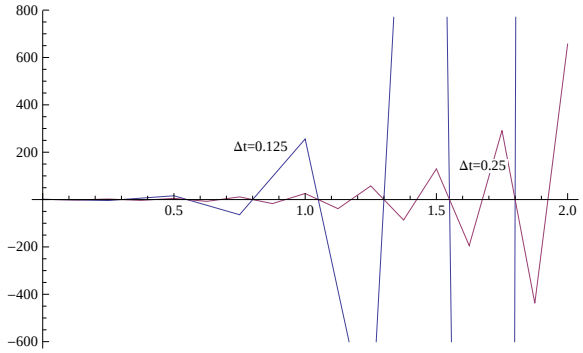

Figure 1.7: The forward Euler approximations to the IVP of Example 4 are oscillating and growing in an unbounded fashion. We say that these approximations are numerically unstable.

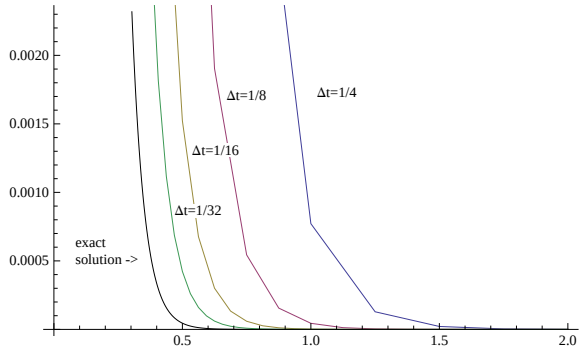

Figure 1.8: A comparison of the exact solution for Example 4 with three approximations using the implicit backward Euler method with $\Delta t = 1/4, 1/8, 1/16$ and $1/32$.

which has an exact solution of $y(t) = e^{-20t}$. In Figure 1.7 we plot the approximate solutions on $[0, 2]$ using the forward Euler method with $\Delta t = \frac{1}{4}, \frac{1}{8}$. Note that for this problem the approximate solution is oscillating and becoming unbounded. In Figure 1.8 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 but due to the specific $f(t, y)$ in this example we don't have to solve a nonlinear equation. At each time we solve $Y_{i+1} = Y_i + \Delta t 20 Y_{i+1}$ which happens to be linear for this problem so we can solve for Y_{i+1} to get $Y_{i+1} = Y_i/(1 + 20\Delta t)$.

Why are the results for the forward Euler method not reliable for this problem whereas they were for Example 2? 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.

1.4 Other approaches to deriving the Euler method

We obtained the backward or forward Euler method by looking at the definition of the derivative and realizing that we could use the slope of a secant line to approximate the derivative. We can also view the derivation of the Euler method in other ways which will be helpful in deriving more accurate methods. Remember that the Euler method is only first order accurate and typically we need schemes that converge at a faster rate. In addition, we saw in Exercise 4 that the forward Euler method does not always converge. 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 (1.7) we have

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

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 Euler's 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 $y'(t)$. This approach is explored in § 2.1.

Another approach is to use numerical integration rules to approximate the integrals obtained when we integrate (1.1a) 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 of $f(t, y)$ 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 forward Euler. 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 way 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 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; 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 (1.1) at a specific time. First look at the errors and try to decide the accuracy of the method. Then use the result (1.13) 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 (1.1a) 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 (1.1). 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.

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.

Chapter 2

Higher order accurate methods

In the last chapter we looked at the Euler method for approximating the solution to the first order IVP (1.1). 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 if the current choice of Δt is acceptable.

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 or backward Euler method. 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 $y(t)$ which makes the methods impractical. Integrating the differential equation (1.1a) 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 use of an interpolating polynomial to derive methods is to represent $y(t)$; then differentiation of the interpolating polynomial gives 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 see an efficient way to implement an implicit method by pairing it with an explicit method to yield the so-called Predictor-Corrector methods.

2.1 Taylor Series Methods

Taylor Series is an extremely useful tool in numerical analysis and 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 drop all terms of $\mathcal{O}(\Delta t^3)$ and higher in (1.7) to obtain

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

so we expect a local n error of $\mathcal{O}((\Delta t)^3)$ and thus expect a global error of $\mathcal{O}((\Delta t)^2)$. Now the problem we have to address when we implement this 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 $y''(t)$ but 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

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

We then substitute this into the Taylor series expansion and solve for $y'(t_i)$ to obtain

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

Using the differential equation we get the difference equation

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

and solving for Y_{i+1} gives

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

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 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 will explore a third order Taylor series method.

Example 5. We want to approximate the solution to

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

using a second order Taylor series scheme. The exact solution to the IVP is $y = \frac{1}{3}e^{t^3}$. Before writing a code for a particular method, it is helpful to first perform some calculations by hand so one can make sure that the method is completely understood and also to have some results to compare the numerical simulation with. To this end, we first calculate 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 above, 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 (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 (2.1), not the equation (2.2). We perform calculations with decreasing values of Δt and compare with results using the forward Euler method. When we compute the numerical rate of convergence we see that the accuracy is $\mathcal{O}(\Delta t^2)$, as expected.

Δt	Error in Euler	Numerical rate	Error in (2.1)	Numerical rate
$\frac{1}{4}$	0.31689		0.12328	
$\frac{1}{8}$	0.20007	0.663	0.41143×10^{-1}	1.58
$\frac{1}{16}$	0.11521	0.796	0.11932×10^{-2}	1.79
$\frac{1}{32}$	0.62350×10^{-2}	0.886	0.32091×10^{-3}	1.89
$\frac{1}{64}$	0.32516×10^{-2}	0.939	0.83150×10^{-4}	1.95
$\frac{1}{128}$	0.16615×10^{-2}	0.969	0.21157×10^{-4}	1.97

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.

2.2 Methods from Integration Formulas

Another approach we used to obtain the Euler method 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 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.

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 (1.1a) 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 and then use it to approximate $y(t_{i+1})$ from

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

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

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

Computationally, we see that we have to do extra work compared with the Euler method because we have to approximate the intermediate value $y(t_i + \Delta t/2)$. Because we are doing more work than the Euler method, we would like to think that the scheme would converge 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 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 1. *Let $f(x, y)$ be continuously differentiable. Then*

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

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 explicit terms through $(\Delta t)^3$ 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; we have

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

Now to combine terms involving $y'(t)$ and $f(t,y)$ in the expression for τ_{i+1} we must expand $f(t_i + \frac{\Delta t}{2}, y(t_i) + \frac{\Delta t}{f}(t_i, y(t_i)))$; because it is a function of two variables instead of one we need to use Proposition 2.4. 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 $k = (\Delta t/2) f(t_i, y(t_i))$. We have

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

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

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

To cancel terms we note that from the differential equation $y'(t) = f(t,y)$ we have $y''(t) = f_t + f f_y$ 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; thus the local truncation error is cubic and we expect the global convergence rate to be quadratic in Δt. The following example demonstrates the numerical accuracy of the Midpoint method.

Example 6. We use the Midpoint method given in (2.3) to approximate the solution of the IVP

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

that we considered in Example 5. 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	0.69664×10^{-1}	
1/8	0.22345×10^{-1}	1.64
1/16	0.63312×10^{-2}	1.82
1/32	0.16827×10^{-2}	1.91
1/64	0.43346×10^{-3}	1.96
1/128	0.10998×10^{-3}	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. Another 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 our differential equation (1.1a) from t_i to t_{i+1} and using this quadrature rule gives

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

This suggests the 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 with higher than first order accuracy 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 must differentiate it and use it as an approximation to $y'(t)$ in the equation $y'(t) = f(t, y)$. This approach leads to the family of implicit methods called Backward Difference Formulas (BDF).

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. This approach leads to families of methods called multistep methods discussed 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}.

2.3.1 Backward difference formulas

Backward difference formulas (BDF) are a family of implicit methods and the backward Euler is a first order BDF. In § 1.4 we showed that if we use a linear

interpolating polynomial $p_1(t)$ to approximate $y(t)$ for $t_i \le t \le t_{i+1}$ then we can differentiate it to get an approximation to $y'(t)$. When we use $p_1'(t)$ to approximate $y'(t_i) = f(t_i, y(t_i))$ we obtained the explicit forward Euler scheme and if we used $y'(t_{i+1}) = f(t_{i+1}, y(t_{i+1}))$ then we obtained the implicit backward Euler scheme. 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.

The backward Euler method is a first order BDF so if we want a higher order approximation an obvious approach would be to use a higher order interpolating polynomial to approximate y. For example, we could use a quadratic polynomial; however we know that fitting a quadratic requires three points. We have the points $(t_i, y(t_i))$ and $(t_{i+1}, y(t_{i+1}))$ but need to choose a third. In BDF formulas information at previously calculated points are used; this has the advantage that no additional function evaluations are required. Thus for a quadratic interpolating the point $\big(t_{i-1}, y(t_{i-1})\big)$ 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

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

Because we want an implicit scheme we use $p_2'(t)$ as an approximation to y' in the equation $y'(t_{i+1}) = f(t_{i+1}, y(t_{i+1}))$; this yields

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

This suggest the BDF

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

or equivalently

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

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

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

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 one.

For our scheme (2.7) we have , $a_{21} = -2$ and $a_{22} = 1/2$. 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, in theory, we should 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.

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 integrate the equation over the given interval the integral of the interpolating polynomial for $f(t, y)$ can be integrated exactly. To see this suppose we want to derive an explicit method where we use the previous information at t_i and t_{i-1}; we do not include the point t_{i+1} because that would result in an implicit method. 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 $\int_{t_i}^{t_{i+1}} y'(t)\, dt$. We have

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

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 § 2.5.

2.3.2 Single step versus multistep methods

Note that the BDF scheme (2.8) differs from other schemes we derived because it uses the history of approximations to y to extrapolate the solution at the next point; for example, (2.7) specifically uses approximations at t_i and t_{i-1}. These types of methods are called **multistep methods** because they use the solution at multiple points of our discretization to approximate the solution at the next point t_{i+1}. This is in contrast to methods such as the Midpoint method which uses only one previous 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 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.

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. He described the Midpoint method in 1895 and demonstrated that the accuracy is quadratic. The 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.

For example, in the simplest case when $s = 1$ we only use y and its slope at t_i; then 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 which make the scheme have 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

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

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

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

where we have used the differential equation $y' = f(t,y)$. Clearly we have $\beta = 1$ and $b_1 = 1$ which is just the forward Euler method so it is the simplest RK method. In the sequel we will dispense with the coefficient β because it always must be equal to one.

We now derive a scheme where $s = 2$, i.e., 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 τ_i 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

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

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:

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

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 $b_1 = 0, b_2 = 1$, $c_2 = a_{21} = 1/2$. There are many other schemes which satisfy these conditions because we only have three constraints and four degrees of freedom, i.e., our 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. For the Heun method we write

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

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 we expect the global error to be $\mathcal{O}(\Delta t^2)$.

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 to do 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.

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. The general form of an s-stage explicit RK is

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

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

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

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 $c_i = \sum_{j=1}^{i-1} a_{ij}$; recall that we set $c_1 = 0$ so that we are forced to use the point t_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 see if 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 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.

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

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. We have

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

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.

Basically the way to use RK for step size control is to 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 would like to 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

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

Then 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

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	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{2856}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

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 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 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. Specifically, 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. 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. Multistep methods require the use of a single step method to obtain additional starting values. We saw that the BDF formula (2.8) is an example of a two-step implicit multistep method.

The general form of an m-step multistep method is

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

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 uses the derivative f evaluated at m prior points but only uses the approximation to $y(t)$ at t_i; i.e., $a_0 = \cdots = a_{m-2} = 0$. Schemes in the *Adams-Moulton* family are commonly used implicit multistep schemes which also use the derivative f evaluated at t_{i+1} plus m prior points but only uses Y_i.

In § 2.3 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 $b_0 = -1/2$. We expect the local truncation error to be $\mathcal{O}(\Delta t^3)$ and the method to be second order. In the exercises, you are asked to rigorously demonstrate that the local truncation error for (2.9) is third order. Because this is a 2-step method

we need Y_1 in addition to Y_0 to start the method. We can use a second order single step method for this purpose; higher order schemes can also be used but of course we have to do extra function evaluations which is wasted work in this case.

To obtain implicit multistep methods we use the point t_{i+1} in additional to previous points to interpolate $f(t, y)$. In the exercises you are asked to derive a 2-step implicit method.

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 methods implicit schemes are used to improve (or correct) the solution that is first obtained (or predicted) by an explicit scheme. The idea is to combine appropriate explicit and implicit schemes to obtain better results. In simulations where a variable step size is needed, we can also use predictor-correct methods to estimate the appropriate step size.

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

$$\begin{aligned} Y^p_{i+1} &= Y_i + \Delta t f(t_i, Y_i) \\ Y_{i+1} &= Y_i + \tfrac{\Delta t}{2}\Big[f(t_{i+1}, Y^p_{i+1}) + f(t_i, Y_i)\Big] \end{aligned} \tag{2.20}$$

It is important to realize that the implicit Trapezoidal method is now implemented like an explicit method because we evaluate $f(t_{i+1}, Y^p_{i+1})$ instead of $f(t_{i+1}, Y_{i+1})$. The predicted solution Y^p_{i+1} 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 Exercise xx you are asked to show that the predictor-corrector pair Euler-Trapezoid is second order.

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

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

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

In the table below we compare the errors and rates of convergence for this PC pair and the third order Adams-Bashforth method defined by the predictor equation above. 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.

Δt	Error in Predictor only	Num. rate	Error in PC pair	Num. rate
1/10	0.20100×10^{-1}		0.153×10^{-2}	
1/20	0.36475×10^{-2}	2.47	$.33482 \times 10^{-3}$	2.19
1/40	0.54518×10^{-3}	2.74	0.55105×10^{-4}	2.60
1/80	0.74570×10^{-4}	2.87	0.79035×10^{-5}	2.80
1/160	0.97513×10^{-5}	2.93	0.10583×10^{-5}	2.90

Using predictor-corrector pairs also provide 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 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 know 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 halved. Otherwise the step is accepted; if the difference in below our minimum prescribed tolerance than 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.
$$\begin{array}{c|cccc} 0 & 0 & 0 & & \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & \\ 1 & -1 & 2 & 0 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \end{array}$$

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. Use a Taylor series to derive a third order accurate explicit difference equation for the IVP (1.1).

3. 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?

4. 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.

5. 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.

6. 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.

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

Chapter 3

Systems of Initial Value Problems

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 provide case studies that illustrate the use of systems of IVPs.

The last 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 Example 4, 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

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

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

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

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 $y(t)$

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

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

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

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.

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

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

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 methods from Chapter 2 can be extended to our system of N equations (3.2).

3.2 Single step methods for systems

We now want to extend single step methods to the system (3.2). To this end, we use the notation $W_{k,i}$ to approximate $w_k(t_i)$ where the first subscript of W refers to the unknown number and the second to the point for which we have an approximation. 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 N 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 unknowns at one time. 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 p 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 7. Consider the system of three IVPs

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

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

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

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

$$\mathbf{F}_1 = \begin{pmatrix} 2(0.4) - 4(.1) \\ 1 + 2.1 - e^{.1} + 2 \\ -1 - 2(.4) + 2.1 + 4(.1) \end{pmatrix} = \begin{pmatrix} 0.4 \\ 3.995 \\ 0.7 \end{pmatrix}$$

so that

$$\mathbf{W}_2 = \begin{pmatrix} -1.0 \\ 0.4 \\ 2.1 \end{pmatrix} + 0.1 \begin{pmatrix} 0.4 \\ 3.995 \\ 0.7 \end{pmatrix} = \begin{pmatrix} -0.96 \\ 0.7995 \\ 2.17 \end{pmatrix}$$

The exact solution at $t = 0.2$ is $(-0.921061, 0.789418, 2.14246)^T$ giving an error vector of $(0.038939, .010082, .02754)^T$ so the standard Euclidean norm of the error normalized by the norm of the solution is 1.98×10^{-2}. Recall that the Euclidean norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ is

$$\|\mathbf{x}\|_2 = \Big[\sum_{i=1}^{n} x_i^2\Big]^{1/2} \tag{3.4}$$

and is often called ℓ_2 norm or the "little l2 norm".

Suppose now that we have a s-stage RK method; recall that for a single first order equation we have s function evaluations for each t_i. If we have p first order IVPs, then we need sp 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, f in (2.12) is now a vector 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 Example 7.

Example 8. We want to approximate the solution to the system given in Example 7 using the Heun method. Recall that for this method the coefficients are $c_2 = 2/3$, $a_{21} = 2/3$, $b_1 = 1/4$ and $b_2 = 3/4$. 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 calculations gives $\mathbf{k}_2 = (.026667, .399773, .08)^T$ so that

$$\mathbf{W}_1 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 0.0 \\ 0.4 \\ 0.1 \end{pmatrix} + \frac{3}{4}\begin{pmatrix} .026667 \\ .399773 \\ .080000 \end{pmatrix} = \begin{pmatrix} -0.980000 \\ 0.39983 \\ 2.085 \end{pmatrix}$$

Similarly for the approximation at 0.2 we have

$$\mathbf{W}_2 = \begin{pmatrix} -0.980000 \\ 0.39983 \\ 2.085 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} .039966 \\ .395983 \\ -.070534 \end{pmatrix} + \frac{3}{4}\begin{pmatrix} .066097 \\ .390402 \\ .0517697 \end{pmatrix} = \begin{pmatrix} -0.9204 \\ .791633 \\ 2.1415 \end{pmatrix}$$

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.

3.3 Multistep methods for systems

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

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

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 only the solution at t_i is used so this saves additional storage because we only have to store m slope values and a single approximation to the solution. So for the system of N equations using an m-step method we must store $(m+1)$ vectors of length N.

As an example, we consider a 2-step Adams-Bashforth method which is an explicit method given by

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

for a single IVP; or for the system of N equations (3.2) we have

$$\mathbf{W}_{i+1} = \mathbf{W}_i + \Delta t\Big[\frac{3}{2}\mathbf{F}(t_i, \mathbf{W}_i) - \frac{1}{2}\mathbf{F}(t_{i-1}, \mathbf{W}_{i-1})\Big]. \tag{3.5}$$

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 Example 7.

Example 9. To apply the 2-step Adams-Bashforth method (3.5) to the system of Example 7 we need values for $\mathbf{W}_1$ because we set $\mathbf{W}_0$ from the initial conditions. Because this method is second order we need a second order scheme to generate an approximation to $\mathbf{W}_1$. In Example 8 we used the Heun method to approximate the solution $\mathbf{W}_2$ and because this is a second order scheme, it is

adequate for our purposes. Here we will use $\Delta t = 0.1$ as before. Consequently we have

$$\mathbf{W}_0 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \quad \text{and} \quad \mathbf{W}_1 = \begin{pmatrix} -.98000 \\ .39982 \\ 2.08500 \end{pmatrix}$$

From Example 8 we have $\mathbf{F}(0, \mathbf{W}_0) = (0.0, 4.0, 1.0)^T$ and also $\mathbf{F}(0.1, \mathbf{W}_1)$=(.39966, $3.95982, -.704659)^T$. Then $\mathbf{W}_2$ is given by

$$\mathbf{W}_2 = \begin{pmatrix} -.980000 \\ .39982 \\ 2.08500 \end{pmatrix} + 0.1 \left[\frac{3}{2} \begin{pmatrix} 0.39966 \\ 3.95983 \\ -0.704659 \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 0.0 \\ 4.0 \\ 1.0 \end{pmatrix} \right] = \begin{pmatrix} -.920051 \\ 0.793795 \\ 1.9293 \end{pmatrix}.$$

3.4 Stability

We have seen that numerical results from some standard methods such as the forward Euler method exhibit oscillatory and unbounded behavior for some problems as we saw in Example 4. However, when we solve the same problem using the same step size with the backward Euler method we got results which appeared to be converging. Why does one numerical method produce reasonable results and the other produces unbounded results? The answer lies in the stability properties of the numerical scheme. Even if the actual differential equation is stable, the numerical scheme may not be stable. Consequently we need to look at the concept of stability of numerical schemes so we are able to choose a scheme which produces stable numerical simulations. 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. Due to time constraints, we will only give some of the major points and illustrate the concept with examples.

We have investigated the accuracy of many of the methods we have derived; specifically we obtained results of the form $\mathcal{O}((\Delta t)^r)$ for the rate of convergence. Of course, we have tacitly assumed that the methods converge but this may not always be the case. We have seen an example where the forward Euler method failed to converge so we might ask ourselves if this is due to the linear accuracy of the method. However, in the plot on the left of Figure 3.1 we present the results of approximating the solution to the same IVP $y'(t) = -20y(t)$, $y(0) = 1$ using a second order RK scheme. As we see from the figure, the numerical results are not converging to the correct solution which is e^{-20t} but rather becoming unbounded. The phenomena we are experiencing here is numerical instability due to too large a step size Δt. The figure on the right shows convergent approximations when we decrease the step size. One might be tempted to just use a very small step size to avoid getting into this problem, but then we may get into problems because roundoff errors are accumulating due to the large number of steps; in addition, using too small a step size results in extra work. Remember that the time frame in realistic problems may be quite long; for example, if we are modeling climate change Consequently, we need to delve a little deeper into what is actually happening.

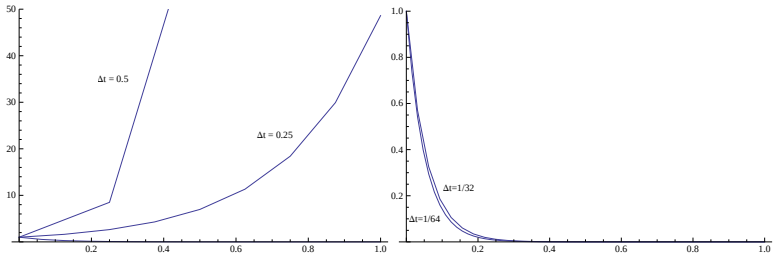

Figure 3.1: Approximations for the IVP $y'(t) = -20y$, $y(0) = 1$ using the Heun method. For the plot on the left the step size Δt is too large and numerical instability is occurring. When we reduce the step size the method converges as the graph on the right demonstrates.

The goal is for our numerical schemes to be convergent so we want to control the global error over $[t_0, T]$. Basically, for convergence we want to look at the global error as $\Delta t \to 0$ and show that this limit goes to zero. 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

$$\lim_{\Delta t \to 0} \max_{1 \le i \le p} |y(t_i) - Y_i| \quad \text{or for the system (3.2)} \quad \lim_{\Delta t \to 0} \max_{1 \le i \le p} \|\mathbf{w}(t_i) - \mathbf{W}_i\|$$

where $p = (T - t_0)/\Delta t$ and $\|\cdot\|$ is a norm on $\mathbb{R}^N$.

We know how to control the local error because this is just the truncation error. Recall that this measures the error in one step of our method if we start on the exact solution curve so it measures how well the difference equation mimics the differential equation. If the local truncation error goes to zero as $\Delta t \to 0$ we say the method is **consistent**; i.e.,

$$\lim_{\Delta t \to 0} \max_{1 \le k \le p} |\tau_k(\Delta t)| \ ,$$

where $\tau_k(\Delta t)$ is the local truncation error at step k using step size Δt. For example, if we computed the local truncation error and found that it was a constant and not dependent upon Δt, then the scheme would not be consistent. A method must be consistent if we have any hope of it being convergent; of course consistency does not guarantee convergence as previous examples have illustrated. The other requirement for convergence turns out to be stability.

Loosely speaking, for stability we want to know that the solution to the difference equation does not grow in an unbounded manner. To see what we mean by this we look at the differential equation

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

which has the exact solution $y(t) = y_0 e^{\lambda t}$ for the initial condition $y(0) = y_0$. Note that here λ is a complex number 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 the term $e^{i\beta t} = \cos(\beta t) + i \sin(\beta t)$ so it 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., $\mathrm{Re}(\lambda) \leq 0$ or the left half plane.

We are going to mimic this analysis first for a difference equation of the form

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

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 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*. We know that its magnitude 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 6. *The region of absolute stability for the difference equation (3.6) is* $\{\lambda \Delta t \in C \mid \zeta(\lambda \Delta t) \leq 1\}$. *A method is called* A**-stable** *if* $\zeta(\lambda \Delta t) \leq 1$ *for the entire left half plane.*

Example 10. We want to determine the region of absolute stability of the forward Euler method and of the backward Euler method and then discuss the results of Example 4 in light of these regions. For the forward Euler method $\zeta(\lambda \Delta t) = 1 + \lambda \Delta t$ so it is A-stable provided $|1 + \lambda \Delta t| \leq 1$. 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 $\Delta t \leq 2/|\lambda|$. For example, in Example 4 $\lambda = -20$ so Δt must satisfy $\Delta t \leq 0.1$. In Figure 1.7 we plotted results for $\Delta t = 1/4$ and $1/8$ which do not satisfy the stability criteria. In Figure 3.2 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. If $\beta \neq 0$ then we have a circle in the complex plane of radius one centered at -1.

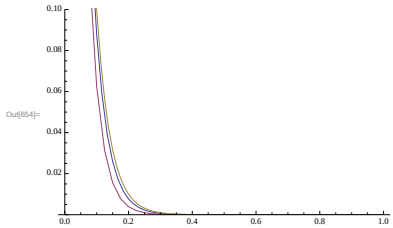

Figure 3.2: Approximations for the IVP $y'(t) = -20y$, $y(0) = 1$ using the forward Euler method with $\Delta t = 1/20, 1/40$, and $1/60$. These values of Δt satisfy the stability condition $|1 + \lambda\Delta t| \leq 1$ so the method is stable and converges. Compare these results with those from Example 4 .

For the backward Euler method $\zeta(\lambda\Delta t) = 1/(1 - \lambda\Delta t)$. As before, we first find the region when λ is real. For $\lambda \leq 0$ have

$$\left|\frac{1}{1-\lambda\Delta t}\right| \leq 1 \Rightarrow 1 - \lambda\Delta t \geq 1 \quad \text{and} \quad -1 + \lambda\Delta t \leq 1 \Rightarrow -\lambda\Delta t \geq 0, \quad \lambda\Delta t \leq 2$$

so we have the entire left plane and the interval $[0, 2]$ on the real axis. We say that the backward Euler method is A-stable. Returning to our Example 4 we saw that the backward Euler appeared to be converging for all values of Δt we used. This is because in this example $\lambda = -20$ and so the stability criteria is always satisfied.

Example 11. In this example we want to investigate the regions of absolute stability for the explicit 2-stage Heun method

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

We have written the scheme as a single equation rather than the standard way of specifying k_i because it will be easier to determine the amplification factor. We apply the difference scheme to $y' = \lambda y$ to get

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

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 $\lambda \leq 0$ we must have $1 + \frac{1}{2}\lambda\Delta t \geq 0$ which says $\Delta t\lambda \geq -2$ so the region 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 forward Euler method.

It can be shown that there is no explicit RK method that has an unbounded region of absolute stability. This is one reason that we need implicit methods.

For an $N \times N$ system of IVPs we consider the linear problem

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

analogous to the problem $y'(t) = \lambda y$ where now 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[1] counted according to multiplicity. So an equivalent statement of stability when A is diagonal is that $\mathrm{Re}(\lambda_i) < 0$, $i = 1, 2, \ldots, N$. 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 can 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 $\partial f_i / \partial w_j$.

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 the 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.

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 starting values $Y_0, Y_1, \ldots, Y_{m-1}$ which are obtained by another method such as a RK method. In 1956 Dahlquist[2] published a seminal work on the stability of linear multistep methods. We first write the m-step multistep method (2.19) where we shift the indices to get

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

as

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

[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.

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

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

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

We seek a solution of the form $Y_i = z^i$ and substitution into the difference equation yields

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

Canceling the lowest order term z^i gives the so-called *characteristic* equations

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

which play an important role in the Dahlquist stability theory. 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. To find the condition on Δt we find the roots σ_i of $\mathcal{Q}(\lambda \Delta t)$ where

$$\begin{aligned} \mathcal{Q}(\lambda \Delta t) &= z^m - \sum_{j=0}^{m-1} a_j z^j - \Delta t \sum_{j=0}^{m} b_j \lambda z^j \\ &= z^m (1 - \lambda \Delta t b_m) - z^{m-1}(a_{m-1} + b_{m-1} \lambda \Delta t) - \cdots - (a_0 + b_0 \lambda \Delta t) \, . \end{aligned}$$

The region of stability is $\{\lambda \Delta t \in \mathbb{C} \mid |\sigma_i| \leq 1\}$.

Example 12. We look at the characteristic polynomial $\rho(z)$ for two methods and see if the root condition is satisfied. If it is, we find the region of absolute stability. First we consider the forward Euler method and see if we get the same result as before. Next we look at a 2-step Adams-Bashforth method.

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 $\mathcal{Q}(\lambda \Delta t) = z - (1 + \lambda \Delta t)$ and thus the region of absolute stability is $|1 + \lambda \Delta t| \leq 1$ which is the condition we got before analyzing the method as a single step method.

The second order Adams-Bashforth method is

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

so its characteristic polynomial $\rho(z) = z^z - z = z(z-1)$ whose roots are $z = 0, 1$ and the root condition is satisfied. To find the region of absolute stability we determine $\mathcal{Q}(\lambda \Delta t)$

$$\mathcal{Q}(\lambda \Delta t) = z^2(1) - z(1 + \frac{3}{2} \lambda \Delta t) - (0 + \frac{1}{2} \lambda \Delta t) = z^2 - z(1 + \frac{3}{2} \lambda \Delta t) - \frac{1}{2} \lambda \Delta t \, .$$

Setting $\zeta = \lambda \Delta t$ the roots of this equation are

$$z^2 - z(1 + \frac{3}{2}\zeta) - \frac{1}{2}\zeta = 0 \Rightarrow \zeta = \frac{2z^2 - 2z}{3z + 1}$$

and we need $|\zeta| \leq 1$ for absolute stability. To get the region in the complex plane we set $z = e^{i\theta}$ for $0 \leq \theta \leq 2\pi$ and use a computer to plot the region.

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 implicit methods tend to be more stable than explicit methods. As we have seen, things are more complicated for multistep methods. We have just touched on the ideas of stability of numerical methods for IVPs; the interested reader is referred to standard graduate texts in numerical analysis for a thorough treatment of stability. An important thing to keep in mind 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.

3.6 Case study - Modeling a viral infection

In this section we look at a simple model of a viral infection and see that with certain assumptions we are lead to a system of two IVPs. In the exercises you are asked to extend this model to include viral mutations. The interested reader is referred to Nowak and My, Mathematical biology of HIV Infections in *Mathematical Biosciences* **106**, 1991.

A viral disease begins with an infection of the body by a small number of viruses. The viruses attempt to enter individual cells and "hijack" them into producing new copies of the virus. If left unchecked, an infected cell will die and release the new copies of the virus to continue the infection. The immune system constantly assesses the body for alien objects such as viruses. When it recognizes an invading virus it records the pattern of *antigens* on the surface of the virus and then produces special cells called *antibodies* which are programmed to destroy all objects that have the same pattern of antigens. Thus, once the immune system has spotted a single virus, it will try to kill all copies of the virus. Consequently, a typical viral disease usually follows one of two patterns; either the virus replicates so quickly that the host body is killed or the immune system is able to kill all the viruses.

To model a viral infection we let $v(t)$ represent the density of viruses which will give the strength of the infection and let $a(t)$ represent the density of antibodies produced by the immune system. We assume that at $t = 0$ we can measure $v(0)$ and $a(0)$. In order to get governing equations for $v(t)$ and $a(t)$ we need to make certain assumptions. First, we assume that the immune system responds to a virus by increasing the number of antibodies linearly, i.e., the number of antibodies produced is proportional to the number of viruses present.

If we let k denote this proportionality constant we have

$$a'(t) = kv(t)\,.$$

We are assuming here that the antibodies are never destroyed.

Now to get an equation for $v(t)$ we assume that we know the rate of growth which assumes knowledge of the probability that the virus will be able to invade a cell at any time and how long the invaded cell will be able to remain alive making copies and how many copies will be made before the cell dies. Of course knowing this rate of growth assumes that the process is understood completely which is not the case but it is a simplifying assumption we make. Also antibodies are busy killing viruses so this results in a decrease. A reasonable assumption is that the decrease is proportional to the product of the number of viruses and the number of antibodies present at any time. This is because we assume that the chance that at a given instance one particular antibody will locate a particular virus and kill it is $p(t)$; then the chance that this particular antibody will locate any virus is $p(t)v(t)$. Because every antibody has the same chance the total number of antibody/virus encounters is $p(t)v(t)a(t)$. To get the governing differential equation we let $r(t)$ be the rate at which new viruses occur at time t. The rate of change in the density of viruses is due to the increase governed by $r(t)v(t)$ and the decrease governed by $p(t)v(t)a(t)$; we have

$$v'(t) = r(t)v(t) - p(t)v(t)a(t)\,.$$

Combining the two differential equations yields the system of IVPs

$$\begin{array}{rcll} a'(t) & = & kv(t) & 0 < t \leq T \\ v'(t) & = & r(t)v(t) - p(t)v(t)a(t) & 0 < t \leq T \\ a(0) & = & a_0 \qquad v(0) = v_0 & \end{array} \tag{3.9}$$

As an example, suppose $k = 0.1$, $r = 0.5$, $p = 0.25$ and assume that no antibodies are initially present, i.e., $a(0) = 0$ and that there is a small density of the virus $v(0) = 0.01$. In Figure 3.3 we approximate the solution to (3.9) by using the second order Heun method. In this case Δt is fifteen minutes and we have run the simulation for 48 hours. As can be seen from the figure the density of the virus peaks to around five at about 15 hours but the antibodies produced are able to kill the virus. If we lower the probability $p(t)$ or the proportionality constant k, or raise the rate $r(t)$ then this may not be the case. We lower the proportionality constant governing the increase in the antibodies and rerun the calculation. We make the assumption that if $v > 20$, the virus has killed the host. In Figure 3.4 we show the result of setting $k = 0.05$ and $k = 0.01$; in the first case the antibodies are able to kill off the virus but in the second they are not and the host is killed before 16 hours!

In the exercises you are asked to extend this model to include viral mutations and perform some studies.

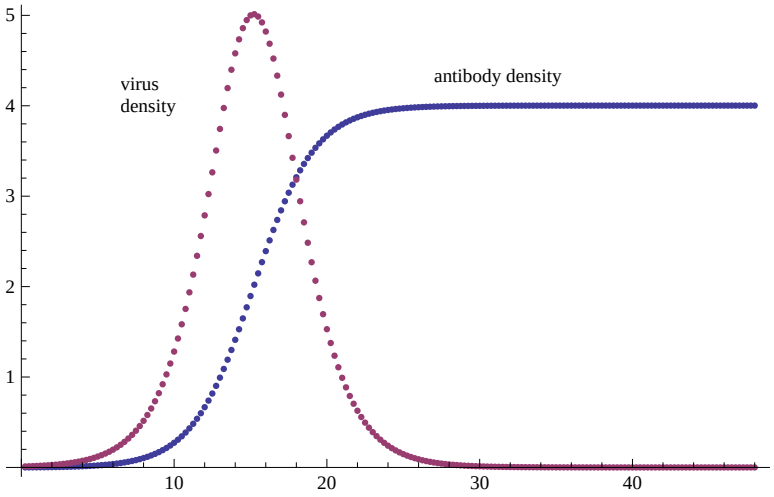

Figure 3.3: Numerical results for approximating the solution to (3.9) with $k = 0.1$, $r = 0.5$, $p = 0.25$, $a(0) = 0$ and $v(0) = 0.01$. The second order Heun method is used.

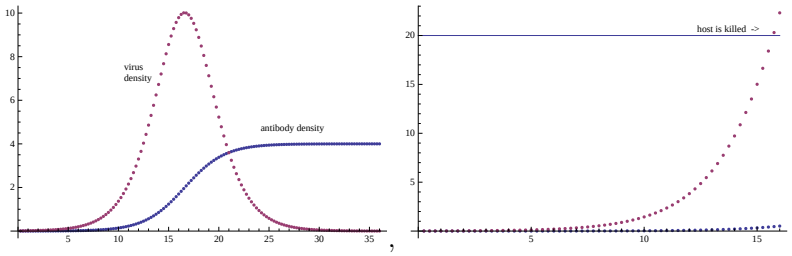

Figure 3.4: Numerical results for approximating the solution to (3.9) with $r = 0.5$, $p = 0.25$, $a(0) = 0$ and $v(0) = 0.01$. For the figure on the left $k = 0.05$ and for the figure on the right $k = 0.01$. The second order Heun method is used. If we assume that the host dies if the virus density reaches 20 then this is the case for the simulations on the right.