1

Basic Numerical Methods

1.1 Introduction

Differential equations form the backbone of various science and engineering problems viz. structural mechanics, image processing, control theory, stationary analysis of circuits, etc. Generally, engineering problems are modeled in terms of mathematical functions or using relationships between the function and its derivatives. For instance, in structural mechanics the governing equation of motion

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = f(t) \tag{1.1}$$

associated with Figure 1.1 is expressed in the form of differential equation with respect to the rate of change in time.

Here m, c, and k are mass, damping, stiffness parameters, respectively, and $f(t)$ is the external force applied on the mechanical system.

There exist various techniques for solving simple differential equations analytically. Modeling of differential equations to compute exact solutions may be found in Refs. [1–3]. But, due to complexity of problems in nature, the existing differential equations are rather cumbersome or complex (nonlinear) in nature. Generally, computation of exact or analytical solutions is quite difficult and in such cases, numerical methods [4–8] and semi-analytical methods [9] are proved to be better. In this regard, the numerical and semi-analytic techniques comprising series or closed-form solutions will be discussed in subsequent chapters. Readers interested with respect to accuracy and stability for various numerical methods may study Higham [10]. With the advancement in numerical computing, various software and programming techniques have also been developed that provide efficient platform for solving differential equations numerically viz. MATLAB, Maple, Mathematica, etc.

This chapter presents the recapitulation of basic numerical techniques for solving ordinary differential equations. Few unsolved problems have also been included at the end for self-validation of the topics. However, before we start

Advanced Numerical and Semi-Analytical Methods for Differential Equations, First Edition.
Snehashish Chakraverty, Nisha Rani Mahato, Perumandla Karunakar, and Tharasi Dilleswar Rao.

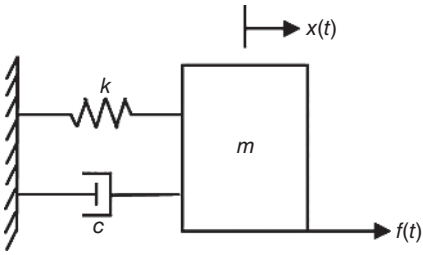

Figure 1.1 Mechanical system.

with numerical methods, we briefly recapitulate the concepts of differential equations.

A *differential equation* is a mathematical relation formulated among function, variables over which it is dependent, and its derivatives. If a function depends on a single variable, then such a differential equation is referred to as ordinary differential equation (ODE), while a function depending on multiple variables result in occurrence of partial differential equation (PDE).

1.2 Ordinary Differential Equation

The general form of an nth order ODE is given by

$$G^n(t,x) = G\left(t, x, \frac{dx}{dt}, \ldots, \frac{d^{n-1}x}{dt^{n-1}}, \frac{d^n x}{dt^n}\right) = 0 \tag{1.2}$$

subject to initial or boundary conditions. The differential equations subject to initial conditions are generally referred to as initial value problems whereas differential equations subject to boundary conditions over a domain Ω where $t \in [t_0, T]$ forms boundary value problems. A detailed discussion on existence and uniqueness of the solution of ODE may be found in Refs. [1–3].

If $G^n(t, x)$ is a linear function of x and its derivatives, then Eq. (1.2) is considered as a linear ODE, otherwise nonlinear. Generally, a nonlinear differential equation consists of product of the dependent variable x with itself or its derivatives. This chapter is evenly divided by focusing on solving ordinary differential equations using Euler, improved Euler, Runge–Kutta, and multistep methods [5, 7]. Laplace and Fourier transform methods for solving differential equations are included in Chapter 2 and numerical solutions of ODE and PDE using difference schemes over grids have been discussed in Chapter 5.

1.3 Euler Method

Let us consider a first-order ODE,

$$\frac{dx}{dt} = F(t,x) \tag{1.3}$$

subject to initial condition $x(t_0) = x_0$.

The numerical methods for solving differential equations are generally obtained initially by partitioning the independent variable $t \in [t_0, T]$ into n finite subintervals at steps $t_0, t_1, \ldots, t_{n-1}, t_n = T$ having equal step size $h = \Delta t = t_i - t_{i-1}$ for $i = 1, 2, \ldots, n-1$.

In terms of Taylor's series expansion, the solution function $x(t)$ of Eq. (1.3) is obtained using

$$\begin{aligned} x(t+\Delta t) &= x(t) + \Delta t \frac{dx}{dt} + \frac{\Delta t^2}{2!}\frac{d^2x}{dt^2} + \frac{\Delta t^3}{3!}\frac{d^3x}{dt^3} + \cdots \\ \Rightarrow x(t+h) &= x(t) + h\frac{dx}{dt} + \frac{h^2}{2!}\frac{d^2x}{dt^2} + \frac{h^3}{3!}\frac{d^3x}{dt^3} + \cdots \end{aligned} \tag{1.4}$$

Equation (1.4) reduces to Euler's equation if we retain the first-order approximation and neglect higher-order terms giving

$$\begin{aligned} x(t+h) &= x(t) + h\frac{dx}{dt} \\ \Rightarrow x(t+h) &= x(t) + hF(t,x). \end{aligned} \tag{1.5}$$

For discrete steps $t_0, t_1, \ldots, t_{n-1}$, Eq. (1.5) may be written as

$$\Rightarrow x_{i+1} = x_i + hF(t_i, x_i) \tag{1.6}$$

for $i = 0, 1, \ldots, n-1$. Here, Eq. (1.6) is known as the Euler method or Euler–Cauchy method. Geometrically, the Euler method at each step t_i results in an approximation of $x(t)|_{t=t_i} \approx x_i$ in terms of tangent to curve at t_i where $i = 0, 1, \ldots, n-1$. The geometrical representation of the Euler method is depicted in Figure 1.2.

The error at each step from Figure 1.2 is obtained using $|x(t_i) - x_i|$ for $i = 0, 1, \ldots, n$. Also, the local truncation error [5, 11] at each step is proportional to h^2 having $O(h^2)$ whereas the global error is of $O(h)$, where "O" stands for order. The algorithm for the Euler method is formulated in Algorithm 1.1.

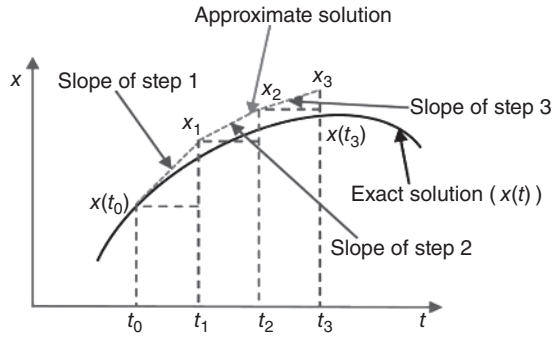

Figure 1.2 Geometrical representation of the Euler method.

Algorithm 1.1: Euler Method

Input: Differential equation, $F(t,x) = \frac{dx}{dt}$ with initial condition $x(t_0) = x_0$; Domain, $t \in [t_0, T]$; Step size, $h = \frac{T-t_0}{n}$ or Number of subintervals, $n = \frac{T-t_0}{h}$

Step (i): *for* $i = 0$ to $n-1$

$t_{i+1} = t_i + h$

$x_{i+1} = x_i + hF(t_i, x_i)$

end *for*

Step (ii): $x(t)|_{t=t_{i+1}} \approx x(t_{i+1}) = x_{i+1}$

Output: Solution $x(t)$

Example 1.1 Solve the differential equation $y' + 2xy = 0$ having initial condition $y(0) = 2$ using the Euler method having step size $h = 0.1$ in the domain $[0, 1]$.

Solution Here, $\frac{dy}{dx} = y' = -2xy = F(x, y)$, $x_0 = 0$, and $y_0 = 2$. The number of subintervals n is obtained using $n = \frac{1-0}{0.1} = 10$. Using the Euler method given in Eq. (1.6) and corresponding Algorithm 1.1, we get

$$y_{i+1} = y_i + hF(x_i, y_i) \Rightarrow y_{i+1} = y_i - 0.2x_i y_i \tag{1.7}$$

for $i = 0, 1, \ldots, n-1$. So, the approximate solutions to four decimal places at each step are given in Table 1.1. The comparison of results with exact solutions at x_i for $i = 1, 2, \ldots, 10$ has also been included in Table 1.1.

The exact solution for Example 1.1 may be obtained as $y(x) = 2e^{-x^2}$. Accordingly, the absolute error in Table 1.1 is computed using $| 2e^{-x_i^2} - y_i |$. The corresponding graphical approximation of Example 1.1 comparative to exact solution $y(x) = 2e^{-x^2}$ is depicted in Figure 1.3.

Generally, the Euler method exhibits inaccuracy with an increase in step size h whereas for small h the method provides better accuracy, but it may not be computationally efficient. As such, higher accuracy must be taken into consideration with higher terms in Eq. (1.4),

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{h^2}{2}\frac{d^2x}{dt^2} + \frac{h^3}{6}\frac{d^3x}{dt^3} + \cdots$$

Improved Euler and Runge–Kutta methods are discussed in successive sections that exhibit better accuracy compared to the Euler method.

Table 1.1 Approximate solution at x_i using the Euler method.

i	Approximate solution y_i	Exact solution $y(x_i)$	Error $\lvert y(x_i) - y_i \rvert$
1	$y_1 = y_0 - 0.2x_0y_0 = 2$	1.9801	0.0199
2	$y_2 = y_1 - 0.2x_1y_1 = 1.96$	1.9216	0.0384
3	$y_3 = y_2 - 0.2x_2y_2 = 1.8816$	1.8279	0.0537
4	$y_4 = y_3 - 0.2x_3y_3 = 1.7687$	1.7043	0.0644
5	$y_5 = y_4 - 0.2x_4y_4 = 1.6272$	1.5576	0.0696
6	$y_6 = y_5 - 0.2x_5y_5 = 1.4645$	1.3954	0.0691
7	$y_7 = y_6 - 0.2x_6y_6 = 1.2887$	1.2253	0.0635
8	$y_8 = y_7 - 0.2x_7y_7 = 1.1083$	1.0546	0.0537
9	$y_9 = y_8 - 0.2x_8y_8 = 0.931$	0.8897	0.0413
10	$y_{10} = y_9 - 0.2x_9y_9 = 0.7634$	0.7358	0.0277

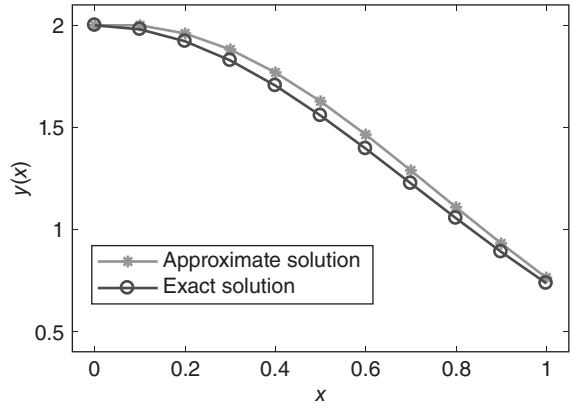

Figure 1.3 Approximate solution of $y' + 2xy = 0$, $y(0) = 2$ using the Euler method and comparison with exact solution.

1.4 Improved Euler Method

Let us again consider the first-order differential equation given in Eq. (1.3). The improved Euler method for solution of Eq. (1.3) is a two-step procedure,

$$x^*_{i+1} = x_i + hF(t_i, x_i), \tag{1.8a}$$

$$x_{i+1} = x_i + \frac{h}{2}[F(t_i, x_i) + F(t_{i+1}, x^*_{i+1})]. \tag{1.8b}$$

In few literature studies, improved Euler method is also termed as modified Euler method or Heun's method. Improved Euler method is also considered as a predictor–corrector method where Eq. (1.8a) acts as the predictor and Eq. (1.8b) acts as the corrector. Eventually, at each step we predict the function value using Eq. (1.8a) and the predicted value is then corrected using Eq. (1.8b). Corresponding algorithm for improved Euler method is formulated in Algorithm 1.2.

Algorithm 1.2: Improved Euler Method

Input: Differential equation, $F(t,x) = \frac{dx}{dt}$ with initial condition, $x(t_0) = x_0$; Domain, $t \in [t_0, T]$; Step size, $h = \frac{T-t_0}{n}$ or Number of subintervals, $n = \frac{T-t_0}{h}$

Step (i): *for* $i = 0$ to $n-1$

$$t_{i+1} = t_i + h$$
$$x^*_{i+1} = x_i + hF(t_i, x_i)$$
$$x_{i+1} = x_i + \frac{h}{2}[F(t_i, x_i) + F(t_{i+1}, x^*_{i+1})]$$

end *for*

Step (ii): $x(t)|_{t=t_{i+1}} \approx x(t_{i+1}) = x_{i+1}$

Output: Solution $x(t)$

The efficiency of Algorithm 1.2 is verified using Example 1.2.

Example 1.2 Solve a first-order differential equation

$$\frac{dy}{dt} = t^2 - y + 2t \tag{1.9}$$

with initial condition $y(0) = 1$ using improved Euler method for step size $h = 0.2$ in the [0, 1].

Solution The exact solution of Eq. (1.9) is

$$y(t) = e^{-t} + t^2. \tag{1.10}$$

Using Eq. (1.8a), we obtain the prediction

$$y^*_{i+1} = y_i + hF(t_i, y_i) = y_i + 0.2(t_i^2 - y_i + 2t_i) \tag{1.11}$$

where $i = 0, 1, \ldots, n-1$. Then, the correction is performed using

$$y_{i+1} = y_i + 0.1[t_i^2 - y_i + 2t_i + t_{i+1}^2 - y_i - 0.2(t_{i+1}^2 - y_i + 2t_{i+1}) + 2t_{i+1}].$$

Table 1.2 Approximate solution at t_i using improved Euler method.

	Approximate solution		Exact solution	Error
i	y_i^*	y_i	$y(t_i)$	$\lvert y(t_i) - y_i \rvert$
1	0.8	0.864	0.8587	0.0053
2	0.7792	0.8397	0.8303	0.0094
3	0.8637	0.9213	0.9088	0.0125
4	1.0491	1.1043	1.0893	0.015
5	1.1043	1.3847	1.3679	0.0168

The approximate value to four decimal places at each step is given in Table 1.2.

The local error [5, 11] for improved Euler method is of $O(h^3)$ whereas the global error is of $O(h^2)$. Runge–Kutta fourth-order method is discussed in next section which gives better accuracy than the previous two methods.

1.5 Runge–Kutta Methods

The Taylor series expansion in the Euler method is truncated till first derivative whereas the Runge–Kutta method retains truncated terms till second-order derivative. As such, the higher-order term improves the accuracy in Runge–Kutta compared to the Euler method. In this regard, we discuss Runge–Kutta second- (midpoint method) and fourth-order methods in Sections 1.5.1 and 1.5.2, respectively.

1.5.1 Midpoint Method

The explicit midpoint method for the first-order differential equation (1.3) is expressed as

$$x_{i+1} = x_i + hF\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}F(t_i, x_i)\right) \tag{1.12}$$

whereas the implicit midpoint method is expressed as

$$x_{i+1} = x_i + hF\left(t_i + \frac{h}{2}, \frac{1}{2}(x_i + x_{i+1})\right) \tag{1.13}$$

It is worth mentioning that Eq. (1.12) or Eq. (1.13) computes the function at midpoints $t_i + \frac{h}{2}$. As such, the Runge–Kutta second-order method is

also referred to as the midpoint method. Also, the explicit second-order Runge–Kutta method is equivalent to improved Euler method.

1.5.2 Runge–Kutta Fourth Order

The fourth-order Runge–Kutta also known as classical Runge–Kutta or RK4 for the differential equation (1.3) is given as

$$x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.14}$$

where

$$k_1 = hF(t_i, x_i),$$
$$k_2 = hF\left(t_i + \frac{h}{2}, x_i + \frac{1}{2}k_1\right),$$
$$k_3 = hF\left(t_i + \frac{h}{2}, x_i + \frac{1}{2}k_2\right)$$

and

$$k_4 = hF(t_i + h, x_i + k_3).$$

Corresponding steps for RK4 are given in Algorithm 1.3.

Algorithm 1.3: Runge–Kutta Fourth-Order Method

Input: Differential equation, $F(t,x) = \frac{dx}{dt}$ with initial condition $x(t_0) = x_0$; Domain, $t \in [t_0, T]$; Step size, $h = \frac{T-t_0}{n}$ or Number of subintervals, $n = \frac{T-t_0}{h}$

Step (i): *for* $i = 0$ to $n-1$

$$t_{i+1} = t_i + h$$
$$k_1 = hF(t_i, x_i)$$
$$k_2 = hF\left(t_i + \frac{h}{2}, x_i + \frac{1}{2}k_1\right)$$
$$k_3 = hF\left(t_i + \frac{h}{2}, x_i + \frac{1}{2}k_2\right)$$
$$k_4 = hF(t_i + h, x_i + k_3)$$
$$x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

end *for*

Step (ii): $x(t)|_{t=t_{i+1}} \approx x(t_{i+1}) = x_{i+1}$

Output: Solution $x(t)$

Example 1.3 Solve the first-order differential equation

$$\frac{dy}{dt} = y + t\sin(t) \tag{1.15}$$

with initial condition $y(0) = 1$ using the RK4 method for step size $h = 0.2$ within the domain [0, 1].

Solution The exact solution of Eq. (1.15) is

$$y(t) = \frac{1}{2}[3e^t - t\sin(t) - (t+1)\cos(t)] \tag{1.16}$$

Using Algorithm 1.3, we obtain

$$y_{i+1} = t_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$k_1 = 0.2[y_i + t_i \sin(t_i)],$$
$$k_2 = 0.2\left[x_i + \frac{k_1}{2} + (t_i + 0.1)\sin(t_i + 0.1)\right],$$
$$k_3 = 0.2\left[x_i + \frac{k_2}{2} + (t_i + 0.1)\sin(t_i + 0.1)\right]$$

and

$$k_4 = 0.2[x_i + k_3 + (t_i + h,)\sin(t_i + h)]$$

for $i = 0, 1, \ldots, n-1$. The approximate value using RK4 rounded to four decimal places is computed and incorporated into Table 1.3.

In case of RK4, the local error is of $O(h^5)$ whereas the global error is of $O(h^4)$ [5, 11]. As such, RK4 is the more efficient numerical method for solving initial value problems compared to Euler and improved Euler methods.

Table 1.3 Approximate solution at t_i using RK4 method.

i	Approximate solution y_i	Exact solution $y(t_i)$	Error $\lvert y(t_i) - y_i \rvert$
1	1.2242	1.2242	0
2	1.5151	1.5151	0
3	1.9035	1.9035	0
4	2.4243	2.4243	0
5	3.1163	3.1164	0.0001

1.6 Multistep Methods

Numerical method is referred to as single-step method when y_{i+1} is computed in a single step. The methods mentioned in Sections 1.3–1.5 are single-step methods whereas a multistep method comprises of two or more successive steps.

1.6.1 Adams–Bashforth Method

The multistep formula for Adams–Bashforth method of fourth order is given by

$$x_{i+1} = x_i + \frac{h}{24}[55F(t_i, x_i) - 59F(t_{i-1}, x_{i-1}) + 37F(t_{i-2}, x_{i-2}) - 9F(t_{i-3}, x_{i-3})]. \tag{1.17}$$

Gerald and Wheatley [5] may be referred for the derivation of Eq. (1.17). Further, the local error for the Adams–Bashforth method is of $O(h^5)$ and the global error is of $O(h^4)$.

1.6.2 Adams–Moulton Method

Adams–Moulton method is a predictor–corrector method where we may predict the value of $x(t)$ for the differential equation (1.3) using Eq. (1.17),

$$x^*_{i+1} = x_i + \frac{h}{24}[55F(t_i, x_i) - 59F(t_{i-1}, x_{i-1}) + 37F(t_{i-2}, x_{i-2}) - 9F(t_{i-3}, x_{i-3})]. \tag{1.18}$$

Then the value of $x^*(t)$ is corrected using the Adams–Moulton method. The multistep formula of the Adams–Moulton method of fourth order may be obtained as

$$x_{i+1} = x_i + \frac{h}{24}[9F(t_{i+1}, x^*_{i+1}) + 19F\left(t_i, x_i\right) - 5F\left(t_{i-1}, x_{i-1}\right) + F\left(t_{i-2}, x_{i-2}\right)] \tag{1.19}$$

for $i = 4, 5, \ldots, n-1$.

Algorithm 1.4: Adams–Moulton Fourth-Order Method

Input: Differential equation, $F(t, x) = \frac{dx}{dt}$ with initial condition, $x(t_0) = x_0$; Domain, $t \in [t_0, T]$; Step size, $h = \frac{T-t_0}{n}$ or Number of subintervals, $n = \frac{T-t_0}{h}$

Step (i): Obtain starting values using Algorithm 1.1 or 1.2 or 1.3
t_i where $i = 0, 1, 2, 3$
for $i = 4$ to $n-1$

Step (ii): Predictor (Adams–Bashforth)

$$t_{i+1} = t_i + h$$

$$x^*_{i+1} = x_i + \frac{h}{24}[55F(t_i, x_i) - 59F\left(t_{i-1}, x_{i-1}\right) + 37F\left(t_{i-2}, x_{i-2}\right) - 9F\left(t_{i-3}, x_{i-3}\right)]$$

Step (iii): Corrector

$$x_{i+1} = x_i + \frac{h}{24}[9F(t_{i+1}, x^*_{i+1}) + 19F(t_i, x_i) - 5F(t_{i-1}, x_{i-1}) + F(t_{i-2}, x_{i-2})]$$

end *for*

Step (iv): $x(t)|_{t=t_{i+1}} \approx x(t_{i+1}) = x_{i+1}$

Output: Solution $x(t)$

Example 1.4 Solve the initial value problem (IVP).

$$\frac{dy}{dx} + y(x-2) = 0, \quad y(0) = 2 \tag{1.20}$$

using the Adams–Moulton method for step size $h = 0.1$ in the domain $[0, 2]$ with starting values $y(0.2) = 2.8$, $y(0.4) = 3.808$, and $y(0.6) = 5.0266$.

Solution The exact solution of Eq. (1.20) is

$$y(t) = 2e^{-\frac{x^2}{2}+2x}. \tag{1.21}$$

Using Algorithm 1.4, the solution is computed and the values are given in Table 1.4.

Finally, the efficiencies of the above numerical methods are verified using Example 1.5.

Table 1.4 Approximate solution using the Adams–Moulton method.

	Approximate solution		Exact solution	Error
i	y^*_i	y_i	$y(x_i)$	$\lvert y(x_i) - y_i \rvert$
4	7.3164	7.0061	7.1933	0.1872
5	10.859	9.1517	8.9634	0.1883
6	9.2127	10.803	10.7311	0.0719
7	12.857	12.439	12.3437	0.0953
8	13.802	13.751	13.6419	0.4073
9	14.581	14.601	14.4855	0.1155
10	14.889	14.897	14.7781	0.1189

Example 1.5 Compute and compare the solution of given IVP

$$y' + y + t = 1, \quad y(0) = 1 \tag{1.22}$$

with respect to Euler, improved Euler, Runge–Kutta fourth order, and Adams–Moulton with step size $h = 0.2$ in the domain [0, 1].

Solution The exact solution for $y' + y + t = 1$, $y(0) = 1$ is $y(t) = 2 - t - e{-}t$. Accordingly, Eq. (1.22) is solved using Algorithms 1.1–1.4 and those are compared in Table 1.5.

Approximate solution obtained using Euler, improved Euler, Runge–Kutta fourth order, and Adams–Moulton are compared with the exact values and depicted in Figure 1.4.

Further, the absolute error $|y_i - y(t_i)|$ has been computed from Table 1.5 and the error plot is illustrated in Figure 1.5.

It may clearly be seen from Table 1.5 and Figure 1.5 that Runge–Kutta fourth order and Adams–Moulton provide better solution compared to the exact values.

Table 1.5 Approximate solution at t_i using the Euler, improved Euler, RK4, and Adams-Moulton methods.

	Approximate solution y_i				
i	**Euler**	**Improved Euler**	**RK4**	**Adams–Moulton**	**Exact solution $y(t_i)$**
1	1	0.98	0.98127	0.98127	0.98127
2	0.96	0.9276	0.92968	0.92968	0.92968
3	0.888	0.84863	0.85118	0.85118	0.85119
4	0.7904	0.74788	0.75067	0.75068	0.75067
5	0.67232	0.62926	0.63211	0.63213	0.63212

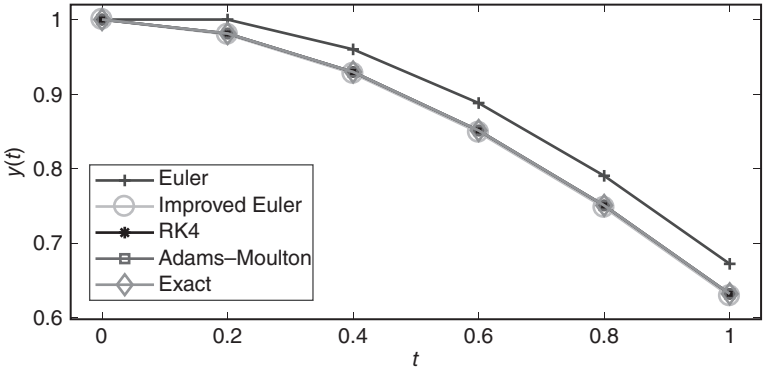

Figure 1.4 Approximate solutions of $y' + y + t = 1, y(0) = 1$.

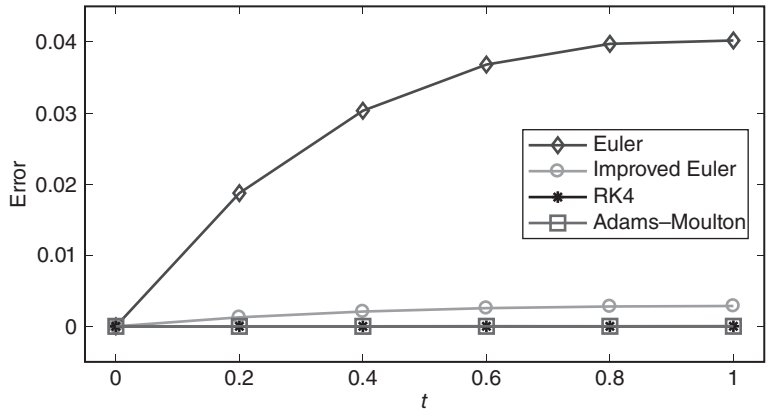

Figure 1.5 Error plot of Example 1.5 using Euler, improved Euler, RK4, and Adams–Moulton methods.

Further, the numerical techniques for first-order ODEs discussed in Sections 1.3–1.6 may be extended to higher order and coupled system of ODEs.

1.7 Higher-Order ODE

Higher-order ODEs may easily be converted to system of first-order ODEs. For instance, consider the nth order ODE given in Eq. (1.2),

$$G^n(t,x) = G\left(t, x, \frac{dx}{dt}, \ldots, \frac{d^{n-1}x}{dt^{n-1}}, \frac{d^n x}{dt^n}\right) = 0$$

subject to initial conditions $x(0) = x_0$ and $x^{(i)}(0) = x_0^{(i)}$ for $i = 1, 2, \ldots, n-1$. This nth order ODE may be transformed to coupled system of first-order ODEs,

$$\left.\begin{aligned} \frac{dy_1}{dt} &= F_1(t, y_1, y_2, \ldots, y_n) = 0 \\ \frac{dy_2}{dt} &= F_2(t, y_1, y_2, \ldots, y_n) = 0 \\ &\vdots \\ \frac{dy_n}{dt} &= F_n(t, y_1, y_2, \ldots, y_n) = 0 \end{aligned}\right\} \tag{1.23}$$

by substituting $y_1 = x$, $y_i = \frac{dy_{i-1}}{dt} = \frac{d^{i-1}x}{dt^{i-1}}$ for $i = 2, 3, \ldots, n$ and $\frac{dy_n}{dt} = \frac{d^n x}{dt^n}$.

Let us now illustrate the numerical methodology for solving higher-order ODE by converting it to system of first-order ODEs through an example problem.

Example 1.6 Solve second-order IVP

$$x'' - x' + 5x = 4 \tag{1.24}$$

having initial conditions $x(1) = 2$ and $x'(1) = -1$ using Euler and RK4 methods with step size $h = 0.2$ in the domain [1, 3].

Solution Equation (1.24) is reduced to coupled system of first-order IVPs using Eq. (1.23) as

$$\left.\begin{aligned} \frac{du}{dt} &= v, u(1) = 2 \\ \frac{dv}{dt} &= v - 5u + 4, v(1) = -1 \end{aligned}\right\} \tag{1.25}$$

where $u = x, v = \frac{dx}{dt}$, and $\frac{dv}{dt} = \frac{d^2x}{dt^2}$.

Above first-order ODEs (1.25) may be solved using methods discussed in Sections 1.3–1.6. In this regard, the Euler method based on Eq. (1.6) for solving system of differential equations transforms to

$$u_{i+1} = u_i + hF_1(t_i, u_i, v_i) \tag{1.26a}$$

and

$$v_{i+1} = v_i + hF_2(t_i, u_i, v_i) \tag{1.26b}$$

where $\frac{du}{dt} = F_1(t, u, v) = v$ and $\frac{dv}{dt} = F_2(t, u, v) = v - 5u + 4$.

Similarly, in case of RK4 the Eq. (1.14) transforms to

$$u_{i+1} = u_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.27a}$$

and

$$v_{i+1} = v_i + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \tag{1.27b}$$

where

$$k_1 = hF_1(t_i, u_i, v_i), \quad l_1 = hF_2(t_i, u_i, v_i)$$

$$k_2 = hF_1\left(t_i + \frac{h}{2}, u_i + \frac{1}{2}k_1, v_i + \frac{1}{2}l_1\right), \quad l_2 = hF_2\left(t_i + \frac{h}{2}, u_i + \frac{1}{2}k_1, v_i + \frac{1}{2}l_1\right)$$

$$k_3 = hF_1\left(t_i + \frac{h}{2}, u_i + \frac{1}{2}k_2, v_i + \frac{1}{2}l_2\right), \quad l_3 = hF_2\left(t_i + \frac{h}{2}, u_i + \frac{1}{2}k_2, v_i + \frac{1}{2}l_2\right)$$

$$k_4 = hF_1(t_i + h, u_i + k_3, v_i + l_3) \quad \text{and} \quad l_4 = hF_2(t_i + h, u_i + k_3, v_i + l_3)$$

for $\frac{du}{dt} = F_1(t, u, v)$ and $\frac{dv}{dt} = F_2(t, u, v)$. Using Eqs. (1.26a), (1.26b), (1.27a), and (1.27b), the solutions of differential equation (1.24) are obtained and incorporated into Table 1.6.

The solutions obtained in Table 1.6 using Euler and RK4 are compared with the exact solution $x(t) = e^{\frac{t-1}{2}}\left(\frac{6}{5}\cos\left(\frac{\sqrt{19}t}{2} - \frac{\sqrt{19}}{2}\right) - \frac{16\sqrt{19}}{95}\sin\left(\frac{\sqrt{19}t}{2} - \frac{\sqrt{19}}{2}\right)\right) + \frac{4}{5}$ in Figure 1.6.

Similar to Euler and RK4 techniques, other numerical methods may be extended for higher order and system of first-order ODEs.

Table 1.6 Approximate solution at t_i using Euler and RK4 methods.

		Approximate solution x_i				
		Euler		RK4		
i	t_i	u_i	v_i	u_i	v_i	Exact solution $x(t_i)$
1	1.2	1.8000	−2.4000	1.6595	−2.3931	1.6596
2	1.4	1.3200	−3.8800	1.0567	−3.5733	1.0567
3	1.6	0.5440	−5.1760	0.2647	−4.2366	0.2645
4	1.8	−0.4912	−5.9552	−0.5858	−4.1244	−0.5866
5	2	−1.6822	−5.8550	−1.3227	−3.0896	−1.3242
6	2.2	−2.8532	−4.5438	−1.7605	−1.1535	−1.7626
7	2.4	−3.7620	−1.7993	−1.7377	1.4617	−1.7401
8	2.6	−4.1219	2.4028	−1.1575	4.3368	−1.1594
9	2.8	−3.6413	7.8053	−0.0230	6.9036	−0.0236
10	3	−2.0803	13.8076	1.5416	8.5351	1.5430

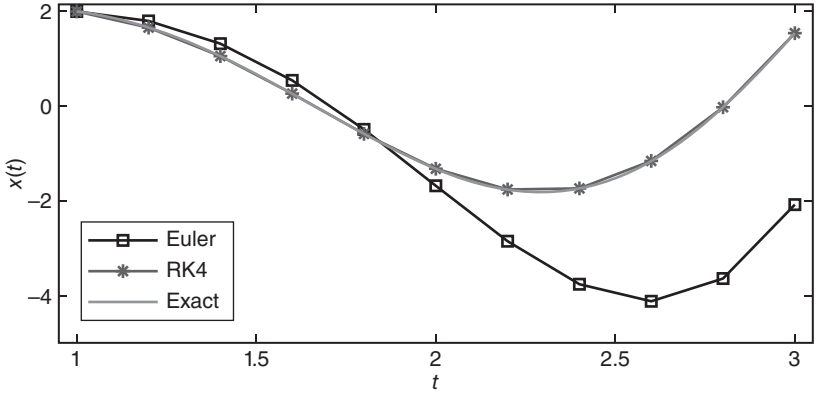

Figure 1.6 Comparison of solutions of Example 1.6 using Euler and RK4 with exact solution.

This chapter is mainly devoted to illustrate the basic ideas of handling initial value ODEs using Euler, improved Euler, RK4, and Adams–Moulton methods. Interested readers may refer the excellent books [4–8] for other numerical methods. Moreover, the methods may be extended to PDEs too.

Exercise

1 Solve the IVPs exactly and compute the error using Euler and improved Euler methods.
(a) $y' - t^2 - 1 = 0, \quad y(0) = 1, \quad h = 0.1, \quad t \in [0, 1]$
(b) $y' + 2y^2 - t + 2 = 0, \quad y(0) = 0, \quad h = 0.25, \quad t \in [0, 2]$
(c) $y' - (y - 2x)^2 - x^3 \sin(x) = 0, \quad y(0) = 2, \quad h = 0.2, \quad x \in [0, 2]$

2 Solve the initial value problem $\frac{dy}{dx} = 3x^3 + 1,\ y(0) = 1,\ h = 0.1,\ t \in [0, 1]$ using RK4 and Adams–Moulton methods.

3 Compute the solution of IVP $y'' + 8xy' + 5xy = 4, y(1) = 2, y'(1) = -1$ using Euler and RK4 for step size $h = 0.1$ in the domain $x \in [1, 2]$.

4 Evaluate $u'' - u' + u = t$ subject to initial conditions $u(0) = 1$ and $u'(0) = 0$ for step size $h = 0.05$ in the domain $[0, 1]$ using the improved Euler method.

5 Compute the system of ODEs $\frac{du}{dt} = -2u + v^2$ and $\frac{dv}{dt} = -30u + 40v$ subject to initial conditions $u(0) = 0$ and $v(0) = 1$ for step size $h = 0.01$ using Euler, improved Euler, RK4, and Adams–Moulton methods.

References

1 Zwilliger, D. (1998). *Handbook of Differential Equations*, 3e. New York: Academic Press.
2 Hartman, P. (2002). *Ordinary Differential Equations*, 2e. Philadelphia: SIAM.
3 Arnold, V.I. (2006). *Ordinary Differential Equations*, 3e. New York: Springer.
4 Atkinson, K.E. and Han, W. (1985). *Elementary Numerical Analysis*, 3e. New York: Wiley.
5 Gerald, C.F. and Wheatley, P.O. (2004). *Applied Numerical Analysis*, 7e. Boston, MA; Munich: Pearson Education India.
6 Bhat, R.B. and Chakraverty, S. (2004). *Numerical Analysis in Engineering*. Pangbourne: Narosa, Alpha Science International Limited.

7 Hoffman, J.D. (2001). *Numerical Methods for Engineers and Scientists*, 2e. New York: Taylor & Francis.
8 Milne, W.E. and Milne, W.E. (1953). *Numerical Solution of Differential Equations*. New York: Wiley.
9 Ganji, D.D. and Talarposhti, R.A. (2018). *Numerical and Analytical Solutions for Solving Nonlinear Equations in Heat Transfer*, Advances in Mechatronics and Mechanical Engineering Book Series. Hershey, PA: IGI Global.
10 Higham, N.J. (2002). *Accuracy and Stability of Numerical Algorithms*, 2e. Philadelphia: SIAM.
11 Kreyszig, E. (2017). *Advanced Engineering Mathematics*, 10e. New Delhi: Wiley.