Chapter 6

Ordinary Differential Equation

6.1 Introduction

Batman fell off while jumping from the *Vampire State Building* to *Votre Brum Cathedral*! Yes, you read it right. He fell off because his *hook thrower* malfunctioned. Lying on the hospital bed, he was trying to piece together the events. He jumped off the *Vampire State Building* and started off the *hook thrower*. According to the design, a laser beam calculates the distance (x) between him and the target (tip of the cathedral tower in this instance) and lets out the hook at right rate.

The computer recalculates this information every 0.001 seconds since he is falling continuously under the influence of gravity and the distance between him and the target is changing. The relation between x and t of the hook line is adjustable. Since, a linear setting is most reliable, *Batman* has been using the following setting for years:

$$\frac{dx}{dt} = 1 + t + x \tag{6.1}$$

It is easy to see that his computer solves this equation every 0.001 seconds with the position of batman at that instance as an initial condition. His computer has been doing it for the past 30 years! So, what went wrong this time? He sent the hook thrower to the original manufacturer for checking. The report came out to be sabotage, possibly by an accomplice of *Joker*. The hook thrower setting has been changed to the following:

$$\frac{dx}{dt} = 1 + t + x^2 \tag{6.2}$$

You can easily see the problem now. The computer is programmed to solve (6.1) but his hook thrower is set to (6.2). Some experts have written the program for solving (6.1) some 30 years ago. In his confident self, Batman never realized that he needed to learn to solve such equations and program them in his computer, not only to avoid such disasters but also to explore the full potential (all settings linear and non-linear) of his hook thrower. So, he decided to learn it and as you can imagine, with *Batman*-ish rigour. He needs to learn very accurate methods to solve them under all conditions and also learn the situations where they will fail. Here we will try to learn with the *Batman*.

The equations of type (6.1) and (6.2) can be generally written as:

$$\frac{dy}{dt} = f(y,t) \tag{6.3}$$

Equations of this form requires one *initial condition* in the form of $y = y_0$ at $t = t_0$ in order to obtain a solution. Initial point can be an arbitrarily chosen fixed point and for engineering problems, it is often governed by practical considerations or measurement at a known point. For *Batman*, it was readjusted every 0.001 seconds. These are known as *initial value problems* (IVPs).

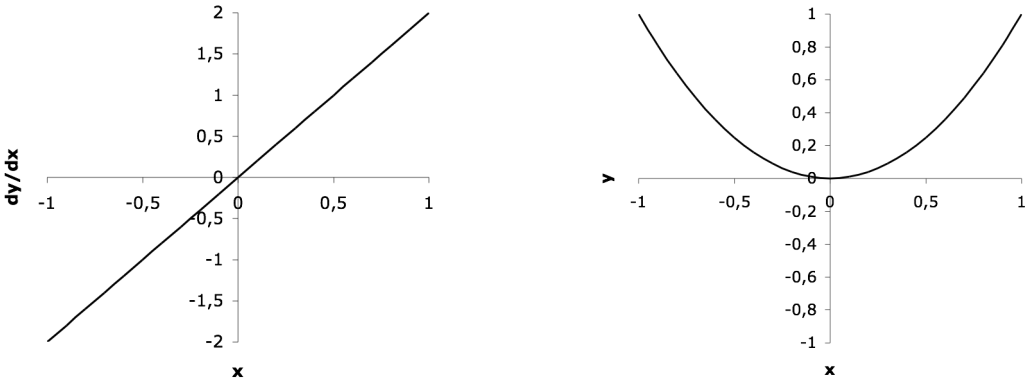
The independent variable does not necessarily have to be time. For example, the parabola expressed in the form is also an IVP where the independent variable x behaves as time variable:

$$\frac{dy}{dx} = 2x, \quad y = 0 \text{ at } x = 0 \tag{6.4}$$

We will spend much of this chapter developing methods for IVPs. We will see later on that this will form the basis for solution of all other forms of differential equations including partial differential equations described in Chapter 7. Before we proceed onto mathematical rigour, let us develop a visual sense of what it means to solve an

IVP. Once again remember that computers cannot carry out any operation other than +, -, × and ÷.

In the simplest words, the equation (6.4) is saying that, "the *slope at any point* (*x*, *y*) *is 2x*". So, for equation (6.4), the slope in the interval [-1, 1] is known apriori (Figure 6.1a). The challenge is to construct the solution *y* vs. *x* (Figure 6.1b) using the information of Figure 6.1a and arithmetic operations.



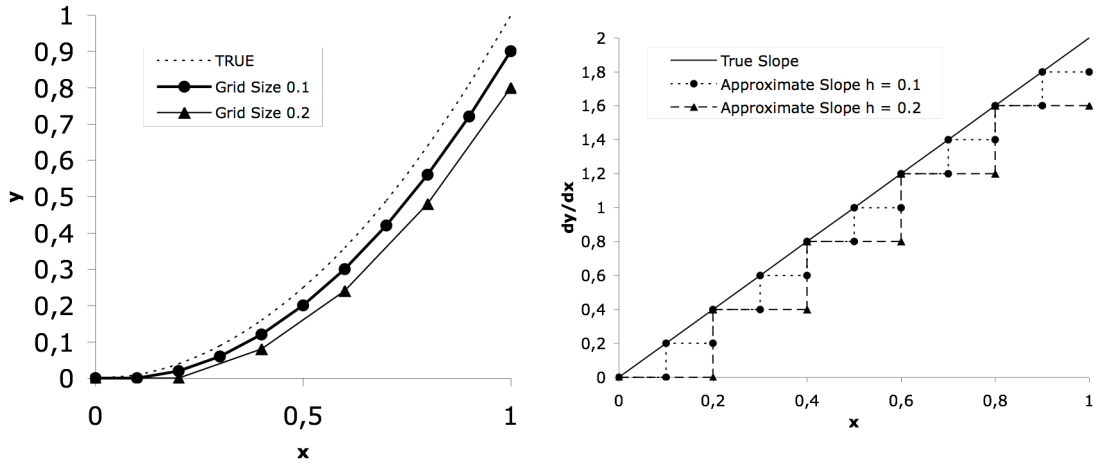
(a) **(b)**

Figure 6.1: Slope function (a) and solution (b) of equation (6.4)

Let us attempt to solve it graphically by using this information. Recall in numerical methods, we can only obtain a discrete approximation of a continuous function (*y*). We have a starting point in the form of initial condition. From the previous chapter, you are already familiar with grids. Let us chose to obtain the discrete approximation of *y* at an *x*-grid size (*h*) of 0.2. Since, *y*-axis is an *axis of symmetry*, let us consider only the 1st quadrant. We know that the solution passes through the point (0, 0), the initial condition (equation 6.4). We take a plane paper and mark this point. For the next point, we know the *x* coordinate as $h = 0.2$. But to locate the *y* coordinate, we need the direction or slope. If we have the slope, we can draw a straight line at that slope from the initial point and a vertical line from $x = 0.2$. The intersection will give us the required point. We know that the slope in the interval (0, 0.2) is not a constant but a function of *x* and therefore, varies at every point. One *key assumption* in the development of numerical methods for IVP is that the *slope remains constant within the grid length*. This is also to say that function joining two points on a grid is a straight line. It is then easy to visualize that finer grid will bring us closer to reality as smaller segments of curves, can be better approximated as straight lines. Next question obviously is *what is the value of this constant slope*. Since the slope is a continuous function, any finite interval can contain infinitely many choices of points where we can evaluate the slope and apply it over the whole interval. In fact, there could be as many choices as the stars there are on the sky. We shall see in this chapter that the primary difference between most of the numerical methods for IVPs is the way the slope is evaluated over a grid length.

For illustration, we will choose the easiest option. That is, whatever the slope is at the starting point of a grid length (an interval) remains constant over that interval. For the first interval, we then calculate the slope at the origin as zero using eq (6.4).

Assuming the slope is constant along one grid length, we go one grid distance (0.2) along the gradient and mark the 2nd point. At the new point, we calculate the slope as 0.4. Once again we travel one grid length along the gradient from this point to mark the 3rd point. At the 3rd point, we pick the gradient as 0.8 and proceed for one more grid. We continue this way for the entire range of x we are interested in. Once we reach the point 1, we can join the points with *straight lines* to get an approximation of the original curve. Using this procedure, the resulting approximation is shown in Figure 6.2a. Since the slope is constant over each interval, it becomes a step function approximation of the continuous slope function (Figure 6.2b). As expected, refining the grid enhances the approximation. We see in Figure 6.2a that refinement of grid size to 0.1 has moved the approximation closer to the true solution.



(a) **(b)**

Figure 6.2: Numerical solution (a) of equation (6.4) by Euler forward method (6.5) and the approximate slope function (b).

One can easily verify that the mathematical equation for obtaining the y_{n+1} from y_n by following the above procedure is given by,

$$y_{n+1} = y_n + 2x_n h \tag{6.5}$$

The steps of obtaining y_{n+1} from y_n can now be formally put to word as follows for the generalized problem (6.3):

- Choose a time step h and the grid $\{t_0, \quad t_1, \quad t_2, \quad \cdot \quad \cdot \quad t_n\}$ on t.
- Compute slopes at the grid points as $f(y_n, t_n)$
- Move one grid length along the slope from the present point y_n

If you put mathematical expressions for the process described above for obtaining y_{n+1} from y_n between any two successive grid points t_n and t_{n+1}, the approximate form of the general equation (6.3) becomes,

$$y_{n+1} = y_n + hf(y_n, t_n) \tag{6.6}$$

Please note, at any point of time y_n, t_n, h and the functional form of f is known. The problem is to compute y_{n+1} at the next grid point. This can be done *explicitly* using equation (6.6) for any arbitrary functional form of f. A small rearrangement of (6.6) leads to,

$$\frac{y_{n+1} - y_n}{h} = f(y_n, t_n) \tag{6.7}$$

Now compare the left hand side of the equation with eq. 5.2. It is easy to see that we approximated the derivative in eq.(6.3) with a *forward difference* approximation and evaluated the functional value of f at the known point (y_n, t_n). The method for solving IVPs given by equations (6.6) or (6.7) is known as *Euler Explicit* or *Euler Forward*.

In the Euler Forward method, the slope at any interval $\{t_n, t_{n+1}\}$ was approximated by the slope at the beginning of the interval or t_n. For the example of parabola, the true slope (straight line) was approximated by step functions. The true and approximate slope in the interval $\{0, 1\}$ is shown in Figure 6.2b for grid sizes of 0.2 and 0.1. A natural question may arise, why not approximate the slope in the interval with, (i) the slope at the end of the interval, i.e., at t_{n+1} or, (ii) average of the two slopes at t_n and t_{n+1}. Application of these two will modify the method of (6.6) to,

Case (i): $$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \tag{6.8}$$

Case (ii): $$y_{n+1} = y_n + h\left(\frac{f(y_n, t_n) + f(y_{n+1}, t_{n+1})}{2}\right) \tag{6.9}$$

The method given by eq. (6.8) although looks very similar to eq. (6.6), is not very easy to apply. In order to illustrate this, let us apply eq. (6.8) to approximate eq. (6.1) and eq. (6.2) of the batman problem.

Eq. (6.1): $y_{n+1} = y_n + h(1 + t_{n+1} + y_{n+1})$ or $y_{n+1} = \frac{y_n + h(1 + t_{n+1})}{(1 - h)}$

Eq. (6.2): $y_{n+1} = y_n + h(1 + t_{n+1} + y_{n+1}^2)$

We observe that the right hand side of the approximation cannot be explicitly computed anymore. In fact, if the functional form is non-linear in y (6.2), one needs to solve a non-linear algebraic equation at every step in order to get a solution. Such methods involving slope evaluation at the end point (t_{n+1}) of the interval is known as *implicit* methods as opposed to the *explicit* methods described earlier where the value at y_{n+1} can be computed explicitly. Thus, the method described by equation (6.8) for solving IVPs is known as *Euler Implicit* or *Euler Backward* method.

In order to have a mathematical understanding of the Case (ii) in equation (6.9), let us integrate the original problem (6.3) between two adjacent grid points,

$$\int_{y_n}^{y_{n+1}} dy = \int_{t_n}^{t_{n+1}} f(y,t)dt \tag{6.10}$$

If we numerically integrate the right hand side using trapezoidal method, we obtain the equation (6.9). So, this method is also known as the Trapezoidal method. You may ask again, why trapezoidal method? Why not Simpson's 1/3rd or 3/8th rule? More generally, why stop by averaging at two points? Why not consider more points to evaluate the slope?

A graphical representation of the iterations of Euler Forward, Euler Backward and Trapezoidal method (equations 6.6, 6.8 and 6.9) is shown in Figure 6.3.

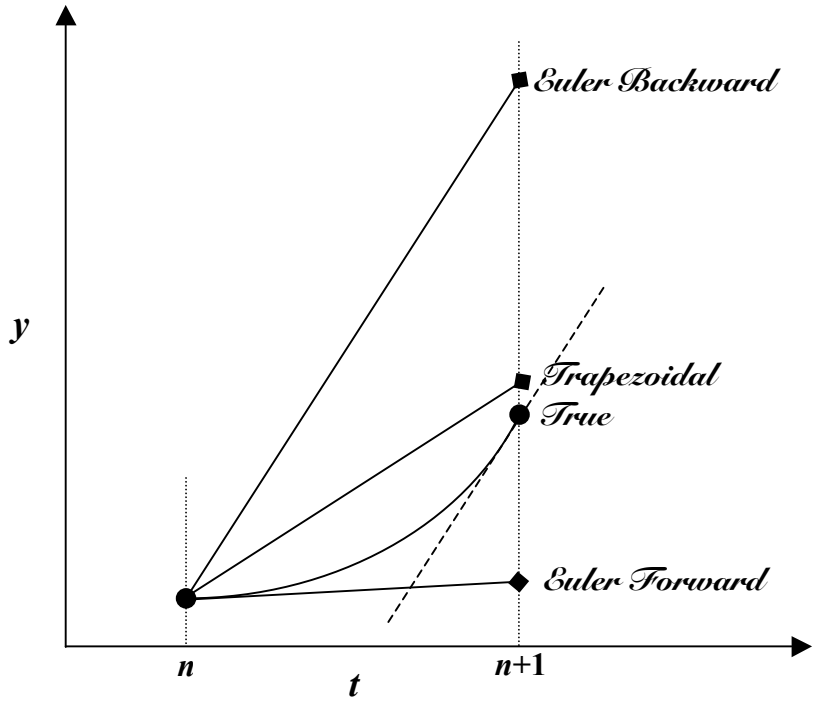


Figure 6.3: Schematic of time stepping scheme for Euler methods and Trapezoidal method. Euler forward method extends the slope at *n*, Euler backward extends the slope at *n*+1 and the Trapezoidal method extends the average of the two slopes.

Yet another way to look at the problem will be to revisit the approximation of the derivative of the left hand side of eq. (6.3). So far, we only did forward difference approximation. In the previous chapter, we have learned many ways to approximate the first derivative using multiple grid points ? Why not use some of those ?

Now, try combining the various options of approximating the right hand side derivatives with the options of single or multiple point functional evaluation of slope *f*, we have a mind boggling combinations in hand. So, before one goes around generating lots of such methods with random combination, let us put some system into madness. More importantly, we have not yet asked the sensitive questions such as,

a) Are all of these methods usable in all cases? (Define usability, convergence!)

b) Accuracy of these methods?

c) Advantages and disadvantages of various methods so as to enable one to choose appropriate method for a given problem (horses for the courses!).

Euler's methods, both implicit and explicit, evaluate the functional value *f* at only one grid point. These are known as *single step* methods. On the other hand, methods given by (6.9) involve evaluation of slope at *more than one* grid points. The possibilities are numerous. These will be called *multi-step* methods.

The classifications of *implicit vs. explicit* and *single step vs. multi-step* are mutually independent. They can be combined to yield four groups of methods: (i) *Single Step Explicit Method* (e.g., Euler explicit), (ii) *Single Step Implicit Method* (e.g., Euler implicit), (ii) *Multi-step Explicit Method* (will be derived in later section), and (iv) *Multi-step Implicit Method* (e.g., trapezoidal method of 6.9, more will be derived in later section).

The group of methods that incorporates approximation of the derivative on the left side of eq. (6.3) using values at more than 2 points is typically called *Backward Difference Formulae* (BDFs). We shall learn their special application at a later section.

In this chapter, we will first learn to formally derive various methods by grouping them appropriately in order to avoid future confusion. Afterwards, we will analyze some of the methods but more importantly, establish generalized protocol for analysis so that any given numerical method can be analyzed and judged on the basis of certain set standards. Lastly, we will apply the methods to solve a system of coupled IVPs, Boundary Value Problems (BVPs) and higher order ordinary differential equations (ODEs).

6.2 Derivation of Methods for IVPs

6.2.1 Single and Multi-Step Methods

In this section we will derive a group of implicit and explicit methods for the general IVP given by equation (6.3). We rewrite the problem as follows:

$$\frac{dy}{dt} = f(y,t) \text{ where } y = y_0 \text{ at } t = t_0 \tag{6.11}$$

We shall seek valid approximations of the following forms for explicit and implicit methods.

$$\text{Explicit: } y_{n+1} = y_n + h\sum_{i=0}^{k} \alpha_i f_{n-i} \tag{6.12}$$

$$\text{Implicit: } y_{n+1} = y_n + h\sum_{i=0}^{k} \beta_i f_{n+1-i} \tag{6.13}$$

where, $k = 0, 1, 2, \ldots, n$ and h is the uniform time step size, $h = t_k - t_{k-1} \quad \forall k$

The criteria for a valid approximation are as follows:

- Each approximation will represent the original equation with well-defined truncation error.
- The truncation error will approach zero in the limit as the grid length $h \to 0$.

This is also known as the *consistency* criteria. Any numerical method derived to satisfy these criteria will be consistent with the original problem, i.e., in the limit $h \to 0$, the approximate equation will approach the original equation. We will discuss consistency, order of method, stability and convergence in more detail in section 6.3.

We illustrate the derivation through an explicit method with $k = 3$. The equation (6.12) then becomes:

$$y_{n+1} = y_n + h\left(\alpha_0 f_n + \alpha_1 f_{n-1} + \alpha_2 f_{n-2}\right) \tag{6.14}$$

Denoting $\frac{dy}{dt} = y'$ and using the original equation $y' = f$, we can rewrite (6.14) as:

$$y_{n+1} = y_n + h\left(\alpha_0 y'_n + \alpha_1 y'_{n-1} + \alpha_2 y'_{n-2}\right) \tag{6.15}$$

We expand y_{n+1}, y'_{n-1} and y'_{n-2} in Taylor's series as follows:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2!}y''_n + \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y''''_n + o\left(h^5\right) \tag{6.16}$$

$$y'_{n-1} = y'_n - hy''_n + \frac{h^2}{2!}y'''_n - \frac{h^3}{3!}y''''_n + o\left(h^4\right) \tag{6.17}$$

$$y'_{n-2} = y'_n - (2h)y''_n + \frac{(2h)^2}{2!}y'''_n - \frac{(2h)^3}{3!}y''''_n + o\left(h^4\right) \tag{6.18}$$

Putting these expressions of (6.16-18) in equation (6.15), we obtain:

$$\begin{aligned} y_n + hy'_n + \frac{h^2}{2!}y''_n + \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y''''_n + o\left(h^5\right) &= y_n + h\alpha_0 y'_n + \\ h\alpha_1\left(y'_n - hy''_n + \frac{h^2}{2!}y'''_n - \frac{h^3}{3!}y''''_n + o\left(h^4\right)\right) &+ \\ h\alpha_2\left(y'_n - (2h)y''_n + \frac{(2h)^2}{2!}y'''_n - \frac{(2h)^3}{3!}y''''_n + o\left(h^4\right)\right) & \end{aligned} \tag{6.19}$$

We now perform some algebraic manipulations on (6.19) and group similar terms on the right hand side together to obtain the following:

$$y_n + hy_n' + \frac{h^2}{2!}y_n'' + \frac{h^3}{3!}y_n''' + \frac{h^4}{4!}y_n'''' + o(h^5) = y_n + h(\alpha_0 + \alpha_1 + \alpha_2)y_n' + h^2(-\alpha_1 - 2\alpha_2)y_n'' + h^3\left(\frac{\alpha_1}{2} + 2\alpha_2\right)y''' + h^4\left(-\frac{\alpha_1}{6} - \frac{4\alpha_2}{3}\right)y_n'''' + o(h^5) \quad (6.20)$$

Notice that the left hand side is an exact representation of y_{n+1}, i.e., the prediction of y at the next time step. Since we have 3 adjustable parameters in α_i's, we can equate coefficients of three terms on the r.h.s with those on the left hand side. So, the set of equations to determine α_i's are as follows:

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 &= 1 \\ \alpha_1 + 2\alpha_2 &= -\frac{1}{2} \\ \frac{\alpha_1}{2} + 2\alpha_2 &= \frac{1}{6} \end{aligned} \quad (6.21)$$

The solution of the above system of equation is $\alpha_0 = \frac{23}{12}, \alpha_1 = -\frac{4}{3}$ and $\alpha_2 = \frac{5}{12}$. Putting these values in (6.14), we obtain the required numerical method as,

$$y_{n+1} = y_n + h\left(\tfrac{23}{12}f_n - \tfrac{4}{3}f_{n-1} + \tfrac{5}{12}f_{n-2}\right) \quad (6.22)$$

The above analysis is general and can be used to derive an arbitrary explicit or implicit method. We can summarize the method to derive explicit or implicit method of any order in the following steps:

- Choose k and write the approximation in terms of α_i's or β_i's as in eq (6.14).
- Replace f with y' using the original equation.
- Expand all the terms using Taylor series except y_n
- Group the similar terms together on the right hand side.
- Equate the coefficients of right hand side terms with the left hand side.
- Compute α_i's or β_i's.

If the derivation appears too cumbersome involving too many algebraic manipulations, we now present a more elegant way of doing the same using a table. However, we caution the students to go through the steps of derivation described above to have a thorough understanding of the process. Using the tabular form without proper understanding may lead to error.

Right after eq. (6.14), we can construct the following table:

Table 6.1: Derivation of explicit multi-step method.

Terms	y_n	hy_n'	h^2y_n''	h^3y_n'''	h^4y_n''''
y_{n+1}	1	1	½	1/6	1/24

y_n	–1	0	0	0	0
f_n	0	$-\alpha_0$	0	0	0
f_{n-1}	0	$-\alpha_1$	α_1	$(-1/2)\alpha_1$	$(1/6)\alpha_1$
f_{n-2}		$-\alpha_2$	$2\alpha_2$	$-2\alpha_2$	$(4/3)\alpha_2$

Note that the entries along the rows are the coefficients of the respective Taylor series expansions. In effect, we have written down the coefficients of the expansion for the following form of equation (6.14)

$$y_{n+1} - y_n - h\left(\alpha_0 y'_n - \alpha_1 y'_{n-1} - \alpha_2 y'_{n-2}\right) = 0 \tag{6.23}$$

This essentially groups the terms together in the columns. Summing the columns and setting to zero give us the required equations. For example, sum of columns 3, 4 and 5 in Table 1 gives us the set of equations in (6.21). Let us highlight the utility of such derivation using the tabular form with an implicit method with $k = 2$. Using (6.13), the required method can be written as:

$$y_{n+1} = y_n + h\left(\beta_0 f_{n+1} + \beta_1 f_n + \beta_2 f_{n-1}\right) \tag{6.24}$$

Now, the following table can be constructed,

Table 6.2: Derivation of a multi-step implicit method.

Terms	y_n	hy'_n	$h^2y''_n$	$h^3y'''_n$	$h^4y''''_n$
y_{n+1}	1	1	1/2	1/6	1/24
y_n	–1	0	0	0	0
f_{n+1}	0	$-\beta_0$	$-\beta_0$	$(-1/2)\beta_0$	$(-1/6)\beta_0$
f_n	0	$-\beta_1$	0	0	0
f_{n-1}	0	$-\beta_2$	β_2	$(-1/2)\beta_2$	$(1/6)\beta_2$

So, the set of equations to solve for obtaining β_i's are:

$$\begin{aligned} \beta_0 + \beta_1 + \beta_2 &= 1 \\ \beta_0 - \beta_2 &= \frac{1}{2} \\ \beta_0 + \beta_2 &= \frac{1}{3} \end{aligned} \tag{6.25}$$

The solution is $\beta_0 = \frac{5}{12}, \beta_1 = \frac{2}{3}$ and $\beta_2 = -\frac{1}{12}$.

At this point, we are in a position to derive arbitrary order methods. The coefficients up to $k = 3$ are given in Table 6.3 for both implicit and explicit methods. The readers may wish to derive these to have some practice, especially for $k = 3$. For $k = 0$, the explicit method is the *Euler Forward* and the implicit method is the *Euler Backward*. All others are generally known as *multi-step* methods. Multi-step explicit methods are also known as *Adams-Bashforth* and the implicit ones are called *Adams-Moulton* methods. Individual methods are identified by their "order", which is equivalent to number of points at which the function is evaluated. For example, the method in

(6.24) is a *3rd order Adams-Moulton* and that in (6.22) is a *3rd order Adams-Bashforth*. Exact nature of this "order" will be clear during the discussion of truncation error in section 6.3.1.

Table 6.3: Single and multi-step explicit and implicit methods up to order 4.

Type	*k*	Method	Order	Name
EXPLICIT	0	$y_{n+1} = y_n + hf_n$	1	Euler Forward
	1	$y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right)$	2	Adams-Bashforth
	2	$y_{n+1} = y_n + h\left(\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right)$	3	
	3	$y_{n+1} = y_n + h\left(\frac{55}{24}f_n - \frac{59}{24}f_{n-1} + \frac{37}{24}f_{n-2} - \frac{3}{8}f_{n-3}\right)$	4	
IMPLICIT	0	$y_{n+1} = y_n + hf_{n+1}$	1	Euler Backward
	1	$y_{n+1} = y_n + h\left(\frac{1}{2}f_{n+1} + \frac{1}{2}f_n\right)$	2	Trapezoidal
	2	$y_{n+1} = y_n + h\left(\frac{5}{12}f_{n+1} + \frac{2}{3}f_n - \frac{1}{12}f_{n-1}\right)$	3	Adams Moulton
	3	$y_{n+1} = y_n + h\left(\frac{3}{8}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right)$	4	

Now that we are comfortable with the derivation of the methods, let us see application of these methods to an IVP. In order to see the effect of various order of the method, let us take the following example.

Example 6.1: Solve the following IVP using 1st to 4th order explicit and implicit methods using $h = 0.5$ and compare their accuracy by evaluating the true error.

$$\frac{dy}{dt} = -0.6y, \quad y = 1 \text{ at } t = 0, \quad y \in (0,5)$$

Solution: The true solution of the above equation is $y = e^{-0.6t}$. We will now apply the methods listed in Table 6.3 to the equation. However, as soon as we try to do that going from $t = 0$ to $t = 0.5$, we observe the following:

- Euler forward, Euler backward and Trapezoidal methods require y-values at $t = 0$ only to obtain the new value of y at $t = 0.5$. (no problem here!)
- All other methods require y-values at $t = -0.5$ and/or -1.0 and/or -1.5. Since, these values are undefined, we can not start the methods!

In general, multi-step methods are *non-self starting*. One obvious solution is to gradually go up using higher and higher order methods. For example, use Euler Forward for the first time step and 2nd to 4th order Adams-Bashforth methods in the subsequent time steps. While this takes care of the problem of starting the higher order methods, it increases the chance of error in the initial time steps by the use of lower order methods. This initial error may also propagate in the future time steps (recall error propagation in Chapter 1). We will address this problem of starting of multi-step methods at a separate section (6.4) later on. For the present problem, in order to make the errors of various methods comparable, we shall apply all of them independently and for equal number of time steps.

We shall evaluate the y-values at t = 0.5, 1.0 and 1.5 using the true solution. Then apply all the methods of Table 6.3 for rest of the time steps to reach t = 5.0. Since, the true solution is known, the true relative error (%) can be calculated at each time step.

Let us first write the time stepping formulae for the explicit methods using the methods listed in Table 6.3. We shall denote the k^{th} order Adams Basforth method as ABk.

Euler Forward (EF): $y_{n+1} = y_n + h(-0.6y_n)$
or $y_{n+1} = y_n(1-0.6h)$

AB2: $y_{n+1} = y_n + \frac{3}{2}h(-0.6y_n) - \frac{1}{2}h(-0.6y_{n-1})$
or $y_{n+1} = y_n(1-0.9h) + 0.3hy_{n-1}$

AB3: $y_{n+1} = y_n + \frac{23}{12}h(-0.6y_n) - \frac{4}{3}h(-0.6y_{n-1}) + \frac{5}{12}h(-0.6y_{n-2})$
$= y_n(1-1.15h) + 0.8hy_{n-1} - 0.25hy_{n-2}$

AB4: $y_{n+1} = y_n + \frac{55}{24}h(-0.6y_n) - \frac{59}{24}h(-0.6y_{n-1}) + \frac{37}{24}h(-0.6y_{n-2}) - \frac{3}{8}h(-0.6y_{n-1})$
$= y_n(1-1.375h) + 1.475hy_{n-1} - 0.925hy_{n-2} + 0.225hy_{n-3}$

The calculations are shown in the following table:

t	True	EF	ε(%)	AB2	ε(%)	AB3	ε(%)	AB4	ε(%)
0	1,0000								
0,5	0,7408								
1	0,5488								
1,5	0,4066								
2	0,3012	0,2846	5,51	0,3059	1,57	0,2997	0,49	0,3017	0,16
2,5	0,2231	0,1992	10,72	0,2292	2,74	0,2214	0,77	0,2236	0,23
3	0,1653	0,1395	15,64	0,1720	4,04	0,1632	1,29	0,1661	0,47
3,5	0,1225	0,0976	20,28	0,1290	5,32	0,1204	1,65	0,1230	0,48
4	0,0907	0,0683	24,68	0,0967	6,63	0,0888	2,14	0,0914	0,79
4,5	0,0672	0,0478	28,83	0,0725	7,95	0,0655	2,52	0,0677	0,69
5	0,0498	0,0335	32,75	0,0544	9,29	0,0483	2,99	0,0504	1,14

We will now write the time stepping formulae for the implicit methods when applied to the problem. We shall denote the k^{th} order Adams Moulton method as AMk.

Euler Backward (EB):

$y_{n+1} = y_n + h(-0.6y_{n+1})$

or $y_{n+1} = \dfrac{y_n}{(1+0.6h)}$

Trapezoidal (TR):

$$y_{n+1} = y_n + \tfrac{1}{2}h\left(-0.6y_n\right) + \tfrac{1}{2}h\left(-0.6y_{n+1}\right)$$

$$\text{or } y_{n+1} = \frac{y_n\left(1-0.3h\right)}{\left(1+0.3h\right)}$$

AM3:

$$y_{n+1} = y_n + \tfrac{5}{12}h\left(-0.6y_{n+1}\right) + \tfrac{2}{3}h\left(-0.6y_n\right) - \tfrac{1}{12}h\left(-0.6y_{n-1}\right)$$

$$= \frac{y_n\left(1-0.4h\right) + 0.05hy_{n-1}}{\left(1+0.25h\right)}$$

AM4:

$$y_{n+1} = y_n + \tfrac{3}{8}h\left(-0.6y_{n+1}\right) + \tfrac{19}{24}h\left(-0.6y_n\right) - \tfrac{5}{24}h\left(-0.6y_{n-1}\right) + \tfrac{1}{24}h\left(-0.6y_{n-2}\right)$$

$$= \frac{y_n\left(1-0.475h\right) + 0.125hy_{n-1} - 0.025hy_{n-2}}{\left(1+0.225h\right)}$$

We tabulate the calculations as before:

t	True	EB	ε(%)	TR	ε(%)	AM3	ε(%)	AM4	ε(%)
0	1,0000								
0,5	0,7408								
1	0,5488								
1,5	0,4066								
2	0,3012	0,3127	3,84	0,3005	0,23	0,3013	0,04	0,3012	0,01
2,5	0,2231	0,2406	7,82	0,2221	0,46	0,2233	0,08	0,2231	0,02
3	0,1653	0,1851	11,95	0,1642	0,68	0,1655	0,11	0,1653	0,02
3,5	0,1225	0,1424	16,25	0,1213	0,91	0,1226	0,15	0,1224	0,03
4	0,0907	0,1095	20,70	0,0897	1,13	0,0909	0,19	0,0907	0,04
4,5	0,0672	0,0842	25,33	0,0663	1,36	0,0674	0,23	0,0672	0,05
5	0,0498	0,0648	30,14	0,0490	1,58	0,0499	0,26	0,0498	0,06

In the above example, let us make a few observations by comparing the methods:

- For the same order methods with same time step size, the implicit method gives smaller error compared to the explicit methods. This is easily seen by comparing the % error columns of EF with EB, AB2 with TR, AB3 with AM3 and AB4 with AM4.
- In the same type of method, i.e., implicit or explicit, higher the order of the method, lower is the error. This is seen by comparing the error columns of the same table. This is due to decrease in the truncation error which we shall discuss in detail at a later section.

- The error always grows with time irrespective of the order and type of the method. Therefore, if one is to calculate many time steps for a problem, it is better to use a method that gives very small error at the start so that the error in the final solution is within reasonable limit. For example, the error using AM4 grew from 0.01-0.06% while that for Euler Backward grew from 3.84-30.14%. You can plot the error with time to find the rate of growth of the error and see how it varies across the methods.

We observe that the equations (6.12) and (6.13) are actually special cases of the following general expression:

$$\sum_{i=0}^{m} \alpha_i y_{n+1-i} = h \sum_{i=0}^{k} \beta_i f_{n+1-i} \tag{6.26}$$

In effect, all the single step and multi-step explicit and implicit methods derived in this section are special cases of (6.26). We get equation (6.12) by setting $\alpha_i = 0$ for $i > 1$ and $\beta_0 = 0$ in (6.26). For (6.12), we only set $\alpha_i = 0$ for $i > 1$. Therefore, all the methods described above are essentially derived from the same concept. The equation (6.26) is the generalized form of all *linear multi-step* methods. In the next section, we derive another group of methods, which also are special cases of (6.26).

6.2.2 Backward Difference Formulae (BDF)

These methods are especially useful for their application to a specific type of problems known as *stiff equations*. The reader have to wait until section (6.5) for a clear understanding of stiff problems and the application of BDF. We will derive the methods here for continuity of discussion in this section. The reader may find it easy to read this along with the previous sub-section because the concepts are same. These are a group of implicit methods. Generalized form of these methods are obtained from (6.26) by putting all $\beta_i = 0$ except β_0, which is set to unity. The objective is to determine the α_i's for maximum accuracy. So, the general form becomes:

$$\sum_{i=0}^{m} \alpha_i y_{n+1-i} = h f_{n+1} \tag{6.27}$$

The derivation follows the same steps as the multi-step method. We illustrate the derivation for $m = 2$ using the tabular form. The equation for $m = 2$ is given by,

$$\alpha_0 y_{n+1} + \alpha_1 y_n + \alpha_2 y_{n-1} = h f_{n+1} \tag{6.28}$$

We tabulate the Taylor Series expansions in Table 6.4.

Table 6.4: Derivation of a BDF.

Terms	y_n	hy'_n	$h^2 y''_n$	$h^3 y'''_n$

y_{n+1}	α_0	α_0	$\frac{1}{2}\alpha_0$	$1/6\alpha_0$
y_n	α_1	0	0	0
y_{n-1}	α_2	$-\alpha_2$	$\frac{1}{2}\alpha_2$	$-1/6\alpha_2$
f_{n+1}	0	−1	-1	−1/2

The equations can now be written as follows by summing the columns of Table 6.4:

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 &= 0 \\ \alpha_0 - \alpha_2 &= 1 \\ \alpha_0 + \alpha_2 &= 2 \end{aligned} \tag{6.29}$$

The solution of the above set of equation is $\alpha_0 = \frac{3}{2}, \alpha_1 = -2$ and $\alpha_2 = \frac{1}{2}$. The method is thus given by,

$$3y_{n+1} - 4y_n + y_{n-1} = 2hf_{n+1} \tag{6.30}$$

This is formally known as the *2nd order Backward Difference Formula*. You may now compare the coefficients of this equation with the 2nd order backward difference approximation of the 1st derivative in Chapter 5.

Using similar procedure, one can derive the implicit BDF of arbitrary order. In practical application however, up to 6th order BDF are useful because methods higher than 6th order are not *stiffly stable*. Full implication of this statement will be clear in section 6.3.2.8. The BDF of up to 6th order are shown in Table 6.5.

Table 6.5: Backward Difference Formulae (BDF) up to order 6.

m	**Method**	**Order**
1	$y_{n+1} - y_n = hf_{n+1}$	1
2	$\frac{3}{2}y_{n+1} - 2y_n + \frac{1}{2}y_{n-1} = hf_{n+1}$	2
3	$\frac{11}{6}y_{n+1} - 3y_n + \frac{3}{2}y_{n-1} - \frac{1}{3}y_{n-2} = hf_{n+1}$	3
4	$\frac{25}{12}y_{n+1} - 4y_n + 3y_{n-1} - \frac{4}{3}y_{n-2} + \frac{1}{4}y_{n-3} = hf_{n+1}$	4
5	$\frac{137}{60}y_{n+1} - 5y_n + 5y_{n-1} - \frac{10}{3}y_{n-2} + \frac{5}{4}y_{n-3} - \frac{1}{5}y_{n-4} = hf_{n+1}$	5
6	$\frac{49}{20}y_{n+1} - 6y_n + \frac{15}{2}y_{n-1} - \frac{20}{3}y_{n-2} + \frac{15}{4}y_{n-3} - \frac{6}{5}y_{n-4} + \frac{1}{6}y_{n-5} = hf_{n+1}$	6

We will now demonstrate the application of BDFs with an example.

Example 6.2: Solve the problem of Example 6.1 using BDFs of order 1 through 4 with a time step of h = 0.5. Evaluate the errors and compare with the multi-step methods.

Solution: In order to be able to compare the accuracies of the various methods, all methods need to be applied for the same number of time steps. Therefore, just like Example 6.1, we shall evaluate y at t = 0.5, 1 and 1.5 using the true solution. Thereafter, we shall apply the BDFs of up to order 4 for the rest of time steps up to t

=5.0. Notice, that the 1st order BDF is Euler Backward. Applying the methods to the IVP, we write the time stepping schemes below:

BDF1 or EB:

$$y_{n+1} = y_n + h(-0.6y_{n+1})$$

$$\text{or } y_{n+1} = \frac{y_n}{(1+0.6h)}$$

BDF2:

$$\tfrac{3}{2}y_{n+1} - 2y_n + \tfrac{1}{2}y_{n-1} = h(-0.6y_{n+1})$$

$$y_{n+1} = \frac{2y_n - \frac{1}{2}y_{n-1}}{(\frac{3}{2}+0.6h)}$$

BDF3:

$$\tfrac{11}{6}y_{n+1} - 3y_n + \tfrac{3}{2}y_{n-1} - \tfrac{1}{3}y_{n-2} = h(-0.6y_{n+1})$$

$$y_{n+1} = \frac{3y_n - \frac{3}{2}y_{n-1} + \frac{1}{3}y_{n-2}}{(\frac{11}{6}+0.6h)}$$

BDF4:

$$\tfrac{25}{12}y_{n+1} - 4y_n + 3y_{n-1} - \tfrac{4}{3}y_{n-2} + \tfrac{1}{4}y_{n-3} = h(-0.6y_{n+1})$$

$$y_{n+1} = \frac{4y_n - 3y_{n-1} + \frac{4}{3}y_{n-2} - \frac{1}{4}y_{n-3}}{(\frac{25}{12}+0.6h)}$$

We tabulate the calculations below:

t	True	BDF1	ε(%)	BDF2	ε(%)	BDF3	ε(%)	BDF4	ε(%)
0	1,0000								
0,5	0,7408								
1	0,5488								
1,5	0,4066								
2	0,3012	0,3127	3,84	0,2993	0,63	0,3016	0,14	0,3011	0,03
2,5	0,2231	0,2406	7,82	0,2196	1,58	0,2240	0,40	0,2229	0,11
3	0,1653	0,1851	11,95	0,1609	2,67	0,1665	0,72	0,1650	0,21
3,5	0,1225	0,1424	16,25	0,1178	3,84	0,1237	1,04	0,1221	0,30
4	0,0907	0,1095	20,70	0,0861	5,04	0,0919	1,35	0,0904	0,37
4,5	0,0672	0,0842	25,33	0,0630	6,25	0,0683	1,64	0,0669	0,44
5	0,0498	0,0648	30,14	0,0461	7,45	0,0507	1,92	0,0495	0,52

In general, we observed the following for the multi-step methods and BDFs:

- Higher order methods are more accurate. For the same order of method applied to the same problem, errors of BDFs are in between the explicit and implicit multi-step methods with same time step.

- Multi-step *Adams-Bashforth*, *Adams-Moulton* and *BDF*s were all *non-self starting*.
- Implicit methods may lead to non-linear algebraic equation at every time step.

There is a group of explicit methods known as *Runge Kutta methods* that try to overcome both the difficulties mentioned in the last two points by evaluating the slope function at intermediate points within a time step. These can also be derived up to arbitrary order. We show the derivation of these methods in the next section.

6.2.3 Runge Kutta Methods

Principle of Runge Kutta methods is to evaluate the slope function at various intermediate points within a time step and finally use a weighted average of all the slope function evaluations to compute the y-value at the next time step. In general terms, the Runge Kutta methods can be represented as follows:

$$y_{n+1} = y_n + h\sum_{i=0}^{p}\omega_i\phi_i \tag{6.31}$$

where, ω_i's are the weight functions and ϕ_i's are the function evaluations at the intermediate points. There are numerous ways to choose these intermediate points. Derivation involves choosing these intermediate points and the ω_i's with the goal of achieving the highest order of accuracy. In the most general form, the ϕ_i's are written as follows:

$$\begin{aligned}
\phi_0 &= f\left(y_n, t_n\right)\\
\phi_1 &= f\left(y_n + h\alpha_1^1\phi_0, t_n + \beta_1 h\right)\\
\phi_2 &= f\left(y_n + h\alpha_1^2\phi_0 + h\alpha_2^2\phi_1, t_n + \beta_2 h\right)\\
\phi_3 &= f\left(y_n + h\alpha_1^3\phi_0 + h\alpha_2^3\phi_1 + h\alpha_3^3\phi_2, t_n + \beta_2 h\right)\\
&\bullet\\
\phi_i &= f\left(y_n + h\sum_{j=1}^{i}\alpha_j^i\phi_{j-1}, t_n + \beta_i h\right)
\end{aligned} \tag{6.32}$$

At this point it will be worthwhile to compare eq 6.31 with the Euler forward method. You will observe that f_n in the Euler forward method have been replaced by $\sum_{i=1}^{p}\omega_i\phi_i$ where, each ϕ_i's are function evaluation at different points. Locations of these points are all within one time step (between t_n and $t_n + h$), which leads to all $\beta_i \le 1$. Instead of one point evaluation in the Euler forward method, Runge Kutta utilizes an weighted average of function evaluations at a number of intermediate points. Therefore, it follows that the sum of the weights (ω_i's) should be unity. Moreover, function evaluation at each ϕ_i depends on previously calculated ϕ values (ϕ_0-ϕ_{i-1}) and can be calculated explicitly. The result of these judiciously chosen intermediate

function evaluation is higher order accuracy like multi-step explicit methods but without the start-up problem.

We will demonstrate the principle of derivation of Runge-Kutta methods by taking the example of the 2nd order method. Using the general form (6.31 and 6.32), the 2nd order Runge Kutta method may be written as:

$$\begin{aligned} y_{n+1} &= y_n + \omega_0 h\phi_0 + \omega_1 h\phi_1 \\ \phi_0 &= f(y_n, t_n) \\ \phi_1 &= f(y_n + \alpha_1 h\phi_0, t_n + \beta_1 h) \end{aligned} \tag{6.33}$$

The objective is to estimate ω_0, ω_1, α_1 and β_1 in order to achieve maximum accuracy. Once again we will use the representation $f_n = f(y_n, t_n)$ and write ϕ_0 and ϕ_1 as follows:

$$\begin{aligned} \phi_0 &= f_n \\ \phi_1 &= f_n + \left(\alpha_1 h f_n \left.\frac{\partial f}{\partial y}\right|_n + \beta_1 h \left.\frac{\partial f}{\partial t}\right|_n \right) + \left(\alpha_1^2 h^2 f_n^2 \left.\frac{\partial^2 f}{\partial y^2}\right|_n \right. \\ &\quad \left. + 2\alpha_1\beta_1 h^2 f_n \left.\frac{\partial f}{\partial y}\right|_n \left.\frac{\partial f}{\partial t}\right|_n + \beta_1^2 h^2 \left.\frac{\partial^2 f}{\partial t^2}\right|_n + o(h^3) \right) \end{aligned} \tag{6.34}$$

Using the expansions of (6.34) in (6.33), we obtain the approximation of y_{n+1} according to 2nd order *Runge-Kutta* method as,

$$\begin{aligned} y_{n+1} = y_n + (\omega_0 + \omega_1) h f_n + \omega_1 h^2 \left(\alpha_1 f_n \left.\frac{\partial f}{\partial y}\right|_n + \beta_1 \left.\frac{\partial f}{\partial t}\right|_n \right) + \omega_1 h^3 \left(\alpha_1 f_n^2 \left.\frac{\partial^2 f}{\partial y^2}\right|_n \right. \\ \left. + 2\alpha_1\beta_1 f_n \left.\frac{\partial f}{\partial y}\right|_n \left.\frac{\partial f}{\partial t}\right|_n + \beta_1^2 \left.\frac{\partial^2 f}{\partial t^2}\right|_n \right) + o(h^4) \end{aligned} \tag{6.35}$$

True value of y_{n+1} is given by the expansion in (6.16). Notice that, we could not replace the function f any more in (6.35) by y' as was done in eq (6.16). This is because of different increments given to y_n and t_n to obtain the intermediate points for function f. However, we need to use some relations to be able to relate the derivatives of y_n with f_n. This is accomplished by the use of total derivative of function f as follows:

$$\frac{d^2 y}{dt^2} = \frac{df}{dt} = \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t} = \left(f\frac{\partial}{\partial y} + \frac{\partial}{\partial t} \right) f \tag{6.36}$$

Thus, using (6.36) in the Taylor series expansion shown in eq (6.16), we can write:

$$y_{n+1} = y_n + h f_n + \frac{h^2}{2!}\left(f_n \left.\frac{\partial f}{\partial y}\right|_n + \left.\frac{\partial f}{\partial t}\right|_n \right) + \frac{h^3}{3!}\frac{d^3 y_n}{dt^3} + o(h^4) \tag{6.37}$$

Equation (6.37) is the true expansion of y_{n+1} whereas (6.35) is the approximation given by 2nd order *Runge-Kutta* method. We can now determine the unknown constants ω_0, ω_1, α_1 and β_1 by equating the coefficients of the similar terms in the two equations. However, we realize immediately that we can get three equations for four unknowns by matching the terms up to $o(h^2)$. This naturally leads to multiple solutions. At the same time, if we want to simulate up to $o(h^3)$, we will land up with six equations with four unknowns which will have no solution. Thus, we settle for the former and have more than one form of 2nd order *Runge Kutta* method. The equations are:

$$\begin{aligned} &\omega_0 + \omega_1 = 1 \\ &\omega_1\alpha_1 = \tfrac{1}{2} \\ &\omega_1\beta_1 = \tfrac{1}{2} \end{aligned} \tag{6.38}$$

Thus, we can find the solution in terms of one independent parameter:

$$\omega_1 = \frac{1}{2\alpha_1},\ \ \beta_1 = \alpha_1 \ \text{ and } \ \omega_0 = 1 - \frac{1}{2\alpha_1} \tag{6.39}$$

By choosing various values of α_1, we can obtain different 2nd order *Runge-Kutta* methods. Some of the commonly used forms are shown in Table 6.6. Following along the similar line, one can derive higher order methods as well. Some examples of 3rd and 4th order *Runge-Kutta* methods are also shown in Table 6.6. We leave it for the readers to go through the derivation or at least convince themselves that these are indeed consistent approximations of the original IVP.

Table 6.6: Commonly used Runge Kutta (RK) Methods or up to order 4.

2nd Order *Runge Kutta* Method	$\alpha_1 = \frac{1}{2} = \beta_1,\ \ \omega_0 = 0,\ \ \omega_1 = 1$ $y_{n+1} = y_n + h\phi_1$ $\phi_0 = f\left(y_n, t_n\right)$ $\phi_1 = f\left(y_n + \frac{1}{2}h\phi_0, t_n + \frac{1}{2}h\right)$
2nd Order *Runge Kutta* Method , also known as *Heun's Predictor-Corrector* Method	$\alpha_1 = 1 = \beta_1,\ \ \omega_0 = \frac{1}{2},\ \ \omega_1 = \frac{1}{2}$ $y_{n+1} = y_n + h\left[\frac{1}{2}\phi_0 + \frac{1}{2}\phi_1\right]$ $\phi_0 = f\left(y_n, t_n\right)$ $\phi_1 = f\left(y_n + h\phi_0, t_n + h\right)$
2nd Order *Runge Kutta* Method, also known as *Ralston's Method*	$\alpha_1 = \frac{3}{4} = \beta_1,\ \ \omega_0 = \frac{1}{3},\ \ \omega_1 = \frac{2}{3}$ $y_{n+1} = y_n + h\left[\frac{1}{3}\phi_0 + \frac{2}{3}\phi_1\right]$ $\phi_0 = f\left(y_n, t_n\right)$ $\phi_1 = f\left(y_n + \frac{3}{4}h\phi_0, t_n + \frac{3}{4}h\right)$

3rd Order *Runge Kutta* Method	$y_{n+1} = y_n + h\left[\frac{1}{6}\phi_0 + \frac{2}{3}\phi_1 + \frac{1}{6}\phi_2\right]$ $\phi_0 = f\left(y_n, t_n\right)$ $\phi_1 = f\left(y_n + \frac{1}{2}h\phi_0, t_n + \frac{1}{2}h\right)$ $\phi_2 = f\left(y_n - h\phi_0 + 2h\phi_1, t_n + h\right)$
4th Order *Runge Kutta* Method	$y_{n+1} = y_n + h\left[\frac{1}{6}\phi_0 + \frac{1}{3}\left(\phi_1 + \phi_2\right) + \frac{1}{6}\phi_3\right]$ $\phi_0 = f\left(y_n, t_n\right)$ $\phi_1 = f\left(y_n + \frac{1}{2}h\phi_0, t_n + \frac{1}{2}h\right)$ $\phi_2 = f\left(y_n + \frac{1}{2}h\phi_1, t_n + \frac{1}{2}h\right)$ $\phi_3 = f\left(y_n + h\phi_2, t_n + h\right)$

We now show the application of these methods for the problem of example 6.1.

Example 6.3: Solve the problem of example 6.1 using the Runge Kutta methods of order 1-4 with a time step of $h = 0.5$. Evaluate the true relative errors at each time step and compare with the multi-step and BDF methods.

Solution: Although it is possible to apply all order Runge Kutta methods from the 1st time step, in order to compare the errors with examples 6.1 and 6.2, we shall apply these methods for the same number of time step. Therefore, similar to the previous examples, we will evaluate the true solution for the first 3 time steps and apply the Runge Kutta method thereafter.

2nd Order Runge Kutta Method: We apply only one of the three (2nd in the row) shown in Table 6.6. We leave the other two for the readers to verify. The method when applied to the problem gives the following time stepping scheme:

$$\phi_0 = -0.6y_n, \;\; \phi_1 = -0.6\left(y_n + h\phi_0\right), \;\; y_{n+1} = y_n + \tfrac{1}{2}h\left(\phi_0 + \phi_1\right)$$

We tabulate the calculations:

t	y (True)	ϕ_0	ϕ_1	y	ε (%)
0	1,0000				
0,5	0,7408				
1	0,5488				
1,5	0,4066				
2	0,3012	-0,2439	-0,1708	0,3029	0,56
2,5	0,2231	-0,1817	-0,1272	0,2257	1,13
3	0,1653	-0,1354	-0,0948	0,1681	1,70
3,5	0,1225	-0,1009	-0,0706	0,1252	2,28
4	0,0907	-0,0751	-0,0526	0,0933	2,85
4,5	0,0672	-0,0560	-0,0392	0,0695	3,44
5	0,0498	-0,0417	-0,0292	0,0518	4,02

Iteration scheme and calculation for the 3rd order Runge Kutta is shown below:

$$\phi_0 = -0.6y_n, \;\; \phi_1 = -0.6\left(y_n + \tfrac{1}{2}h\phi_0\right), \;\; \phi_2 = -0.6\left(y_n - h\phi_0 + 2h\phi_1\right)$$
$$y_{n+1} = y_n + h\left(\tfrac{1}{6}\phi_0 + \tfrac{2}{3}\phi_1 + \tfrac{1}{6}\phi_2\right)$$

t	y (True)	ϕ_0	ϕ_1	ϕ_2	y	ε (%)
0	1,0000					
0,5	0,7408					
1	0,5488					
1,5	0,4066					
2	0,3012	-0,2439	-0,21	-0,1927	0,3011	0,04
2,5	0,2231	-0,1806	-0,15	-0,1427	0,2229	0,09
3	0,1653	-0,1338	-0,11	-0,1057	0,1651	0,13
3,5	0,1225	-0,0991	-0,08	-0,0783	0,1222	0,17
4	0,0907	-0,0733	-0,06	-0,0579	0,0905	0,21
4,5	0,0672	-0,0543	-0,05	-0,0429	0,0670	0,26
5	0,0498	-0,0402	-0,03	-0,0318	0,0496	0,30

4th Order Runge Kutta Method:

$$\phi_0 = -0.6y_n,\quad \phi_1 = -0.6\left(y_n + \tfrac{1}{2}h\phi_0\right),\quad \phi_2 = -0.6\left(y_n + \tfrac{1}{2}h\phi_1\right),\quad \phi_3 = -0.6\left(y_n + h\phi_2\right)$$

$$y_{n+1} = y_n + h\left(\tfrac{1}{6}\phi_0 + \tfrac{1}{3}\phi_1 + \tfrac{1}{3}\phi_2 + \tfrac{1}{6}\phi_3\right)$$

t	y (True)	ϕ_0	ϕ_1	ϕ_2	ϕ_3	y	ε (%)
0	1,0000						
0,5	0,7408						
1	0,5488						
1,5	0,4066						
2	0,3012	-0,2439	-0,2074	-0,2128	-0,1801	0,3012	0,00
2,5	0,2231	-0,1807	-0,1536	-0,1577	-0,1334	0,2231	0,01
3	0,1653	-0,1339	-0,1138	-0,1168	-0,0988	0,1653	0,01
3,5	0,1225	-0,0992	-0,0843	-0,0865	-0,0732	0,1225	0,01
4	0,0907	-0,0735	-0,0625	-0,0641	-0,0542	0,0907	0,01
4,5	0,0672	-0,0544	-0,0463	-0,0475	-0,0402	0,0672	0,02
5	0,0498	-0,0403	-0,0343	-0,0352	-0,0298	0,0498	0,02

Errors in the Runge Kutta methods are clearly comparable to those of the implicit multi-step methods. They do not have the start-up problem either. If you started to wonder why one would need the other methods, wait until we analyze all the methods in various ways in the next section. We will establish different kinds of errors associated with the methods. This will not only lead to better understanding of the methods but will also help us to determine the suitability of a method for a given problem.

A natural question that may arise is whether Runge Kutta methods of order higher than 4th are possible. The answer is yes but there are a few drawbacks. To understand this, let us take a look at the Table 6.6. We have evaluated the slope function ϕ at 2 points for the 2nd order method, at 3 points for the 3rd order method and at 4 points for the 4th order method. That is say, for an n^{th} order R-K method with n = 2 to 4, we need to evaluate the slope function at n points or stages. This cannot be done with the R-K methods of higher order and therefore, the methods become uneconomical. It has been shown (Butcher, 1965) that to attain orders of n = 5 and 6, one needs to evaluate the slope function on at least 6 and 7 points, respectively. With 8 and 9 point evaluation of slope function, maximum order possible are 6 and 7, respectively. For even higher number of points $(m \geq 10)$, the maximum attainable

order of accuracy is $\leq (m-2)$. This is the reason for the popularity of the R-K methods of up to order 4. The higher orders are almost never used.

Exercise 6.2

1. Solve the differential equation $dy/dx = x^2y - 2y$ with $y(0)=1$ over the interval $x=0$ to 1.0 with $h = 0.1$ using the following methods:

(a) Trapezoidal Method

(b) Heun's method without iteration

(c) Heun's method with iteration

(d) 4^{th} order Runge-Kutta method

(e) Solve analytically and compare the true relative errors of the above methods at every time step.

2. Solve the differential equation $dy/dx = 10 \sin(\pi x)$ with the initial condition $y(0)=0$ and step length of 0.2 using

(a) the 4^{th} order R-K method

(b) 4^{th} order Adams Moulton. Use values from (a) for startup.

3. Solve the differential equation $dy/dt = -100\ y + 99\ e^{-t}$ with the initial condition $y(0)=2$ using, (a) Euler's forward (explicit) method, and (b) Euler backward (implicit) method, to obtain the value of y at t=1.0. Use time steps of 0.01, 0.02 and 0.025. Find the analytical solution and graphically compare the errors for these time steps.

4. Consider the form of IVP in eq 6.10. Now, use Simpson's $1/3^{rd}$ rule and $3/8^{th}$ rule (Table 5.4) to obtain two different methods. Can you categorize these methods ? (Explicit vs. implicit, single step vs. multi-step, etc.)

5. Table 6.6 shows one form of 3^{rd} order R-K method. Are there other forms ? Can you derive a 3^{rd} order general R-K method with variable coefficients (like one floating variable we found in the 2^{nd} order R-K method).

6.3 Error Analysis of Methods for IVPs

6.3.1 Truncation Error

Concept of truncation error was first introduced in Chapter 1 as the error introduced by approximating an infinite series by a finite one. We will now formalize the concept in the context of numerical methods for ordinary differential equation. We will define various types of truncation error and formalize a process by which one can calculate the truncation error or establish the order of the error for any given method.

We will analyze truncation error in two ways. In all the initial value problems, value of y at the initial point (y_0) is known exactly. Numerical methods calculate the value of y_{n+1} given the value of y_n. That is, y_1 is calculated from y_0 after a time step of h, y_2 is calculated from y_1, and so on. We will first calculate the error incurred in one time step and this will be called the *local truncation error* (*LTE*). We will then calculate the error involved in the approximation of the original differential equation (6.11) when we approximate it using a truncated series similar to (6.14). This will be called the *global truncation error* (*GTE*). We will now highlight these calculations through examples.

Notice, that at any time step, true value of y_{n+1} is given by the Taylor's series expansion shown in eq (6.16). We rewrite it here:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2!}y''_n + \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y''''_n + o\left(h^5\right) \tag{6.40}$$

Equations of the form (6.26) only calculate an approximation value of y_{n+1}. Let us denote the approximate value as $\tilde{y}_{n+1}$. If we consider the 3rd order Adams Bashforth method, this approximate value is given by the following truncated series:

$$\tilde{y}_{n+1} = y_n + h\left(\tfrac{23}{12}f_n - \tfrac{4}{3}f_{n-1} + \tfrac{5}{12}f_{n-2}\right) \tag{6.41}$$

We have already seen (6.19) that the right hand side can be represented as an infinite series using the exact relation of the original ODE $y'_i = f_i$ and expanding y'_{n-1} and y'_{n-2} in Taylor's series. Putting the values $\alpha_0 = \frac{23}{12}, \alpha_1 = -\frac{4}{3}$ and $\alpha_2 = \frac{5}{12}$ on the right hand side of eq (6.19), the eq (6.41) can be written as:

$$\tilde{y}_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{6}y''' - \frac{h^4}{3}y''''_n + o\left(h^5\right) \tag{6.42}$$

Recall from Chapter 1, the definition of error as the difference between the true value and the approximate value. Therefore, an expression of the *local truncation error* (*LTE*) can be obtained by deducting eq (6.42) from (6.40) as follows:

$$LTE = y_{n+1} - \tilde{y}_{n+1} = \frac{3}{8}h^4 y''''_n + o\left(h^5\right) \tag{6.43}$$

Notice that all the terms of order h^4 and higher survives as their coefficients do not match between the true and approximate series. One is often interested in the first

non-zero term or the leading order term of the truncation error. If the expression contains h^p as the leading order term we would term the method to have a local truncation error of order p. In this case, we say, "this Adams-Bashforth method has a 4th order local truncation error or the method has 4th order accuracy locally." It is important to note that we could also obtain this leading order local truncation error term by summing the terms on the last column of Table 6.1. This shows that, if we derive the method using the tabular form, we can also obtain the local truncation error in the process of derivation.

Let us now try to understand what the approximation means in the context of the whole IVP. Let us redefine the original mathematical problem (6.11) in the following functional form:

$$\begin{aligned} &F(y,t)=0 \\ &\text{where, } F(y,t)=\frac{dy}{dt}-f(y,t) \end{aligned} \tag{6.44}$$

Let us also define its approximation given by a numerical method as,

$$\tilde{F}(y,t)=0 \tag{6.45}$$

For example, in the case of Euler forward, the approximation will be,

$$\tilde{F}(y,t)=\frac{y_{n+1}-y_n}{h}-f_n \tag{6.46}$$

The *global truncation error* (*GTE*) is defined as:

$$GTE=F(y,t)-\tilde{F}(y,t) \tag{6.47}$$

Let us illustrate this using the 3rd order Adams-Bashforth formula (6.41). The approximation can be written in the functional form as:

$$\frac{y_{n+1}-y_n}{h}=\left(\tfrac{23}{12}f_n-\tfrac{4}{3}f_{n-1}+\tfrac{5}{12}f_{n-2}\right) \tag{6.48}$$

Using the exact relation of the original ODE $y_i'=f_i$ and expanding y_{n-1}' and y_{n-2}' in Taylor's series we obtain,

$$\begin{aligned} &\left[y_n+hy_n'+\frac{h^2}{2!}y_n''+\frac{h^3}{3!}y_n'''+\frac{h^4}{4!}y_n''''+o\left(h^5\right)\right]-y_n \\ &\quad -h\left[\frac{23}{12}f_n-\frac{4}{3}\left(y_n'-hy_n''+\frac{h^2}{2!}y_n'''-\frac{h^3}{3!}y_n''''+o\left(h^4\right)\right)\right. \\ &\quad \left.+\frac{5}{12}\left(y_n'-(2h)y_n''+\frac{(2h)^2}{2!}y_n'''-\frac{(2h)^3}{3!}y_n''''+o\left(h^4\right)\right)\right]=0 \end{aligned} \tag{6.49}$$

Doing the usual algebraic manipulations by grouping the similar terms, we obtain the following:

$$\left.\frac{dy_n}{dt}\right|_n - f_n + \frac{9}{46}h^3 y_n''' + o\left(h^4\right) = 0 \tag{6.50}$$

Therefore, we can write the approximate functional form as:

$$\tilde{F}(y,t) = \left.\frac{dy_n}{dt}\right|_n - f_n + \frac{9}{46}h^3 y_n''' + o\left(h^4\right) \tag{6.51}$$

Deducting the above eq (6.51) from eq (6.44) we obtain the global truncation error as:

$$GTE = F(y,t) - \tilde{F}(y,t) = -\frac{9}{46}h^3 y_n''' - o\left(h^4\right) \tag{6.52}$$

Therefore, the leading order term of the global truncation error is 3rd order. Recall that we called equation (6.48) as 3rd order Adams-Bashforth method in Table 6.3. All the methods are named by the order of the leading order term in the global truncation error.

It is important to note that the local truncation error is one order higher than the global truncation error. This is generally the case for a consistent scheme for IVP. This can also be understood from the fact that in order to obtain global truncation error, the derivative is simulated by $\left(\frac{y_{n+1} - y_n}{h}\right)$ and as a result, the truncation error is also divided by h. By the same logic, local truncation error for the 2nd order Runge-Kutta method came out to be $o(h^3)$. We leave it to the readers to show that the global truncation error is indeed $o(h^2)$ and thus the name "2nd order".

Since the truncation error is proportional to h^p, one can anticipate the error to decrease with h. This way one can get arbitrarily close to the true solution by decreasing h. Once we fix an error target of ε, we can always choose an h that satisfies the criterion. Therefore, h becomes a function of ε and we write it as $h(\varepsilon)$. At this point, we are ready to formally define consistency.

Definition 6.1: A numerical method for an IVP is *consistent* if the global truncation error (*GTE*) is such that for any $\varepsilon > 0$ there exists a $h(\varepsilon) > 0$ for which $|GTE| < \varepsilon$.

This also means that the global truncation error approaches zero and the approximate equation approaches the original equation in the limit $h \to 0$. In other words, $\lim_{h \to 0} GTE = 0$ and $\lim_{h \to 0} \tilde{F}(y,t) = F(y,t)$. If the leading order term of a numerical method is $o\left(h^p\right)$, it is consistent if $p \geq 1$.

Let us now see the implication of truncation error on the accuracy of the method. We observed that the leading order term in the truncation error for all consistent numerical method derived so far could be represented as:

$$GTE = Kh^p \tag{6.53}$$

where, K is a constant for any time step. Since the time step h is small, higher order terms will decrease more rapidly. Thus the leading order term is likely to dominate

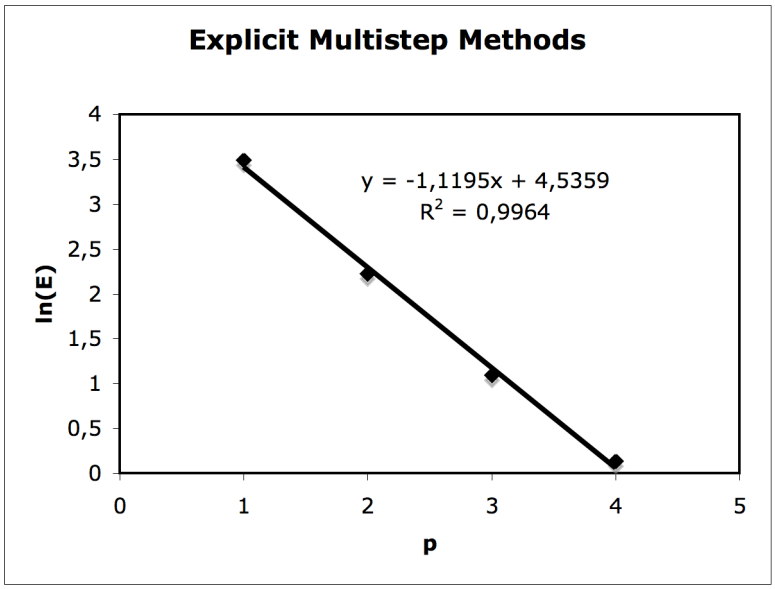
the absolute magnitude of the truncation error. We can define the order of accuracy of a numerical method as follows:

Definition 6.2: A consistent numerical method is said to have an order of accuracy of p, if p is the largest positive integer such that, $|GTE| \le Kh^p$ for $0 < h \le H$ where, K and H are constants.

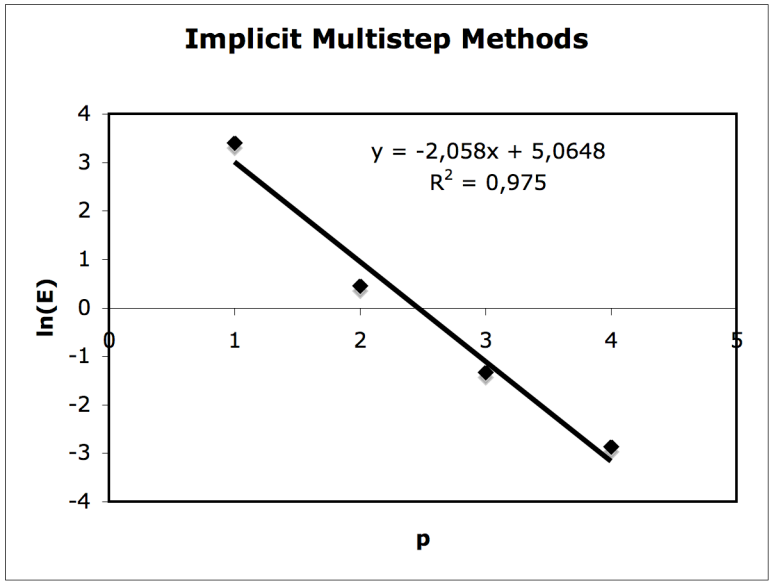
From the definition, one can write:

$$\ln(GTE) = \ln(K) + p\ln(h) \tag{6.54}$$

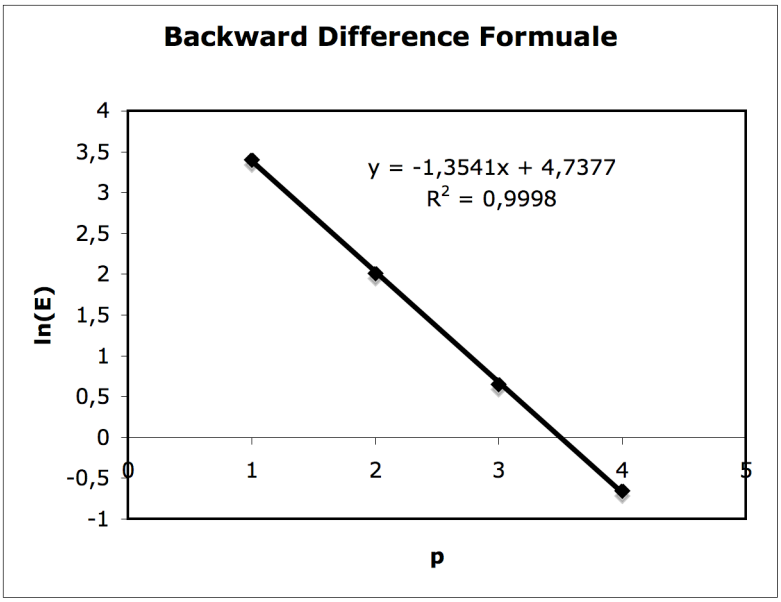
Thus, for a constant h, we would expect a linear relationship between ln(GTE) and p with ln(h) as the slope. This means that the truncation error will decrease with increasing order of the methods at a rate of ln(h) or higher order method has less error. In practice however, the slope will be steeper than ln(h) due to the effect of higher order terms. Graphically this can be easily shown for the methods we have discussed so far using the values calculated in examples 6.1-3. In computation, total error is a composite of the effect of truncation error and machine round off error. So, we can only calculate the total error by comparing the numerical solution with the true solution. Let us plot the natural logarithme of errors at $t = 5$ for explicit, implicit, BDFs and Runge Kutta methods with p (Fig 6.4 a-d).



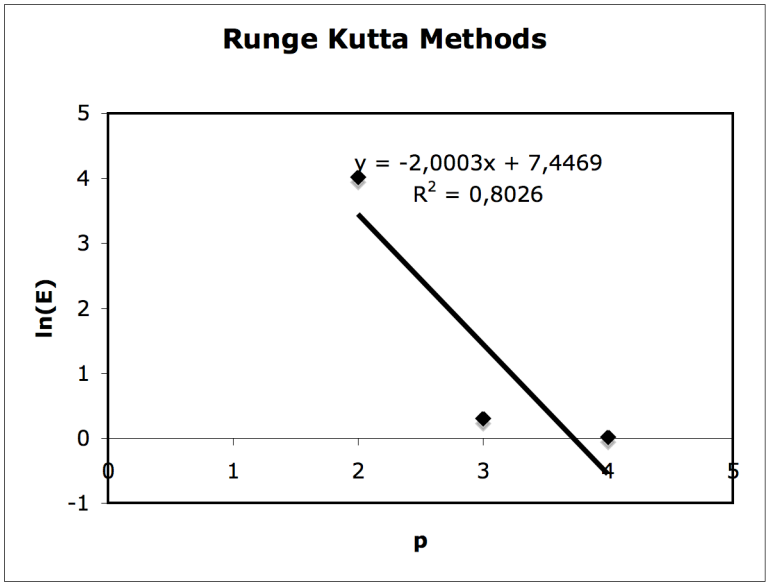
(a)


Implicit Multistep Methods
y = -2,058x + 5,0648
R2 = 0,975
ln(E)
p
4
3
2
1
0
-1
-2
-3
-4
0
1
2
3
4
5

(b)


Backward Difference Formuale
y = -1,3541x + 4,7377
R2 = 0,9998
ln(E)
p
4
3,5
3
2,5
2
1,5
1
0,5
0
-0,5
-1
0
1
2
3
4
5

(c)



(d)

Figure 6.4: Plot of ln(E) with the order p at t = 5.0 for the examples 6.1-6.3 with h = 0.5. The slope and intercept of the fitted line is shown. Notice that all the slopes are steeper than ln(h) = -0.69. The ln(E) starts to become asymptotic as it comes closer to zero. This is seen for the Runge Kutta methods.

Similarly, for the same method (constant p), we will expect a linear relationship between ln(GTE) and ln(h) with the slope being p. This means that refining the grid will also decrease the truncation error at the rate of p. This provides a numerical way to estimate the order of a method. We demonstrate this by taking the example 6.1 with Trapezoidal method.

Example 6.4: Numerically determine the order of the Trapezoidal method by taking the example problem 6.1.

Solution: We solve the problem using Trapezoidal method with time steps of 0.1, 0.25, 0.5, 0.75, 1.0, 1.5 and 2.0. Since, the chosen h values have the least common factor as 6.0, we plot the natural logarithme of error at t = 6.0 with the natural logarithme of the time step h. The values are shown in the table below:

True Solution at t = 6.0 is $y = 1.0e^{-0.6\times 6.0} = 0.027323722$.

h	y at t = 6.0	ε (%)	Ln (h)	Ln (ε)
0,1	0,027294213	0,107999995	-2,302585093	-2,225624094
0,25	0,027139288	0,674998897	-1,386294361	-0,393044222
0,5	0,026586001	2,699928551	-0,693147181	0,993225310
0,75	0,025664033	6,074169180	-0,287682072	1,804045219
1	0,024374074	10,795192350	0,000000000	2,379100882
1,5	0,020700401	24,240188767	0,405465108	3,188011949
2	0,015625	42,81525868	0,693147181	3,756894550

We now fit a straight line through the points in the plot of ln(h) vs. ln(ε) (Figure 6.5). The slope of the fitted line is 1.998 and we know that the Trapezoidal method is 2nd

order accurate globally. This is a convenient way of determining the order of the global truncation error of a method.

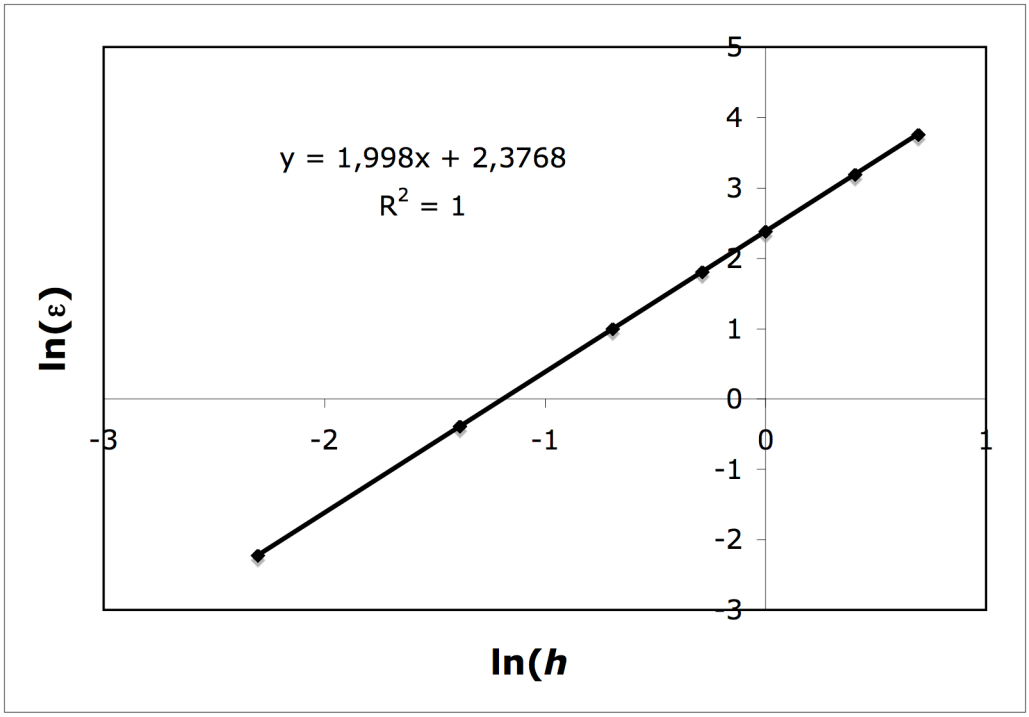
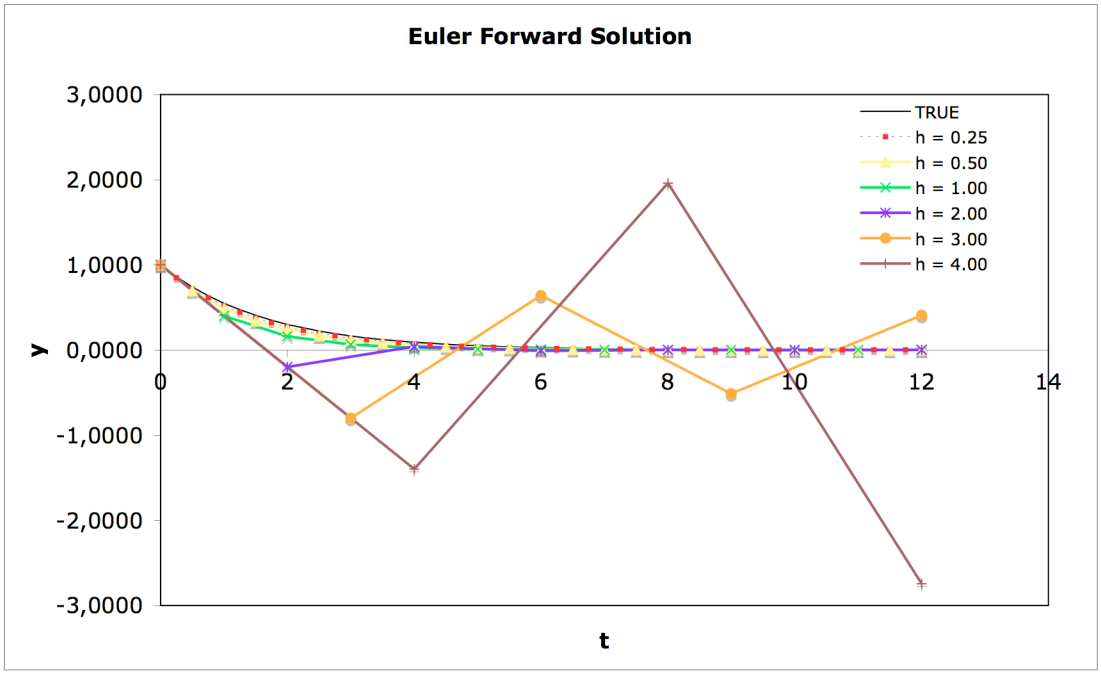
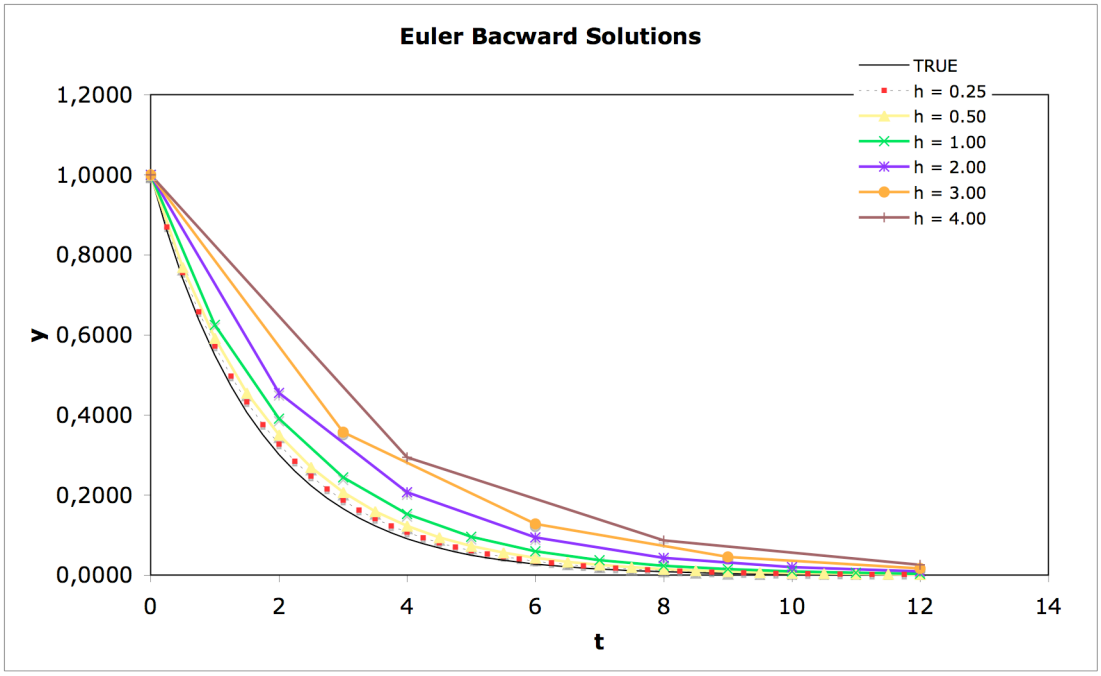


Figure 6.5: Plot of ln(ε) vs. ln(h) for the Trapezoidal method. Theoretical slope of the fitted line is 2.0. Here it is seen as 1.998.

In this section, we were concerned with the accuracy of a numerical method with respect to the true solution. Sometimes, a *consistent* numerical method may produce oscillating or diverging result (e.g. goes to infinity) even though the true solution may be finite and monotonous. In order to visualize this, let us consider the problem of example 6.1. We solved it using Euler forward method with the time step sizes of h = 0.25, 0.5, 1.0, 2.0, 3.0 and 4.0. The solutions are compared with true solution in Figure 6.6a. The solution is well behaved up to h = 1.0. At h = 2.0, the solution oscillated for the first few time steps but gradually died down as the time progressed. The magnitude of oscillations are much larger at h = 3.0 although it is decreasing as the time progressed. At h = 4.0, the magnitude of oscillations are increasing with time. Compare this observation with the solution using Euler backward method with the same time step sizes (Figure 6.6b). No oscillation was observed for any of the time steps. Only the error increases with the time step size which is expected from the truncation error analysis. At this point, recall that both Euler forward and Euler backward are first order method. But there is nothing in the truncation error analysis that indicated the oscillation observed in the Euler forward method. This behaviour is due to the stability of a numerical method and often a more important aspect than the truncation error. We introduce it in the next section.



(a)



(b)

Figure 6.6: Solution of example problem 6.1 with different time steps using (a) Euler Forward and (b) Euler Backward methods. The time step sizes above 2.0 shows oscillations in the solutions using Euler forward method. Magnitude of these oscillations decreases with time up to $h = 3.0$ but increases for $h = 4.0$.

6.3.2 Stability Analysis

In this section, we will attempt to understand why some numerical methods produce oscillations while some others don't for the same IVP using same time steps. Notice that in Figure 6.6a, the oscillations keep on increasing with time at h = 4.0. Eventually, it may (will) produce unbounded solution in a few time steps although the true solution is well defined. This leads to the concept of stability.

The true solution or the analytical solution is stable when it does not grow unbounded. For such a problem, if a numerical method produces solution that is bounded for all possible choices of time steps, the method is *unconditionally stable*. If the numerical method produces unbounded solution for all possible choices of time steps, the method is *unconditionally unstable*. However, majority of the numerical methods derived in this section are *conditionally stable* which produces bounded solution only for certain time step values. At this juncture, we will work with these working definitions of stability. In the last subsection, we will formalize the definition of stability and absolute stability.

A given numerical method can be stable for one IVP but unstable for another with the same time step. Also, the conditions of stability may be different for two different IVPs using the same numerical method. Therefore, we need a model problem, which we can use to compare various numerical methods. Let us introduce the origin of the model problem for the general IVP (equation 6.11). Since the solution starts from the fixed initial point (y_0, t_0), let us look at the behaviour of the equation in the neighbourhood of (y_0, t_0) by expanding the function $f(y, t)$ using Taylor's series,

$$\begin{aligned}\frac{dy}{dt} = f(y_0,t_0) + (y-y_0)\left.\frac{\partial f}{\partial y}\right|_{(y_0,t_0)} + (t-t_0)\left.\frac{\partial f}{\partial t}\right|_{(y_0,t_0)} + \frac{(y-y_0)^2}{2}\left.\frac{\partial^2 f}{\partial y^2}\right|_{(y_0,t_0)} \\ + (y-y_0)(t-t_0)\left.\frac{\partial^2 f}{\partial y\,\partial t}\right|_{(y_0,t_0)} + \frac{(t-t_0)^2}{2}\left.\frac{\partial^2 f}{\partial t^2}\right|_{(y_0,t_0)} + \cdots\end{aligned} \tag{6.55}$$

If we only collect the linear terms and ignore the quadratic and higher order terms, the above equation may be written as follows:

$$\frac{dy}{dt} = \alpha + \beta t + \lambda y + \cdots \tag{6.56}$$

where, α, β and λ are constants. Out of the three terms on the right, the last term leads to an exponential solution. The above equation can be generalized as,

$$\frac{dy}{dt} = \lambda y + \gamma(t)$$

where, $\gamma(t)$ is any function of t. For any approximate solution ($\tilde{y}$) of the above equation computed using a numerical method, we can write,

$$\frac{d\tilde{y}}{dt} = \lambda \tilde{y} + \gamma(t)$$

Since the error (ε) is defined as $\varepsilon = y - \tilde{y}$, by deducting the approximate equation from the true equation, we obtain:

$$\frac{d\varepsilon}{dt} = \lambda\varepsilon$$

Therefore, the error follows the equation without the function $\gamma(t)$. The function $\gamma(t)$ has no effect on the error because it does not contain the dependent variable and can be evaluated exactly at every time step. Since, the major effect on the growth and decay of the error is provided by the λy term, the model problem typically used for the analysis of numerical methods is,

$$\frac{dy}{dt} = \lambda y \tag{6.57}$$

In order to obtain the model equation, the full expression of (6.55) was linearized. Stability analysis performed using this model equation belongs to the group of *linear stability analysis*. In this book, we will restrict ourselves to *linear stability analysis* using model eq (6.57). We will introduce non-linear stability analysis in chapter 8.

To generalize the applicability of stability analysis, we will allow λ to take imaginary values, that is,

$$\lambda = \lambda_r + i\lambda_i \tag{6.58}$$

The analytical solution with the initial condition $y = y_0$ at $t = 0$ is

$$y = y_0 e^{\lambda t} \tag{6.59}$$

It is a common practice to plot the stability regions in the $\lambda_r h$ - $\lambda_i h$ plane although the time step h has no meaning in the context of analytical solution. One looks at the analytical solution (6.59) on a discrete grid as,

$$y_n = y_0 e^{\lambda h\left(\frac{t}{h}\right)} = y_0 e^{\lambda h n} \tag{6.60}$$

Since, we will deal with a constant time step of h, it does not alter anything in terms of the analytical solution but helps in future comparison of stability regions of the numerical methods.

It is easy to see that y will grow unbounded for all positive values of real part of λ ($\lambda_r > 0$) irrespective of whether λ_i is positive or negative. Therefore, the analytical solution of the model problem is bounded for $\lambda_r \leq 0$ and often most significant for the majority of the engineering problems. This region of bounded analytical solution is shown as the shaded region (grey) in Figure 6.7.

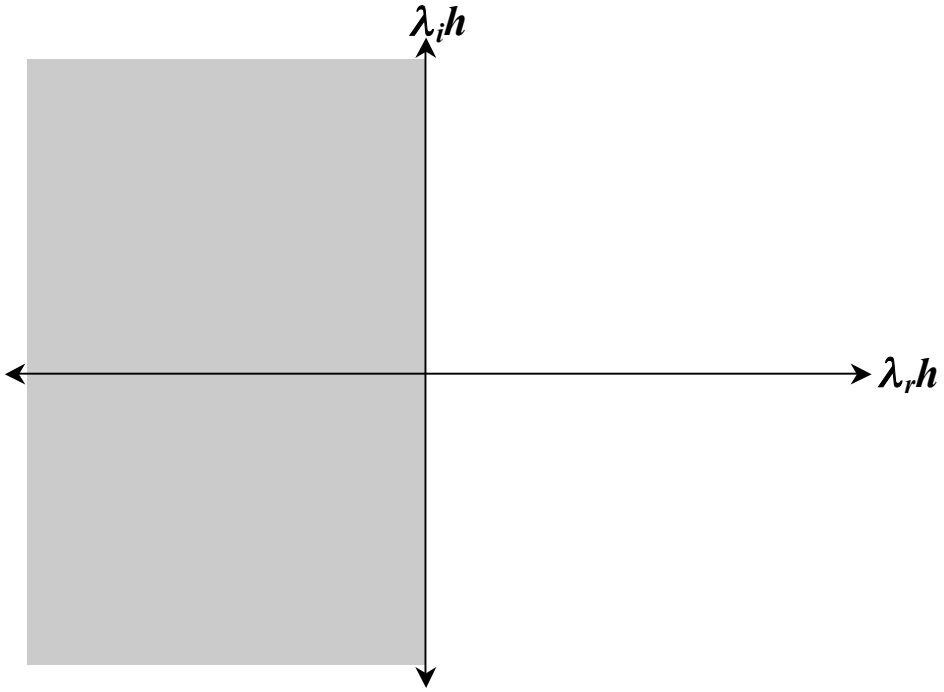


Figure 6.7: Region of bounded analytical solution for the model problem.

For λ_r>0, the analytical solution grows unbounded. The terminology "*unconditionally stable*" is often used for the methods that are stable for all $\lambda_r \le 0$, i.e., the stability region includes the complete shaded region of Figure 6.7 where the analytical solution is bounded. We will compute the stability regions of the numerical methods and compare them with the model problem. If the region includes the complete shaded region of Figure 6.7 for a method, we will call that method as unconditionally stable. For other methods with conditional stability, we will compute the condition or the ranges of λ_r and λ_i for which the methods are stable. Although, the analytical solution of the model problem grows unbounded for $\lambda_r > 0$, some numerical methods can still produce stable solution without oscillations and match the true solution reasonably well. Therefore, while formulating the stability regions of the numerical methods, we will not restrict ourselves to $\lambda_r \le 0$.

We will now present the detailed stability analysis of three methods, Euler Forward, Euler Backward and Trapezoidal. This will be followed by a presentation of a general framework that can be used to obtain stability regions of all the methods. We will give one example formulation from each group of methods. At the end, we will formalize the definition of stability and characterize types of stability.

6.3.2.1 Euler Forward

The Euler Forward method is given in the Table 6.3 as follows:

$$y_{n+1} = y_n + hf_n \tag{6.61}$$

Applying this method to the model problem of eq (6.57), we obtain,

$$y_{n+1} = y_n + \lambda h y_n = \left(1 + \lambda_r h + i\lambda_i h\right) y_n = \sigma y_n \tag{6.62}$$

where, $\sigma = \frac{y_{n+1}}{y_n} = 1 + \lambda h = 1 + \lambda_r h + i\lambda_i h$ is the amplification in the value of y in one time step, going from y_n to y_{n+1}. This σ is known as the *amplification factor*. By recursive application of eq (6.62), one can easily relate the value of y at any time step n with the initial condition y_0 as follows:

$$y_n = \sigma^n y_0 \tag{6.63}$$

It is clear from the above equation that for any finite initial condition y_0, the solution will be bounded if and only if the absolute magnitude of the *amplification factor* is less than unity. That is to say,

$$|\sigma| = \left|(1 + \lambda_r h) + i\lambda_i h\right| < 1 \tag{6.64}$$

The above can be easily written in the following form:

$$(1 + \lambda_r h)^2 + (\lambda_i h)^2 < 1 \tag{6.65}$$

Another way to look at eq (6.64) is,

$$\sigma = \Lambda e^{i\phi} \text{ where, } \Lambda = \sqrt{(1 + \lambda_r h)^2 + (\lambda_i h)^2} \text{ and } \phi = \tan^{-1}\left(\frac{\lambda_i h}{1 + \lambda_r h}\right) \tag{6.66}$$

Since $\left|e^{i\phi}\right| = 1$, the condition $|\sigma| < 1$ essentially means $\Lambda < 1$ which leads to eq (6.65). This equation gives the condition imposed on the time step h for the stability of the Euler Forward method and provides the stability region. When plotted on the $\lambda_r h$ - $\lambda_i h$ plane, this is the interior of a unit circle centered at (-1, 0). The stability region is shown in Figure 6.8.

When λ is real and negative (necessary for bounded analytical solution), *i.e.*, $\lambda_i = 0$ and $\lambda_r = -\lambda$, we obtain the range of time step as $0 < h < \frac{2}{\lambda}$ for which the method will be stable. At this point, one can relate to the example 6.4. Since, the $\lambda = 0.6$, the maximum limit of h is 3.33. Instability was observed for $h = 4.0$ but not for 3.0 (Figure 6.6a, b). For purely imaginary λ, *i.e.*, $\lambda_r = 0$, there is no value of the time step h other than zero that will satisfy eq (6.65). This can be seen visually from Figure 6.8 as the stability region is tangent to the imaginary axis. Therefore, the method is unconditionally unstable for purely imaginary λ.

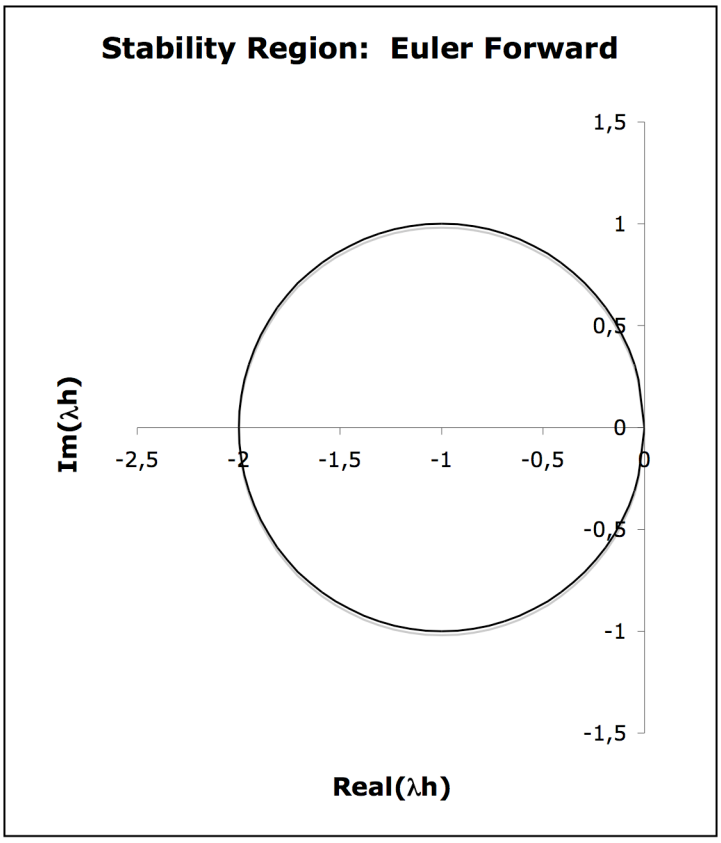


Figure 6.8: Stability Region of Euler Forward Method. The method is stable inside the circle.

6.3.2.2 Euler Backward Method

Application of Euler backward method on the model problem leads to the following discrete equation:

$$y_{n+1} = y_n + hf_{n+1} = y_n + h\lambda y_{n+1} \tag{6.67}$$

This leads to the following amplification factor:

$$\frac{y_{n+1}}{y_n} = \sigma = \frac{1}{1-\lambda h} = \frac{1}{1-\lambda_r h - i\lambda_i h} = \frac{1}{\Lambda e^{i\phi}} \tag{6.68}$$

where, $\Lambda = \sqrt{(1-\lambda_r h)^2 + (-\lambda_i h)^2}$ and $\phi = \tan^{-1}\left(\frac{-\lambda_i h}{1-\lambda_r h}\right)$. In this case, the stability condition $|\sigma| < 1$ essentially means $\frac{1}{\Lambda} < 1$ since, $\left|e^{-i\phi}\right| = 1$. The stability region is therefore given by the condition:

$$\sqrt{(1-\lambda_r h)^2 + (-\lambda_i h)^2} > 1 \text{ or } (\lambda_r h - 1)^2 + (\lambda_i h)^2 > 1 \tag{6.69}$$

The above relation is always true for $\lambda_r < 0$ which is also required for the bounded analytical solution of the model problem. Therefore, the Euler backward method is unconditionally stable for the model problem. The stability region also includes some parts on the right side when $\lambda_r > 0$. The equation 6.69 suggests that the method is stable everywhere outside the circle whose center is at (1, 0). This is shown in Figure 6.9. More importantly, the method is stable for all purely imaginary λ.

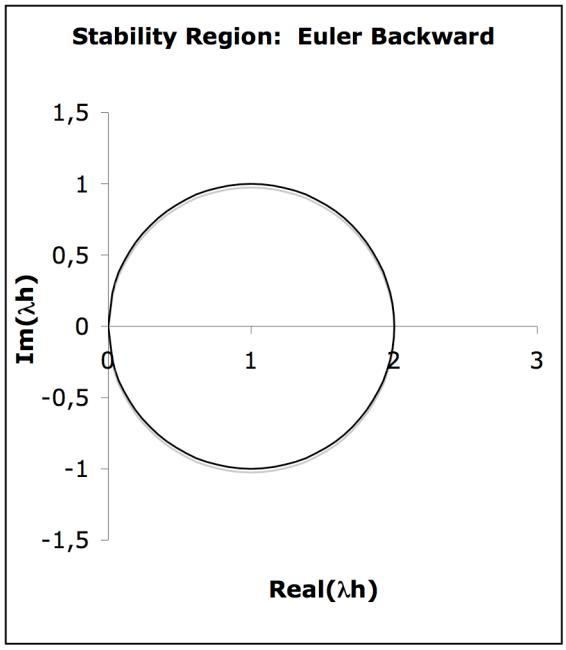


Figure 6.9: Stability region of Euler Backward method. The method is stable outside the circle.

6.3.2.3 Trapezoidal Method

Application of Trapezoidal method to the model problem leads to the following expression for the amplification factor:

$$\frac{y_{n+1}}{y_n} = \sigma = \frac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h} = \frac{1+\frac{1}{2}\lambda_r h + i\frac{1}{2}\lambda_i h}{1-\frac{1}{2}\lambda_r h - i\frac{1}{2}\lambda_i h} = \frac{\Lambda_1 e^{i\phi}}{\Lambda_2 e^{i\varphi}} \tag{6.70}$$

where, $\Lambda_1 = \sqrt{\left(1+\frac{1}{2}\lambda_r h\right)^2 + \left(\frac{1}{2}\lambda_i h\right)^2}$, $\Lambda_2 = \sqrt{\left(1-\frac{1}{2}\lambda_r h\right)^2 + \left(-\frac{1}{2}\lambda_i h\right)^2}$, $\phi = \tan^{-1}\left(\frac{\frac{1}{2}\lambda_i h}{1+\frac{1}{2}\lambda_r h}\right)$ and $\varphi = \tan^{-1}\left(\frac{-\frac{1}{2}\lambda_i h}{1-\frac{1}{2}\lambda_r h}\right)$. Since, $\left|e^{i\phi}\right| = 1$ and $\left|e^{i\varphi}\right| = 1$, the stability condition $|\sigma| < 1$ leads to the following inequality:

$$\frac{\sqrt{\left(1+\frac{1}{2}\lambda_r h\right)^2 + \left(\frac{1}{2}\lambda_i h\right)^2}}{\sqrt{\left(1-\frac{1}{2}\lambda_r h\right)^2 + \left(-\frac{1}{2}\lambda_i h\right)^2}} < 1 \text{ or } \lambda_r h < 0 \tag{6.71}$$

For negative λ_r ($\lambda_r < 0$), the numerator is always less than the denominator except for the case of $\lambda_r = 0$ where it is unity.

For the higher order methods, it is not easy to visualize the stability conditions so easily or derive explicit expressions as we did for Euler methods and trapezoidal method. We will describe a generalized framework that can be used for all multi-step methods and BDFs to generate the stability region.

6.3.2.4 Adams Bashforth Methods

We will illustrate the formulation for the equation of stability region with the 3rd order Adams Bashforth method. Application of the 3rd Order Adams Bashforth method to the model problem leads to the following approximation:

$$y_{n+1} = y_n + h\left(\tfrac{23}{12}\lambda y_n - \tfrac{4}{3}\lambda y_{n-1} + \tfrac{5}{12}\lambda y_{n-2}\right) \tag{6.72}$$

We have defined the amplification factor as $\sigma = \dfrac{y_{n+1}}{y_n}$ for all values of n. Recursive application of this relation leads to the following identities:

$$\frac{y_{n+1}}{y_{n-2}} = \sigma^3,\ \frac{y_n}{y_{n-2}} = \sigma^2 \text{ and } \frac{y_{n-1}}{y_{n-2}} = \sigma \tag{6.73}$$

Using these relations, eq (6.72) becomes:

$$\lambda h = \lambda_r h + i\lambda_i h = \frac{\sigma^3 - \sigma^2}{\left(\tfrac{23}{12}\sigma^2 - \tfrac{4}{3}\sigma + \tfrac{5}{12}\right)} \tag{6.74}$$

The stability of the method requires σ to be less than unity. In a complex plane, the boundary of the region defined by $\sigma = 1$ is given by the identity $\sigma = e^{i\theta}$, $\theta \in (0,\ 2\pi)$. The equation (6.74) thus leads to the following stability region:

$$\lambda_r h + i\lambda_i h = \frac{(\cos 3\theta - \cos 2\theta) + i(\sin 3\theta - \sin 2\theta)}{\left(\tfrac{23}{12}\cos 2\theta - \tfrac{4}{3}\cos\theta + \tfrac{5}{12}\right) + i\left(\tfrac{23}{12}\sin 2\theta - \tfrac{4}{3}\sin\theta\right)} \tag{6.75}$$

Equating the real and imaginary part, it is now easy to calculate the values of $\lambda_r h$ and $\lambda_i h$. If we chose θ in (0, 2π) and plot them, we get the stability region. We show the stability regions of Adams Bashforth methods up to 4th order (Figure 6.10). We have also included the stability region of the Euler forward method in the figure for comparison with the multi-step explicit methods. We leave it to the readers to derive and plot the stability region of the 2nd and 4th order Adams Bashforth methods following the procedure illustrated above and verify with the figure. It is important to note that the stability region of the 3rd order Adams Bashforth method includes some part of the y-axis which means that it is conditionally stable for certain values of purely imaginary λh. This method thus becomes usable for purely imaginary λ while Euler explicit method was unusable.

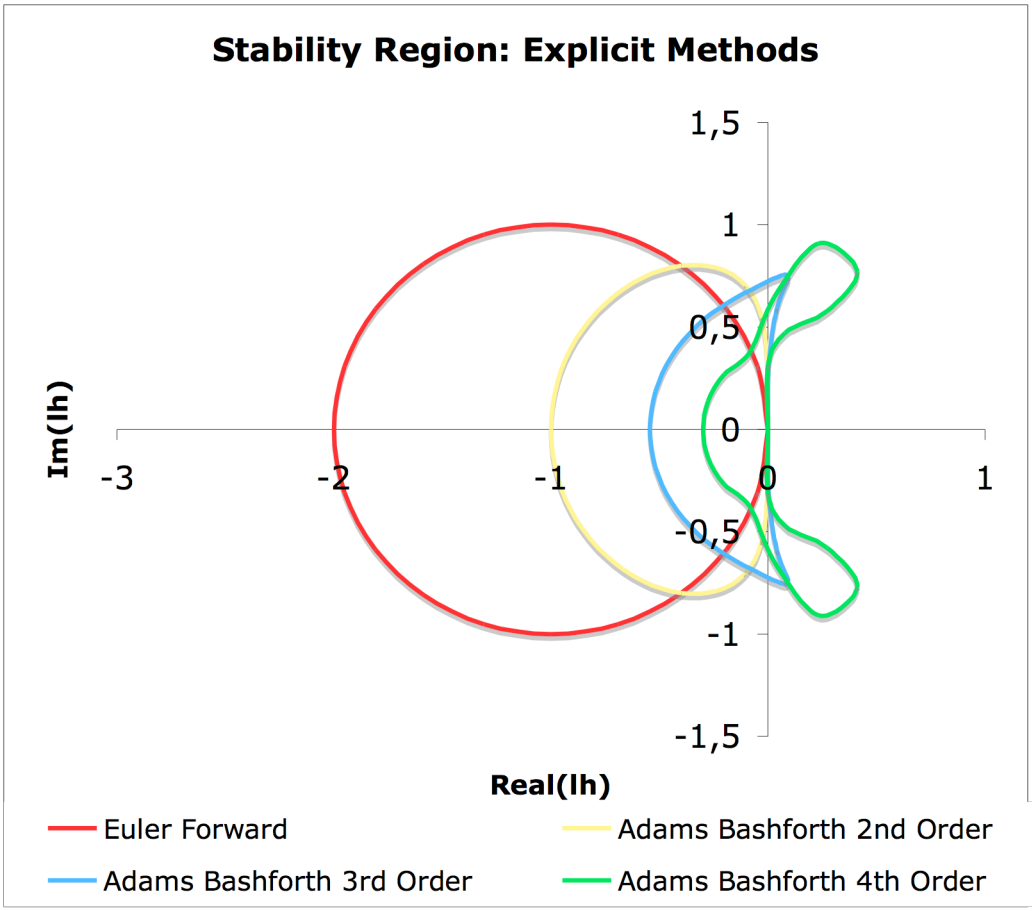


Figure 6.10: Stability regions of the Adams Basforth methods in comparison with Euler forward. The methods are stable inside the enclosed regions.

6.3.2.5 Adams Moulton Methods

We will illustrate the formulation of the equation for stability region for the group of multistep implicit methods using 3rd order Adams Moulton method as example. Approximation of the model problem using this method is as follows:

$$y_{n+1} = y_n + h\left(\tfrac{5}{12}\lambda y_{n+1} + \tfrac{2}{3}\lambda y_n - \tfrac{1}{12}\lambda y_{n-1}\right) \tag{6.76}$$

Following a procedure similar to the explicit methods (6.3.2.4), we obtain the following identity:

$$\lambda_r h + i\lambda_i h = \frac{\sigma^2 - \sigma}{\left(\tfrac{5}{12}\sigma^2 + \tfrac{2}{3}\sigma - \tfrac{1}{12}\right)} = \frac{\left(\cos 2\theta - \cos\theta\right) + i\left(\sin 2\theta - \sin\theta\right)}{\left(\tfrac{5}{12}\cos 2\theta + \tfrac{2}{3}\cos\theta - \tfrac{1}{12}\right) + i\left(\tfrac{5}{12}\sin 2\theta + \tfrac{2}{3}\sin\theta\right)} \tag{6.77}$$

The stability regions for the 3rd and 4th order implicit methods are shown in Figure 6.11. The reader should compare the stability regions of explicit and implicit methods. The stability region of an implicit method is much larger than the explicit method of the same order. This allows larger time step h for the same value of λ. For the multi-step methods (explicit and implicit), size of the stability region decreases with increasing order of the method.

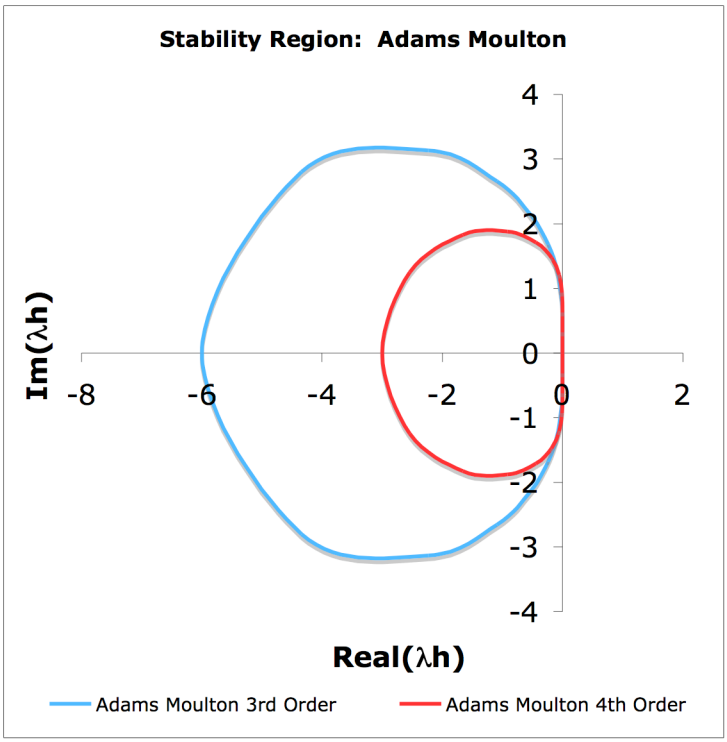


Figure 6.11: Stability regions of 3rd and 4th order Adams Moulton methods. The methods are stable within the enclosed region.

6.3.2.6 Backward Difference Formulae

Recall that the first order BDF is the Euler Backward method and we have already seen the stability region for it. Let's illustrate the formulation of equations for stability regions for BDFs with the 3rd order method (Table 6.5). The discrete approximation is given by:

$$\tfrac{11}{6} y_{n+1} - 3y_n + \tfrac{3}{2} y_{n-1} - \tfrac{1}{3} y_{n-2} = hf_{n+1} \tag{6.78}$$

Application of this approximation to the model equation followed by recursive application of the relation $\sigma = \dfrac{y_{n+1}}{y_n}$ leads to,

$$\begin{aligned} \lambda_r h + i\lambda_i h &= \left(\frac{11}{6} - \frac{3}{\sigma} + \frac{3}{2\sigma^2} - \frac{1}{3\sigma^3} \right) \\ &= \left(\tfrac{11}{6} - 3\cos\theta + \tfrac{3}{2}\cos 2\theta - \tfrac{1}{3}\cos 3\theta \right) - i\left(3\sin\theta - \tfrac{3}{2}\sin 2\theta + \tfrac{1}{3}\sin 3\theta \right) \end{aligned} \tag{6.79}$$

Following in a similar line, the equations for the stability regions can be formulated for all the BDF. These are shown in Figure 6.12 for BDF up to order 6. In order to get a sense of the relative sizes, all 6 regions are shown in one plot but regions of orders 1-3 appear too small. In order to get a clearer picture, orders 1-3 are shown once again on a separate plot (Figure 6.13). Just like the Euler backward method, the

stability regions for all the BDF is outside of the enclosed curve. Therefore, the stability region of a BDF can be visualized as a hole in the entire two dimensional space where the method is not stable. The BDF of orders 1 and 2 have no part of the curves on the left hand side of the y-axis. Therefore, these are stable for all $\lambda_r < 0$. This is not so for BDF of orders 3-6 where some part of the curve crosses over but note that purely real values along the axis on the left are excluded. It is easy to see that out of all the methods described so far, the BDFs have the best stability properties. Moreover, BDF of up to order 6 are unconditionally stable for all purely real $\lambda < 0$. Therefore, any time step size becomes usable without having to worry about the stability. This property makes them especially suitable for a special group of problems known as stiff equations. These will be discussed later in section 6.5.

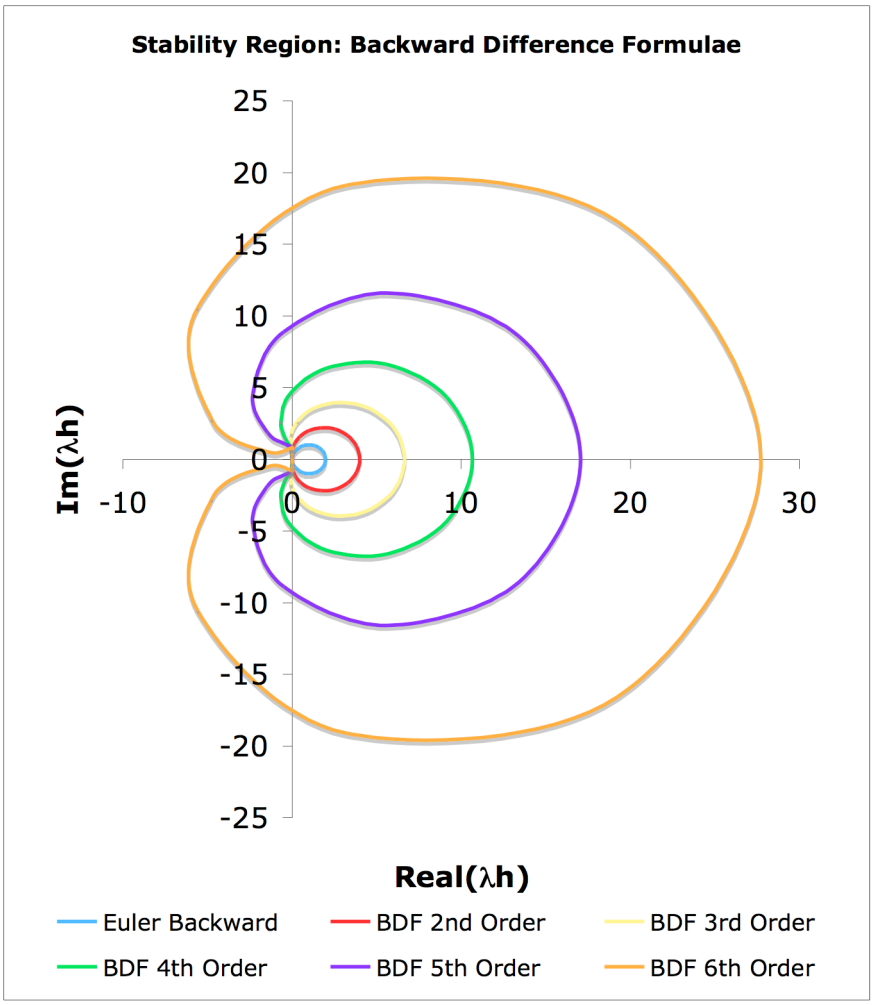


Figure 6.12: Stability Regions of Backward Difference Formulae up to order six. All the regions are shown in one plot for comparison. The methods are stable outside the enclosed curves. These may also be seen as the regions of instability (or holes in the stability region). Size of the hole increases with the order.

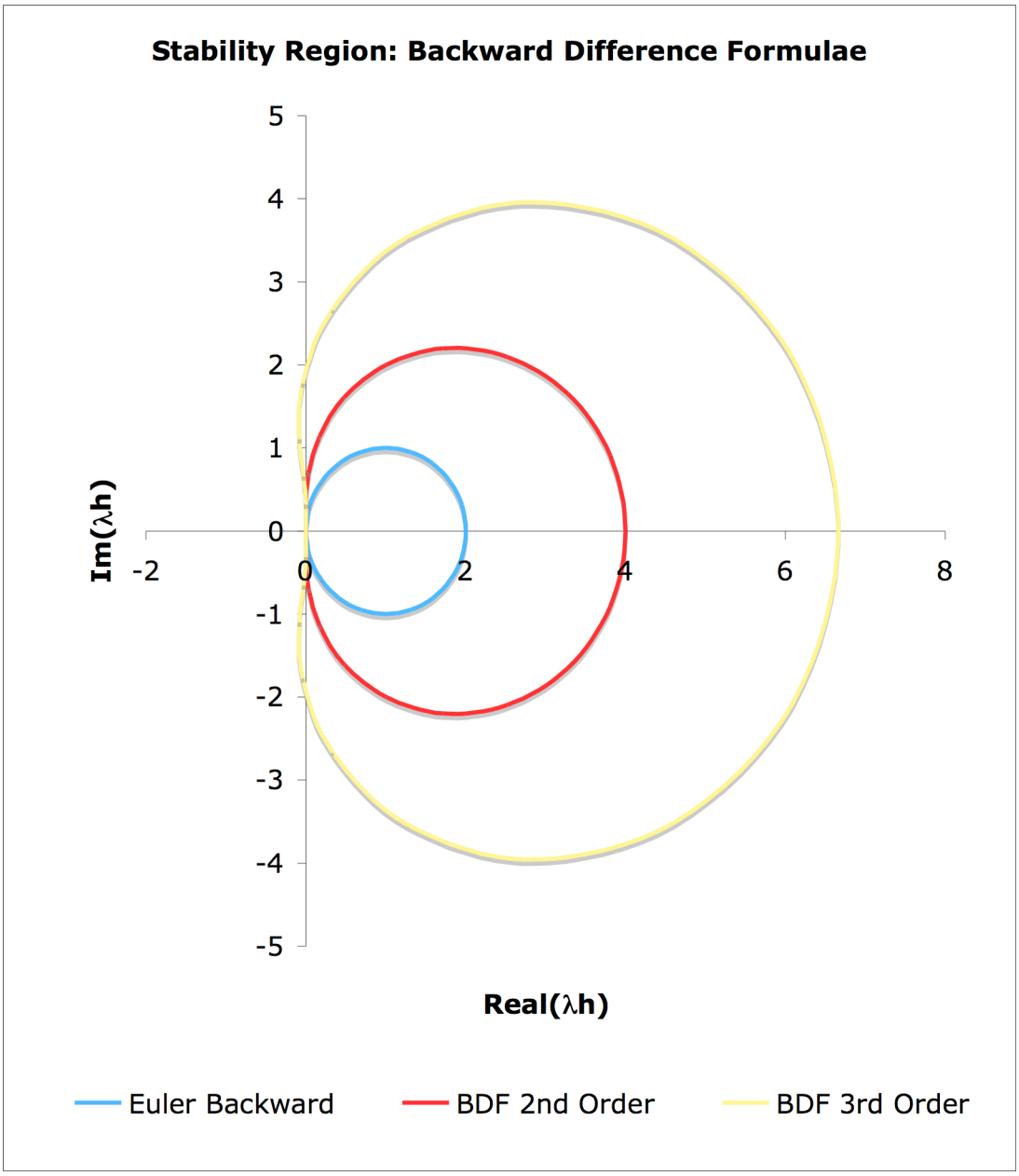


Figure 6.13: Stability Regions of BDF of orders 1-3. No part of the curve is on the right of the y-axis for the 1st and 2nd order methods. The 3rd order method includes some part of the region left of imaginary axis which expands (Figure 6.12) as we move up the order of BDF.

6.3.2.7 Runge Kutta Methods

We will illustrate the formulation of the equation for stability region with a 2nd order Runge Kutta Method (Table 6.6). The method we chose is given below:

$$y_{n+1} = y_n + \tfrac{1}{2}\phi_0 + \tfrac{1}{2}\phi_1, \quad \phi_0 = hf\left(y_n, t_n\right), \quad \phi_1 = hf\left(y_n + \phi_0, t_n + h\right) \tag{6.80}$$

Application of this method to the model problem leads to the following approximation for y_{n+1}:

$$y_{n+1} = y_n + \tfrac{1}{2} h\lambda y_n + \tfrac{1}{2} h\lambda\left(y_n + h\lambda y_n\right) \tag{6.81}$$

Using the amplification factor and the stability limit for the amplification factor, one obtains,

$$1 + \lambda h + \tfrac{1}{2}(\lambda h)^2 = \frac{y_{n+1}}{y_n} = \sigma = e^{i\theta} = \cos\theta + i\sin\theta, \quad \theta \in (0,\ 2\pi) \tag{6.82}$$

At different values of θ, it is required to solve the polynomial equation for the complex roots. While it is easy for a quadratic equation, the 3rd and 4th order Runge Kutta method involves solution of 3rd and 4th order polynomials, respectively. Any method described in chapter 3 for solving non-linear equations and polynomials can be used. Recall that complex roots occur in conjugate pairs. So, determining only half of the distinct roots are enough to plot the full region. Plotting of these roots on the complex plane provides the stability region.

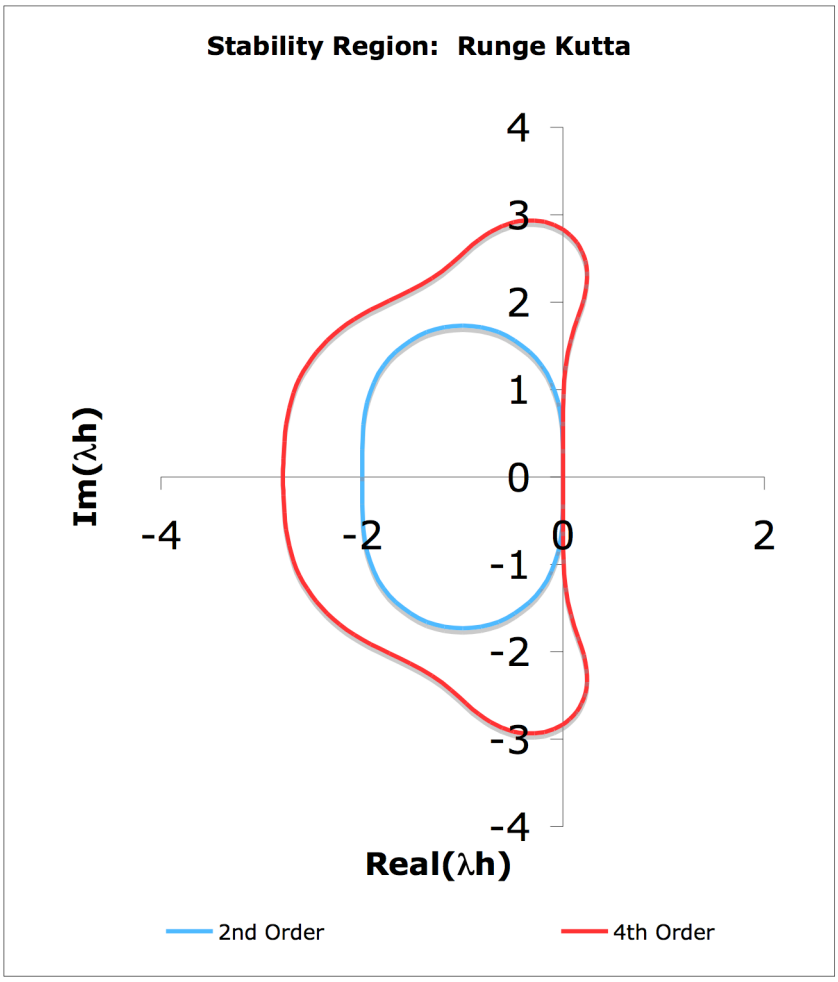


Figure 6.14: Stability regions of 2nd and 4th order Runge Kutta methods. The methods are stable inside the enclosed regions.

The stability regions for the 2nd and 4th order Runge Kutta methods are shown in Figure 6.14. A few points to note are:

- Stability region of the 4th order method includes part of the imaginary axis and therefore stable for purely imaginary λ.

- For all the multi-step methods (explicit and implicit) and BDF discussed so far, the stability region becomes smaller as the order of the method increases. It is opposite for the case of the Runge Kutta. The 4th order method has a larger stability region compared to the 2nd order (Figure 6.14).

- The stability regions as expected are smaller compared to same order implicit methods and BDFs. However, they are bigger compared to the multi-step explicit methods of same order.

We have now learned how to analyze a numerical method for initial value problems for truncation error and stability. We have shown these analyses for all the groups of methods that were developed in the beginning of this chapter. It is now time to formalize the definition of stability. With the help of consistency and stability, we will also formalize the definition of convergence.

6.3.2.8 Formal Definition of Stability and Convergence

Before we venture into formalizing the definition of stability of a numerical method, let us look at the stability of the original IVP given by equation (6.3). In the first subsection, we have shown that the solution of the model problem becomes unbounded for lr > 0. Let us first define the stability for a general solution of the original equation (6.3). For this, let us consider two well defined solutions of the original equation in the interval $[x_0, X]$ with two different initial conditions, $y = u(x)$ for $u(x_0) = u_0$ and $y = v(x)$ for $v(x_0) = v_0$. The stability can be defined as follows:

Definition 6.3: The solution $y = u(x)$ is *stable* on the interval $[x_0, X]$ if for every $\varepsilon > 0$, there exists $\delta > 0$ such that for all v_0 satisfying $|u_0 - v_0| < \delta$, the inequality $|u(x) - v(x)| < \varepsilon$ is true for all $x \in [x_0, X]$. The solution is *asymptotically stable* if it is stable in $[x_0, \infty)$ and $\underset{x \to \infty}{Lim} |u(x) - v(x)| = 0$

In simple terms, the above is saying that a finite perturbation in the initial condition will produce a finite change in the solution as well and the change will approach zero when the variable approaches infinity. It is now easy to see that the solution to the model equation is unstable for $\lambda_r > 0$ and asymptotically stable for $\lambda_r < 0$. At this point, do not confuse about the existence of the solution. The IVP (original as well as the model problem) still possess a unique solution for $\lambda_r > 0$. Now, we shall consider stability of the numerical method in the sense of its capability to simulate the analytical solution as closely as possible.

We shall start with the general linear multi-step method described by equation (6.26). We have already seen that it is a general representation of Euler Methods, Trapezoidal Method, Adams Bashforth, Adams Moulton and BDF. For the convenience of

writing, we rewrite it here after generalizing the limits of the indices for a general linear multi-step method consisting of (n+1) steps:

$$\sum_{i=0}^{n+1} \alpha_i y_i = h \sum_{i=0}^{n+1} \beta_i f_i \tag{6.83}$$

Notice that, i ranges from a future time step (n+1) to as far back as possible (up to 0). So, to startup this method, we will require the set of values $\{y_0, y_1, \cdots y_n\}$ as initial conditions. These have to be computed by a different method or assumed. We shall first consider the sensitivity of this numerical method to small changes in the initial condition, which may occur due to use of different methods for generating these values. A method is stable if it can attenuate the changes in the initial conditions in the final solution. This is known as *zero stability*. For this purpose, we shall consider solutions in the interval $[x_0, X]$ where a unique solution exists for the original IVP and assume two initial conditions $\{y_0, y_1, \cdots y_n\}$ and $\{z_0, z_1, \cdots z_n\}$ generated by two different methods. The *zero stability* is defined as follows:

Definition 6.4: The linear multi-step method (6.26) is *zero stable* if there exists a constant K such that the following inequality holds,

$$|y_{n+1} - z_{n+1}| \le K \max\{|y_0 - z_0|, |y_1 - z_1|, \cdots |y_n - z_n|\} \text{ as } h \to 0$$

Let us now see how we can assess the zero stability for a given method. We have earlier defined the amplification factor $\left|\frac{y_{n+1}}{y_n}\right| = \sigma$ for the model problem. For the general problem $y' = f$, let us define another amplification factor for the functional value as $\left|\frac{f_{n+1}}{f_n}\right| = \xi$. Note that both σ and ξ can be complex. In terms of the amplification factors, the method can be written as follows:

$$y_0 \sum_{i=0}^{n+1} \alpha_i \sigma^i = h f_0 \sum_{i=0}^{n+1} \beta_i \xi^i \tag{6.84}$$

The y_0 and f_0 are the initial conditions and the slope function value at the initial condition, respectively. Therefore, both are known and constant. For a fixed time step $h > 0$, the above defines two polynomials in the complex space involving σ and ξ. These polynomials can be written as follows:

$$\begin{aligned} P(\sigma) &= y_0 \sum_{i=0}^{n+1} \alpha_i \sigma^i = \sum_{i=0}^{n+1} a_i \sigma^i \text{ where } a_i = y_0 \alpha_i \\ Q(\xi) &= h f_0 \sum_{i=0}^{n+1} \beta_i \xi^i = \sum_{i=0}^{n+1} b_i \xi^i \text{ where } b_i = h f_0 \beta_i \end{aligned} \tag{6.85}$$

The $P(\sigma)$ and $Q(\xi)$ are known as *first* and second *characteristic* polynomials, respectively. It can be shown mathematically (the proof is beyond the scope of this book) that the zero stability implies that all the roots of the first characteristic polynomial lies within a unit disc ($|\sigma| < 1$) and if any root lies on the unit circle

($|\sigma| = 1$), it is a simple root. This can now be used easily to evaluate the *zero stability* of any of the linear multi-step methods described in this chapter.

A numerical method is said to be convergent if the computed value at any time step approaches the true solution as $h \to 0$. The *consistency* and *zero stability* are necessary conditions for the convergence of a linear multi-step method. For any consistent linear multi-step method, zero stability is also a sufficient condition for the method to be convergent.

Although *zero stability* helps in assessing the convergence of a linear multi-step method, it is not very useful in the context of practical application because it defines the stability in the asymptotic limit $h \to 0$. So, we need to concentrate on cases with fixed $h > 0$ and assess how the approximate solution behaves with respect to the true solution. For this purpose, we shall apply the general linear multi-step method (6.26) to our model problem (6.57),

$$\sum_{i=0}^{n+1} \alpha_i y_i = h \sum_{i=0}^{n+1} \beta_i (\lambda y_i) \tag{6.86}$$

Using the definition of the amplification factor we obtain the *stability polynomial* as,

$$R(\sigma) = \sum_{i=0}^{n+1} y_0 (\alpha_i - \lambda h \beta_i) \sigma^i = \sum_{i=0}^{n+1} c_i \sigma^i = 0 \text{ where } c_i = y_0 (\alpha_i - \lambda h \beta_i) \tag{6.87}$$

The reader can now compare this polynomial with the ones derived earlier in the section for various methods. For any fixed h, the coefficient c_i's are constants. In that case, the method will be stable if all the roots lie within the unit disc ($|\sigma| < 1$). If any root is outside the disc ($|\sigma| > 1$), the method is unstable. The case $|\sigma| = 1$ indicates the boundary. Any root lying on this boundary must be a simple root for the method to be stable. In the earlier subsections, our interest was to find this boundary and therefore, we had used the values of $|\sigma| = 1$ to draw it. In this subsection, our interest is to classify various kinds of stability that may be observed depending on the nature of the roots of this characteristic polynomial. We will define some of them here. Let us assume $\{\sigma_1, \sigma_2, \cdots \sigma_n\}$ are the roots of the stability polynomial.

Definition 6.5: A linear multi-step method is *absolutely stable* in an open set C_A of the complex plane if the roots of the stability polynomial satisfy $|\sigma_i| < 1$ for all i and for all $\lambda h \in C_A$. The open set C_A is the *region of absolute stability* for the method.

According to this definition, all the methods described in this chapter are absolutely stable. In the subsections 6.3.2.1-7, we had drawn the regions of their absolute stability. At this point, the reader must note that the *absolutely stability* automatically ensures *zero stability* (Can you say why ?).

At this point, recall our working definitions of stability (unconditional and conditional). We would now like to relate that to our formal definitions. We define the following for this,

Definition 6.6: A linear multi-step method is called *A-stable* if its region of *absolute stability* includes the whole left half of the complex plane ($h\lambda_r < 0$) shown as the shaded region of Figure 6.7.

Now, it is easy to see that the methods we defined as "unconditionally stable" are in fact *A-stable*. Whereas, both "unconditionally stable" and "conditionally stable" are included in the definition of *absolute stability*. Therefore, the *A-stable* methods are subset of the *absolutely stable* methods. For example, none of the explicit methods are *A-stable*. The implicit methods and BDF of order larger than 2 are also not *A-stable*. However, all the methods are *absolutely stable*. In fact, it is possible to show that (Dahlquist 1963) highest order of an implicit linear multi-step method cannot exceed 2. Recognizing that the definition of *A-stability* is too restrictive, there were many attempts to define a few more relaxed options. One notable amongst them is $A(\alpha)$-stability with $\alpha \in \left(0, \frac{\pi}{2}\right)$. We shall refrain from describing all such definitions as they are beyond the scope of this book. However, we will define one last type called *stiff stability* that we shall use at a later section.

Definition 6.7: A linear multi-step method is *stiffly stable* if its region of *absolute stability* includes the following two sets C_1 and C_2:

$$C_1 = \{\lambda h \in C \ : \ \lambda_r h < -r\}$$
$$C_2 = \{\lambda h \in C \ : \ -r < \lambda_r h < 0, \ -s < \lambda_i h < s\}$$

If you want to visualize it, think of a wedge on the left half of the complex plane with the tip placed at the origin. If you moved away towards left from the axis beyond $-r$, the complete imaginary range is included in the region of absolute stability. If you are approaching along the real axis from the left and towards the origin, the region of absolute stability is getting constricted to a maximum limit of +/- s along the imaginary axis as you approach closer than $-r$. In this case, the complete left half of the real axis is included in the region. Now compare and see that all the BDF of up to 6^{th} order are *stiffly stable.*

In addition to the error analysis described above, another kind of analysis known as *phase error analysis* becomes important if one is dealing with equations that have periodic solutions. When the truncation error accumulates in a periodic solution (think of a wave!), it manifests itself into two forms. It affects the amplitude as well as the phase of the wave. The stability is a measure of increase (or decrease) of the amplitude. The phase error is a measure of the phase change of the wave compared to its analytical (true) solution. It is important to note that both are caused by accumulation of truncation error. We only observe their effects in different forms through different analysis. Periodic solution occurs in many science and engineering scenario such as waves, earthquake, prey-predator interactions, etc. We address the analysis of phase error in the next section.

6.3.3 Phase Error

We will introduce the concept of phase error by applying Euler forward method to an example problem similar to the model equation but with purely imaginary λ. Let us consider the following equation:

$$\frac{dy}{dt} = i\lambda y, \;\; y(0) = y_0 \tag{6.88}$$

The analytical solution of the above equation is of course $y = y_0 e^{i\lambda t}$. Application of the Euler forward method to the problem leads to the following approximate solution:

$$y_{n+1} = y_n(1 + i\lambda h) \tag{6.89}$$

Note, that y_n is a complex number. At any time step n, if $y_n = a + ib$, putting this in eq (6.89), we obtain,

$$y_{n+1} = (a - \lambda hb) + i(b + \lambda ha) \tag{6.90}$$

The true or analytical solution at any time step is computed as,

$$y_n = y_0 e^{i\lambda hn} = y_0(\cos\lambda hn + i\sin\lambda hn) \tag{6.91}$$

Therefore, for a given λ and y_0, both true and approximate solution by Euler explicit method can be computed from relations (6.90) and (6.91). For illustration purposes, we take $\lambda = 1$, $y_0 = 1$ and $h = \pi/10$.

Figure 6.15 shows the real and imaginary parts of the solution for one full period. The amplitude of the approximate solutions for both real and imaginary parts differs from the true solution and the difference increases with time. This was expected from the stability analysis as Euler explicit method is unconditionally unstable for pure imaginary λ. In fact, the increase in the amplitude can be computed using the amplification factor derived in stability analysis. However, their phases are also different, i.e., crests and troughs are at different locations and the starting points of the waves for imaginary part at zero time are also different. This is known as the *phase error* which neither the truncation error analysis nor the stability analysis provided any measure for. Amplitude and phase $\left[\tan^{-1}\frac{\mathrm{Im}(y_n)}{\mathrm{Re}(y_n)}\right]$ of the true and approximate solutions are shown in Figures 6.16 and 6.17, respectively. For the behaviour of a periodic solution, it is important to look at the phase diagram by plotting real vs. imaginary part of the solution (Figure 6.18). For the analytical solution, it is a unit circle. We observe that the true solution becomes a spiral while the true solution is a closed curve (circle in this case). This means that the numerical method has both amplitude and phase errors. If the phase error is absent, the approximate solution will remain as a circle but with larger or smaller radius depending on whether the amplitude error amplifies or attenuates. We will now show how to estimate the phase error for a numerical method.

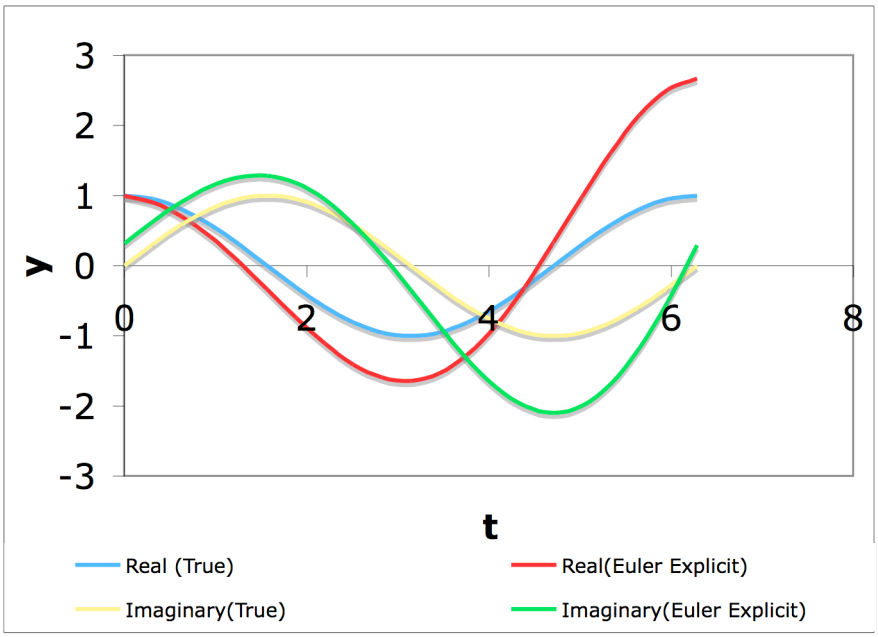


Figure 6.15: Real and imaginary parts of a periodic solution by Euler Forward method.

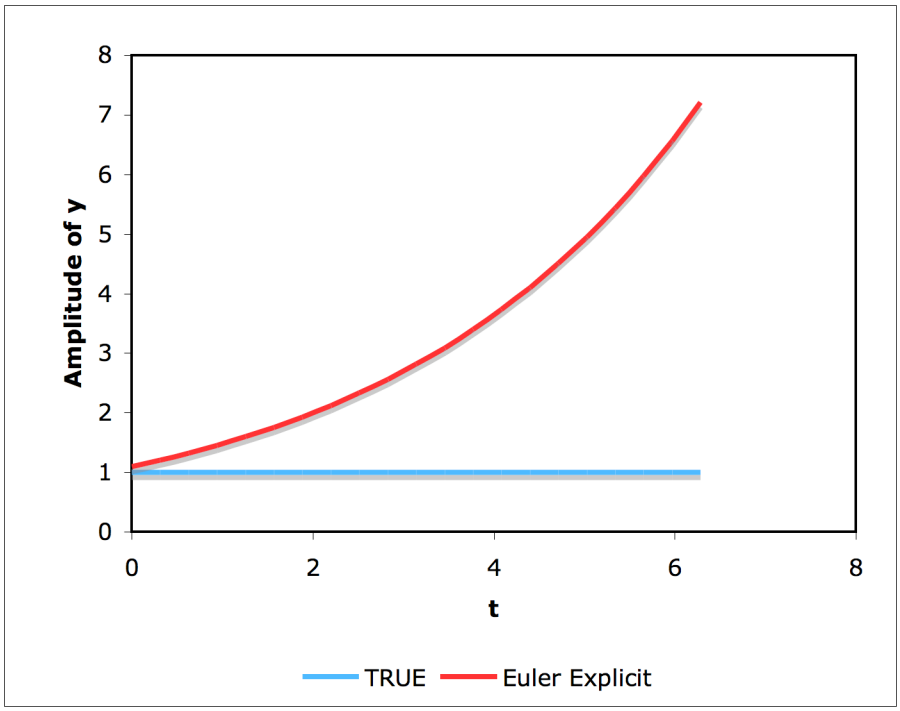


Figure 6.16: Amplitude error in Euler Forward Method.

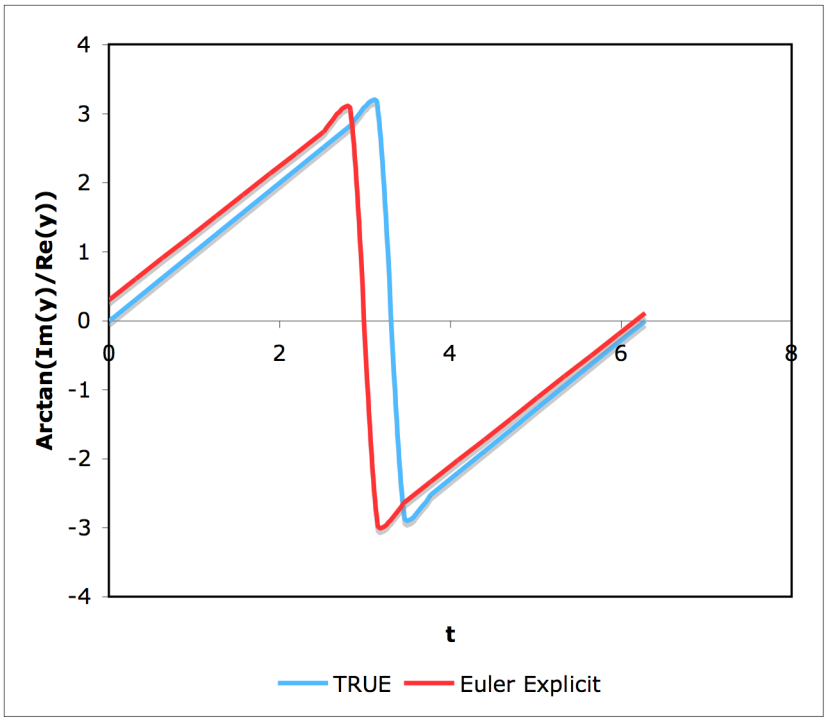


Figure 6.17: Phase error in Euler Forward method.

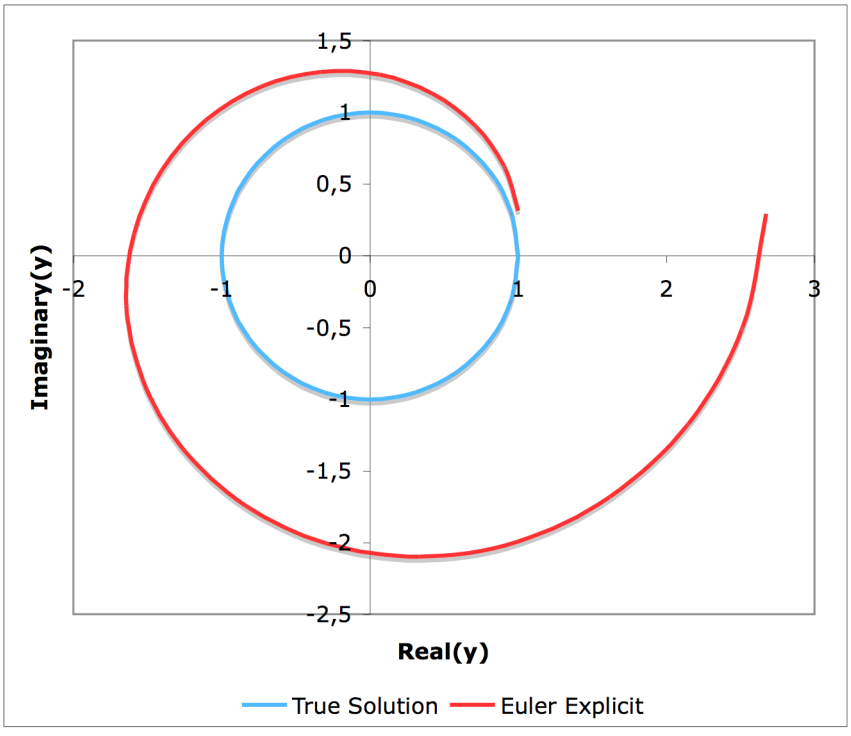


Figure 6.18: True and Approximate solution cycles.

In order to obtain a measure of the phase error, we shall use eq (6.88) as the model equation. Using the analytical solution, we obtain the true amplification factor as:

$$\sigma_{True} = \frac{y_{n+1}}{y_n} = \frac{y_0 e^{i\lambda h(n+1)}}{y_0 e^{i\lambda hn}} = e^{i\lambda h} \tag{6.92}$$

Amplitude of the amplification factor is $|\sigma_{True}| = \sqrt{\cos^2 \lambda h + \sin^2 \lambda h} = 1$ and the phase is $\theta_{True} = \tan^{-1}\left(\frac{\sin \lambda h}{\cos \lambda h}\right) = \lambda h$. For the Euler forward method, the amplification factor from eq (6.62) can be seen as $\sigma = 1 + i\lambda h$. Therefore, the amplitude of the approximate amplification factor is,

$$|\sigma| = \sqrt{1 + (\lambda h)^2} > |\sigma_{True}| = 1 \tag{6.93}$$

Since, magnitude of the approximate amplification factor is larger than the true amplification factor, the amplitude increases with each time step. This reconfirms the instability of the Euler explicit method for purely imaginary λ. The phase of the approximate amplification factor can be computed as,

$$\theta = \tan^{-1}\left(\frac{\mathrm{Im}(\sigma)}{\mathrm{Re}(\sigma)}\right) = \tan^{-1} \lambda h \tag{6.94}$$

We now define phase error (*PE*) as,

$$PE = \theta_{True} - \theta = \lambda h - \tan^{-1} \lambda h \tag{6.95}$$

Using the Taylor's series, we can write:

$$\tan^{-1} \lambda h = \lambda h - \frac{(\lambda h)^3}{3!} + \frac{(\lambda h)^5}{5!} - \frac{(\lambda h)^7}{7!} + \frac{(\lambda h)^9}{9!} \cdots \tag{6.96}$$

Therefore, we obtain the following for the phase error of the Euler explicit method:

$$PE = \frac{(\lambda h)^3}{3!} - \frac{(\lambda h)^5}{5!} + \frac{(\lambda h)^7}{7!} - \frac{(\lambda h)^9}{9!} \cdots \tag{6.97}$$

In the leading order term, λh is raised to the power of 3. Following a convention similar to truncation error, we would refer to this as 3rd order accuracy. Therefore, the Euler explicit method is 3rd order accurate for phase error. Moreover, the first term is positive and summation of the infinite series is also positive. Therefore, θ_{True} is always larger than θ and the waves of the approximate solution will be lagging behind the true solution. This will be referred as *phase lag*.

Let us look at the phase error of a few more methods namely, Euler implicit, Trapezoidal and 2nd order Runge Kutta. The amplification factors for these methods when applied to the model problem for the phase error can be obtained from the corresponding expressions derived in the stability analysis by replacing λ with $-i\lambda$. The amplification factors, amplitude and phase for these methods are listed below:

Euler Implicit: $\sigma = \dfrac{1}{1 - i\lambda h},\ |\sigma| = \dfrac{1}{\sqrt{1 + (\lambda h)^2}},\ \theta = \tan^{-1}(\lambda h)$ (6.98)

Trapezoidal Method: $\sigma = \dfrac{1 + i\frac{1}{2}\lambda h}{1 - i\frac{1}{2}\lambda h},\ |\sigma| = 1,\ \theta = 2\tan^{-1}\left(\dfrac{\lambda h}{2}\right)$ (6.99)

2nd Order Runge Kutta: $\sigma = 1 - \dfrac{(\lambda h)^2}{2} + i\lambda h,\ |\sigma| = \sqrt{1 + \dfrac{(\lambda h)^4}{4}},\ \theta = \tan^{-1}\left(\dfrac{\lambda h}{1 - \dfrac{(\lambda h)^2}{2}}\right)$ (6.100)

From the above relations, the leading order terms for the PE may be calculated as $\dfrac{(\lambda h)^3}{3}$, $\dfrac{(\lambda h)^3}{12}$ and $-\dfrac{(\lambda h)^3}{6}$ for Euler implicit, trapezoidal and 2nd order Runge Kutta methods, respectively. Readers should verify these by following the procedure similar to the one shown for Euler forward.

It is easy to see that the magnitude of the amplification factor for Euler implicit method is always less than unity. Therefore, application of this method to a periodic solution will always lead to attenuation of the peaks. Trapezoidal method on the other hand has no amplitude error. The 2nd order Runge Kutta method was unstable for the purely imaginary λ and leads to increase in amplitude but to a lesser extent compared to the Euler explicit method.

The phase error for the Euler implicit method is same as that of the explicit. Although, the order of the phase error for trapezoidal method is same as the Euler methods but the coefficient is 1/4th. So, the phase error is less but both are positive and leads to phase lag. The 2nd order Runge Kutta method on the other hand leads to phase lead. Therefore, peaks and troughs of the approximate solution will occur earlier than the true solution.

Following in the similar lines, the phase error of the other methods can also be computed. We leave these to the readers as exercise!

We have looked at ways to analyze the methods for different kinds of error. In the next section, we will look at a few application perspectives of different methods presented in sections 6.1 for different kinds of problems.

Exercise 6.3

1. Consider a 2nd order central difference approximation of the derivative in equation (6.3) and the slope function evaluated at *n*. The method becomes $y_{n+1} = y_{n-1} + 2hf_n$. What is the order of the method ? Is it consistent ? What is the region of stability for this method ?

2. In exercise 6.2, consider the methods derived in problem 7 using the Simpson's rules. Evaluate their truncation errors, stability properties and phase error.

3. Perform stability analyses and phase error analysis of the 3rd order R-K method shown in table 6.6.

6.4 Application of Various Methods

In the section 6.2 we have introduced the reader with different numerical methods to solve a first order ordinary differential equation and in section 6.3 we have shown different kinds of errors introduced in these methods due to approximation. We have also shown the ways to compute these errors. In this section, we touch upon some aspects of application of these methods for various types of problems:

- Decision of the order of the method is guided by how accurate solution one needs. For a preliminary design problem, it often suffices to use a fast lower order scheme with single precision computation. However, for the final solution, one often requires higher order schemes with more accuracy. For engineering applications, it is generally sufficient to use a scheme with 4th order global truncation error. It is also possible to combine several numerical methods to increase the accuracy. This will be discussed later in this section.

- If the smaller times steps (large amount of computations) are not a concern (typically for small and simple problems), it is easier to use and implement explicit and R-K methods. However, often the stability concerns force the time step to be so small that the computation time becomes large even for simple problems.

- Implicit methods and BDFs have much larger stability regions and allow larger time steps. These can save significant amount of computation times.

- If the solution is periodic and/or imaginary, it is required to choose a method which has at least some parts of its *region of absolute stability* along the imaginary axis and has low phase error.

In the following section, we cover a problem specific to multi-step method that is encountered when one tries to apply these methods.

6.4.1 Startup of non-self starting Multi-Step methods

We have stated in section 6.2 as well as shown in Example 6.1 that all the multi-step methods of order greater than 2 are non-self starting. In a first order IVP, the value of the independent variable is known at the initial point in terms of the initial condition. Typically, we designate it as y_n and try to compute y_{n+1}. However, at the start n is zero. Thus, only y_0 is known when we try to compute y_1. Let us illustrate the problem with the following example:

Example 6.5: Solve $\frac{dy}{dt} = -0.6y; \quad y(0) = 1$ using 4th order Adams Bashforth method.

Solution: The fourth order Adams Bashforth method is given by:

$$y_{n+1} = y_n + h\left(\tfrac{55}{24} f_n - \tfrac{59}{24} f_{n-1} + \tfrac{37}{24} f_{n-2} - \tfrac{3}{8} f_{n-3}\right)$$

Using the functional form $f = -0.6y$, the above relation for the first four time steps translates to:

$$y_1 = 1 + h\left(\tfrac{5.5}{24} y_0 - \tfrac{5.9}{24} y_{-1} + \tfrac{3.7}{24} y_{-2} - \tfrac{0.3}{8} y_{-3}\right)$$
$$y_2 = 1 + h\left(\tfrac{5.5}{24} y_1 - \tfrac{5.9}{24} y_0 + \tfrac{3.7}{24} y_{-1} - \tfrac{0.3}{8} y_{-2}\right)$$
$$y_3 = 1 + h\left(\tfrac{5.5}{24} y_2 - \tfrac{5.9}{24} y_1 + \tfrac{3.7}{24} y_0 - \tfrac{0.3}{8} y_{-1}\right)$$
$$y_4 = 1 + h\left(\tfrac{5.5}{24} y_3 - \tfrac{5.9}{24} y_2 + \tfrac{3.7}{24} y_1 - \tfrac{0.3}{8} y_0\right)$$

From the initial condition $y_0 = y(0) = 1$ the values of y is known at the zero time and values at the negative time steps y_{-1}, y_{-2} and y_{-3} are undefined or not known. So, y_1, y_2 and y_3 cannot be computed from the above equations. The expression of y_4 contains y values with positive subscripts but in order to use it, y_1, y_2 and y_3 have to be known. So, the fourth order Adams Bashforth method becomes un-usable unless we find some other ways to compute the values of y_1, y_2 and y_3. This is the reason all multi-step methods are non-self starting as they require the help of some other method to compute the dependent variable values at the initial time steps. Numbers of such values to be computed by other methods vary depending on the type and order of the method. For the explicit methods, the number of time steps to be computed by other methods is $(n - 1)$ for $n > 1$ where n is the order of the global truncation error of the method. For the implicit methods, the number is $(n - 2)$. There are several ways one can attempt to start these methods. These are discussed below along with their advantages and disadvantages.

Use of Lower Order Methods

It is clear from the Example 6.5 that if one computes y_1, y_2 and y_3 using lower order methods, it will be possible to use the 4th order Adams Bashforth method to compute y_4 and subsequent values of y at future time steps. One procedure is shown in the example below:

Example 6.6: Solve the problem of example 6.1 using 4th order Adams Bashforth (AB) method with $h = 0.5$. Use Euler forward for the first time step, 2nd order AB for the second, 3rd order AB for the 3rd and 4th order AB for all the successive time steps. Compare the true relative error at $t = 5$ with the error obtained in example 6.1 using 4th order AB where true solution was used for the first three time steps.

Solution: We already have seen the computation formulae for these methods in example 6.1. We tabulate the computations below:

Time (t)	Method Used	True Solution	Numerical Solution	Error (%)
0		1,0000	1,0000	
0,5	EF	0,7408	0,7000	5,51
1	AB2	0,5488	0,5350	2,52
1,5	AB3	0,4066	0,3824	5,95

2	AB4	0,3012	0,3028	0,53
2,5		0,2231	0,2079	6,81
3		0,1653	0,1716	3,84
3,5		0,1225	0,1100	10,20
4		0,0907	0,0988	8,95
4,5		0,0672	0,0560	16,68
5		0,0498	0,0588	18,19

The error at t = 5.0 is 18.2% compared to 1.14% obtained using true solution for the first three time steps. It is easy to see the larger error in the three initial time steps.

One obviously sacrifices accuracy since, lower order methods are less accurate and larger errors incurred in the initial time steps would progress through the solution in the future time steps. However, it is possible to reduce such errors by application of Richadson's extrapolation. From the truncation error analysis notice that, approximation ($\tilde{y}_{n+1}$) by any method (of local truncation error order p) can be represented as follows in terms of true value (y_{n+1}) :

$$\tilde{y}_{n+1} = y_{n+1} + c_1 h^p + c_2 h^{p+1} + c_3 h^{p+2} + \cdots \tag{6.101}$$

where, c_i's are not functions of h. They only involve constants and higher derivatives evaluated at a fixed point. Some examples of such expression are shown below

Euler Forward: $\tilde{y}_{n+1} = y_{n+1} - \frac{h^2}{2} y_n'' - \frac{h^3}{6} y_n''' - \frac{h^4}{24} y_n'''' + o\left(h^5\right)$

Euler Backward: $\tilde{y}_{n+1} = y_{n+1} + \frac{h^2}{2} y_n'' + \frac{h^3}{3} y_n''' + \frac{h^4}{8} y_n'''' + o\left(h^5\right)$

3rd order Adams Basforth: $\tilde{y}_{n+1} = y_{n+1} - \frac{3}{8} h^4 y_n'''' + o\left(h^5\right)$

Let us denote the approximate value $\tilde{y}_{n+1}$ calculated using a time step h as $\tilde{y}_{n+1}^{h}$. Similarly, using 6.101, we can write

$$\tilde{y}_{n+1}^{h/2} = y_{n+1} + c_1 \left(\frac{h}{2}\right)^p + c_2 \left(\frac{h}{2}\right)^{p+1} + c_3 \left(\frac{h}{2}\right)^{p+2} + \cdots \tag{6.102}$$

$$\tilde{y}_{n+1}^{h/4} = y_{n+1} + c_1 \left(\frac{h}{4}\right)^p + c_2 \left(\frac{h}{4}\right)^{p+1} + c_3 \left(\frac{h}{4}\right)^{p+2} + \cdots \tag{6.103}$$

Eliminating c_1 from (6.101) and (6.102), we can get an approximation of y_{n+1} which is (p+1) order accurate.

$$\tilde{y}_{n+1}^{h,h/2} = \frac{2^p \tilde{y}_{n+1}^{h/2} - \tilde{y}_{n+1}^{h}}{\left(2^p - 1\right)} = y_{n+1} - \left(\frac{1}{2\left(2^p - 1\right)}\right) c_2 h^{p+1} - \left(\frac{3}{4\left(2^p - 1\right)}\right) c_3 h^{p+2} - \cdots \tag{6.104}$$

Since, the new approximation was obtained from two approximations with time steps h and $h/2$, we denote it as $\tilde{y}_{n+1}^{h,h/2}$. Similarly, we can obtain another approximation of y_{n+1} with (p+1) order accuracy by eliminating c_1 from the p^{th} order accurate approximations with time steps $h/2$ (6.102) and $h/4$ (6.103):

$$\tilde{y}_{n+1}^{h/2,h/4} = \frac{2^p \tilde{y}_{n+1}^{h/4} - \tilde{y}_{n+1}^{h/2}}{\left(2^p - 1\right)} = y_{n+1} - \left(\frac{1}{2\left(2^p - 1\right)}\right) c_2 \left(\frac{h}{2}\right)^{p+1} - \left(\frac{3}{4\left(2^p - 1\right)}\right) c_3 \left(\frac{h}{2}\right)^{p+2} - \cdots \quad (6.105)$$

It is now possible to use two (p+1) order accurate estimates of y_{n+1} to obtain a (p+2) order accurate estimate by eliminating c_2 from (6.104) and (6.105):

$$\tilde{y}_{n+1}^{h,h/2,h/4} = \frac{2^{p+1} \tilde{y}_{n+1}^{h/2,h/4} - \tilde{y}_{n+1}^{h,h/2}}{\left(2^{p+1} - 1\right)} = y_{n+1} + \frac{3}{8\left(2^p - 1\right)\left(2^{p+1} - 1\right)} c_3 h^{p+2} + \cdots \quad (6.106)$$

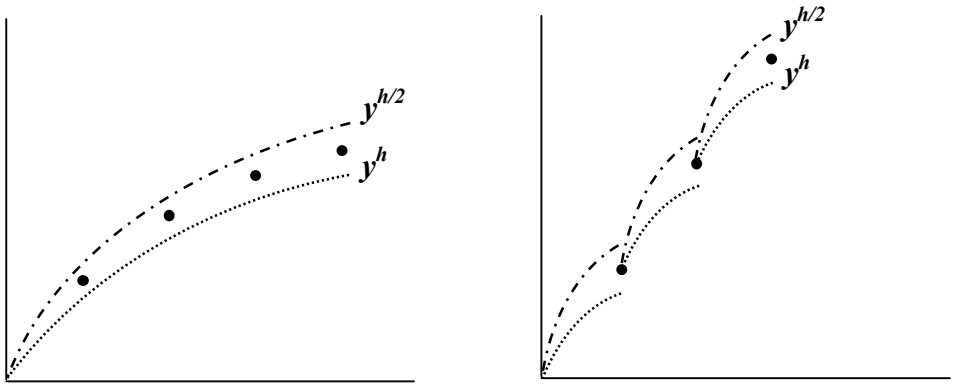
Formulae for the Euler forward method are $\tilde{y}_{n+1}^{h,h/2} = \frac{4\tilde{y}_{n+1}^{h/2} - \tilde{y}_{n+1}^{h}}{3}$, $\tilde{y}_{n+1}^{h/2,h/4} = \frac{4\tilde{y}_{n+1}^{h/4} - \tilde{y}_{n+1}^{h/2}}{3}$ and $\tilde{y}_{n+1}^{h,h/2,h/4} = \frac{8\tilde{y}_{n+1}^{h/2,h/4} - \tilde{y}_{n+1}^{h,h/2}}{7}$ where, $\tilde{y}_{n+1}^{h}$, $\tilde{y}_{n+1}^{h/2}$ and $\tilde{y}_{n+1}^{h/4}$ are approximations of y_{n+1} obtained using Euler forward method with time steps of h, $h/2$ and $h/4$, respectively.

If one also had an approximation $y_{n+1}^{h/8}$ with a time step size of $h/8$, one could obtain another p+2 order accurate estimate $\tilde{y}_{n+1}^{h/2,h/4,h/8}$. Using two ($p$+2) order accurate estimates $\tilde{y}_{n+1}^{h,h/2,h/4}$ and $\tilde{y}_{n+1}^{h/2,h/4,h/8}$, it would be possible to compute another estimate which is (p+3) order accurate. This way, it is possible to compute estimates at any time step up to arbitrarily higher order accuracy.

We had derived the Richardson's extrapolation formulae (6.104-6.106) using the *LTE* of a method. The same expression can be derived using similar steps if one considers p as the *GTE* of the method instead of LTE. We leave it to the reader to check the veracity of the statement! Recall that for any p^{th} order method, the *LTE* is p+1. Therefore, for the same method, we can obtain two different Richardson's extrapolation formulae by using *LTE* and *GTE*. The natural question is which error shall one use ? By definition, the *LTE* gives the error in one time step if the step is taken from a true point. The order of *LTE* is used when the extrapolated values have to be generated for one time step (or a maximum of 2) from the initial point. This is typically the case for the startup of a 2nd or 3rd order multi-step methods. If the extrapolated values are required at a number of time step, use the order of *GTE*. Overall, if you are in doubt, use the order of *GTE*.

There are two ways to apply the Richadson's extrapolation scheme. In the example 6.6, one required to calculate y_1, y_2 and y_3 using a lower order method. It is possible to compute these values at all three times using say Euler Forward with time steps of h, $h/2$ and $h/4$. Then, use the computed values for estimation of more accurate values using Richardson's extrapolation. The extrapolated value at one time step is not used as the starting point for next time step. This is called *passive Richardson's* extrapolation. In the other application, one computes values of y_1 using the lower

order method with time steps h, $h/2$, $h/4$, etc. and then applies Richardson's extrapolation to compute a more accurate value of y_1. This extrapolated value at y_1 is then used as starting point for computation of y_2 using lower order method. This is called *active Richardson's extrapolation*. A graphical schematic representation of these two schemes are shown below (Figure 6.19). We show the startup using Euler Forward method with both active and passive Richardson's extrapolation in Example 6.7.



(a) Passive (b) Active

Figure 6.19: Schematic showing (a) Passive and (b) Active Richardson's extrapolation concept. Two different dotted lines showing values obtained using the lower order method with time step sizes of h and $h/2$. Dots are the higher order values obtained using Richardson's extrapolation.

Example 6.7: Solve the same example problem of 6.1 using 3rd order Adams Bashforth Method with $h = 5.0$. Use Euler forward method along with Richardson's extrapolation (both active and passive) for the first three time steps to obtain same order of local truncation error as the 3rd order Adams Bashforth method. Compare the errors at $t = 5.0$ with the case of using progressively higher order method (example 6.6).

Solution: We observe that the order of LTE of Euler Forward method is 2 and that of 3rd order AB is 4. Therefore, we will require two time step refinements in order to obtain 4th order approximation using Richardson's extrapolation from Euler Forward solution. If one uses GTE, the improvement of orders is from 1st to 3rd but one would still require two time steps refinements. It is the improvement in orders that matters,

not the absolute magnitude of the orders. The absolute magnitude of the order only determines the extrapolation formulae. Since, the primary time step h is 0.5, we will use 0.25 and 0.125 as the subsequent refined time steps of $h/2$ and $h/4$ respectively.

We shall use the following relation ships to obtain 3rd order accurate estimate of y_{n+1}:

$$\tilde{y}_{n+1}^{h,h/2} = \frac{4\tilde{y}_{n+1}^{h/2} - \tilde{y}_{n+1}^{h}}{3}$$

$$\tilde{y}_{n+1}^{h/2,h/4} = \frac{4\tilde{y}_{n+1}^{h/4} - \tilde{y}_{n+1}^{h/2}}{3}$$

If we use *GTE*, the formulae will be $\tilde{y}_{n+1}^{h,h/2} = 2\tilde{y}_{n+1}^{h/2} - \tilde{y}_{n+1}^{h}$ and $\tilde{y}_{n+1}^{h/2,h/4} = 2\tilde{y}_{n+1}^{h/4} - \tilde{y}_{n+1}^{h/2}$.

From the above two estimates, we can formulate the following for the 4th order estimate (recall the *LTE* of 3rd order AB is of order 4):

$$\tilde{y}_{n+1}^{h,h/2,h/4} = \frac{8\tilde{y}_{n+1}^{h/2,h/4} - \tilde{y}_{n+1}^{h,h/2}}{7}$$

If we use the *GTE*, the formula will be $\tilde{y}_{n+1}^{h,h/2,h/4} = \frac{4\tilde{y}_{n+1}^{h/2,h/4} - \tilde{y}_{n+1}^{h,h/2}}{3}$.

To start 3rd order AB, we require solutions at $t = 0.5$ and t = 1.0 to be evaluated using the Euler forward and Richardson's extrapolation.

Let us first see the application of passive extrapolation. We solve the equation using $h = 0.5$, 0.25 and 0.125 using Euler forward with the given initial condition $y(0) = 1$. Then compute higher order estimates of y at $t = 0.5$ and t = 1.0 using the values at those time steps in the three expressions shown above. The computation is tabulated below:

t	$y^{(0.5)}$	$y^{(0.25)}$	$y^{(0.125)}$	$y^{(0.5,0.25)}$	$y^{(0.25,0.125)}$	$y^{(0.5,0.25,0.125)}$
0	1	1	1			
0,125			0,9250			
0,25		0,8500	0,8556			
0,375			0,7915			
0,5	0,7000	0,7225	0,7321	0,7300	0,7353	0,7360
0,625			0,6772			
0,75		0,6141	0,6264			
0,875			0,5794			
1	0,4900	0,5220	0,5360	0,5326	0,5406	0,5417

The computation using active extrapolation has to be done in two steps. First step to t = 0.5 is same in active and passive extrapolation and therefore the values from the above table will be used. For the second time step to t = 0.5, the Euler Forward has to be applied using the extrapolated value at 0.5 as initial condition. This is shown in the table below:

t	$y^{(0.5)}$	$y^{(0.25)}$	$y^{(0.125)}$	$y^{(0.5,0.25)}$	$y^{(0.25,0.125)}$	$y^{(0.5,0.25,0.125)}$

0,5	0,7360	0,7360	0,7360			
0,625			0,6808			
0,75		0,6256	0,6298			
0,875			0,5825			
1	0,5152	0,5318	0,5389	0,5373	0,5412	0,5418

Using these extrapolated values, calculation of 3rd order Adams Basforth method is shown in the Table below. The identifications are:

$y^{(True)}$**:** True Solution
$y^{(1)}$**:** Solution obtained using progressively higher order explicit methods similar to example 6.6.

$y^{(2)}$**:** Solution obtained using passively extrapolated values at $t = 0.5$ and 1.0.
$y^{(3)}$**:** Solution obtained using actively extrapolated values at $t = 0.5$ and 1.0.

t	$y^{(True)}$	$y^{(1)}$	error(%)	$y^{(2)}$	error(%)	$y^{(3)}$	error(%)
0	1,0000	1,0000		1,0000		1,0000	
0,5	0,7408	0,7000	5,51	0,7360	0,64	0,7360	0,64
1	0,5488	0,5350	2,52	0,5417	1,29	0,5418	1,28
1,5	0,4066	0,3824	5,95	0,3997	1,70	0,3997	1,70
2	0,3012	0,2890	4,05	0,2945	2,21	0,2946	2,20
2,5	0,2231	0,2089	6,38	0,2173	2,60	0,2173	2,60
3	0,1653	0,1566	5,27	0,1602	3,07	0,1602	3,06
3,5	0,1225	0,1140	6,92	0,1182	3,47	0,1182	3,47
4	0,0907	0,0850	6,34	0,0872	3,92	0,0872	3,91
4,5	0,0672	0,0621	7,55	0,0643	4,32	0,0643	4,32
5	0,0498	0,0461	7,32	0,0474	4,76	0,0474	4,76

When we compare the results at $t = 5.0$, it has decreased from 7.32% to 4.76% due to application of Richardson's extrapolation. This brings it much closer to what was obtained (2.99%) using true solution for the first 3 time steps in example 6.1. There is not much difference in actively and passively extrapolated values in this case because the difference was only for one time step.

Use of Runge Kutta Method

In order to avoid the extra error incurred by lowering the time step, one may use the same order Runge-Kutta method for the initial time steps. However, one need to pay attention to the time step size for stability. For example, if one wants to apply the 4th order Adam's Moulton method for solving an ODE and 4th order Runge-Kutta method for start-up, the time step chosen should be small enough such that the R-K method is stable even though the AM method being an implicit one may afford a larger time step. Example of such application is shown below:

Example 6.8: Solve the problem of example 6.1 using 4th order Adams Moulton method. Use 4th order Runge Kutta method for startup.

Solution: The iteration formulae for the 4th order Runge Kutta method is shown in example 6.3 and that for 4th order Adams Moulton method in example 6.1. We tabulate the calculated values below:

Time (t)	Method Used	True Solution	Numerical Solution	Error (%)
0	1,0000		1,0000	
0,5	0,7408	RK4	0,7408	0,00
1	0,5488	RK4	0,5488	0,01
1,5	0,4066	AM4	0,4066	0,00
2	0,3012		0,3012	0,01
2,5	0,2231		0,2231	0,02
3	0,1653		0,1653	0,03
3,5	0,1225		0,1224	0,04
4	0,0907		0,0907	0,04
4,5	0,0672		0,0672	0,05
5	0,0498		0,0498	0,06

In this case, the 4th order Runge Kutta gives the best starting option resulting in minimum errors. However, this may not be the case if the time step is outside the region of absolute stability for the 4th order Runge Kutta method.

6.4.2 Combination of Mehods: Predictor Corrector Schemes

One of the problem encountered in the application of any implicit method is the necessity of solving non-linear equations at each time step whenever the right hand side function f is a non-linear function of the dependent variable. This becomes necessary because one of the function evaluation is at the time step n+1. This problem does not occur in the explicit methods. A group of methods combine explicit and implicit methods in such a way so as to utilize the computational simplicity of the explicit method as well as nice stability property of the implicit method. Basic ideas of these methods are as follows:

- *Predict* the value of y_{n+1} using an explicit method.
- *Correct* the value of y_{n+1} using the implicit method. For this, the predicted value of y_{n+1} is used to evaluate f_{n+1}. This way, the solution of non-linear equation arising due to evaluation of f at n+1 is avoided.

One of the most commonly used single step predictor corrector method is Heun's method which can be written as follows:

Predictor: $y_{n+1}^{p} = y_n + hf\left(y_n, t_n\right)$ (6.107)

Corrector: $y_{n+1}^{c} = y_n + \frac{h}{2}\left[f\left(y_{n+1}^{p}, t_{n+1}\right) + f\left(y_n, t_n\right)\right]$ (6.108)

You may Compare the above with the 2nd order Runge Kutta methods (Table 6.6) and convince yourself about their similarity. Any combination of linear multi-step explicit and implicit methods of same order (Table 6.3) can be used as predictor and corrector, respectively. This essentially means that any combination of Adams Bashforth and Adams Moulton methods of same order can serve as a set of predictor and corrector, respectively. They are often referred as Adams method.

There are two options for the application of corrector on a predicted value: *single* and *iterative*. In the *single* corrector application, the corrector equation is applied only once on the predicted value. However, more commonly, the corrector equation is applied repeatedly (*iteratively*) until the value stop changing or the approximate relative error satisfies the preset error criterion. During each corrector application, the previously corrected value is used as the predicted value. The final truncation error and stability property of a predictor-corrector method is same as the corrector formula provided the time step used for the predictor formula is within its stability limit.

Example 6.9: Solve the IVP $\frac{dy}{dt} = t^2 y - 2y;\ y(0) = 1;\ t \in (0,0.5)$ using (a) Heun's method without single corrector application and, (b) Heun's method with iteration with stopping criterion of 1% approximate relative error). Use $h = 0.25$.

Solution:

(a) Heun's method with single corrector application:

Application of Heun's method to the IVP leads to the following equations:

Predictor: $y_{n+1}^p = y_n + h\left(t_n^2 y_n - 2y_n\right)$

Corrector: $y_{n+1}^c = y_n + \frac{h}{2}\left[\left(t_{n+1}^2 y_{n+1}^p - 2y_{n+1}^p\right) + \left(t_n^2 y_n - 2y_n\right)\right]$

Computations are tabulated below:

t_n	y_n	f_n	y_{n+1}^p	f_{n+1}^p	y_{n+1}^c
0	1	-2	0.5	-0,9688	0,6289
0.25	0,6289	-1,2185	0,3243	-0,5675	0,4057
0.5	0,4057				

(b) Heun's method with iterative corrector application:

Application of Heun's method to the IVP leads to the following predictor corrector equations. The index *i* indicates the iterative sequence of corrector application:

Predictor: $y_{n+1}^p = y_n + h\left(t_n^2 y_n - 2y_n\right)$

Corrector: $y_{n+1}^{c,i+1} = y_n + \frac{h}{2}\left[\left(t_{n+1}^2 y_{n+1}^{c,i} - 2y_{n+1}^{c,i}\right) + \left(t_n^2 y_n - 2y_n\right)\right];\ y_{n+1}^{c,0} = y_{n+1}^p$

The computations are shown below. For the first time step:

t_n	y_n	f_n	i	$y_{n+1}^{c,i}$	$f_{n+1}^{c,i}$	$y_{n+1}^{c,i+1}$	ε (%)
0,0000	1,0000	-2,0000	0	0,5000	-0,9688	0,6289	
0,0000	1,0000	-2,0000	1	0,6289	-1,2185	0,5977	5,2234
0,0000	1,0000	-2,0000	2	0,5977	-1,1580	0,6052	1,2492
0,0000	1,0000	-2,0000	3	0,6052	-1,1727	0,6034	0,3035
0,2500	0,6034	-1,1691	0	0,3111	-0,5445	0,3892	20,0606
0,2500	0,6034	-1,1691	1	0,3892	-0,6811	0,3721	4,5897

0,2500	0,6034	-1,1691	2	0,3721	-0,6512	0,3759	0,9940
0,5	0,3759						

For the multi-step predictor corrector methods, we will come across the start-up problem once again. Solutions are also similar to the once discussed earlier. Since, a predictor corrector scheme has the same order of accuracy as that of the corrector equation, one can improve that accuracy by the application of Richardson's extrapolation. Alternatively, one also use gradually increasing orders of predictor corrector method or Runge-Kutta method.

All the methods that we used to solve the single initial value problems are equally applicable for a system of IVPs that often need to be solved simultaneously in many a engineering applications. In the next section, we will illustrate these applications and discuss the special problems that occur in such applications.

Exercise 6.4

1. Solve the differential equation $dy/dt = -y + e^{-t}$ with the initial condition $y(0)=1$ using the following methods with $h = 0.1$ for t in (0,1):

(a) 4^{th} order Adams Bashforth with Trapezoidal method and Richardson's extrapolation for startup.

(b) Predictor Corrector method with 4^{th} order Adams methods. Use 4^{th} order Runge Kutta for startup.

2. A physical phenomenon is governed by the differential equation:

$$\frac{dv}{dt} = -0.2v - 2v^2\cos(2t) \text{ subject to the initial condition } v(0) = 1.$$

(a) Using 4^{th} order Adams-Bashforth and Adams-Moulton Predictor-Corrector method, compute v at $t = 1$ with a time step of 0.1. Use Heun's method along with Richardson's extrapolation for start-up. Use the corrector iteratively for both the methods.

(b) Using Heun's method with iterative application of corrector, solve the system with $h = 0.05, 0.1, 0.2, 0.25, 0.5$. Numerically determine the order of the method with error at t = 1

6.5 Higher Order and System of IVPs

Many engineering applications involve simultaneous solution of a set of ordinary differential equations of first order. These typically occur when one analyzes some property of a network such as flow, pressure, current, concentration, etc. However, these can also occur from an ODE of higher order as we shall see in the next section.

To illustrate such an application, let us recall the example 2.6 involving interconnected tanks. In chapter 2, we solved the steady state problem. What if we are interested in the unsteady state of the system and analyze the time the system takes to reach steady state after a shock loading. In that case, we will need to solve the following set of ordinary differential equations:

$$V_1 \frac{dC_1}{dt} = QC_0 + W_1 + Q_{21}C_2 - Q_{12}C_1 - k_1 V_1 C_1$$

$$V_2 \frac{dC_2}{dt} = Q_{12}C_1 + W_2 + Q_{32}C_3 - Q_{21}C_2 - Q_{23}C_2 - k_2 V_2 C_2$$

$$V_3 \frac{dC_3}{dt} = Q_{23}C_2 + W_3 - Q_{32}C_3 - k_3 V_3 C_3 - QC_3$$

The three ODEs shown above can not be solved in isolation since, the right hand side functions are dependent on each other through the dependent variables C_1, C_2 and C_3. This set of equation can be written as follows:

$$\frac{d\mathbf{C}}{dt} = \mathbf{AC} + \mathbf{b} \tag{6.109}$$

where, $\mathbf{C} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} \left(-\frac{Q_{12}}{V_1} - k_1\right) & \frac{Q_{21}}{V_1} & 0 \\ \frac{Q_{12}}{V_2} & \left(-\frac{Q_{21}}{V_2} - \frac{Q_{23}}{V_2} - k_2\right) & \frac{Q_{32}}{V_2} \\ 0 & \frac{Q_{23}}{V_3} & \left(-\frac{Q_{32}}{V_3} - \frac{Q}{V_3} - k_3\right) \end{bmatrix}$ and

$$\mathbf{b} = \begin{bmatrix} \frac{W_1 + QC_0}{V_1} \\ \frac{W_2}{V_2} \\ \frac{W_3}{V_3} \end{bmatrix}$$

In this example, the functions on the right hand side were linear functions of the dependent variable and therefore, it was easy to express them in matrix form. However, it is easily possible that the functions are non-linear. For example, if the reaction in each tank is second order, the above set of equation becomes:

$$V_1 \frac{dC_1}{dt} = QC_0 + W_1 + Q_{21}C_2 - Q_{12}C_1 - k_1 V_1 C_1^2$$

$$V_2 \frac{dC_2}{dt} = Q_{12}C_1 + W_2 + Q_{32}C_3 - Q_{21}C_2 - Q_{23}C_2 - k_2 V_2 C_2^2$$

$$V_3 \frac{dC_3}{dt} = Q_{23}C_2 + W_3 - Q_{32}C_3 - k_3 V_3 C_3 - QC_3^2$$

It is not possible to write these equations in the matrix form anymore. Therefore, a more general way of writing such a set of equations with *m* dependent variables will be:

$$\begin{aligned}
&\frac{dy_1}{dt} = f_1(y_1, y_2 \cdots y_m, t) \\
&\frac{dy_2}{dt} = f_2(y_1, y_2 \cdots y_m, t) \\
&\cdots\cdots\cdots \\
&\frac{dy_m}{dt} = f_m(y_1, y_2 \cdots y_m, t)
\end{aligned} \tag{6.110}$$

In the vector form, they can be written as:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t) \tag{6.111}$$

where, **y** is the vector of the dependent variables and **f** is the vector containing the right hand side functions.

The higher order initial value problems can always be expressed as a system of equations. Let us consider the following example of a second order initial value problem:

$$x^2 y'' - xy' + y = \frac{1}{x} \quad x \in (1, \infty) \quad y(1) = 0 \quad y'(1) = 0$$

Let us define two variables *u* and *v* as follows:

$u = y$ and $v = y'$

Using this definition, the 2nd order initial value problem can be decomposed into a system of equation as follows:

$$\begin{aligned}
&u' = v \\
&v' = \frac{v}{x} - \frac{u}{x^2} + \frac{1}{x^3} \\
&u(1) = 0 \quad v(1) = 0
\end{aligned}$$

This is a complete system of IVPs in variables $u(x)$ and $v(x)$ with the initial conditions defined appropriately. The values of $u(x)$ will provide the required solution. An added advantage is that the gradient is calculated simultaneously as $v(x)$ which may be required in many engineering problems. The method illustrated in the above example can be generalized for an IVP or arbitrary order. Let us consider an IVP of order *m*. We shall denote the m^{th} derivative as $y^{(m)}$. A general form of such an IVP is:

$$y^{(m)} = f\left(y^{(m-1)}, y^{(m-2)}, \cdots y', y; t\right) \tag{6.112}$$

with m initial conditions as $y = a_0,\ y' = a_1, y'' = a_2, \cdots y^{(m-1)} = a_{m-1}$ where $a_0, a_1 \cdots a_{m-1}$ are constants or functions of the independent variable t. Let us first define a set of variable $\{u_1, u_2, u_3, \cdots u_m\}$ as follows:

$$u_1 = y,\ u_2 = y', u_3 = y'', \cdots u_m = y^{(m-1)} \tag{6.113}$$

Using the new variables, the m^{th} order IVP (6.112) yields the following system of 1^{st} order IVPs:

$$\begin{aligned}
&u_1' = u_2 \\
&u_2' = u_3 \\
&u_3' = u_4 \\
&\cdots\cdots\cdots\cdots \\
&u_{m-1}' = u_m \\
&u_m' = f\left(u_1, u_2, u_3, \cdots u_m; t\right)
\end{aligned} \tag{6.114}$$

Initial Conditions: $u_1^0 = a_0,\ u_2^0 = a_1, u_3^0 = a_2, \cdots u_m^0 = a_{m-1}$

The above set of equation is of the same form as the set of equation (6.110). Therefore, using this framework, an arbitrary m^{th} order initial value problem can be decomposed into a system of m first order initial value problems.

In the vector form, the equation (6.111) is of the same form as our general single variable equation given by (6.3). Therefore, all the methods that have been derived for the single variable can be directly applied to (6.111) as well. The only difference is that the scalar arithmetic operations are now replaced by the vector operations. In order to illustrate this, let us apply Euler forward method to (6.111):

$$\mathbf{y^{n+1}} = \mathbf{y^n} + h\mathbf{f^n} \tag{6.115}$$

In the expanded variable form, the above can be written as:

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \cdots \\ y_m^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^n \\ y_2^n \\ \cdots \\ y_m^n \end{bmatrix} + h \begin{bmatrix} f_1(y_1^n, y_2^n \cdots y_m^n, t^n) \\ f_2(y_1^n, y_2^n \cdots y_m^n, t^n) \\ \cdots \\ f_m(y_1^n, y_2^n \cdots y_m^n, t^n) \end{bmatrix} \tag{6.116}$$

The time step identifier in the above expressions is indicated in the superscript since the subscript is occupied by the variable identifier. This notational structure will be used throughout this section. The initial condition of these problems is known in the form of a vector as follows:

$$\mathbf{y^0} = \begin{bmatrix} y_1^0 \\ y_2^0 \\ \cdots \\ y_m^0 \end{bmatrix} \tag{6.117}$$

Using the initial condition, one can easily compute the solution vector at the later time steps through a series of vector operation. One such application is shown in Example 6.10.

Example 6.10: Solve the following IVP using Euler Forward and 4th order Runge Kutta methods:

$$f''' + \alpha f f'' + \beta\left(1 - f'^2\right) = 0;\ f(0) = 0\ \ f'(0) = 0\ \ f''(0) = 5.0\ \ x \in (0,1)$$

Show two complete time step operations using $h = 0.05$, $\alpha = 1$ and $\beta = 1$.

Solutions: Let us assume $u = f$, $v = f'$ and $w = f''$. Using these, the 3rd order IVP is transformed into the following systems of equation:

$$u' = v$$

$$v' = w$$

$$w' = -\alpha u w - \beta\left(1 - v^2\right)$$

Initial Conditions: $u(0) = 0,\ \ v(0) = 0$ and $w(0) = 5.0$

Before we start, let us denote a function $g = -\alpha u w - \beta\left(1 - v^2\right)$

(a) Application of Euler Forward method to the above set of equations provides the computational scheme as:

$$u^{n+1} = u^n + h v^n$$

$$v^{n+1} = v^n + h w^n$$

$$w^{n+1} = w^n + h g^n$$

where, $g^n = \left[-\alpha u^n w^n - \beta\left(1 - v^{n^2}\right)\right]$

We now tabulate the calculations below:

t	*u*	*v*	*w*
0,0000	0,0000	0,0000	5,0000
0,0500	0,0000	0,2500	4,9500
0,1000	0,0125	0,4975	4,9031
0,1500	0,0374	0,7427	4,8624
0,2000	0,0745	0,9858	4,8309
0,2500	0,1238	1,2273	4,8115
0,3000	0,1852	1,4679	4,8071
0,3500	0,2586	1,7083	4,8203
0,4000	0,3440	1,9493	4,8539
0,4500	0,4414	2,1920	4,9104
0,5000	0,5510	2,4375	4,9922

0,5500	0,6729	2,6871	5,1018
0,6000	0,8073	2,9422	5,2411
0,6500	0,9544	3,2042	5,4124
0,7000	1,1146	3,4749	5,6175
0,7500	1,2883	3,7557	5,8582
0,8000	1,4761	4,0486	6,1361
0,8500	1,6785	4,3554	6,4528
0,9000	1,8963	4,6781	6,8097
0,9500	2,1302	5,0186	7,2083
1,0000	2,3811	5,3790	7,6498

(b) Application of 4th order Runge Kutta method leads to the following computational scheme:

For *u*:

$$\phi_0 = v^n$$
$$\phi_1 = v^{n+\frac{1}{2}} = v^n + \tfrac{h}{2} w^n$$
$$\phi_2 = v^{n+\frac{1}{2}} = v^n + \tfrac{h}{2} w^n$$
$$\phi_3 = v^{n+1} = v^n + h w^n$$
$$u^{n+1} = u^n + h\left(\tfrac{1}{6}\phi_0 + \tfrac{1}{3}\phi_1 + \tfrac{1}{3}\phi_2 + \tfrac{1}{6}\phi_3\right)$$
or $$u^{n+1} = u^n + h\left(v^n + \tfrac{h}{2} w^n\right)$$

For *v*:

$$\phi_0 = w^n$$
$$\phi_1 = w^{n+\frac{1}{2}} = w^n + \tfrac{h}{2} g^n$$
$$\phi_2 = w^{n+\frac{1}{2}} = w^n + \tfrac{h}{2} g^n$$
$$\phi_3 = w^{n+1} = w^n + h g^n$$
$$v^{n+1} = v^n + h\left(\tfrac{1}{6}\phi_0 + \tfrac{1}{3}\phi_1 + \tfrac{1}{3}\phi_2 + \tfrac{1}{6}\phi_3\right)$$
or $$v^{n+1} = v^n + h\left(w^n + \tfrac{h}{2} g^n\right)$$

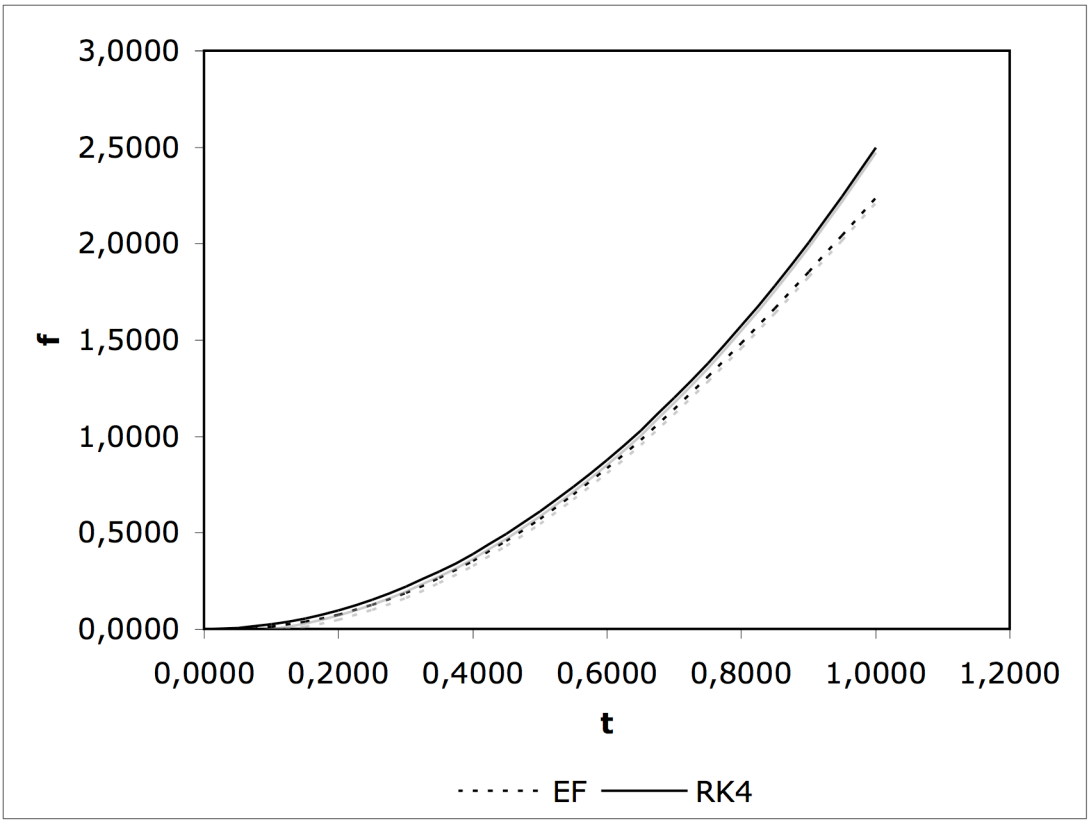
For *w*:

$$\phi_0 = -\alpha u^n w^n - \beta\left(1 - v^{n^2}\right) = g^n$$
$$\phi_1 = -\alpha\left(u^n + \tfrac{h}{2} v^n\right)\left(w_n + \tfrac{1}{2}\phi_0\right) - \beta\left[1 - \left(v^n + \tfrac{h}{2} w^n\right)^2\right]$$
$$\phi_2 = -\alpha\left(u^n + \tfrac{h}{2} v^n\right)\left(w_n + \tfrac{1}{2}\phi_1\right) - \beta\left[1 - \left(v^n + \tfrac{h}{2} w^n\right)^2\right]$$
$$\phi_3 = -\alpha\left(u^n + h v^n\right)\left(w_n + \phi_2\right) - \beta\left[1 - \left(v^n + h w^n\right)^2\right]$$
$$w_{n+1} = w_n + h\left[\tfrac{1}{6}\phi_0 + \tfrac{1}{3}\left(\phi_1 + \phi_2\right) + \tfrac{1}{6}\phi_3\right]$$

The computations are tabulated below:

t	u	v	ϕ_0	ϕ_1	ϕ_2	ϕ_3	w	g
0,0000	0,0000	0,0000					5,0000	-1,0000
0,0500	0,0063	0,2488	-1,0000	-0,9844	-0,9844	-0,9375	4,9510	-0,9691
0,1000	0,0249	0,4951	-0,9691	-0,9169	-0,9172	-0,8291	4,9055	-0,8769
0,1500	0,0558	0,7393	-0,8769	-0,7848	-0,7865	-0,6563	4,8665	-0,7249
0,2000	0,0988	0,9817	-0,7249	-0,5932	-0,5981	-0,4303	4,8370	-0,5142
0,2500	0,1539	1,2229	-0,5142	-0,3492	-0,3594	-0,1652	4,8196	-0,2464
0,3000	0,2211	1,4636	-0,2464	-0,0618	-0,0789	0,1233	4,8162	0,0771
0,3500	0,3003	1,7045	0,0771	0,2579	0,2346	0,4185	4,8285	0,4552
0,4000	0,3916	1,9465	0,4552	0,5975	0,5731	0,7040	4,8577	0,8866
0,4500	0,4950	2,1905	0,8866	0,9426	0,9303	0,9636	4,9043	1,3707
0,5000	0,6106	2,4374	1,3707	1,2775	1,3031	1,1802	4,9686	1,9070
0,5500	0,7387	2,6882	1,9070	1,5849	1,6930	1,3341	5,0502	2,4959
0,6000	0,8794	2,9438	2,4959	1,8455	2,1076	1,3983	5,1486	3,1384
0,6500	1,0330	3,2052	3,1384	2,0384	2,5626	1,3317	5,2625	3,8368
0,7000	1,1999	3,4731	3,8368	2,1403	3,0846	1,0686	5,3905	4,5946
0,7500	1,3803	3,7484	4,5946	2,1248	3,7138	0,5023	5,5303	5,4171
0,8000	1,5746	4,0317	5,4171	1,9621	4,5084	-0,5378	5,6788	6,3125
0,8500	1,7833	4,3235	6,3125	1,6170	5,5504	-2,3208	5,8315	7,2934
0,9000	2,0068	4,6242	7,2934	1,0477	6,9542	-5,2656	5,9817	8,3792
0,9500	2,2454	4,9337	8,3792	0,2012	8,8795	-10,0204	6,1194	9,6011
1,0000	2,4998	5,2517	9,6011	-0,9922	11,5544	-17,5908	6,2289	11,0097

A graphical comparison of the solution obtained by two methods is shown in the figure below:



The multi-step explicit method will experience start-up problems similar to the single variable case but the same strategies (section 6.4.1) can be applied with vector operations.

Let us now attempt to apply an implicit method to a set of equation. We will use the Euler backward method for simplicity but it will illustrate all the problems typically associated with such applications. Application of Euler backward method to (6.111) gives,

$$\mathbf{y}^{\mathbf{n+1}} = \mathbf{y}^{\mathbf{n}} + h\mathbf{f}^{\mathbf{n+1}} \tag{6.118}$$

In the expanded vector form, this looks like:

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \cdots \\ y_m^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^{n} \\ y_2^{n} \\ \cdots \\ y_m^{n} \end{bmatrix} + h \begin{bmatrix} f_1(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^n) \\ f_2(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^n) \\ \cdots \\ f_m(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^n) \end{bmatrix} \tag{6.119}$$

The above is similar to eq (3.47) where one can solve for the vector $\mathbf{y}^{\mathbf{n+1}}$ using any of the methods described in section 3.7. However, before we proceed with the solution, let us look at the cases when the functions in **f** are linear and when they are non-linear, separately. For the linear functions, we have seen (6.109) that the set of equations can be expressed as:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{y} + \mathbf{b} \tag{6.120}$$

Application of Euler backward method results in,

$$\mathbf{y}^{\mathbf{n+1}} = \mathbf{y}^{\mathbf{n}} + h\mathbf{A}\mathbf{y}^{\mathbf{n+1}} + h\mathbf{b}^{\mathbf{n+1}} \tag{6.121}$$

Notice that the vector **b** not only has constants but may also contain functions of the independent variable *t*. In that case, the value of *t* has to be set at *n*+1 time step. Some algebraic manipulation leads to the following matrix equation:

$$[\boldsymbol{I}\text{-}h\mathbf{A}]\mathbf{y}^{\mathbf{n+1}} = \mathbf{y}^{\mathbf{n}} + h\mathbf{b}^{\mathbf{n+1}} \tag{6.122}$$

This set of linear equation needs to be solved at every step where the coefficient matrix (***I***-*h***A**) remain constant for uniform time steps. Therefore, if a *LU decomposition* (Section 2.2.1.3) of the coefficient matrix is carried out once, solutions at all the time steps (as long as *h* remain constant) can be easily computed by solving two triangular matrix equations.

For non-linear functions, there is no option other than solving the non-linear set of equations given by (6.119). One can apply either Fixed Point Iteration or Newton-Raphson method. For faster convergence, it is a common practice to use the later. Application of Newton Raphson method to (6.119) gives,

$$\left[\mathbf{I} - h\mathbf{J}_{\mathbf{k}}^{\mathbf{n+1}}\right]\left(\mathbf{y}_{\mathbf{k+1}}^{\mathbf{n+1}} - \mathbf{y}_{\mathbf{k}}^{\mathbf{n+1}}\right) = -\mathbf{y}_{\mathbf{k}}^{\mathbf{n+1}} + h\mathbf{f}_{\mathbf{k}}^{\mathbf{n+1}} + \mathbf{y}^{\mathbf{n}} \tag{6.123}$$

where, *k* is the Newton Raphson iteration index and **J** is the Jacobian given by,

$J = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial y_2} & \cdots & \frac{\partial f_m}{\partial y_m} \end{bmatrix}$. The Jacobian has to be evaluated at $\mathbf{y_k^{n+1}} = \begin{Bmatrix} y_{1,k}^{n+1} & y_{2,k}^{n+1} & \cdots & y_{m,k}^{n+1} \end{Bmatrix}$. At each time step, the Newton Raphson iterations are started by setting $\mathbf{y_0^{n+1}} = \mathbf{y^n}$. The Jacobian is evaluated at this point. Thereafter, most common implementations do not alter the Jacobian at every NR iterations. It is recalculated only intermittently during every time step. If the Newton Raphson iteration slows down, it is an indication to recalculate the Jacobian. Near the convergence, the NR is slow even after recalculating the Jacobian. Since, the Jacobian is always calculated near the convergence at every time step, the same Jacobian can also be used for starting the iteration at the next time step if the startup condition $\mathbf{y_0^{n+1}} = \mathbf{y^n}$ is used. If the Jacobian is held constant for a few iterations, it is always more economical (with respect to computation time) to perform a *LU* decomposition. The following example illustrates one such application.

Example 6.11: Solve the problem of example 6.10 using Euler Backward method with Newton Raphson iterations for the system of equation. Use a tolerance of 0.1% on the maximum norm of approximate relative error vector. Show one complete iteration.

Solutions: The problem in the vector form can be written as:

$$\mathbf{y}' = \mathbf{f}$$

$$\mathbf{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} v \\ w \\ -\alpha uw - \beta\left(1 - v^2\right) \end{bmatrix}$$

The Jacobian for the functions is,

$$\mathbf{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\alpha w & 2\beta v & -\alpha u \end{bmatrix}$$

Applying the Euler Backward time stepping (superscript index *n*) along with Newton Raphson iterations (subscript index *k*) similar to 6.123 we obtain,

$$\begin{bmatrix} 1 & -h & 0 \\ 0 & 1 & -h \\ \alpha h w_k^{n+1} & -2\beta h v_k^{n+1} & 1 + \alpha h u_k^{n+1} \end{bmatrix} \begin{bmatrix} u_{k+1}^{n+1} - u_k^{n+1} \\ v_{k+1}^{n+1} - v_k^{n+1} \\ w_{k+1}^{n+1} - w_k^{n+1} \end{bmatrix} = \begin{bmatrix} -u_k^{n+1} + h v_k^{n+1} + u^n \\ -v_k^{n+1} + h w_k^{n+1} + v^n \\ -w_k^{n+1} - \alpha h u_k^{n+1} w_k^{n+1} - \beta h\left(1 - v_k^{n+1^2}\right) + w^n \end{bmatrix}$$

The above is in the form of **Ax** = **b** where the notations are as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & -h & 0 \\ 0 & 1 & -h \\ \alpha h w_k^{n+1} & -2\beta h v_k^{n+1} & 1+\alpha h u_k^{n+1} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} u_{k+1}^{n+1} - u_k^{n+1} \\ v_{k+1}^{n+1} - v_k^{n+1} \\ w_{k+1}^{n+1} - w_k^{n+1} \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -u_k^{n+1} + h v_k^{n+1} + u^n \\ -v_k^{n+1} + h w_k^{n+1} + v^n \\ -w_k^{n+1} - \alpha h u_k^{n+1} w_k^{n+1} - \beta h\left(1 - v_k^{n+1^2}\right) + w^n \end{bmatrix}$$

Notice that the vector **x** is the incremental improvement of the solution at every Newton Raphson iterations. To start the Newton Raphson iterations, we set $u_0^{n+1} = u^n$, $v_0^{n+1} = v^n$ and $w_0^{n+1} = w^n$. Two such iterations are shown below. We leave the other iterations for the readers to fill up.

Initial vector is given as $\mathbf{y^0} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$. With this, the initial Jacobian matrix is calculated as $\mathbf{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -5 & 0 & 0 \end{bmatrix}$.

Since, the matrix $\mathbf{A} = \boldsymbol{I}\text{-}h\mathbf{J}$, we obtain $\mathbf{A} = \begin{bmatrix} 1 & -0.05 & 0 \\ 0 & 1 & -0.05 \\ 0.25 & 0 & 1 \end{bmatrix}$. We are not going to recalculate Jacobian until the solution vector becomes stagnant i.e., difference between successive iterations is less than error tolerance. Therefore, we compute an *LU* decomposition of **A** using Doolittle's algorithme:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.25 & 0.0125 & 1 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} 1 & -0.05 & 0 \\ 0 & 1 & -0.05 \\ 0 & 0 & 1.000625 \end{bmatrix}$$

Using the initial conditions as starting values for the Newton Raphson iterations, we compute the vector $\mathbf{b} = \begin{bmatrix} 0 \\ 0.25 \\ -0.05 \end{bmatrix}$. Now we are ready to compute the improvement vector **x** using the LU decomposition with the right side vector b. We also compute the new **y** vector by adding **x** to the existing **y** the error vector containing the approximate relative errors. These are:

$$\mathbf{x} = \begin{bmatrix} 0.0124 \\ 0.2473 \\ -0.0531 \end{bmatrix}, \quad \mathbf{y}_1^1 = \begin{bmatrix} 0.0124 \\ 0.2473 \\ 4.9469 \end{bmatrix} \text{ and } \mathbf{e} = \begin{bmatrix} NA \\ NA \\ 1.06 \end{bmatrix}.$$

Since the error criterion is not satisfied, we recompute the vector **b** using the new values of **y** as:

$$\mathbf{b} = \begin{bmatrix} 0 \\ 2.78 \times 10^{-17} \\ 0.0031 \end{bmatrix}$$

Once again, using the *LU* decomposition we compute vectors **x**, **y** and **e**:

$$\mathbf{x} = \begin{bmatrix} 7.72 \times 10^{-6} \\ 0.0002 \\ 0.0031 \end{bmatrix}, \quad \mathbf{y}_2^1 = \begin{bmatrix} 0.0124 \\ 0.2475 \\ 4.95 \end{bmatrix} \text{ and } \mathbf{e} = \begin{bmatrix} 0.0625 \\ 0.0625 \\ 0.0625 \end{bmatrix}$$

Although the error criterion is satisfied we cannot exit the Newton Raphson iterations until we recompute the Jacobian and verify whether the convergence has really taken place. Or else, we have to reiterate with the new Jacobian. So, the new Jacobian was recomputed as:

$$\mathbf{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4.95 & 0.495 & -0.0124 \end{bmatrix}$$

The new **A, L** and **U** were recomputed using the new Jacobian :

$$\mathbf{A} = \begin{bmatrix} 1 & -0.05 & 0 \\ 0 & 1 & -0.05 \\ 0.2475 & -0.02475 & 1.0006 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.2475 & -0.0124 & 1 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} 1 & -0.05 & 0 \\ 0 & 1 & -0.05 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the new **y** and the initial conditions, we recomputed the vector $\mathbf{b} = \begin{bmatrix} 1.73 \times 10^{-18} \\ -2.78 \times 10^{-17} \\ 1.93 \times 10^{-6} \end{bmatrix}$.

Next we compute vectors **x**, **y** and **e** as:

$$\mathbf{x} = \begin{bmatrix} 4.83\times10^{-9} \\ 9.66\times10^{-8} \\ 1.93\times10^{-6} \end{bmatrix},\ \mathbf{y}_3^{\mathbf{1}} = \begin{bmatrix} 0.0124 \\ 0.2475 \\ 4.95 \end{bmatrix} \text{ and } \mathbf{e} = \begin{bmatrix} 3.9\times10^{-5} \\ 3.9\times10^{-5} \\ 3.9\times10^{-5} \end{bmatrix}$$

Since the error satisfies the specified criterion, we terminate the Newton Raphson iterations and set $\mathbf{y^1} = \begin{bmatrix} 0.0124 \\ 0.2475 \\ 4.95 \end{bmatrix}$. We proceed similarly to compute $\mathbf{y^2}$ and so on. Notice that the last computed Jacobian could be used as the starting Jacobian for the next iteration.

The problem of startup of the multi-step method also occurs in this case and are dealt with similar to single equations (section 6.3.1). The predictor corrector method can save significant amount of computation time for the system of IVPs. In a typical application, the explicit method is used for the prediction of $\mathbf{y}^{n+1}$ and the predicted vector ($\mathbf{y_p^{n+1}}$) is used as a starting point for the Newton Raphson iteration for implicit methods, *i.e.,* $\mathbf{y_0^{n+1}} = \mathbf{y_p^{n+1}}$.

We have seen the application of the methods to systems of IVPs. However, we also need to understand how the error analyses shown for single variable IVPs translate to multiple equations. Intuitively, it would be the easy to visualize that the order of truncation error of any method would remain the same. In the next section, we will show the stability analysis for a system of IVPs and in the process, introduce the concept of *Stiff Equations*.

6.5.1 Stability of a System of IVPs and Stiff Systems

First we need a model problem for the analyses of stability for a system of IVPs. Following the same logic (section 6.3.2) as of single variable IVP, we formulate a system of IVPs where the left hand side consists of the derivatives of dependent variables and the right hand side functions are **linear** combinations of all the dependent variables. We neglect all terms consisting of functions of independent variable and constants. We also do not consider the non-linear terms of the dependent variables. This can be expressed as follows:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{y} \tag{6.124}$$

Application of Euler forward method to the model problem yields,

$$\mathbf{y^{n+1}} = \left[\mathbf{I} + h\mathbf{A}\right]\mathbf{y^n} \text{ or } \mathbf{y^{n+1}} = \left[\mathbf{I} + h\mathbf{A}\right]^{\mathbf{n+1}}\mathbf{y}^0 \tag{6.125}$$

For the above iteration to be stable and eventually converge, the limit $\underset{n \to \infty}{Lt} \left[\mathbf{I} + h\mathbf{A}\right]^{n+1}$ must approach zero. Following the logic of section 2.3.2, this would mean that all the eigenvalues of the matrix are less than unity. This is automatically satisfied if the spectral radius or the largest eigenvalue of the matrix [**I** + *h***A**] is less than unity.

Alternatively, taking the norms of the vectors and the matrix, one can define an amplification factor similar to the single IVP:

$$\left|\mathbf{y}^{n+1}\right| \le \left\|\mathbf{I} + h\mathbf{A}\right\| \left|\mathbf{y}^{n}\right| \text{ or } \sigma = \frac{\left|\mathbf{y}^{n+1}\right|}{\left|\mathbf{y}^{n}\right|} \le \left\|\mathbf{I} + h\mathbf{A}\right\| \tag{6.126}$$

The amplification factor according to the definition of stability must be less than unity. Since $\sigma \le \left\|\mathbf{I} + h\mathbf{A}\right\|$, using the relation between the spectral radius and matrix norms (Gelfund's formula, eq. 2.38), it is possible to write a necessary condition for convergence as follows:

$$\rho\left(\mathbf{I} + h\mathbf{A}\right) \le 1 \tag{6.127}$$

If the maximum eigenvalue of the matrix **A** is λ_{max}, the above relation translates to:

$$\left|1 + h\lambda_{\max}\right| \le 1 \tag{6.128}$$

Since, the eigenvalue can be real or imaginary, the above relation is similar to (6.64) and defines a similar stability region. To illustrate the difficulty with stiff system, let us assume that the λ_{max} is real. In that case, the stability condition becomes:

$$h \le \frac{2}{\lambda_{\max}} \tag{6.129}$$

It is easy to see that the time step for the whole system of equation is restricted by the λ_{max}. In many engineering systems, this may happen due to one or two equations being too restrictive while rest of the equations allows much larger time steps. Such a system is typically characterized by a large ($\lambda_{max}/\lambda_{min}$) ratio and is called a *stiff system*. Although there is no universally accepted limit that can be placed on this ratio, in general, a system may be stable if $\frac{\lambda_{\max}}{\lambda_{\min}} > 100$. For a general problem where the right hand side functions are non-linear, a first order approximation (linearization) leads to Jacobian as the equivalent of **A** matrix of the model problem. Therefore, one can estimate the largest and smallest eigenvalues of the Jacobian at the initial point to evaluate the stiffness before one sets out to solve the system. Before we proceed to discuss the solution methods for stiff systems, let us look at an example to get a visual understanding of a stiff system and the problems associated with them. We consider the following systems of equation:

$$u' = -50u$$
$$v' = -50u - 0.1v + t$$
$$u(0) = 1,\ \ v(0) = 0,\ \ t \in (0,\infty)$$

The above set of equations can be expressed in the form of 6.109. The eigenvalues of the matrix **A** are 50 and 0.1. Let us consider a problem with k_1 = 50 and k_2 = 0.1. Therefore, $\frac{\lambda_{\max}}{\lambda_{\min}} = 500$. To have a visual sense of a *stiff system*, let us plot the analytical solutions (Figure 6.20). Analytical solution of the system is $u = e^{-50t}$ and $v = 1.002e^{-50t} + 98.998e^{-0.1t} + 10t - 100$.

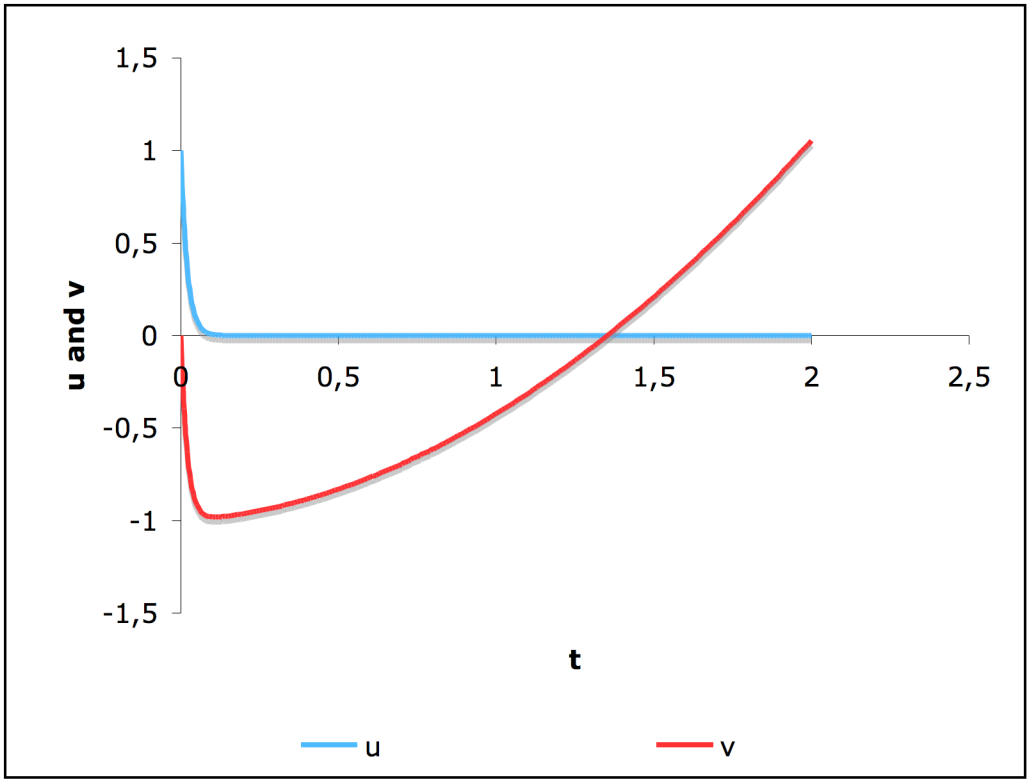


Figure 6.20: Analytical solution of a stiff system consisting of one rapidly decaying solution and another slowly growing solution.

Both u and v initially decreases rapidly. Thereafter, v increases and u decreases slowly. Essentially, there are two different time scales, one very fast and the other slow. To get an understanding of how fast the initial decrease is, let us look at a few values (Table 6.7).

Table 6.7: Values of u and v of Figure 6.20 at the initial stages.

t	*u*	*v*
0,0000	1,0000	0,0000
0,0100	0,6065	-0,3932
0,0200	0,3679	-0,6312
0,0300	0,2231	-0,7750

0,0400	0,1353	-0,8616
0,0500	0,0821	-0,9135
0,0600	0,0498	-0,9443
0,0700	0,0302	-0,9623
0,0800	0,0183	-0,9725
0,0900	0,0111	-0,9779
0,1000	0,0067	-0,9803

Within 0.05, the values of both u and v have decreased by about 92%. For rest of the time u decreases only by 8% and v increases gradually. In order to simulate the rapid initial decrease we will require a few closely spaced grid points (or small time steps) within the first 0.05 time units. Thereafter, the grid points may be sparsed or the time steps may be large.

If we want to apply Euler Forward method to solve it, maximum allowable time step throughout the time span will be given by 2/50 = 0.04. To compute a solution that is free from any oscillation, we need to use an even smaller time step. This we have observed in Example 6.4 that oscillations start to occur at time steps much below the stability limit although they damped out eventually. We will not be able to increase the time step beyond 0.04 even after the initial rapid decrease stage because of the stability limitation of the numerical method used although the actual problem will allow a larger time step after 0.05 time units.

It is clear from the above example that in a stiff system, the step size requirement is guided by one critical time scale and this severely affects the computational efficiency for the whole system. Most of the stiff system requires smaller time steps in the initial phase (when the solution has sharp gradients) but can easily use a larger time step if the numerical method allows it from the stability condition. The explicit method such as Euler Forward do not allow larger time steps even when the problem might. The solution of course would be to use methods that are capable of handling changing time steps and stable for a wide range of time step values. Cleary, all *A-stable* methods would permit this. The Euler backward and Trapezoidal methods are *A-stable* and can be used for stiff problems. Recall that the *stiffly stable* methods also allow larger time steps as their stability region is wedge shaped on the left half of the plane. The region of stability for *stiffly stable* methods expands as we move away from the origin towards $-\infty$ along the real axis. The whole group of BDF's up to order six are *stiffly stable* and are used to solve the stiff system with highest efficiency. These are also known as *Gear methods*. All of them allow changing time steps to a large extent. It is possible to start with the first order BDF with very small time step. Gradually time steps can be increased as well as higher order BDF can be used. When used judiciously, these allow increasing time steps as well as solve the startup problem of the multi-step BDFs. For example, if we start with a 1st order BDF with a time step of h, we need to use it for **at least** two time steps before we can use a 2nd order BDF with time step of $2h$. Subsequently, the second order BDF have to be used for **at least** 3 time steps before one can use a 3rd order BDF with a time step of $4h$ and the 3rd order BDF with a time step of $4h$ have to be used for **at least** 4 time steps before one can use the 4th order BDF with a time step of $8h$. In general, if one wants to increase the time step by a factor of k when going to a higher order method, the 1st order method have to be used for **at least** k times. Thereafter, each method has to be used for **at least** (m+1) times where m is the order of the method.

However, one often needs to continue with the smaller time steps for longer than the minimum required number. Essentially, the following may be used as guidelines for changing time steps:

- Have sufficient number of closely spaced grid points where the solution is changing rapidly.
- When changing to a higher order method with larger time step, make sure the values at the required past time nodes are available from the lower order computation with smaller time steps.
- The BDF are essentially weighted average approximation of values at different time nodes. When changing to higher order multi-step method, avoid using values from the nodes where the solution is changing rapidly. This may result in oscillation (not instability!). It is wise to continue with smaller time steps for some more steps so that enough values from the slowly changing region is generated for the startup of the multi-step method.

An application of BDF with change of time step and method is shown in the example 6.12.

It is of course possible to change time steps whenever the problem allows it but in that case, the BDF shown in Table 6.4 will not be applicable. Because in such a case, different time step sizes will be included in the same formulae. For example, (t_{n-3}-t_{n-2}) and (t_{n-2}-t_{n-1}) will be different if the time step was changed at t_{n-2}. Recall that the BDF in Table 6.4 was derived based on the assumption of uniform grid sizes. The BDF were formulated by approximating dy/dt on a uniform mesh involving a number of past grid points starting from n+1. We need to recompute new coefficients for the BDF for non-uniform grid sizes by obtaining a backward difference approximation of the derivative on a non-uniform grid. Computation of new coefficients has to be done at every time step. Most of the modern implementations of the *Gears method* are based on recomputation of coefficients on a non-uniform grid at every time step. We will discuss the difference approximation on a non-uniform grid in Chapter 8.

Before we can start the solution, we will need to decide on an initial time step h0 for a stiff problem to be able to clearly depict the rapidly changing part of the solution. There are several suggestions in the literature. For the stiff problems, a reasonable choice is Enright et al. (1975):

$$h_0 = \frac{1}{|\lambda_{\max}|} \tag{6.130}$$

where, $\lambda_{\max}$ is the eigenvalue of the maximum absolute magnitude of matrix **A** if the functions are linear (equation 6.120) or of the Jacobian for a general non-linear problem.

Example 6.12: Solve the following system of IVPs using Gears method or BDF up to order 4. Solve up to t = 4.0 can graphically compare the solution with the true solution.

$$u' = -50u$$
$$v' = -50u - 0.1v + t$$
$$u(0) = 1, \;\; v(0) = 0, \;\; t \in (0,\infty)$$

Solution: We will first write the computation schemes using BDF of up to 6th order for the above problem. We denote the k^{th} order BDF by BDFk:

BDF1:

$$u^{n+1} - u^n = h\left(-50u^{n+1}\right)$$
$$u^{n+1} = \frac{u^n}{(1+50h)}$$
$$v^{n+1} - v^n = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$
$$v^{n+1} = \frac{v^n + h\left[-50u^{n+1} + t^{n+1}\right]}{(1+0.1h)}$$

BDF2:

$$\tfrac{3}{2}u^{n+1} - 2u^n + \tfrac{1}{2}u^{n-1} = h\left(-50u^{n+1}\right)$$
$$u^{n+1} = \frac{2u^n - \frac{1}{2}u^{n-1}}{\left(\frac{3}{2}+50h\right)}$$
$$\tfrac{3}{2}v^{n+1} - 2v^n + \tfrac{1}{2}v^{n-1} = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$
$$v^{n+1} = \frac{2v^n - \frac{1}{2}v^{n-1} + h\left[-50u^{n+1} + t^{n+1}\right]}{\left(\frac{3}{2}+0.1h\right)}$$

BDF3:

$$\tfrac{11}{6}u^{n+1} - 3u^n + \tfrac{3}{2}u^{n-1} - \tfrac{1}{3}u^{n-2} = h\left(-50u^{n+1}\right)$$
$$u^{n+1} = \frac{3u^n - \frac{3}{2}u^{n-1} + \frac{1}{3}u^{n-2}}{\left(\frac{11}{6}+50h\right)}$$
$$\tfrac{11}{6}v^{n+1} - 3v^n + \tfrac{3}{2}v^{n-1} - \tfrac{1}{3}v^{n-2} = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$
$$v^{n+1} = \frac{3v^n - \frac{3}{2}v^{n-1} + \frac{1}{3}v^{n-2} + h\left[-50u^{n+1} + t^{n+1}\right]}{\left(\frac{11}{6}+0.1h\right)}$$

BDF4:

$$\tfrac{25}{12}u^{n+1} - 4u^n + 3u^{n-1} - \tfrac{4}{3}u^{n-2} + \tfrac{1}{4}u^{n-3} = h\left(-50u^{n+1}\right)$$

$$u^{n+1} = \frac{4u^n - 3u^{n-1} + \tfrac{4}{3}u^{n-2} - \tfrac{1}{4}u^{n-3}}{\left(\tfrac{25}{12} + 50h\right)}$$

$$\tfrac{25}{12}v^{n+1} - 4v^n + 3v^{n-1} - \tfrac{4}{3}v^{n-2} + \tfrac{1}{4}v^{n-3} = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$

$$v^{n+1} = \frac{4v^n - 3v^{n-1} + \tfrac{4}{3}v^{n-2} - \tfrac{1}{4}v^{n-3} + h\left[-50u^{n+1} + t^{n+1}\right]}{\left(\tfrac{25}{12} + 0.1h\right)}$$

BDF5:

$$\tfrac{137}{60}u^{n+1} - 5u^n + 5u^{n-1} - \tfrac{10}{3}u^{n-2} + \tfrac{5}{4}u^{n-3} - \tfrac{1}{5}u^{n-4} = h\left(-50u^{n+1}\right)$$

$$u^{n+1} = \frac{5u^n - 5u^{n-1} + \tfrac{10}{3}u^{n-2} - \tfrac{5}{4}u^{n-3} + \tfrac{1}{5}u^{n-4}}{\left(\tfrac{137}{60} + 50h\right)}$$

$$\tfrac{137}{60}v^{n+1} - 5v^n + 5v^{n-1} - \tfrac{10}{3}v^{n-2} + \tfrac{5}{4}v^{n-3} - \tfrac{1}{5}v^{n-4} = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$

$$v^{n+1} = \frac{5v^n - 5v^{n-1} + \tfrac{10}{3}v^{n-2} - \tfrac{5}{4}v^{n-3} + \tfrac{1}{5}v^{n-4} + h\left[-50u^{n+1} + t^{n+1}\right]}{\left(\tfrac{137}{60} + 0.1h\right)}$$

BDF6:

$$\tfrac{49}{20}u^{n+1} - 6u^n + \tfrac{15}{2}u^{n-1} - \tfrac{20}{3}u^{n-2} + \tfrac{15}{4}u^{n-3} - \tfrac{6}{5}u^{n-4} + \tfrac{1}{6}u^{n-4} = h\left(-50u^{n+1}\right)$$

$$u^{n+1} = \frac{6u^n - \tfrac{15}{2}u^{n-1} + \tfrac{20}{3}u^{n-2} - \tfrac{15}{4}u^{n-3} + \tfrac{6}{5}u^{n-4} - \tfrac{1}{6}u^{n-4}}{\left(\tfrac{49}{20} + 50h\right)}$$

$$\tfrac{49}{20}v^{n+1} - 6v^n + \tfrac{15}{2}v^{n-1} - \tfrac{20}{3}v^{n-2} + \tfrac{15}{4}v^{n-3} - \tfrac{6}{5}v^{n-4} + \tfrac{1}{6}v^{n-4} = h\left[-50u^{n+1} - 0.1v^{n+1} + t^{n+1}\right]$$

$$v^{n+1} = \frac{6v^n - \tfrac{15}{2}v^{n-1} + \tfrac{20}{3}v^{n-2} - \tfrac{15}{4}v^{n-3} + \tfrac{6}{5}v^{n-4} - \tfrac{1}{6}v^{n-4} + h\left[-50u^{n+1} + t^{n+1}\right]}{\left(\tfrac{49}{20} + 0.1h\right)}$$

Let us now choose an initial time step size. The set of equation can be written similar to (6.109) as,

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = -\begin{bmatrix} 50 & 0 \\ 50 & 0.1 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ t \end{bmatrix}$$

Characteristics equation for the coefficient matrix on the right is given by,

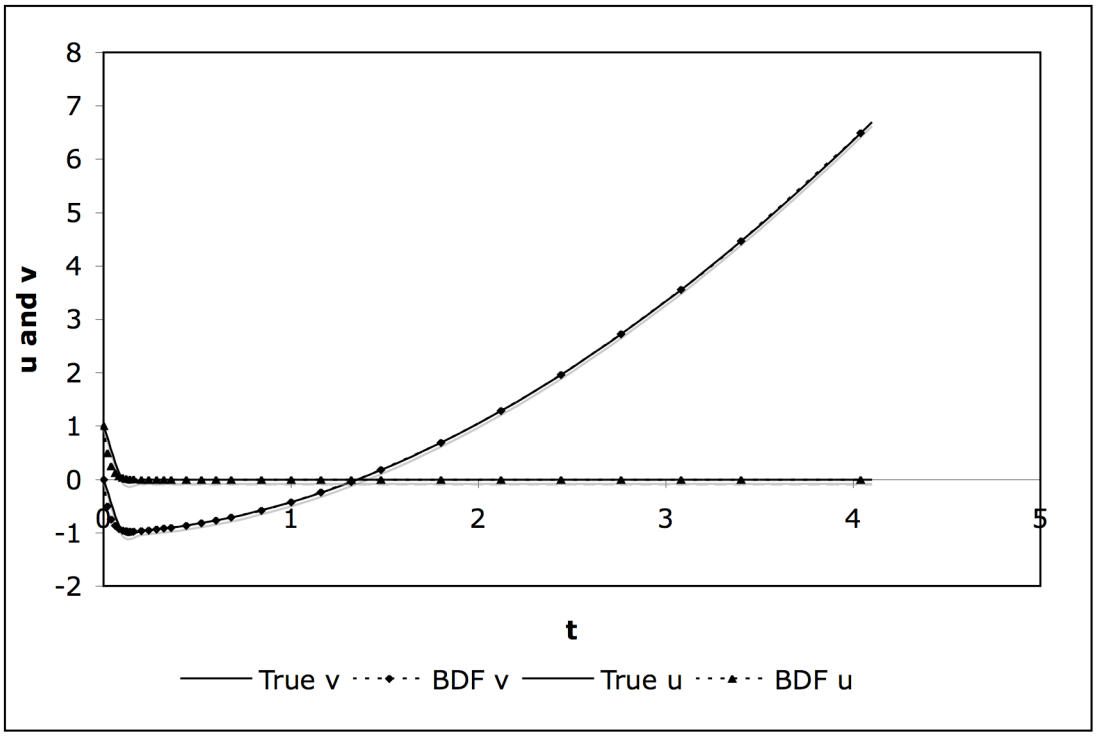
$$(50 - \lambda)(0.1 - \lambda) = 0$$

The eigenvalues are therefore, 50 and 0.1. According to equation 6.130, a suitable choice for the initial time step would be 1/50 = 0.02. Computations are shown in the table below:

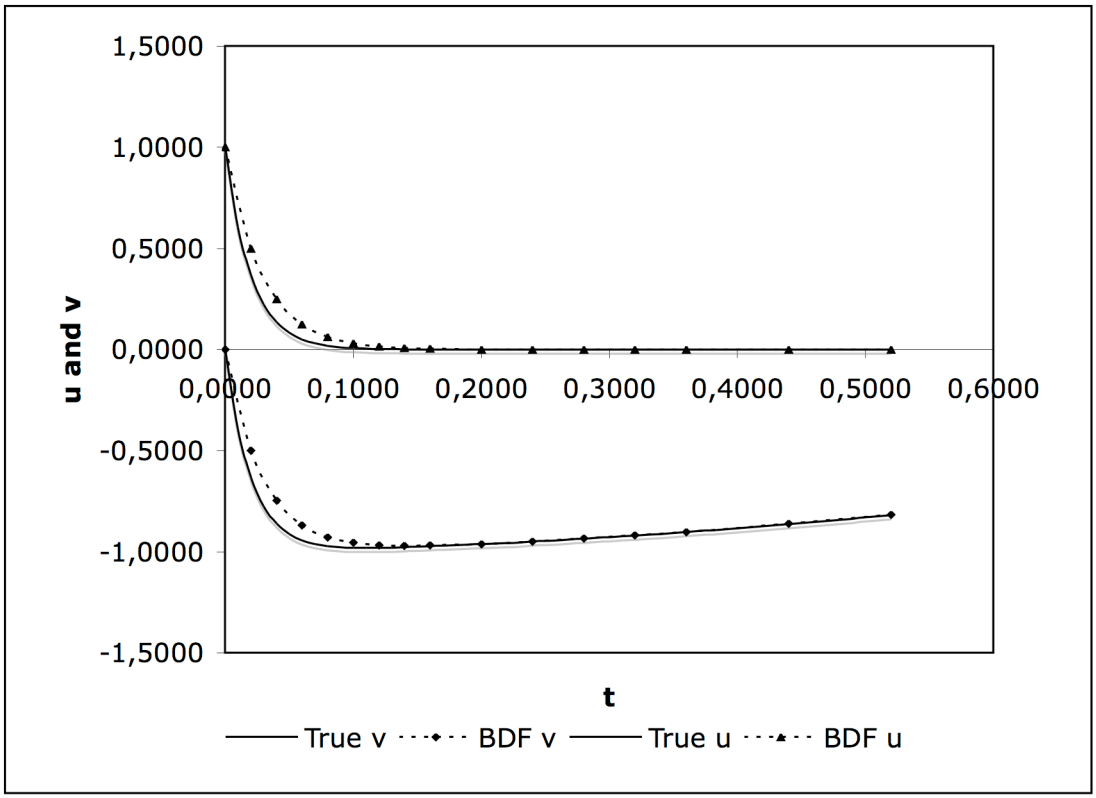
Method	***h***	***t***	***u***	***v***
		0	1,0000	0,0000
BDF1	0,02	0,02	0,5000	-0,4986

	0,02	0,04	0,2500	-0,7463
	0,02	0,06	0,1250	-0,8684
	0,02	0,08	0,0625	-0,9274
	0,02	0,1	0,0313	-0,9548
	0,02	0,12	0,0156	-0,9661
	0,02	0,14	0,0078	-0,9691
	0,02	0,16	0,0039	-0,9679
BDF2	0,04	0,2	0,00E+00	-0,9606
	0,04	0,24	-5,58E-04	-0,9485
	0,04	0,28	-3,19E-04	-0,9341
	0,04	0,32	-1,02E-04	-0,9182
	0,04	0,36	-1,30E-05	-0,9008
BDF3	0,08	0,44	7,53E-05	-0,8617
	0,08	0,52	2,39E-05	-0,8167
	0,08	0,6	-7,84E-06	-0,7656
	0,08	0,68	-5,86E-06	-0,7085
BDF4	0,16	0,84	-1,11E-05	-0,5765
	0,16	1	8,01E-07	-0,4213
	0,16	1,16	2,27E-06	-0,2431
	0,16	1,32	-6,67E-07	-0,0424
	0,16	1,48	-5,57E-07	0,1805
BDF5	0,32	1,8	-4,43E-06	0,6915
	0,32	2,12	3,76E-07	1,2873
	0,32	2,44	9,37E-07	1,9651
	0,32	2,76	-5,92E-07	2,7223
	0,32	3,08	-5,26E-08	3,5564
	0,32	3,4	2,44E-07	4,4651
BDF6	0,64	4,04	-8,34E-08	6,4966

The comparison plot with the true solution is shown below:



In order to see how it simulated the steeply varying solutions of u and v, the initial portion (0,0.5) of the solution is plotted below



In this example, BDF1 was carried out for more time steps than what is required for starting BDF2. This was required to cover the steeply changing region as well as to avoid using values from this region for starting BDF2. For the same reason, BDF2 was also carried out for more time steps than the minimum requirement of 3.

So far we have discussed ordinary differential equations where the independent variable was either time or behaved like time in the sense that the variable had an initial starting point and progressed in one direction. All the initial conditions for the higher order differential equations were specified at the starting point. This is not always practical. More often than not, conditions are known at two or more different spatial location by actual measurements or by physical location of a boundary. In the next section, we look at these higher order ODEs.

Exercise 6.5

1. The organic pollutant (L) and dissolved oxygen deficit (D) in a river are given by the Streeter-Phelps equation as follows:

$$U\frac{dL}{dx} = -k_d L$$

$$U\frac{dD}{dx} = k_d L - k_a D$$

where, $U = 10$, $k_d = 0.5$, $k_a = 0.2$, at $x = 0$, $L_o = 12$ and $D_o = 3$, and x is measured along the length of the river. Using 4th order Runge-Kutta method, compute L and D at 10 km by taking $\Delta x = 0.5$ km.

6.6 Boundary Value Problems

In this section, we will look at the ordinary differential equations of order 2 or more where the conditions are specified at more than one point. The points at which these conditions are specified are typically the solution boundaries or the ranges of the independent variable. Therefore, the independent variable is a space variable on which a grid can be laid and the conditions are specified at the two grid points at the boundaries. These are commonly known as boundary value problems (BVPs). For example, a second order general boundary value problem may look like,

$$p(x,y)\frac{d^2y}{dx^2} + q(x,y)\frac{dy}{dx} + r(x,y) = 0 \tag{6.131}$$

Boundary Conditions: $y(0) = y_0, \quad y(L) = y_L$

If p and q are only functions of x and r is a linear function of y, it will be a linear boundary value problem. For a non-linear boundary value problem, the functions p and q can also contain derivative of y. Any one of the boundary conditions may also be specified in terms of derivative of y. Boundary value problem can also be higher order. For example, the problem 6.9 may have the 3rd condition of double derivative specified at infinity.

There are two fundamentally different approaches to solving these equations. One involves decomposing the equation into a set of first order IVP and known as shooting method. The second approach involves laying a grid on the solution space and approximating the equation and the boundary conditions by finite difference approximations derived in chapter 5. The later is known as direct method. Both of these approaches rely on the methods already derived and tested in various chapters of this book.

6.6.1 Shooting Method

We have seen in section 6.5 that any higher order ODE can be decomposed into a system of 1st order ODE. Let us illustrate how we can use that to solve a boundary value problem. Consider the following 2nd order boundary value problem:

$$p(x)y'' + q(x)y' + r(x)y = s(x) \tag{6.132}$$

where, $x \in (0,l)$ and the boundary conditions are $y(0) = a$ and $y(l) = b$. Using $u = y$ and $v = y'$, we can decompose the original equation as follows:

$$u' = v$$
$$v' = -\frac{q(x)v}{p(x)} - \frac{r(x)u}{p(x)} + \frac{s(x)}{p(x)} \tag{6.133}$$
$$u(0) = a \quad v(0) = ?$$

It is clear that we require the 2nd initial condition $v(0)$ that we do not have for the solution of the set of the ODEs. Instead, we have a condition $u(l) = b$ that we can not use directly to start the solution. The principle of shooting method is to assume a $v(0)$ and estimate u(l). For an arbitrary choice of $v(0)$, the estimate of $u(l)$ is unlikely to match b (If it does, you may head for Las Vegas with some money and come back a millionaire!). However, one thing is easy to see that different choice of $v(0)$ will lead to different estimates of $u(l)$. So, one can say that $u(l)$ is a function of the independent variable $v(0)$. Therefore, the problem boils down to computing the value of the independent variable $v(0)$ for which the function $u(l)$-b is zero. This is equivalent to computing the root of the equation $u(l)$-$b = 0$. Theoretically, any method of chapter 3 can be used for this purpose. We shall use the secant method in the shooting method. Let us start with two initial guesses of the independent variables as $v_1(0)$ and $v_2(0)$ and assume the corresponding estimates as $u_1(l)$ and $u_2(l)$, respectively. One can use any of the methods described in this chapter for the solution of the system of IVPs as long as the time steps are within the limits of stability. One can compute the next guess $v_3(0)$ as follows:

$$v_3(0) = v_2(0) - u_2(l)\frac{v_2(0) - v_1(0)}{u_2(l) - u_1(l)}$$

The general iteration scheme is as follows:

$$v_{k+1}(0) = v_k(0) - u_k(l)\frac{v_k(0) - v_{k-1}(0)}{u_k(l) - u_{k-1}(l)} \tag{6.134}$$

The iteration can be terminated when the true relative error $\left|\frac{b - u_k(l)}{b}\right| \times 100$ satisfies the preset error criterion. If the condition specified in the original problem at l is a gradient condition *i.e.,* $y'(l) = b$, the method can be easily modified. In that case, the original condition $v(l) = b$ is to be matched for various values of $v(0)$. So, the dependent variable $u(l)$ is replaced by $v(l)$ in the secant method iteration scheme. An application of shooting method is shown in example 6.13.

The problem considered to illustrate the shooting method is linear and as a result, the decomposed system of IVPs are also linear. If the BVP is non-linear, the system of IVPs are also non-linear. Once an initial condition is assumed, the shooting method essentially follows the same methods described in the previous section for solving the set of equation. Once solved, the same secant method have to be applied to compute the next initial condition irrespective of whether the system is linear or non-linear.

Example 6.13: Solve the following boundary value problem using shooting method with 2nd order Runge Kutta method (Ralston's) and $h = 0.25$. Stop the secant method when the true absolute error in the boundary condition is less than 10^{-6}.

$$y'' + y + x = 0; \qquad x \in [0,1]; \qquad y(0) = y(1) = 0$$

Solution: The equation can be decomposed into the following system of two equations with definitions $u = y$ and $v = y'$,

$$u' = v$$
$$v' = -u - x$$
$$u(0) = u(1) = 0$$

Applying Ralston's method to the above system, we obtain the following numerical schemes for u and v:

For u

$$\phi_0 = v_n$$
$$\phi_1 = v_{n+\frac{3}{4}h} = v_n + \tfrac{3}{4}h\left(-u_n - x_n\right)$$
$$u_{n+1} = u_n + h\left(\tfrac{1}{3}\phi_0 + \tfrac{2}{3}\phi_1\right)$$

For v

$$\phi_0 = -u_n - x_n$$
$$\phi_1 = -u_{n+\frac{3}{4}h} - x_{n+\frac{3}{4}h} = -u_n - \tfrac{3}{4}hv_n - x_{n+\frac{3}{4}h}$$
$$v_{n+1} = v_n + h\left(\tfrac{1}{3}\phi_0 + \tfrac{2}{3}\phi_1\right)$$

Notice that the functional values at the intermediate steps were computed using Euler forward method. The initial condition for the system is $u(0) = 0$ but the $v(0)$ is not known. We need to assume two values for this initial condition for applying secant method. Let us assume 0 and 1 as the starting values. We show the computations for these initial conditions.

Initial conditions $u(0) = 0$ and $v(0) = 0$

x	ϕ_0	ϕ_1	u	ϕ_0	ϕ_1	v
0			0,0000			0,0000
0,25	0,0000	0,0000	0,0000	0,0000	-0,1875	-0,0313
0,5	-0,0313	-0,0781	-0,0156	-0,2500	-0,4316	-0,1240
0,75	-0,1240	-0,2148	-0,0618	-0,4844	-0,6486	-0,2725
1	-0,2725	-0,4015	-0,1514	-0,6882	-0,8246	-0,4673

Initial conditions $u(0) = 0$ and $v(0) = 1$

x	ϕ_0	ϕ_1	u	ϕ_0	ϕ_1	v
0			0,0000			1,0000
0,25	1,0000	1,0000	0,2500	0,0000	-0,3750	0,9375
0,5	0,9375	0,8438	0,4688	-0,5000	-0,8633	0,7520
0,75	0,7520	0,5703	0,6265	-0,9688	-1,2972	0,4550
1	0,4550	0,1969	0,6972	-1,3765	-1,6493	0,0654

Using the secant method equation 6.134, we can compute the new initial condition as

$v(0) = 0.1784$

The solution with the new initial conditions are:

x	ϕ_0	ϕ_1	u	ϕ_0	ϕ_1	v
0			0,0000			0,1784
0,25	0,1784	0,1784	0,0446	0,0000	-0,2210	0,1416
0,5	0,1416	0,0863	0,0708	-0,2946	-0,5086	0,0323
0,75	0,0323	-0,0748	0,0610	-0,5708	-0,7643	-0,1427
1	-0,1427	-0,2948	2,78E-17	-0,8110	-0,9718	-0,3722

Since, the error is less than the tolerance, we can stop. The u column contains the required solution. We will compare this solution with the direct method and true solution in example 6.14.

6.6.2 Direct Method

Philosophy of this method is approximation of the original equation using difference approximations of the derivatives derived in chapter 5. Before starting this method, two decisions have to be made based on the practical considerations. These are as follows:

- *Grid Size*: The choice is governed by how closely spaced are the points where the solution is required in the entire solution space. This can be evenly spaced (uniform) or unevenly spaced (non-uniform). Near the boundary, it is often required to use closely spaced grid points because of rapidly changing solution (high gradient). The entire solution space is divided into integer number of segments.

- *Order of Approximation*: This is governed by the accuracy of the solution desired. It is not entirely independent of the grid size. A lower order approximation with closely spaced grid points may yield the same accuracy as that of a higher order method with sparser grid points.

Once the approximation is made, the boundary problem transforms into a system of linear algebraic equation that can be solved using any of the methods described in Chapter 2. Let us outline the method using 2nd order approximation for the derivatives and a grid size of h such that $l = nh$ where, n is an integer. The grid is shown in Figure 6.21.

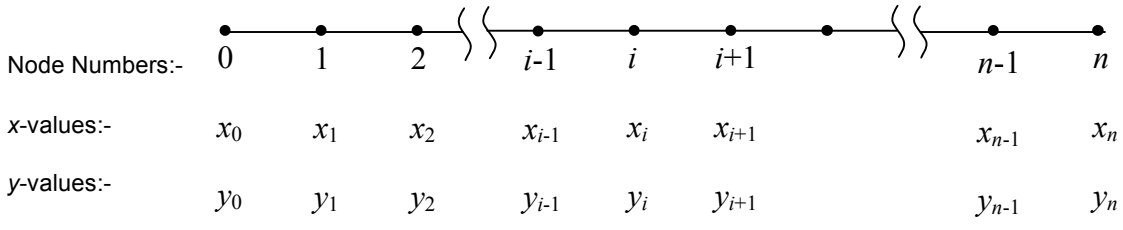


Figure 6.21: A typical grid along with nomenclatures.

The grid points are numbered as 0 to n. The independent variable values at the grid points are $\{x_0, x_1, x_2 \cdots x_n\}$ and the dependent variable values are $\{y_0, y_1, y_2 \cdots y_n\}$. Therefore, $x_i = x_0 + ih$ where, $i = 0,1,2 \cdots n$. Notice that $y_0 = a$ and $y_n = b$ are known from the boundary conditions. Therefore, the problem is to determine the values of $\{y_1, y_2 \cdots y_{n-1}\}$. For the grid point i, the following 2nd order difference approximation can be written using the approximations derived in Chapter 5 for double and single derivatives:

$$p(x_i)\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + q(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + r(x_i)y_i = s(x_i) \tag{6.135}$$

This equation can be rearranged to form:

$$\left(\frac{p(x_i)}{h^2} - \frac{q(x_i)}{2h}\right)y_{i-1} + \left(-\frac{2p(x_i)}{h^2} + r(x_i)\right)y_i + \left(\frac{p(x_i)}{h^2} + \frac{q(x_i)}{2h}\right)y_{i+1} = s(x_i) \tag{6.136}$$

For any point i, the above equation involves values at one preceding nodal point at (i-1) and at one successive nodal point at (i+1). Therefore the above equation can be written only for the interior nodes, i.e., i = 2 to n-1. This is convenient because the equation is valid at the interior nodes while at the boundary nodes, one should apply the boundary conditions. For the convenience, we denote the coefficients of (6.136) as:

$$\alpha_i = \frac{p(x_i)}{h^2} - \frac{q(x_i)}{2h}, \beta_i = -\frac{2p(x_i)}{h^2} + r(x_i) \text{ and } \gamma_i = \frac{p(x_i)}{h^2} + \frac{q(x_i)}{2h} \tag{6.137}$$

The set of equations for the interior nodes becomes,

$$\begin{aligned}
&i = 1: \quad \alpha_1 y_0 + \beta_1 y_1 + \gamma_1 y_2 = s(x_1) \\
&i = 2: \quad \alpha_2 y_1 + \beta_2 y_2 + \gamma_2 y_3 = s(x_2) \\
&\cdots\cdots\cdots \\
&i = n-1: \quad \alpha_{n-1} y_{n-2} + \beta_{n-1} y_{n-1} + \gamma_{n-1} y_n = s(x_{n-1})
\end{aligned} \tag{6.138}$$

One can now add the two boundary conditions $y_0 = a$ and $y_n = b$ to the above set of equations to yield the following matrix equation:

$$\begin{bmatrix}
\beta_1 & \gamma_1 & 0 & \bullet & 0 & 0 \\
\alpha_2 & \beta_2 & \gamma_2 & \bullet & 0 & 0 \\
0 & \alpha_3 & \beta_3 & \bullet & 0 & 0 \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
0 & 0 & 0 & \alpha_{n-2} & \beta_{n-2} & \gamma_{n-2} \\
0 & 0 & 0 & 0 & \alpha_{n-1} & \beta_{n-1}
\end{bmatrix}
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bullet \\ y_{n-2} \\ y_{n-1} \end{bmatrix} =
\begin{bmatrix} s(x_1) - a \\ s(x_2) \\ s(x_3) \\ \bullet \\ s(x_{n-2}) \\ s(x_{n-1}) - b \end{bmatrix} \tag{6.139}$$

The coefficient matrix is tridiagonal and therefore, the Thomas algorithme can be used for solution. Since, the equation considered is in the general form, the 2nd order central difference approximation of a 2nd order boundary value problem generally yields a tridiagonal matrix. The tri-diagonal structure is disrupted if one uses forward or backward difference 2nd order approximations instead of the central difference. If one expects a steep downward gradient in the solution domain, a backward difference approximation typically produces better result while a steep upward gradient calls for a forward difference approximation. This is because the central difference puts equal weights on both sides of the central node, which may lead to erroneous result when a steep gradient levels out sharply to become parallel to the axis. Let us now look at an application of direct method to solve a boundary value problem.

Example 6.14: Solve the boundary value problem of example 6.13 using direct methods with 2nd order central difference approximation and a grid size of 0.25. Compute the true relative errors at the interior node for the solutions obtained by direct method and by the shooting method in example 6.13.

Solution: With a grid size of 0.25, we will have four intervals and five nodal points in the entire domain (0,1) Let us number the nodes as 0-4. The variable values are y_0, y_1, y_2, y_3 and y_4. Out of these two values are known in terms of boundary conditions. These are $y_0 = y_4 = 0$. We have to determine the other three values.

We will now write a 2nd order central difference approximation of the differential equation for an arbitrary interior node *i*.

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{(0.25)^2} + y_i + x_i = 0$$

or $16y_{i-1} - 31y_i + 16y_{i+1} = -x_i$

The equations for node numbers 1-3 using the boundary condition values $y_0 = y_4 = 0$ are as follows:

$$\begin{aligned} -31y_1 + 16y_2 &= -0.25 \\ 16y_1 - 31y_2 + 16y_3 &= -0.5 \\ 16y_2 - 31y_3 &= -0.75 \end{aligned}$$

The tridiagonal system of equation can be solved using Thomas algorithme. The calculation is tabulated below:

l	*d*	*u*	*b*	*α*	*β*	*y*
	-31	16	-0,25	-31,0000	-0,2500	0,0443
16	-31	16	-0,5	-22,7419	-0,6290	0,0702
16	-31		-0,75	-19,7433	-1,1926	0,0604

The true solution of the equation is given by $y = \frac{\sin x}{\sin 1} - x$. The true relative errors in the shooting method and direct method are shown in the table below:

x	y	TRUE	Shooting	Error	Direct	Error
0	y_0	0	0	0	0	0
0,25	y_1	0,044013654	0,044602079	1,336913042	0,044274014	0,591542815
0,5	y_2	0,069746964	0,070791527	1,497647491	0,070155902	0,586317053
0,75	y_3	0,060056166	0,061018808	1,60290267	0,060403046	0,577592438
1	y_4	0	2,77556E-17	0	0	0

In this particular problem, direct method provides more accurate solution compared to shooting method although both central difference and Ralston's method are 2nd order accurate and both the methods were used with same h. However, this is not a universal rule that the direct method would always produce better result. It depends on the type of the problem.

One last thing we will discuss in this section is the case of derivative boundary conditions. Let us assume the right hand side boundary condition involves derivative of the dependent variable, i.e., $y'_n = b$.

The set of equation for the interior nodes (6.134) remain unaltered. To this, we add the left hand boundary condition $y_0 = a$ and a difference approximation of the right hand boundary condition $y'_n = b$. There are many ways to make the difference approximation. We list a few here.

First order backward difference

$$\frac{y_n - y_{n-1}}{h} = b \tag{6.140}$$

This preserves the tridiagonal structure of the coefficient matrix but changes (reduces) the order of approximation at the boundary node.

Second order backward difference

$$\frac{y_{n-2} - 4y_{n-1} + 3y_n}{2h} = b \tag{6.141}$$

This preserves the order of the approximation but tridiagonal structure is disturbed and the Thomas algorithme cannot be applied any more.

Ghost Node

Let us put an imaginary node (n+1) at a distance h from the node n (Figure 6.21). The n^{th} node has now become and interior node and thus the approximation of the ODE can be written for this node:

$$\alpha_n y_{n-1} + \beta_n y_n + \gamma_n y_{n+1} = s(x_n) \tag{6.142}$$

We also know that the boundary condition $y'_n = b$ is applicable at the n^{th} node. Therefore, we can write a 2nd order central difference approximation of the boundary condition,

$$\frac{y_{n+1} - y_{n-1}}{2h} = b \tag{6.143}$$

Combining the above two equations, we obtain:

$$(\alpha_n + \gamma_n) y_{n-1} + \beta_n y_n = s(x_n) - 2hb\gamma_n \tag{6.144}$$

The above equation can be added to the set of equation (6.**) to yield,

$$\begin{bmatrix} \beta_1 & \gamma_1 & 0 & \bullet & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \bullet & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \alpha_{n-1} & \beta_{n-1} & \gamma_{n-1} \\ 0 & 0 & 0 & 0 & \alpha_n + \gamma_n & \beta_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bullet \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} s(x_1) - a \\ s(x_2) \\ s(x_3) \\ \bullet \\ s(x_{n-1}) \\ s(x_n) - 2hb\gamma_n \end{bmatrix} \tag{6.145}$$

This preserves the tridiagonal structure as well as the order of approximation but increases the dimension of the matrix equation to be solved by one.

If the derivative boundary condition is at the left side boundary (at x_0), the above procedures can be modified by replacing backward difference approximations by the forward differences or assuming a ghost node to the left at x_{-1}. We illustrate the formulation of the set of equation with gradient boundary condition in example (6.15).

Example 6.15: Formulate the system of equations with 2nd order backward difference as well as with ghost node for the ODE in example 6.13 with the following boundary conditions,

$$y(0) = 0, \quad y'(1) = -0.3722$$

Solution: The unknown variable values are y_1, y_2, y_3 and y_4. We outline both backward difference and ghost node approach for formulating the equations.

2nd Order Backward Difference

The equations for nodes 1-3 are similar to example 6.14 as:

$$\begin{aligned} -31y_1 + 16y_2 &= -0.25 \\ 16y_1 - 31y_2 + 16y_3 &= -0.5 \\ 16y_2 - 31y_3 + 16y_4 &= -0.75 \end{aligned}$$

The fourth equation is obtained by writing the 2nd order backward difference approximation of the right hand boundary condition $y'(1) = -0.3722$ as:

$$y_2 - 4y_3 + 3y_4 = 2(0.25)(-0.3722) = -0.1861$$

Therefore the set of equation to solve is

$$
\begin{aligned}
-31y_1 + 16y_2 &= -0.25 \\
16y_1 - 31y_2 + 16y_3 &= -0.5 \\
16y_2 - 31y_3 + 16y_4 &= -0.75 \\
y_2 - 4y_3 + 3y_4 &= -0.1861
\end{aligned}
$$

Notice that the matrix is not a tridiagonal any more and the Thomas algorithme cannot be used. We leave it to the readers to solve it using a suitable algorithme from Chapter 2.

Ghost Node

We assume a ghost node (5) at distance 0.25 from the right hand boundary node (4). The assumed value of the variable at this node is y_5. Therefore, the set of equation for the interior nodes 1-4 can be written as:

$$
\begin{aligned}
-31y_1 + 16y_2 &= -0.25 \\
16y_1 - 31y_2 + 16y_3 &= -0.5 \\
16y_2 - 31y_3 + 16y_4 &= -0.75 \\
16y_3 - 31y_4 + 16y_5 &= -1
\end{aligned}
$$

We now write a 2nd order central difference approximation of the boundary condition at node 4:

$$
\begin{aligned}
&y_5 - y_3 = 2(0.25)(-0.3722) = -0.1861 \\
&\text{or} \quad y_5 = y_3 - 0.1861
\end{aligned}
$$

Using the value of y5 in the last equation we get the following set of equations:

$$
\begin{aligned}
-31y_1 + 16y_2 &= -0.25 \\
16y_1 - 31y_2 + 16y_3 &= -0.5 \\
16y_2 - 31y_3 + 16y_4 &= -0.75 \\
32y_3 - 31y_4 &= 1.9776
\end{aligned}
$$

We get back the tridiagonal structure of the equations. We leave it to the readers to solve both the systems of equation and verify which one provides better solution.

Exercise 6.6

1. Determine approximations of the smallest characteristic value of λ for the following problem using, 0.3333 and 0.250,

$$y'' + \lambda y = 0 \qquad y(0) = y(1) = 0$$

(a) $h = 1/2$ and solve analytically.

(b) $h = 1/3$ and solve analytically.

(c) $h = 1/5$ and use inverse power method with an initial vector of (1, 1, 1) and relative error in eigenvalue, $\varepsilon < 0.1\%$.

(d) Compare each of the three results with the true value of π^2 and comment on your results.

2. Let us consider the following differential equation:

$$\frac{d^2T}{dx^2} + a(x)\frac{dT}{dx} + b(x)T = f(x)$$

where, a(x), b(x), and f(x) are functions given by,

$$a(x) = x^2 \quad b(x) = \frac{1}{x}, \quad \text{and} \quad f(x) = e^{-x}$$

x is between (0,1). T(0) = 10 and dT/dx = 0 at x=1. Discretize the above equation using 2nd order finite difference approximation and formulate the set of linear simultaneous equations. Incorporate the boundary conditions such that the accuracy of the scheme is preserved. Use $\Delta x = 0.1$. Plot the solution.

3. Solve the differential equation $d^2y/dx^2 - dy/dx - 2y + 2x = 3$ with the boundary conditions y(0)=0 and y(1)=1 using the shooting method with (a) 4th order Runge Kutta, and (b) Heun's predictor corrector method. For both cases, use $\Delta x=0.2$.

4. Consider the problem of example 6.10. Change the 3rd boundary condition to $f''(-\infty) = 0$. Use shooting method with 4th order Runge Kutta method to solve the problem.

6.7 Summary

This chapter was intended to provide practical information for the use of the methods in engineering application. We have carefully avoided (or stayed away from) many methods found in the literature. However, most of them will fall into one of the categories presented here. In fact many options of linear multi-step methods exist. Some of them we had introduced as part of exercises (6.2, problem 7). The basic understanding developed here should help the reader to comprehend many more methods that exist for the solution of IVP but not included in this chapter. Importance was given to understand the basics of stability and different kinds of errors that creeps in various applications. We have established that a numerical method developed for a basic first order IVP is good enough to solve arbitrary order ordinary differential equation. Understanding of IVP developed here will once again form the basis for the solution of partial differential equations in the next chapter.