Chapter 2

Introduction to Initial Value Problems

The purpose of this chapter is to study the simplest numerical methods for approximating the solution to a first order initial value problem (IVP). Because the methods are simple, we can easily derive them plus give graphical interpretations to gain intuition about our approximations. Once we analyze the errors made in replacing the continuous differential equation by a difference equation, we will see that the methods only converge linearly which is quite slow. This is the motivation for looking at higher accurate methods in the next chapter. We will look at several numerical examples and verify the linear convergence of the methods and we will see that in certain situations one of the methods tends to oscillate and even “blow up” while the other always provides reliable results. This will motivate us to study the numerical stability of methods.

We begin by looking at the prototype IVP that we consider in this chapter and the next. The differential equation for this IVP is first order and gives information on the rate of change of our unknown; in addition, an initial value for the unknown is specified. Later, in Chapter 4, we consider higher order IVPs and we will see that higher order IVPs can be written as a system of first order IVPs so that all the methods we study in this chapter and the next can be easily extended to systems. Before we investigate methods to approximate the solution of this prototype IVP we consider conditions which guarantee that the analytic solution exists, is unique and depends continuously on the data. In the sequel we will only be interested in approximating the solution to such problems.

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

The simplest methods for approximating the solution to our prototype IVP are
the forward and backward Euler methods which we derive by approximating the
derivative in the differential equation at a point by the slope of a secant line. In
§ 2.2.2 we demonstrate the linear convergence of the method by introducing the
concepts of local truncation error and global error. The important differences in
explicit and implicit methods are illustrated by comparing these two Euler methods.
In § 2.3 we present some 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.

Lastly, we demonstrate numerically that the forward Euler method gives unre-
liable results in certain situations whereas the backward Euler always appears to
give reliable results. This leads us to introduce the concept of numerical instability
and to see why this is required for a numerical simulation to converge to the exact
solution.

2.1 Prototype initial value problem

A problem commonly encountered is an initial value problem where we seek a func-
tion whose value is known at some initial time and whose derivative is specified for
subsequent times. The following problems are examples of first order IVPs for $y(t)$:

$I : \begin{cases} y'(t) = \sin \pi t & 0 < t \leq 4 \\ y(0) = 0. \end{cases}$  
$II : \begin{cases} y'(t) + y^2(t) = t & 2 < t \leq 10 \\ y(2) = 1. \end{cases}$  

(2.1)

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

\[
\text{General IVP: find } y(t) \text{ satisfying} \\
\frac{dy}{dt} = f(t, y) \quad t_0 < t \leq T \quad (2.2a) \\
y(t_0) = y_0. \quad (2.2b)
\]

Here $f(t, y)$ is the given derivative of $y(t)$ and $y_0$ is the known value at the initial
time $t_0$. For example, for the IVP II in (2.1) we have $f(t, y) = t - y^2$, $t_0 = 2,
T = 10$ and $y_0 = 1$. Note that both linear and nonlinear differential equations are
included in the general equation (2.2a).

For certain choices of $f(t, y)$ we can find an analytic solution to (2.2). In the
simple case when $f = f(t)$, i.e., $f$ is a function of $t$ and not both $t$ and $y$, we can
2.1. PROTOTYPE INITIAL VALUE PROBLEM

solve the ODE exactly if \( \int f(t) \, dt \) can be evaluated. We expect that the solution to the differential equation (2.2a) is not unique; actually there is a family of solutions which satisfy the differential equation. To be able to determine a unique solution we must specify \( y(t) \) at some point such as its initial value. If \( f(t, y) \) is more complicated than simply a function of \( t \) then other techniques are available to find the analytic solution in certain circumstances. These techniques include methods such as separation of variables, using an integrating factor, etc. Remember that when we write a code to approximate the solution of the IVP (2.2) we always want to test the code on a problem where the exact solution is known so it is useful to know some standard approaches; alternately one can use the method of manufactured solutions discussed in Example 1.3. The following example illustrates when the method of separation of variables can be used to solve (2.2a) analytically; other techniques are explored in the exercises.

**Example 2.1. Method of separation of variables for finding the analytic solution of (2.2).**

Consider the differential equation \( y'(t) = -ty(t) \) and find its general solution using the method of separation of variables; illustrate the family of solutions graphically. Verify that the solution satisfies the differential equation and then impose the initial condition \( y(0) = 2 \) to determine a unique solution to the IVP.

Because \( f(t, y) \) is a function of both \( y \) and \( t \) we can not directly integrate the differential equation with respect to \( t \) to obtain the solution. However, rewriting the equation as

\[
\frac{dy}{y} = -tdt \quad \Rightarrow \quad \int \frac{dy}{y} \, dt = -\int t \, dt
\]

allows us to integrate to get the general solution

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

This technique can be applied whenever we can “separate variables”, i.e., bring all the terms involving \( y \) to one side of the equation and those involving \( t \) to the other side. Note that the general solution to this differential equation involves an arbitrary constant \( C \) and thus there is an infinite family of solutions which satisfy the differential equation. A family of solutions is illustrated in the figure below; note that as \( t \to \pm\infty \) the solution approaches zero.
We can always verify that we haven’t made an error in determining the solution by demonstrating that it satisfies the differential equation. Here we have

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

so the equation is satisfied.

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

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

Even if we are unable to determine the analytic solution to (2.2), we can still gain some qualitative understanding of the behavior of the solution. This is done by the visualization technique of plotting the tangent line to the solution at numerous points \((t,y)\); recall that the slope of the tangent line to the solution curve is given and is just \( f(t,y) \). Mathematical software with graphical capabilities often provide commands for automatically drawing a direction field with arrows which are scaled to indicate the magnitude of the slope; typically they also offer the option of drawing some solutions or streamlines. Using direction fields to determine the behavior of the solution is illustrated in the following example.

**Example 2.2. Direction fields**

Draw the direction fields for the ODE

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

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

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

To see which streamline corresponds to the solution with \( y(0) = 1 \) we locate the point \((0,1)\) and follow the tangents; this solution is indicated by a thick line in the direction field plot below. If a different initial condition is imposed, then we get a different streamline.

![Direction Field Example](image-url)
chaptertitle{Prototype Initial Value Problem}

Before we discuss methods for approximating the solution of the IVP (2.2) we first need to ask ourselves if our prototype IVP actually has an analytic solution, even if we are unable to find it. We are only interested in approximating the solution to IVPs which have a unique solution. However, even if we know that a unique solution exists, we may still have unreliable numerical results if the solution of the IVP does not depend continuously on the data. If this is the case, then small changes in the data can cause large changes in the solution and thus roundoff errors in our calculations can produce meaningless results. In this situation we say the IVP is ill-posed or ill-conditioned, a situation we would like to avoid. Luckily, most differential equations that arise from modeling real-world phenomena are well-posed.

The conditions that guarantee well-posedness of a solution to (2.2) are well known and are presented in Theorem 2.1. Basically the theorem requires that the derivative of \( y(t) \) (given by \( f(t, y) \)) be continuous and, moreover, this derivative is not allowed to change too quickly as \( y \) changes. A basic problem in calculus is to determine how much a continuous function changes as the independent variables change; clearly we would like a function to change a small amount as an independent variable changes but this is not always the case. The concept of Lipschitz continuity\(^1\) gives a precise measure of this “degree of continuity”. To understand this concept first think of a linear function \( g(x) = ax + b \) and consider the effect changing \( x \) has on the dependent variable \( g(x) \). We have

\[
|g(x_1) - g(x_2)| = |ax_1 + b - (ax_2 + b)| = |a| |x_1 - x_2|.
\]

This says that as the independent variable \( x \) varies from \( x_1 \) to \( x_2 \) the change in the dependent variable \( g \) is governed by the slope of the line, i.e., \( a = g'(x) \). For a general function \( g(x) \) Lipschitz continuity on an interval \( I \) requires that the magnitude of the slope of the line joining any two points \( x_1 \) and \( x_2 \) in \( I \) must be bounded by a real number. Formally, a function \( g(x) \) defined on a domain \( D \subset \mathbb{R}^1 \) is Lipschitz continuous on \( D \) if for any \( x_1 \neq x_2 \in D \) there is a constant \( L \) such that

\[
|g(x_1) - g(x_2)| \leq L|x_1 - x_2|,
\]

or equivalently

\[
|g(x_1) - g(x_2)| = |x_1 - x_2| \leq L.
\]

Here \( L \) is called the Lipschitz constant. This condition says that we must find one constant \( L \) which works for all points in the domain. Clearly the Lipschitz constant is not unique; for example, if \( L = 5 \), then \( L = 5.1, 6, 10, 100 \), etc. also satisfy the condition. If \( g(x) \) is differentiable then an easy way to determine the Lipschitz constant is to find a constant such that \( |g'(x)| \leq L \) for all \( x \in D \). The linear function \( g(x) = ax + b \) is Lipschitz continuous with \( L = |a| = |g'(x)| \). 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 \( D \) because \( g'(x) = 1/(2\sqrt{x}) \) is not bounded on \( [0, 1] \).

\(^{\text{1}}\)Named after the German mathematician Rudolf Lipschitz (1832-1903).
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. Consequently, Lipschitz continuity is a stronger requirement than continuity but a weaker one than differentiability.

For the existence and uniqueness result for (2.2), we need \( f(t,y) \) to be Lipschitz continuous in \( y \) so we need to extend the above definition by just holding \( t \) fixed. Formally, for fixed \( t \) we have that a function \( g(t,y) \) defined for \( y \) in a prescribed domain is Lipschitz continuous in the variable \( y \) if for any \( (t,y_1), (t,y_2) \) there is a constant \( L \) such that
\[
|g(t,y_1) - g(t,y_2)| \leq L|y_1 - y_2|.
\]

We are now ready to state the theorem which guarantees existence and uniqueness of a solution to (2.2) as well as guaranteeing that the solution depends continuously on the data; i.e., the problem is well-posed. Note that \( y(t) \) is defined on \( [t_0, T] \) whereas \( f(t,y) \) must be defined on a domain in \( \mathbb{R}^2 \). Specifically the first argument \( t \) is in \( [t_0, T] \) but \( y \) can be any real number so that \( D = \{(t,y) \mid t \in [t_0,T], y \in \mathbb{R}_1\} \); a shorter notation for expressing \( D \) is \( D = [t_0,T] \times \mathbb{R}_1 \) which we will employ.

**Theorem 2.1. Existence and uniqueness for IVP (2.2).**

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 (2.3). Then the IVP (2.2) has a unique solution in \( D \) and moreover, the problem is well-posed.

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

### 2.2 Discretization

Even if we know that a solution to (2.2) exists for some choice of \( f(t,y) \), we may not be able to find the closed form solution to the IVP; that is, a representation of the solution in terms of a finite number of simple functions. Even for the simplified case of \( f(t,y) = f(t) \) this is not always possible. For example, consider \( f(t) = \sin t^2 \) which has no explicit formula for its antiderivative. In fact, a symbolic algebra software package like Mathematica gives the antiderivative of \( \sin t^2 \) in terms of the Fresnel Integral which is represented by an infinite power series near the origin; consequently there is no closed form solution to the problem. Although there are numerous techniques for finding the analytic solution of first order differential equations, we are unable to easily obtain closed form analytic solutions for many equations. When this is the case, we must turn to a numerical approximation to
the solution where we give up finding a formula for the solution at all times and instead find an approximation at a set of distinct times.

Probably the most obvious approach to discretizing a differential equation is to approximate the derivatives in the equation by difference quotients to obtain a difference equation which involves only differences in function values. The solution to the difference equation will not be a continuous function but rather a discrete function which is defined over a finite set of points. When plotting the discrete solution one often draws a line through the points to get a continuous curve but remember that interpolation must be used to determine the solution at points other than grid points. In this chapter we concentrate on two of the simplest difference equations for (2.2a).

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

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

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

Our task is to find a means for approximating the solution at each of these discrete values and our hope is that as we perform more calculations with \(N\) getting large, or equivalently \(\Delta t \to 0\), our approximate solution will approach the exact solution in some sense. In the left plot in Figure 2.1 we plot an exact solution (the continuous curve) to a specific IVP and a discrete approximation for \(\Delta t = 0.5\). The approximate solution is plotted only at the points where the difference equation is enforced. From this plot we are unable to say if our discrete solution appears to approach the continuous solution; it is important to realize that when we test our code results at a single time step do not confirm that the results are correct. However, in the figure on the right we plot this discrete solution plus three additional discrete solutions where we decrease \(\Delta t\) by half for each approximation.

By observing the plot and using the “eyeball norm” we can convince ourselves that as \(\Delta t \to 0\) our discrete solution approaches the analytic solution. One of our goals is to make this statement precise and to determine the rate at which our approximate solution converges to the exact solution.

### 2.2.1 The Euler methods

We will see that there are many approaches to deriving discrete methods for our IVP (2.2) but the two simplest methods use the slope of a secant line to approximate the derivative in (2.2a). These methods are called the forward and backward Euler methods named after Leonhard Euler.

The methods can be derived from several different viewpoints but here we use the secant line approximation

\[
y'(t) \approx \frac{y(t_{n+1}) - y(t_n)}{\Delta t} \quad \text{for} \quad t \in [t_n, t_{n+1}]. \tag{2.4}
\]

\(^2\)Euler (1707-1783) was a Swiss mathematician and physicist.
in the figure on the right the discrete solutions for \( \Delta t = 0.5 \) are plotted. From this plot, it is not possible to say that the discrete solution is approaching the exact solution. However, in the figure on the right the discrete solutions for \( \Delta t = 0.5, 0.25, 0.125, \) and 0.625 are plotted. From this figure, the discrete approximations appear to be approaching the exact solution as \( \Delta t \) decreases.

Using this difference quotient to approximate \( y'(t_n) \) gives one method and using it to approximate \( y'(t_{n+1}) \) gives the other method.

When we write a difference equation we need to use different notation for the exact solution \( y(t) \) and its discrete approximation; to this end, we let \( Y^n \) denote the approximation to \( y(t_n) \). Clearly \( Y^0 = y_0 \) which is the given initial condition (2.2b). We now want to write a difference equation which will allow us to calculate \( Y^1 \approx y(t_1) \). We use the differential equation evaluated at \( t_0 \), i.e., \( y'(t_0) = f(t_0, y_0) \), and the approximation for \( y'(t_0) \) from (2.4) with \( n = 0 \) and \( t = t_0 \), i.e., \( y'(t_0) \approx (Y^1 - Y^0)/\Delta t \), to get the difference equation

\[
\frac{Y^1 - Y^0}{\Delta t} = f(t_0, Y^0).
\]

We have a starting point \( Y^0 = y_0 \) from our initial condition and thus we can solve for our approximation to \( y(t_1) \) from

\[
Y^1 = Y^0 + \Delta tf(t_0, Y^0).
\]  

(2.5)

Once \( Y^1 \) is obtained we can repeat the process to obtain a difference equation for \( Y^2 \). This procedure is known as the forward Euler method and is defined by the following formula.

**Forward Euler:**

\[
Y^{n+1} = Y^n + \Delta tf(t_n, Y^n), \quad n = 0, 1, 2, \ldots, N - 1
\]  

(2.6)

The term “forward” is used in the name because we write the equation at the point \( t_n \) and difference forward in time to \( t_{n+1} \); note that this implies that the given slope \( f \) is evaluated at the known point \( (t_n, Y^n) \).
The forward Euler method uses the tangent line at \( t_n \) to extrapolate the solution at \( t_{n+1} \); a graphical interpretation is shown in Figure 2.2. To start the method, consider the slope of the tangent line at \((t_0, Y_0) = (t_0, y_0)\) which is a point that lies on the solution curve as well as on the tangent line. The tangent line has slope \( y'(t_0) = f(t_0, Y_0) \); if \( Y^1 \) denotes the point on the tangent line corresponding to \( t_1 \) then the point \((t_1, Y^1)\) satisfies the tangent line equation \( Y^1 - Y_0 = f(t_0, Y_0)(t_1 - t_0) \) which is just Euler’s equation for the approximation to \( y(t_1) \). Now for the second step we don’t have a point on the solution curve to compute the tangent line but if \( \Delta t \) is small, then \( Y^1 \approx y(t_1) \) and \( f(t_1, Y^1) \approx f(t_1, y(t_1)) = y'(t_1) \). So we write the equation passing through \((t_1, Y^1)\) with slope \( f(t_1, Y^1) \) and evaluate it at \( t_2 \) to get \( Y^2 - Y^1 = \Delta t f(t_1, Y^1) \) which again is just the formula for \( Y^2 \) from \( (2.6) \).

It is important to realize that after the first step we do not have the exact slope \( f(t_n, y(t_n)) \) of the tangent line to the solution curve but rather the approximation \( f(t_n, Y^n) \).

We derived the forward Euler method using the secant line approximation \( (2.4) \) for \( y'(t_n) \). When we use this quotient to approximate \( y'(t_{n+1}) \) a very different situation arises. At the first step we now approximate \( y'(t_1) \) by the slope of the secant line

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

so substituting this approximation into the differential equation \( y'(t_1) = f(t_1, y(t_1)) \).
leads to the difference equation
\[ \frac{Y_1 - Y_0}{\Delta t} = f(t_1, Y_1) \Rightarrow Y_1 = Y_0 + \Delta t f(t_1, Y_1). \]

It is important to realize that this equation is inherently different from (2.5) because we must evaluate \( f(t, y) \) at the unknown point \((t_1, Y_1)\). In general, this leads to a nonlinear equation to solve for each \( Y^n \) 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 for the unknown \( Y^1 \). This method is called the backward Euler method because we are writing the equation at \( t_{n+1} \) and differencing backwards in time to \( t_n \).

The difference between the forward and backward Euler schemes is so important that we use this characteristic to broadly classify methods. The forward Euler scheme given in (2.6) is called an explicit scheme because we can write the unknown explicitly in terms of known values. The backward Euler method given in (2.7) is called an implicit scheme because the unknown is written implicitly in terms of known values and itself. The terms explicit/implicit are used in the same manner as explicit/implicit differentiation. In explicit differentiation a function to be differentiated is given explicitly in terms of the independent variable such as \( y(t) = t^3 + \sec t \); in implicit differentiation the function is given implicitly such as \( y^3 + \sin y - t^2 = 4 \) and we want to compute \( y'(t) \). In the exercises you will get practice in identifying schemes as explicit or implicit.

In the case when \( f(t, y) \) is linear in \( y \) we can get a general formula for \( Y^n \) in terms of \( Y^0 \) and \( \Delta t \) for both the forward and backward Euler methods by applying the formulas recursively. This means that we can compute an approximation to \( y(t_n) \) without computing approximations at \( y(t_{n-1}), \ldots, y(t_1) \) which we normally have to do. The following example illustrates this for the forward Euler method and in the exercises you are asked to find the analogous formula for the backward Euler method. In the next example we fix the time step and compare the relative error for a range of final times; in Exercise 2.5 we fix the final time and reduce the time step.

**Example 2.3.** General solution to forward euler difference equation for linear equations

Consider the IVP
\[ y'(t) = -\lambda y \quad 0 < t \leq T, \quad y(0) = y_0 \]
whose exact solution is \( y(t) = y_0 e^{-\lambda t} \). Find the general solution for \( Y^n \) in terms of \( Y^0 \) and \( \Delta t \) for the forward Euler method.

For the forward Euler method we have
\[
Y^1 = Y^0 + \Delta t(-\lambda Y^0) = (1 - \lambda \Delta t)Y^0.
\]
Similarly
\[
Y^2 = (1 - \lambda \Delta t)Y^1 = (1 - \lambda \Delta t)^2Y^0
\]
Continuing in this manner gives
\[
Y^n = (1 - \lambda \Delta t)^nY^0.
\]

**Example 2.4. Error in the forward euler method for fixed time**

In this example, we fix the time step \( \Delta t \) and compare the relative error for a range of times using the formula in Example 2.3. Set \( \lambda = 5 \), \( y_0 = 2 \) and \( \Delta t = 1/20 \) in Example 2.3 and compute the relative error at \( t = 0.2, 0.4, \ldots, 1.0 \).

For this choice of \( \Delta t \) and \( \lambda \) the forward Euler formula we derived in Example 2.3 reduces to \( Y^n = 2(0.95^n) \). We give the relative error as a percent; it is computed by taking the actual error, normalizing by the exact solution and converting to a percent. The table below gives the approximations.

<table>
<thead>
<tr>
<th>( t )</th>
<th>Percentage Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>2.55%</td>
</tr>
<tr>
<td>0.4</td>
<td>5.04%</td>
</tr>
<tr>
<td>0.6</td>
<td>7.47%</td>
</tr>
<tr>
<td>0.8</td>
<td>9.83%</td>
</tr>
<tr>
<td>1.0</td>
<td>12.1%</td>
</tr>
</tbody>
</table>

As can be seen from the table the relative error increases as the time increases which we expect because the errors are being accumulated; this will be discussed in detail in the next section. It is important to compute a relative error because if the exact solution is near zero then the absolute error may be small while the relative error is large.

**Example 2.5. Error in the forward euler method as time step decreases**

In this example we fix the time at which the error is calculated and vary \( \Delta t \) using the formula derived in Example 2.3 for the forward Euler method with \( \lambda = 5 \) and \( y_0 = 2 \). We compute the absolute and relative errors at \( t = 1 \) for \( \Delta t = 1/10, \ldots, 1/320 \) and discuss the results. Then we determine how many times do have to halve the time step if we want the relative error to be less than 1%.

We use the expressions for \( Y^n \) from Example 2.3 to determine the approximations to \( y(1) = 2e^{-5} \approx 1.3476 \times 10^{-2} \).
2.2.2 Discretization errors

When we implement a numerical method on a computer the error we make is due to both round-off and discretization error. Rounding error is due to using a computer 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 will be illustrated in Example 2.8.

We are mainly concerned with discretization error here and when we derive error estimates we will assume that no rounding error exists. In Figure 2.3 we illustrate approximations to a known exact solution using the forward Euler method. 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 the tangent line approximation for $y'(t_0)$ and thus 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 source of error 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 $t_n$ is the magnitude of the actual error at the point whereas the local truncation error or local discretization error is the error made because we solve the difference equation rather than the actual differential equation. The exact solution to the differential equation does not satisfy the difference equation exactly; the local error is just this remainder. Thus to measure the local truncation error we plug the exact solution into the difference equation and calculate the remainder. An equivalent way to view the local truncation error is to determine the difference in $y(t_{n+1})$ and the approximation obtained by assuming all previous values in the difference equation are known exactly. The local error for the forward Euler method at $t_2$ is illustrated in the plot on the right in Figure 2.4.
2.2. DISCRETIZATION

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 2.3: 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.

and is the difference in $y(t_2)$ and $\tilde{Y}^2 = y(t_1) + \Delta t f(t_1, y(t_1))$, i.e., the remainder when the exact solution is substituted into the difference equation at $t_1$. The global error at $t_2$ is illustrated in the plot on the left of the figure and is just the difference in $y(t_2)$ and $\tilde{Y}^2$.

We now want to determine the local truncation error for the forward Euler method so we substitute the exact solution to (2.2a) into the difference equation (2.6) and calculated the remainder. If $\tau_{n+1}$ denotes the local truncation error at the $(n+1)$st time step then

$$
\tau_{n+1} = y(t_{n+1}) - \left[ y(t_n) + \Delta t f(t_n, y(t_n)) \right].
$$

Our strategy is to first quantify the local truncation error in terms of $\Delta t$ and then use this result to determine the global error. In order to to combine terms in (2.8) we need all terms to be evaluated at the same point $(t_n, y(t_n))$. The only term not at this point is the exact solution $y(t_{n+1})$ so we use a Taylor series with remainder (see (??) in Appendix) because the expansion is in terms of $y(t_n)$ and its derivatives at $t_n$. We have

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

Substituting this into the expression (2.8) for the truncation error yields

$$
\tau_{n+1} = \left[ y(t_n) + \Delta t f(t_n, y(t_n)) + \frac{(\Delta t)^2}{2!} y''(\xi) \right] - \left[ y(t_n) + \Delta t f(t_n, y(t_n)) \right]
$$
30 CHAPTER 2. INTRODUCTION TO INITIAL VALUE PROBLEMS

\[ y(t_0) \quad Y^0 \]
\[ \quad \quad Y^1 \]
\[ \quad \quad \quad \quad Y^2 = Y^1 + \Delta t f(t_1, Y^1) \]
\[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \tau = \max_{1 \leq n \leq N} |\tau_n| \leq \frac{M}{2} (\Delta t)^2. \]  

where we have used the differential equation at \( t_n \), i.e., \( y'(t_n) = f(t_n, y(t_n)) \). If \( y''(t) \) is bounded on \([0, T]\), say \( |y''(t)| \leq M \) and \( T = t_0 + N \Delta t \), then we have

\[ y'(t_1) \]
\[ \quad \quad Y^1 \]
\[ \quad \quad \quad \quad Y^2 = y(t_1) + \Delta t f(t_1, y(t_1)) \]

\[ \Rightarrow \]

We say that the local truncation error for Euler’s method is order \( (\Delta t)^2 \) which we write as \( O(\Delta t^2) \). 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. This means that if we compute the local error for \( \Delta t \) then the local error using \( \Delta t/2 \) will be reduced by approximately \((1/2)^2 = 1/4\). Remember, however, that this is not the global error but rather the error made because we have used a finite difference quotient to approximate \( y'(t) \).

We now turn to estimating the global error in the forward Euler method. We should expect to only be able to find an upper bound for the error because if we can find a formula for the exact error, then we can calculate this and add it to the approximation to get the exact solution. The proof for the global error for the forward Euler method is a bit technical but it is the only global error estimate that we will derive because the methods we consider will follow the same relationship between the local and global error as the Euler method.

Figure 2.4: A comparison of the global error (left figure) and the local truncation error (right figure) at \( t_2 \) for the forward Euler method. The global error is the total error made whereas the local truncation error is the error due to the discretization of the differential equation.
2.2. DISCRETIZATION

Our goal is to demonstrate that the global discretization error for the forward Euler method is $O(\Delta t)$ which says that the method is first order, i.e., linear in $\Delta t$. At each step we make a local error of $O(\Delta t^2)$ 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 $\Delta t$. 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 $\Delta t^2$ so for $N$ steps we have $NC(\Delta t)^2 = (T - t_0)/\Delta t |C\Delta t|^2 = O(\Delta t)$.

Theorem 2.2 provides a formal statement and proof for the global error of the forward Euler method. Note that one hypothesis of Theorem 2.2 is that $f(t,y)$ must be Lipschitz continuous in $y$ which is also the hypothesis of Theorem 2.1 which guarantees existence and uniqueness of the solution to the IVP (2.2) so it is a natural assumption. We also assume that $y(t)$ possesses a bounded second derivative because we will use the local truncation error given in (2.9); however, this condition can be relaxed but it is adequate for our needs.

**Theorem 2.2.** GLOBAL ERROR ESTIMATE FOR THE FORWARD EULER METHOD

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 (2.3) with Lipschitz constant $L$. Also assume that there is a constant $M$ such that

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

Then the global error at each point $t_n$ satisfies

$$|y(t_n) - Y_n| \leq C \Delta t$$

where $C = M e^{TL}/2L (e^{TL} - 1)$;

thus the forward Euler method converges linearly.

**Proof.** Let $E_n$ represent the global discretization error at the specific time $t_n$, i.e., $E_n = |y(t_n) - Y_n|$. The steps in the proof can be summarized as follows.

**Step I.** Use the definition of the local truncation error $\tau_n$ to demonstrate that the global error satisfies

$$E_n \leq K E_{n-1} + \tau \quad \text{for } K = 1 + \Delta t L \quad (2.10)$$

where $\tau$ is the maximum of all $|\tau_n|$.

**Step II.** Apply (2.10) recursively and use the fact that $E_0 = 0$ to get

$$E_n \leq \tau \sum_{i=0}^{n-1} K^i. \quad (2.11)$$
CHAPTER 2. INTRODUCTION TO INITIAL VALUE PROBLEMS

Step III. Recognize that the sum in (2.11) is a geometric series whose sum is known to get

\[ E_n \leq \frac{\tau}{\Delta t L} [(1 + \Delta t L)^n - 1]. \]  

(2.12)

Step IV. Use the Taylor series expansion of \( e^{\Delta t L} \) near zero to bound \((1 + \Delta t L)^n\) by \( e^{\Delta t L} \) which in turn is less than \( e^{\Delta t L} \).

Step V. Use the bound (2.9) for \( \tau \) to get the final result

\[ E_n \leq \frac{M \Delta t^2}{2 \Delta t L} (e^{\Delta t L} - 1) = C \Delta t \] where \( C = \frac{M}{2L} (e^{\Delta t L} - 1). \)  

(2.13)

We now give the details for each step. For the first step we use the fact that the local truncation error is the remainder when we substitute the exact solution into the difference equation; i.e.,

\[ \tau_n = y(t_n) - y(t_{n-1}) - \Delta t f(t_{n-1}, y(t_{n-1})). \]

To get the desired expression for \( E_n \) we solve for \( y(t_n) \) in the above expression, substitute into the definition for \( E_n \) and use the triangle inequality; then we use the forward Euler scheme for \( Y^n \) and (2.3). We have

\[
E_n = \left| \left( \tau_n + y(t_{n-1}) + \Delta t f(t_{n-1}, y(t_{n-1})) \right) - Y^n \right|
\]

\[
= \left| \tau_n + y(t_{n-1}) + \Delta t f(t_{n-1}, y(t_{n-1})) - \left[ Y^{n-1} + \Delta t f(t_{n-1}, Y^{n-1}) \right] \right|
\]

\[
\leq \left| \tau_n \right| + \left| y(t_{n-1}) - Y^{n-1} \right| + \Delta t \left| f(t_{n-1}, y(t_{n-1})) - f(t_{n-1}, Y^{n-1}) \right|
\]

\[
\leq \left| \tau_n \right| + E_{n-1} + \Delta t L \left| y(t_{n-1}) - Y^{n-1} \right| = \left| \tau_n \right| + (1 + \Delta t L) E_{n-1}.
\]

In the final step we have used the Lipschitz condition (2.3) which is a hypothesis of the theorem. Since \( |\tau_n| \leq \tau \), we have the desired result.

For the second step we apply (2.10) recursively

\[
E_n \leq KE_{n-1} + \tau \leq KE_{n-2} + \tau = K^2 E_{n-2} + (K + 1) \tau
\]

\[
\leq K^3 E_{n-3} + (K^2 + K + 1) \tau
\]

\[
\leq \ldots
\]

\[
\leq K^n E_0 + \tau \sum_{i=0}^{n-1} K^i.
\]

Because we assume for analysis that there are no roundoff errors, \( E_0 = |y_0 - Y_0| = 0 \) we are left with \( \tau \sum_{i=0}^{n-1} K^i \). For the third step we simplify the sum by noting that it is a geometric series of the form \( \sum_{k=0}^{n-1} a r^k \) with \( a = \tau \) and \( r = K \). From calculus we know that the sum is given by \( a(1 - r^n)/(1 - r) \) so that if we use the fact that \( K = 1 + \Delta t L \) we arrive at the result (2.12)

\[
E_n \leq \tau \left( \frac{1 - K^n}{1 - K} \right) = \tau \left( \frac{K^n - 1}{K - 1} \right) = \frac{\tau}{\Delta t L} \left[ (1 + \Delta t L)^n - 1 \right].
\]
To justify the fourth step we know that for real \( z \) the Taylor series expansion 
\[ e^z = 1 + z + z^2/2! + \cdots \] 
near zero implies that \( 1 + z \leq e^z \) so that \((1 + z)^n \leq e^{nz}\). If we set \( z = \Delta tL \) we have 
\[ (1 + \Delta tL)^n \leq e^{n\Delta tL} \] 
so that 
\[ E_n \leq \frac{\tau}{nL} (e^{n\Delta tL} - 1). \]

For the final step we know from the hypothesis of the theorem that \( |y''(t)| \leq M \) so \( \tau \leq M\Delta t^2/2 \). Also \( n \) in \( E_n \) is the number of steps taken from \( t_0 \) so \( n\Delta t = t_n \leq T \) where \( T \) is the final time and so \( e^{n\Delta tL} \leq e^{TL} \). Combining these results gives the desired result (2.13).

In general, the calculation of the local truncation error is straightforward (but sometimes tedious) whereas the proof for the global error estimate is much more involved. However, for the methods we consider, if the local truncation error is \( O(\Delta t^r) \) then we expect the global error to be one power of \( \Delta t \) less, i.e., \( O(\Delta t^{r-1}) \). One can show that the backward Euler method also has a local truncation error of \( O(\Delta t^2) \) so we expect a global error of \( O(\Delta t) \). You are asked to demonstrate the local truncation rigorously in the exercises. The calculation is slightly more complicated than for the forward Euler method because the term \( f(t_{n+1}, y(t_{n+1})) \) must be expanded in a Taylor series with two independent variables whereas for the forward Euler method we did not have to expand \( f(t_n, y(t_n)) \) because it was already at the desired point \((t_n, y(t_n))\).

In the following section we look at some specific examples of the IVP (2.2) and use both forward and backward Euler methods; we will demonstrate that our numerical rate of convergence agrees with the theoretical rate. However, we should keep in mind that \( O(\Delta t) \) is a very slow rate of convergence and ultimately we need to derive methods which converge more quickly to the solution.

### 2.3 Numerical computations

In this section we provide some numerical simulations for IVPs of the form (2.2) using both the forward and backward Euler methods. We begin by looking at some examples of IVPs which include exponential, logistic, and logarithmic growth/decay. Before providing the simulation results we discuss the computer implementation of both the forward and backward Euler methods. For the simulations presented we choose problems with known analytic solutions so that we can compute numerical rates of convergence and compare with the theoretical result given in Theorem 2.2. To do this we compute approximate solutions for a sequence of step sizes where \( \Delta t \to 0 \) and then compute a numerical rate of convergence using \( \S 1.6 \). As expected, the numerical rate of convergence for both methods is linear; however, we will see that the forward Euler method does not provide reliable results for all choices of \( \Delta t \) for some problems. Reasons for this failure will be discussed in \( \S ?? \).
2.3.1 Growth and decay models

In modeling various phenomena, a function often obeys a particular growth or decay pattern. Common growth rates are (i) exponential where \( y(t) = ae^{bt} \), (ii) logarithmic where \( y(t) = a + b\ln t \) or \( y(t) = a + b\log_e t \), and (iii) logistic where \( y(t) = a/(b+e^{-rx}+c) \). These growth patterns are illustrated in Figure 2.3.1. Each of these patterns correspond to the solution of a differential equation which arises when modeling a particular phenomenon. We will discuss two of these here; logarithmic growth will be explored in the exercises as well as Gaussian behavior.

Exponential growth and decay

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

\[
p'(t) = r_0 p(t) \quad t \in (t_0, T]
\]

along with the initial condition

\[
p(0) = p_0,
\]

where \( r_0 \) is the given proportionality constant. We can solve this differential equation by using the technique of separation of variables. We have

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

Thus we see that if the population at any time \( t \) is proportional to the amount present at that time, then it behaves exponentially where the initial population is a multiplicative constant and the proportionality constant \( r_0 \) is the rate of growth.
if it is positive; otherwise it is the rate of decay. In the exercises you are asked to explore an exponential growth model for bread mold.

**Logistic growth and decay**

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

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

(2.15)

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 $\left( 1 - \frac{p}{K} \right)$ 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 $\left( 1 - \frac{p}{K} \right)$ 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 (2.15), 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

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

(2.16)

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

**Logarithmic growth**

Logarithmic growth has a period of rapid increase followed by a period where the growth rate slows. This is in contrast to exponential growth which has a slow initial growth rate and then grows more rapidly. Unlike logistic growth, there is no
upper bound for the growth. The IVP whose solution obeys the logarithmic growth
\[ y(t) = a + b \ln t \]
is
\[ y'(t) = \frac{b}{t} \quad 1 < t \leq T, \quad y(1) = a. \quad (2.17) \]
An application of logarithmic for modeling strength of earthquakes is explored in the exercises.

2.3.2 Computer implementation

The computer implementation of the forward Euler method requires a single time loop. The information which changes for each IVP is the interval \([t_0, T]\), the initial condition, the given slope \(f(t, y)\) and the exact solution for the error calculation. We incorporate separate functions for \(f(t, y)\) and the exact solution and input the other variables as well as the step size \(\Delta t\). The solution should not be stored for all time; if it is needed for plotting, etc., then the time and solution should be written to a file to be used later. For a single equation it does not take much storage to keep the solution at each time step but when we encounter systems and problems in higher dimensions the storage to save the entire solution can become prohibitively large so it is not good to get in the practice of storing the solution at each time.

The following pseudocode gives an outline of one approach to implement the forward Euler method using a uniform time step. Note that in this algorithm we have specified \(N\), the number of time steps and computed \(\Delta t\) by \((T - t_0) / N\). The only problem with this is when the time interval is, for example, \(\pi\), then \(N\Delta t\) may not be exactly equal to \(\pi\) due to roundoff so for the error you must compare the approximate solution at computed time \(t\) with the exact solution there rather than at \(T\). Alternately, if we specify the initial and final times along with \(\Delta t\), then \(N\) can be computed from \(N = (T - t_0) / \Delta t\). Even then \(N\Delta t\) may not be exactly \(T\) so care should be taken.

**Algorithm 1. FORWARD EULER METHOD**

**Define:** external function for the given slope \(f(t, y)\) and exact solution for error calculation

**Input:** the initial time, \(t_0\), the final time \(T\), the initial condition \(y = y_0\), and the number of uniform time steps \(N\)

\[ \Delta t = (T - t_0) / N \]
\[ t = t_0 \]

for \(i=1:N\)

\[ m = f(t, y) \]
\[ y = y + \Delta t \; m \]
\[ t = t + dt \]

output \(t, y\)
compute error at final time $t$

When we modify this code to incorporate the backward Euler scheme we must add an interior loop which solves the resulting nonlinear equation.

### 2.3.3 Numerical examples using backward and forward Euler methods

We now consider examples involving the exponential and logistic growth/decay models discussed in § 2.3.1. We apply both the forward and backward Euler methods for each problem and compare the results and work involved. The global error is computed at a fixed time and we expect that as $\Delta t$ decreases the global error at that fixed time decreases linearly. To confirm that the numerical approximations are valid, for each pair of successive errors we use (1.6) to calculate the numerical rate. Then we can easily see if the numerical rates approach one as $\Delta t \to 0$. In one example we compare the results from the forward and backward Euler methods for an exponential decay problem and see that in this case the forward Euler method gives numerical approximations which oscillate and grow unexpectedly whereas the backward Euler method provides reliable results. In the last example, we compare the local truncation error with the global error for the forward Euler method.

---

**Example 2.6. Using backward and forward Euler methods for exponential growth**

Consider the exponential growth problem

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

whose exact solution is $p(t) = 2e^{0.8t}$. We compute approximations at $t = 1$ using a sequence of decreasing time steps for both the forward and backward Euler methods and demonstrate that the numerical rate of convergence is linear by using the formula (1.6) and by using a graphical representation of the error. We then ask ourselves if the time at which we compute the results affect the errors or the rates.

We first compute approximations using the forward Euler method with $\Delta t = 1/2, 1/4, 1/8, 1/16$ and plot the exact solution along with the approximations. Remember that the approximate solution is a discrete function but we have connected the points for illustration purposes. These results are plotted in the figure below and we can easily see that the error is decreasing as we decrease $\Delta t$. 

To verify that the global error is $O(\Delta t)$ we compare the discrete solution to the exact solution at the point $t = 1$ where we know that the exact solution is $e^x = 4.45108$; we tabulate our approximations $P_n$ to $p(t)$ at $t = 1$ and the global error in the table below for $\Delta t = 1/2, 1/4, \ldots, 1/128$. By looking at the errors we see that as $\Delta t$ is halved the error is approximately halved so this suggests linear convergence; the calculation of the numerical rate of convergence makes this result precise because we see that the sequence {.805,.891,.942,.970,.985,.992} tends to one. In the table the approximations and errors are given to five digits of accuracy.

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>1/2</th>
<th>1/4</th>
<th>1/8</th>
<th>1/16</th>
<th>1/32</th>
<th>1/64</th>
<th>1/128</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>p(1) - P_n</td>
<td>$</td>
<td>0.53108</td>
<td>0.30388</td>
<td>0.16390</td>
<td>0.085333</td>
<td>0.043568</td>
</tr>
<tr>
<td>num. rate</td>
<td>0.805</td>
<td>0.891</td>
<td>0.942</td>
<td>0.970</td>
<td>0.985</td>
<td>0.992</td>
<td></td>
</tr>
</tbody>
</table>

We can also demonstrate graphically that the convergence rate is linear by using a log-log plot. Recall that if we plot a polynomial $y = ax^r$ on a log-log scale then the slope is $r$.\footnote{Using the properties of logarithms we have $\log y = \log a + r \log x$ which implies $Y = mX + b$ where $Y = \log y$, $X = \log x$, $m = r$ and $b = \log a$.}
Since the error is $E = C\Delta t^r$, if we plot it on a log-log plot we expect the slope to be $r$ and in our case $r = 1$. This is illustrated in the log-log plot below where we computed the slope of
the line for two points.

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

| \( \Delta t \) | \( P^n \) | \( |p(0.5) - P^n| \) | num. rate |
|-------|----------|-----------------|----------|
| 1/2   | 2.8000   | 0.18365         | 0.825    |
| 1/4   | 2.8800   | 0.10365         | 0.902    |
| 1/8   | 2.9282   | 0.055449        | 0.948    |
| 1/16  | 2.9549   | 0.028739        | 0.973    |
| 1/32  | 2.9600   | 0.014638        | 0.986    |
| 1/64  | 2.9763   | 0.0073884       | 0.993    |
| 1/128 | 2.9799   | 0.0037118       |          |

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

\[
P^{n+1} = P^n + 0.8\Delta t P^{n+1},
\]

where \( P^n \approx p(t_n) \). Thus we do not need to use Newton’s method for this particular problem but rather just solve the equation

\[
P^{n+1} = \frac{1}{1 - 0.8\Delta t} P^n.
\]

If we have a code that uses Newton’s method it should get the same answer in one step because it is solving a linear problem rather than a nonlinear one. The results are tabulated below. Note that the numerical rate of convergence is also approaching one but for this method it is approaching one from above whereas using the forward Euler scheme for this problem the convergence was from below, i.e., through values smaller than one. The amount of work required for the backward Euler method is essentially the same as the forward Euler for this problem because the derivative \( f(t, p) \) is linear in the unknown \( p \).

| \( \Delta t \) | \( P^n \) | \( |p(1) - P^n| \) | num. rate |
|-------|----------|-----------------|----------|
| 1/2   | 5.5556   | 1.1045          | 1.355    |
| 1/4   | 4.8828   | 0.43173         | 1.146    |
| 1/8   | 4.6461   | 0.19503         | 1.067    |
| 1/16  | 4.5441   | 0.093065        | 1.032    |
| 1/32  | 4.4966   | 0.045498        | 1.015    |
| 1/64  | 4.4736   | 0.022499        | 1.008    |
| 1/128 | 4.4623   | 0.011188        |          |
Example 2.7. Using backward and forward Euler methods for logistic growth.

Consider the logistic model
\[ p'(t) = 0.8 \left( 1 - \frac{p(t)}{100} \right) p(t) \quad 0 < t \leq 10 \quad p(0) = 2. \]

We want to implement both the forward and backward Euler schemes and demonstrate that we get linear convergence. Also we compare the results from this example with those from Example 2.6 of exponential growth.

The exact solution to this problem is given by (2.16) with \( K = 100 \), \( r_0 = 0.8 \), and \( p_0 = 2 \). Before generating any simulations we should think about what we expect the behavior of this solution to be compared with the exponential growth solution in the previous example. Initially the population should grow at the same rate because \( r_0 = 0.8 \) which is the same growth rate as in the previous example. However, the solution should not grow unbounded but rather always stay below the carrying capacity \( p = 100 \). The approximations at \( t = 1 \) for a sequence of decreasing values of \( \Delta t \) are presented below along with the calculated numerical rates. The exact value at \( t = 1 \) is rounded to 4.3445923. Again we see that the numerical rate approaches one.

<table>
<thead>
<tr>
<th>( \Delta t )</th>
<th>1/2</th>
<th>1/4</th>
<th>1/8</th>
<th>1/16</th>
<th>1/32</th>
<th>1/64</th>
<th>1/128</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>p(1) - P^n</td>
<td>)</td>
<td>0.47799</td>
<td>0.27063</td>
<td>0.14497</td>
<td>0.075179</td>
<td>0.038302</td>
</tr>
<tr>
<td>num. rate</td>
<td>0.821</td>
<td>0.901</td>
<td>0.947</td>
<td>0.973</td>
<td>0.993</td>
<td>0.993</td>
<td></td>
</tr>
</tbody>
</table>

Below we plot the approximate solution for \( \Delta t = 1/16 \) on \([0,10]\) for this logistic growth problem and the previous exponential growth problem. Note that the exponential growth solution increases without bound whereas the logistic growth solution never exceeds the carrying capacity of \( K = 100 \). Also for small time both models give similar results.

We now turn to implementing the backward Euler scheme for this problem. At each step we have the nonlinear equation
\[ P^{n+1} = P^n + \Delta t f(t_{n+1}, P^{n+1}) = P^n + 0.8 \Delta t \left( P^{n+1} - \frac{(P^{n+1})^2}{100} \right) \]
for $P^{n+1}$. Thus to determine each $P^{n+1}$ we have to employ a method such as Newton’s method. To find the root $z$ of the nonlinear equation $g(z) = 0$ (a function of one independent variable) each iteration of Newton’s method is given by

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

for the iteration counter $k = 1, 2, \ldots$ and where an initial guess $z^0$ is prescribed. For our problem, to compute the solution at $t_{n+1}$ we have the nonlinear equation

$$g(z) = z - P^n - 8\Delta t \left( z - \frac{z^2}{100} \right) = 0$$

where $z = P^{n+1}$. Our goal is to approximate the value of $z$ which makes $g(z) = 0$ and this will be our approximation $P^{n+1}$. For an initial guess $z^0$ we simply take $P^n$ because if $\Delta t$ is small enough and the solution is smooth then the approximation at $t_{n+1}$ will be close to the solution at $t_n$. To implement Newton’s method we also need the derivative $g'$ which for us is just

$$g'(z) = 1 - 8\Delta t \left( 1 - \frac{z}{50} \right).$$

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

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

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>1/4</th>
<th>1/8</th>
<th>1/16</th>
<th>1/32</th>
<th>1/64</th>
<th>1/128</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>p(1) - P_n</td>
<td>$</td>
<td>0.3699</td>
<td>0.1693</td>
<td>0.08123</td>
<td>0.03881</td>
</tr>
<tr>
<td>num. rate</td>
<td>1.127</td>
<td>1.060</td>
<td>1.029</td>
<td>1.014</td>
<td>1.007</td>
<td></td>
</tr>
</tbody>
</table>

Typically, three to four Newton iterations were required to satisfy this convergence criteria. It is well known that Newton’s method typically converges quadratically (when it converges) so we should demonstrate this. To this end, we look at the normalized difference in successive iterates. For example, for $\Delta t = 1/4$ at $t = 1$ we have the sequence 0.381966, 4.8198 $10^{-3}$, 7.60327 $10^{-7}$, 1.9105 $10^{-15}$ so that the difference at one iteration is approximately the square of the difference at the previous iteration indicating quadratic convergence.

**Example 2.8.** Numerically unstable computations for the forward Euler method.

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

$$y'(t) = -20y(t) \quad 0 < t \leq 2, \quad y(0) = 1$$
which has an exact solution of \( y(t) = e^{-20t} \).

In the first plot below we graph the approximate solutions on \([0, 2]\) using the forward Euler method with \( \Delta t = \frac{1}{4} \) and \( \frac{1}{8} \). Note that for this problem the approximate solution is oscillating and becoming unbounded.

In the next plot we graph approximations using the backward Euler method along with the exact solution. As can be seen from the plot, it appears that the discrete solution is approaching the exact solution as \( \Delta t \to 0 \). Recall that the backward Euler method is an implicit scheme whereas the forward Euler method is an explicit scheme.

Why are the results for the forward Euler method not reliable for this problem whereas they were for previous examples? In this example the numerical approximations are not converging as \( \Delta t \to 0 \); the reason for this is a stability issue which we address in the next section and in \( \S \) ???. When we determined the theoretical rate of convergence we tacitly assumed that the method converged; which of course for the forward Euler method it does not.

**Example 2.9.** Comparing the local and global errors for the forward Euler method.

We consider the IVP

\[
y'(t) = \cos(t)e^{\sin t} \quad 0 < t \leq \pi \quad y(0) = 0
\]
whose exact solution is $e^{\sin t}$. The goal of this example is to demonstrate that the local truncation error for the forward Euler method is second order, i.e., $O(\Delta t^2)$ and to compare the local and global errors at a fixed time.

The local truncation error at $t_n$ is computed from the formula

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

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

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>local error</th>
<th>num. rate</th>
<th>global error</th>
<th>num. rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>7.713 $10^{-4}$</td>
<td>1.99</td>
<td>2.835 $10^{-2}$</td>
<td>1.03</td>
</tr>
<tr>
<td>1/16</td>
<td>1.947 $10^{-4}$</td>
<td>2.00</td>
<td>1.391 $10^{-2}$</td>
<td>1.02</td>
</tr>
<tr>
<td>1/32</td>
<td>4.879 $10^{-4}$</td>
<td>2.00</td>
<td>6.854 $10^{-3}$</td>
<td>1.02</td>
</tr>
<tr>
<td>1/64</td>
<td>1.220 $10^{-4}$</td>
<td>2.00</td>
<td>3.366 $10^{-3}$</td>
<td>1.00</td>
</tr>
<tr>
<td>1/128</td>
<td>3.052 $10^{-5}$</td>
<td></td>
<td>1.681 $10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

One different aspect about this problem is that in previous examples the stopping point was an integer multiple of $\Delta t$ whereas here the interval is $[0, \pi]$. The computer implementation of the method must check if the time the solution is being approximated at is greater than the final time. If it is, then $\Delta t$ must be reset to the difference in the final time and the previous time so that the last step is taken with a smaller time step; otherwise you will be comparing the solution at different final times. Alternately, one can compare the results at the final time even if it is slightly less or more than $\pi$ as long as the exact solution is evaluated at the same time.

### 2.4 Consistency, stability and convergence

In Example 2.8 we saw that the forward Euler method failed to provide reliable results for some values of $\Delta t$ but in Theorem 2.2 we proved that the global error satisfied $|y(t_n) - Y^n| \leq C\Delta t$ at each point $t_n$. At first glance, the numerical results seem to be at odds with the theorem we proved. However, with closer inspection we realize that to prove the theorem we tacitly assumed that $Y^n$ is the exact solution of the difference equation. This is not the case because when we implement methods small errors are introduced due to round off. We want to make sure that the solution of the difference equation that we compute remains close to the one we would get if exact arithmetic was used. The forward Euler method in Example 2.8 exhibited numerical instability because round off error accumulated so that the computed solution of the difference equation was very different from its exact solution; this resulted in the solution oscillating and becoming unbounded. However, recall that the backward Euler method gave reasonable results for this same problem so the
round off errors here did not unduly contaminate the computed solution to the
difference equation.

Why does one numerical method produce reasonable results and the other pro-
duces meaningless results even though their global error is theoretically the same?
The answer lies in the stability properties of the numerical scheme. In this section
we formally define convergence and its connection with the concepts of consistency
and stability. When we consider families of methods in the next chapter we will
learn techniques to determine their stability properties.

Any numerical scheme we use must be consistent with the differential equa-
tion we are approximating. \( Y^n \) satisfies the difference equation exactly but the
exact solution \( y(t_n) \) yields a residual which we called the local truncation error.
Mathematically we require that

\[
\lim_{\Delta t \to 0} \max_{1 \leq n \leq N} |\tau_n(\Delta t)| = 0 \tag{2.18}
\]

where, as before, \( \tau_n(\Delta t) \) is the local truncation error at time \( t_n \) using step size \( \Delta t \)
and \( N = (T - t_0)/\Delta t \). To understand this definition we can think of forming two
sequences. The first is a sequence of values of \( \Delta t \) which approach zero monoton-
ically such as 0.1, 0.05, 0.025, 0.0125, \ldots and the second is a sequence where the
kth term is the maximum truncation error in \([t_1, t_N]\) where the \( \Delta t \) used is the value in
the kth term of the first sequence. Then the method is consistent if the limit of the
sequence of errors goes to zero. Both the forward and backward Euler methods are
consistent with (2.2) because we proved that the maximum local truncation error
is \( O(\Delta t^2) \) for all \( t_n \). If the local truncation error is constant then the method is
not consistent. Clearly we only want to use difference schemes which are consistent
with our IVP (2.2). However, the consistency requirement is a local one and does
not guarantee that the method is convergent as we saw in Example 2.8.

We now want to determine how to make a consistent scheme convergent. In-
tuitively we know that for a scheme to be convergent the discrete solution at each
point must get closer to the exact solution as the step size reduces. As with con-
sistency, we can write this formally and say that a method is convergent if

\[
\lim_{\Delta t \to 0} \max_{1 \leq n \leq N} |y(t_n) - Y^n| = 0
\]

where \( N = (T - t_0)/\Delta t \).

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

\[
\text{Consistency} + \text{Stability} = \text{Convergence}
\]
To investigate the effects of the (incorrect) computed solution to the difference equation, we let $\tilde{Y}^n$ represent the computed solution to the difference equation which has an actual solution of $Y^n$ at time $t_n$. We want the difference between $y(t_n)$ and $\tilde{Y}^n$ to be small. At a specific $t_n$ we have

$$|y(t_n) - \tilde{Y}^n| = |y(t_n) - Y^n + Y^n - \tilde{Y}^n| \leq |y(t_n) - Y^n| + |Y^n - \tilde{Y}^n|,$$

where we have used the triangle inequality. Now the first term $|y(t_n) - Y^n|$ is governed by making the local truncation error sufficiently small (i.e., making the equation consistent) and the second term is controlled by the stability requirement. So if each of these two terms can be made sufficiently small then when we take the maximum over all points $t_n$ and take the limit as $\Delta t$ approaches zero we get convergence.

In the next chapter we will investigate the stability of methods and demonstrate that the forward Euler method is only stable for $\Delta t$ sufficiently small whereas the backward Euler method is numerically stable for all values of $\Delta t$. Recall that the forward Euler method is explicit whereas the backward Euler is implicit. This pattern will follow for other methods, that is, explicit methods will have a stability requirement whereas implicit methods will not. Of course this doesn’t mean we can take a very large time step when using implicit methods because we still have to balance the accuracy of our results.
2.1. Classify each difference equation as explicit or implicit. Justify your answer.

a. \( Y_{n+1} = Y_n - 1 + 2\Delta t f(t_n, Y_n) \)

b. \( Y_{n+1} = Y_n - 1 + \frac{\Delta t}{3} \left[ f(t_{n+1}, Y_{n+1}) + 4f(t_n, Y_n) + f(t_{n-1}, Y_{n-1}) \right] \)

c. \( Y_{n+1} = Y_n + \frac{\Delta t}{2} \left[ f(t_n, Y_n) - \frac{\Delta t}{2} f(t_n, Y_n) - \frac{\Delta t}{2} f(t_n, Y_n + \frac{\Delta t}{2}) \right] \)

d. \( Y_{n+1} = Y_n + \frac{\Delta t}{3} \left[ f(t_n, Y_n) + 3f(t_n + \frac{\Delta t}{2}, Y_n + \frac{\Delta t}{3}) \right] \)

2.2. Assume that the following set of errors was obtained from three different methods for approximating the solution of an IVP of the form \((2.2)\) at a specific time. First look at the errors and try to decide the accuracy of the method. Then use the result \((??)\) 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.

<table>
<thead>
<tr>
<th>(\Delta t)</th>
<th>Errors Method I</th>
<th>Errors Method II</th>
<th>Errors Method III</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>0.23426 \times 10^{-2}</td>
<td>0.27688</td>
<td>0.71889 \times 10^{-3}</td>
</tr>
<tr>
<td>1/8</td>
<td>0.64406 \times 10^{-3}</td>
<td>0.15249</td>
<td>0.49840 \times 10^{-6}</td>
</tr>
<tr>
<td>1/16</td>
<td>0.16833 \times 10^{-3}</td>
<td>0.80353 \times 10^{-1}</td>
<td>0.32812 \times 10^{-7}</td>
</tr>
<tr>
<td>1/32</td>
<td>0.43215 \times 10^{-4}</td>
<td>0.41292 \times 10^{-1}</td>
<td>0.21048 \times 10^{-8}</td>
</tr>
</tbody>
</table>

2.3. Suppose the solution to the DE \( y'(t) = f(t, y) \) is a concave up function on \([0, T]\). Will the forward Euler method give an underestimate or an overestimate to the solution? Why?

2.4. Show that if we integrate the IVP \((2.2a)\) from \(t_n\) to \(t_{n+1}\) and use a right Riemann sum to approximate the integral of \(f(t, y)\) then we obtain the backward Euler method.

2.5. Suppose we integrate the IVP \((2.2a)\) from \(t_n\) to \(t_{n+1}\) as in \((3.5)\) and then use the midpoint rule

\[
\int_a^b g(x) \, dx \approx (b - a) \, g \left( \frac{a + b}{2} \right)
\]

to approximate the integral of \(f(t, y)\). What approximation to \(y(t_{n+1})\) do you get? What is the difficulty with implementing this method?
2.6. In Figure 2.2 we gave a graphical interpretation of the forward Euler method. Give a graphical interpretation of the backward Euler method and compare with that of the forward Euler method.

2.7. Consider the IVP $y'(t) = -\lambda y(t)$ with $y(0) = 1$. Apply the backward Euler method to this problem and show that we have a closed form formula for $Y^n$, i.e.,

$$Y^n = \frac{1}{(1 + \Delta t)^n}.$$

For the backward Euler method we have

$$Y^1 = Y^0 + \Delta t(-\lambda Y^1) \Rightarrow Y^1 = \frac{1}{(1 + \lambda \Delta t)} Y^0.$$

Similarly

$$Y^2 = \frac{1}{(1 + \lambda \Delta t)} Y^1 = \frac{1}{(1 + \lambda \Delta t)^2} Y^0.$$

Continuing in this manner gives

$$Y^n = \frac{1}{(1 + \lambda \Delta t)^n} Y^0.$$

2.8. Derive the backward Euler method by using the Taylor series expansion for $y(t_n - \Delta t)$.

2.9. Consider approximating the solution to the IVP

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

using the backward Euler method. In this case the given slope $y'(t)$ is linear in $y$ so the resulting difference equation is linear. Use backward Euler to approximate the solution at $t = 0.5$ with $\Delta t = 1/2, 1/4, \ldots, 1/32$. Compute the error in each case and the numerical rate of convergence.

2.10. What IVP has a solution which exhibits the logarithmic growth $y(t) = 2 + 3 \ln t$ where the initial time is prescribed at $t = 1$?

2.11. Determine an approximation to the solution at $t = 0.5$ to the IVP

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

using the forward Euler method with $\Delta t = 1/4$. Compute the local and global errors at $t = 1/4$ and $t = 1/2$. The exact solution is $y(t) = (e^{2t} - 1)/(e^{2t} + 1)$.

---

**Computer Exercises**
2.12. Consider the exponential growth problem

\[ p'(t) = 4p(t) \quad 0 < t \leq 1, \quad p(0) = 2 \]

whose exact solution is \( p(t) = 2e^{4t} \).

This problem has a growth rate which is five times that of the rate in Example 2.6 so we expect it to grow much faster. Tabulate the errors for \( \Delta t = \frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{128} \) and compare with those from the previous example. Compare the numerical errors as well as the actual magnitude of the errors. Why do you think that this problem has larger errors.

2.13. Consider the IVP

\[ y'(t) = \cos^2(t) \cos^2(2y) \quad \frac{\pi}{2} < t \leq \pi, \quad y\left(\frac{\pi}{2}\right) = \pi. \]

This nonlinear differential equation is separable and the IVP has the exact solution \( y(t) = \frac{1}{2} \left[ \arctan \left( t + \frac{1}{2} \sin(2t) - \frac{\pi}{2} \right) + 2\pi \right] \). Compute the solution using forward and backward Euler methods and demonstrate that the convergence is linear. For the backward Euler method incorporate Newton’s method and verify that it is converging quadratically. For each method compute and tabulate the numerical rates using successive values of \( N = 10, 20, 40, \ldots, 320 \). Discuss your results and compare with theory.

2.14. Write a code which implements the forward Euler method to solve an IVP of the form (2.2). Use your code to approximate the solution of the IVP

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

which has an exact solution \( y(t) = (e^{2t} - 1)/(e^{2t} + 1) \). 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 \( \Delta t \) and demonstrate that your method converges with accuracy \( O(\Delta t) \); 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 \( \Delta t \) and determine the rate of convergence of the local error; justify your answer by calculating the numerical rate of convergence for successive pairs of errors. Compare your results with those obtained in (a).

2.15. 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 \( \Delta t \) 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?

<table>
<thead>
<tr>
<th>( t )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0.55</td>
</tr>
<tr>
<td>2</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>2.25</td>
</tr>
<tr>
<td>5</td>
<td>7.5</td>
</tr>
<tr>
<td>7</td>
<td>16.25</td>
</tr>
<tr>
<td>8</td>
<td>19.5</td>
</tr>
<tr>
<td>10</td>
<td>22.75</td>
</tr>
</tbody>
</table>

d. Use the forward Euler method with \( \Delta t \) 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. INTRODUCTION TO INITIAL VALUE PROBLEMS