CHAPTER 10

ODEs: BOUNDARY VALUE PROBLEMS

SOLVED EXAMPLE PROBLEMS

for

NUMERICAL METHODS FOR SCIENTISTS AND ENGINEERS With Pseudocodes

By Zekeriya ALTAÇ

April 2025

EXAMPLE 10.1: Linear two-point BVP

Consider following two-point BVP:

$$y'' + \frac{27}{(x^2+9)^2} y(x) = 0, \quad y'(0) = \frac{1}{3}, \quad y(4) = \frac{4}{5}$$

To find a numerical solution to thisproblem, (a) divide the solution interval, [0, 4], into uniform M subintervals and derive the general difference equation; (b) implement the BCs and express the difference equations in matrix form; and (c) solve the resulting linear system for $M = 10$, 20, and 40 and compare your estimates with the true solution given by $y = x/\sqrt{x^2+9}$.

SOLUTION:

Step 1. Gridding. A one-dimensional uniform grid, shown in **Figure 10.1**, is obtained by dividing the solution interval ($0 \leqslant x \leqslant 4$) into M uniformly spaced subintervals ($h = 4/M$). Since the nodal value at the right boundary is known ($y(4) = 4/5$), for the right (base) node, we may set $y_M = 4/5$. The nodal value of the left boundary is unknown due to the Neumann BC. Thus, the total number of unknowns in this example is M. The numerical approximations for the unknown function corresponding to the nodal points are labeled as $y(x_i) = y_i$ for $i = 0, 1, 2, \ldots, (M-1)$, where $x_i = ih$.

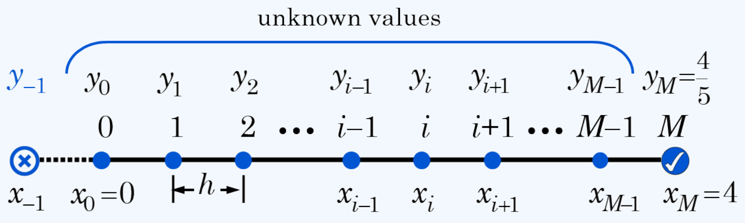


Figure 10.1

Step 2. Discretizing. The ODE is satisfied for all nodes on [0,4); hence, for any x_i, we may write

$$y_i'' + \frac{27}{(x_i^2+9)^2} y_i = 0, \quad x_i \in [0,4)$$

Substituting the CDF for y_i'' leads to

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \frac{27}{(x_i^2+9)^2} y_i = 0$$

Multiplying both sides by h^2 of the above equation and rearranging, the *general difference equation* is found as

$$y_{i-1} + d_i y_i + y_{i+1} = 0, \quad \text{for } i = 0, 1, \ldots, M-1$$

where $d_i = -2 + 27h^2/(x_i^2+9)^2$.

Step 3. Implementing BCs. The difference equations for all unknown nodes are derived from the general difference equation as follows:

$$\begin{array}{ll}
\text{For } i = 0 & \boxed{y_{-1}} + d_0 y_0 + y_1 = 0 \\
\text{For } i = 1 & y_0 + d_1 y_1 + y_2 = 0 \\
\text{For } i = 2 & y_1 + d_2 y_2 + y_3 = 0 \\
\quad\vdots & \quad\vdots \\
\text{For } i = M-2 & y_{M-3} + d_{M-2} y_{M-2} + y_{M-1} = 0 \\
\text{For } i = M-1 & y_{M-2} + d_{M-1} y_{M-1} + \boxed{y_M} = 0
\end{array}$$

The fictitious node value, y_{-1}, at the left boundary is eliminated by discretizing the left BC with the CDF as $y_0' \cong (y_1 - y_{-1})/2h = 1/3$, which gives $y_{-1} = y_1 - 2h/3$. Substituting this into the difference equation with $i = 0$ (after collecting the unknown nodes on the left) leads to

$$d_0\, y_0 + 2y_1 = 2h/3$$

At the right boundary, the nodal value of $y_M = 4/5$ is plugged into the difference equation obtained with $i = M - 1$. Likewise, rearranging this difference equation gives

$$y_{M-2} + d_{M-1}\, y_{M-1} = -4/5$$

Step 4. Constructing a Linear System. The total of M equations and M unknowns can be expressed in matrix form as

$$\begin{bmatrix} d_0 & 2 & & & & \\ 1 & d_1 & 1 & & & \\ & 1 & d_2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & d_{M-2} & 1 \\ & & & & 1 & d_{M-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{M-2} \\ y_{M-1} \end{bmatrix} = \begin{bmatrix} 2h/3 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -4/5 \end{bmatrix}$$

which can be easily programmed for any value of M.

Step 5. Solving the Linear System. The coefficient matrix of the resulting linear system of equations is in tridiagonal matrix form. The system is solved using Thomas' algorithm.

Table 10.1

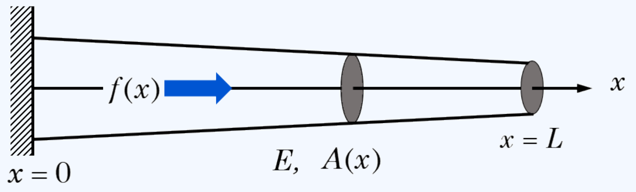
	True	$M = 10$		$M = 20$		$M = 40$	
x	Solution	Estimate	Abs. Error	Estimate	Abs. Error	Estimate	Abs. Error
0	0	0.005882	5.88×10^{-3}	0.001464	1.46×10^{-3}	0.000366	3.66×10^{-4}
0.4	0.132164	0.139058	6.89×10^{-3}	0.133877	1.71×10^{-3}	0.132591	4.28×10^{-4}
0.8	0.257663	0.265075	7.41×10^{-3}	0.259505	1.84×10^{-3}	0.258122	4.60×10^{-4}
1.2	0.371391	0.378769	7.38×10^{-3}	0.373225	1.83×10^{-3}	0.371849	4.58×10^{-4}
1.6	0.470588	0.477451	6.86×10^{-3}	0.472295	1.71×10^{-3}	0.471015	4.26×10^{-4}
2	0.554700	0.560698	6.00×10^{-3}	0.556193	1.49×10^{-3}	0.555073	3.73×10^{-4}
2.4	0.624695	0.629613	4.92×10^{-3}	0.625920	1.22×10^{-3}	0.625001	3.06×10^{-4}
2.8	0.682318	0.686042	3.72×10^{-3}	0.683246	9.28×10^{-4}	0.682550	2.32×10^{-4}
3.2	0.729537	0.732021	2.48×10^{-3}	0.730156	6.19×10^{-4}	0.729692	1.55×10^{-4}
3.6	0.768221	0.769457	1.24×10^{-3}	0.768529	3.08×10^{-4}	0.768298	7.70×10^{-5}
4	0.8	0.8		0.8		0.8	

Discussion: The true absolute errors of the numerical estimates obtained with $M = 10$, 20, and 40 and the true solution are comparatively tabulated in **Table 10.1**. Fairly good quality estimates were obtained even with a small number of panels. The average of the true errors is about 5.28×10^{-3}, 1.31×10^{-3}, and 3.28×10^{-4} for $M = 10$, 20, and 40, respectively. If we further refine the mesh (i.e., reduce h), the numerical estimate should converge to the exact solution. For a second-order central difference method, the error should behave like $\mathcal{O}(h^2)$. Notice that when M is doubled, the global error decreases by a factor of approximately four since $\mathcal{O}((h/2)^2) \approx \mathcal{O}(h^2/4)$. This can also be verified by running the method with different values of h and plotting $\log(error)$ vs. $\log(h)$ to confirm the expected slope ($\approx$2 for the second-order accurate scheme).

The coefficient matrix is usually tridiagonal and symmetric positive definite for linear BVPs, which ensures stable and efficient solution methods (e.g., Thomas algorithm). Efficiency is high for linear BVPs due to the tridiagonal structure.

EXAMPLE 10.2: Linear two-point BVP in conservative form

Consider a $L=1$-meter-long bar which is fixed at one end and subjected to a distributed axial force along its length, as shown in the figure below.



The governing equation for a bar under axial load, with varying cross-sectional area $A(x)$, Young's modulus $E(x)$, and distributed force $f(x)$, written in conservative form is:

$$\frac{d}{dx}\left(EA(x)\frac{du}{dx}\right) = f(x), \qquad 0 \leqslant x \leqslant L$$

where $u(x)$ is the axial displacement (m), $A(x) = 0.01 - 0.005\,x$ (m^2) is the cross-sectional area, $E = 200 \times 10^9$ (Pa) is the Young's modulus, $f(x) = 4000\,x\,(1-x)$ is the distributed body force per unit length (N/m). The pertinent boundary conditions can be expressed as follows:

$$\text{At fixed end, } x = 0 \quad u(0) = 0$$

$$\text{At free end, } x = L \quad \sigma(L) = A(L)E\left.\frac{du}{dx}\right|_{x=L} = 0 \quad \Rightarrow \quad u'(L) = 0.$$

Apply the FVM to find a numerical solution to this problem. (a) Divide the physical domain, [0, 1], into uniform M subintervals and derive the general difference equation; (b) implement the BCs and express the difference equations in matrix form; and (c) solve the resulting linear system for $M = 10$ and 20 and compare your estimates with the true solution given by

$$u(x) = \left(\frac{10}{3}\ln(1-0.5x) + \frac{x}{9}\left(4x^2 + 3x + 12\right)\right) \times 10^{-6}, \quad \text{(m)}$$

SOLUTION:

This ODE is in conservation form, which makes it ideal for *Finite Volume Method*. We follow the numerical solution steps in the following order:

Step 1. Gridding: Gridding is carried out by dividing the physical domain (rod length) into M uniform cells of size $h = L/M$. Nodal points are placed on cell boundaries at x_i, as shown in **Figure 10.2**. The positions of the cell boundaries and midpoints are obtained by $x_i = i\,h$ $x_{i-1/2} = x_i - h/2$ and $x_{i+1/2} = x_i + h/2$, respectively. Note that the cell boundary nodes coincide with the physical boundaries.

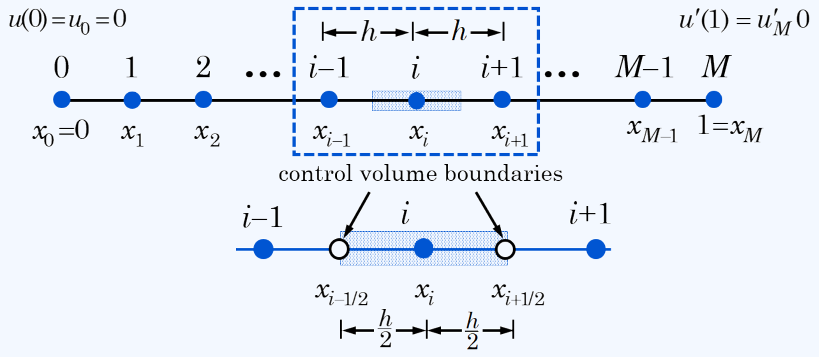


Figure 10.2: Gridding of the bar.

Step 2. Discretizing: To discretize, first the governing ODE is integrated over the finite cell volume, $x_{i-1/2} \leqslant x \leqslant x_{i+1/2}$:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{d}{dx}\left(EA(x)\frac{du}{dx}\right)dx = \int_{x_{i-1/2}}^{x_{i+1/2}} f(x)dx$$

The axial stiffness, $EA(x)$, is continuous throughout the bar; thus, the integrations over the control volume can be carried out as follows:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{d}{dx}\left(EA(x)\frac{du}{dx}\right)dx = \left(EA(x)\frac{du}{dx}\right)_{x_{i+1/2}} - \left(EA(x)\frac{du}{dx}\right)_{x_{i-1/2}}$$

Evaluating the axial stiffness, $EA(x)$, and replacing the derivatives at $x_{i-1/2}$ and $x_{i+1/2}$ with the CDFs yields

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{d}{dx}\left(EA(x)\frac{du}{dx}\right)dx \cong EA_{i+1/2}\left(\frac{u_{i+1}-u_i}{h}\right) - EA_{i-1/2}\left(\frac{u_i-u_{i-1}}{h}\right)$$

where $A_{i-1/2} = A(x_{i-1/2})$ and $A_{i+1/2} = A(x_{i+1/2})$.

Making use of Eq. (10.42), the term on the right-hand side becomes:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} f(x)dx \cong \frac{h}{2}(f_{i-1/2}+f_{i+1/2})$$

where $f_{i-1/2} = f(x_{i-1/2})$ and $f_{i+1/2} = f(x_{i+1/2})$.

Finally, the general form of the discretized equations for interior nodes can be written as

$$b_i\, u_{i-1} + d_i\, u_i + a_i\, u_{i+1} = c_i, \quad \text{for } i = 1, \ldots, (M-1)$$

where $b_i = EA_{i-1/2}/h$, $a_i = EA_{i+1/2}/h$, $d_i = -(b_i + a_i)$, and $c_i = h(f_{i-1/2} + f_{i+1/2})/2$.

Step 3. Implementing BCs: The left and right BCs are Dirichlet and Neumann types, respectively. At the left boundary node, we simply set $a_0 = c_0 = 0$ and $d_0 = 1$ in the general form of the difference equation, yielding $u_0 = 0$. To obtain the finite volume equation for the right boundary, we integrate the ODE over $[x_{M-1/2}, x_M]$.

$$\int_{x_{M-1/2}}^{x_M} \frac{d}{dx}\left(EA(x)\frac{du}{dx}\right)dx - \int_{x_{M-1/2}}^{x_M} f(x)dx \cong EA_M\left.\frac{du}{dx}\right)_{x_M} - EA_{M-1/2}\left.\frac{du}{dx}\right)_{x_{M-1/2}} - \frac{h}{2}f_M = 0$$

Note that $(du/dx)_{x_M} = 0$ due to the right BC and the integral of $f(x)$ is replaced with the rectangle rule. The velocity gradient at $x_{M-1/2}$ is approximated with the CDF as $(du/dx)_{x_{M-1/2}} \cong (u_M - u_{M-1})/h$. Substituting these approximations into the above equation yields

$$0 - EA_{M-1/2}\frac{u_M - u_{M-1}}{h} = \frac{h}{2}f_{M-1/2}$$

or rearranging and simplifying similar terms leads to

$$b_M\, u_{M-1} + d_M\, u_M = c_M$$

where $b_M = EA_{M-1/2}/h$, $d_M = -EA_{M-1/2}/h$, and $c_M = (h/2)f_{M-1/2}$.

Step 4. Constructing a linear system: The finite volume discretized equations together constitute a linear system of $M + 1$ equations. They can be expressed in matrix form as

$$\begin{bmatrix} d_0 & a_0 & & & \\ b_1 & d_1 & a_1 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{M-1} & d_{M-1} & a_{M-1} \\ & & & b_M & d_M \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{M-1} \\ u_M \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{M-1} \\ c_M \end{bmatrix}$$

Step 5. Solving the linear system: The solution of the resulting tridiagonal system for $M = 10$, 20, 40, and 100 gives numerical estimates for the ODE.

Table 10.2: Numerical results and errors; all values were multiplied by 10^6.

	True	For $M = 10$		For $M = 20$	
x	Solution	Estimate	Abs. Error	Estimate	Abs. Error
0	0	0	0	0	0
0.1	−0.0338665	−0.0338718	5.26×10^{-6}	−0.0338679	1.33×10^{-6}
0.2	−0.0676462	−0.0676826	3.64×10^{-5}	−0.0676553	9.15×10^{-6}
0.3	−0.0997298	−0.0998255	9.57×10^{-5}	−0.0997538	2.40×10^{-5}
0.4	−0.1287007	−0.1288861	1.85×10^{-4}	−0.1287471	4.64×10^{-5}
0.5	−0.1533847	−0.1536925	3.08×10^{-4}	−0.1534618	7.71×10^{-5}
0.6	−0.1729165	−0.1733822	4.66×10^{-4}	−0.1730331	1.17×10^{-4}
0.7	−0.1868319	−0.1874933	6.61×10^{-4}	−0.1869975	1.66×10^{-4}
0.8	−0.1951965	−0.1960933	8.97×10^{-4}	−0.1954209	2.24×10^{-4}
0.9	−0.1987900	−0.1999629	1.17×10^{-3}	−0.1990835	2.93×10^{-4}
1	−0.1993795	−0.2008676	1.49×10^{-3}	−0.1997518	3.72×10^{-4}

Discussion: The true and numerical solutions, as well as the absolute true errors, all multiplied by 10^6, are comparatively presented in **Table 10.2**. The mean absolute errors are found to be 5.31×10^{-10} and 1.33×10^{-10} for $M = 10$ and 20, respectively. The FVM is inherently a second-order accurate method; therefore, when the interval size is halved, the global errors are reduced by a factor of four on average.

The accuracy in finite-volume (and other) methods near the boundaries is affected by the type of BC and how they are treated. For instance, Dirichlet BC is typically imposed strongly in FVMs. This means that the boundary value is directly substituted at the boundary node. As a result, the boundary value is exact (numerically) and does not introduce a local approximation error at that point. However, the first interior node may experience a lower order of accuracy due to one-sided flux approximations or lack of symmetry in stencils. In this regard, the absolute errors near the left boundary are much smaller than those in the rest of the domain.

Neumann or Robin BCs involve the derivative of the solution, so they are more complex to implement. They require discretizations of the derivatives at the boundary, typically *via* finite difference approximations (e.g., central or one-sided differences). This introduces approximation error directly at the boundary point, unless high-order methods are used. The error at a Dirichlet or Robin boundary is not zero, even if the numerical scheme is consistent because the boundary condition itself is approximated. That is why the largest absolute errors in this example are observed at the right boundary.

The numerical estimates improve with an increasing number of cells as a result of the reduction in the truncation error as the cell size decreases. To improve accuracy near a Neumann or Robin boundary, you may use second-order (or higher) one-sided approximations for derivatives, in which case you lose the tridiagonality of the coefficient matrix. Implementing fictitious nodes with extrapolation will preserve a second-order approximation. Lastly, always check the behavior of the boundary error by performing a grid refinement study.

EXAMPLE 10.3: Two-point nonlinear BVP with Newton's method.

In astrophysics, the *Lane-Emden equation* models the structure of polytropic stars—spherically symmetric, self-gravitating fluid spheres in hydrostatic equilibrium. The equation arises from combining the equations of hydrostatic equilibrium, mass conservation, and a polytropic equation of state. The dimensionless form of the equation is given by

$$\frac{d^2\rho}{dr^2} + \frac{2}{r}\frac{d\rho}{dr} + \rho^5(r) = 0$$

subject to the following BCs:

$$\begin{array}{lll} \text{At } r = 0, & \rho'(0) = 0 & \text{density max. at the center} \\ \text{At } r = 3, & \rho(3) = 1/2 & \text{density is reduced to } 1/2 \end{array}$$

where $\rho(r)$ is the dimensionless density as a function of dimensionless radius, and n is the polytropic index.

Apply Newton's method to find a numerical solution to the Lane-Embden equation for the case with polytropic index $n=5$. (a) Divide the physical domain, [0, 3], into uniform M subintervals and derive the general difference equation; (b) implement the BCs and express the difference equations in matrix form; and (c) solve the resulting nonlinear system for $M=10$, 20, and 100 to obtain numerical estimates within $\varepsilon = 10^{-6}$ tolerance. Compare your estimates with the true solution given by $\rho(r) = 1/\sqrt{1 + r^2/3}$.

SOLUTION:

To implement the numerical solution steps, the nonlinear BVP is expressed in the following form:

$$\rho'' = f(r, \rho, \rho') = -2\frac{\rho'}{r} - \rho^5, \qquad \rho'(0) = 0, \quad \rho(3) = 1/2.$$

where Neumann and Dirichlet BCs are imposed on the left and right boundaries, respectively, and the primes denote derivatives with respect to r.

Step 1: Gridding. The physical domain (that is, [0,3]) is divided into M subintervals ($h = \Delta r = 3/M$), and the grid nodes are placed on both sides of each subinterval, as shown in **Figure 10.3**. The nodes are enumerated from 0 to M, and the abscissas are found by $r_i = ih$ for all i. Note that the right boundary node is marked as *known*, and a fictitious node is placed at r_{-1} due to the Neumann BC.

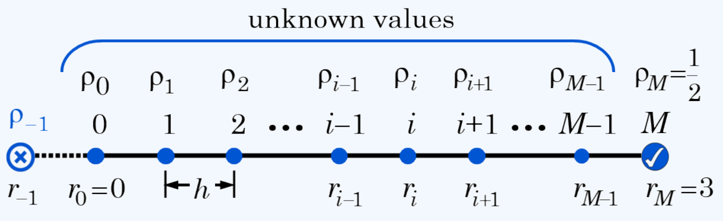


Figure 10.3

Step 2. Discretizing. The ODE, valid for an arbitrary $r = r_i$, is discretized using the central difference formulas and expressed as a set of non-linear equations as follows:

$$g_i(\boldsymbol{\rho}) = \frac{\rho_{i+1} - 2\rho_i + \rho_{i-1}}{h^2} - f\left(r_i, \rho_i, \frac{\rho_{i+1} - \rho_{i-1}}{2h}\right) = 0, \quad i = 0, 1, \ldots, M-1$$

where $\boldsymbol{\rho} = (\rho_1, \rho_2, \ldots\rho_M)$ denotes the discrete set of nodal density.

Step 3. Implementing BCs. For the left and right boundary nodes, the BCs are expressed as $\rho_0' = 0$ and $\rho_M = 1/3$, respectively. However, we note that $\rho'(r)/r$ yields the 0/0 indeterminate form as $r \to 0$. Thus, applying L'Hopital's rule, we find $\rho'(r)/r \to \rho''(0)$. Then, the ODE for $r = 0$ becomes $3\rho_0'' + \rho_0^5 = 0$ or we may write $\rho_0'' = f(r_0, \rho_0, \rho_0') = -\rho_0^5/3$, which gives the following discretized form:

$$\frac{\rho_{-1} - 2\rho_0 + \rho_1}{h^2} = f\left(r_0, \rho_0, \rho_0'\right) = -\frac{1}{3}\rho_0^5$$

Next, to eliminate the fictitious node value ρ_{-1}, we apply the CDF to $\rho_0' = 0$, which gives $\rho_{-1} = \rho_1$. Thus, for $i = 0$ (left boundary node), the difference equation can be written as

$$g_0 = -\rho_0 + \rho_1 + \frac{h^2}{6}\rho_0^5 = 0$$

For $i = M$ (right boundary node), adopting the tridiagonal system conventions, we set $b_M = a_M = 0$, $d_M = 1$, and $c_M = 1/2$.

Step 4. Constructing a nonlinear system. The resulting nonlinear difference equations are expressed as $\mathbf{g}(\boldsymbol{\rho}) = 0$. Then, the nonlinear equation can be explicitly written as

$$\begin{aligned}
g_0 &= -\rho_0 + \rho_1 + \frac{h^2}{6}\rho_0^5 = 0 \\
g_i &= \rho_{i-1} - 2\rho_i + \rho_{i+1} + h^2\left(\frac{2}{r_i}\frac{\rho_{i+1} - \rho_{i-1}}{2h} + \rho_i^5\right) = 0, \quad \text{for } i = 1, 2, \ldots, M-1 \\
g_M &= \rho_M - \boxed{1/2} = 0
\end{aligned}$$

It should be noted that the $\rho_M = 1/2$ condition is boxed to emphasize implementation of the BC.

Step 5. Solving the system. The nonlinear system requires a set of initial guesses. Due to the ρ^5 term, the ODE is highly nonlinear and requires a smart initial guess to be able to achieve fast convergence. Since the density is normalized, it is maximum at $r = 0$ (i.e., $\rho = 1$). So we may generate the initial guess from a straight line that connects the points (0,1) and (3,1/2); i.e., $\rho^{(0)}(r) = 1 - r/6$. Then, the system is solved using the Newton's method, which was covered in Section 4.8.

An improved solution (the current estimate) $\boldsymbol{\rho}^{(p+1)}$ is obtained as follows:

$$\boldsymbol{\rho}^{(p+1)} = \boldsymbol{\rho}^{(p)} + \boldsymbol{\delta}^{(p)}$$

where $\boldsymbol{\delta}^{(p)}$ is the displacement vector, which is obtained from solving the following linear system:

$$\mathbf{J}(\boldsymbol{\rho}^{(p)})\,\boldsymbol{\delta}^{(p)} = -\mathbf{g}(\boldsymbol{\rho}^{(p)})$$

where $\mathbf{J}(\boldsymbol{\rho})$ is the Jacobian matrix for this system, leading to a tridiagonal matrix given as

$$\mathbf{J}\left(\boldsymbol{\rho}^{(p)}\right) = \begin{bmatrix}
d_0 & a_0 & & & & & \\
b_1 & d_1 & a_1 & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & b_i & d_i & a_i & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & b_{M-1} & d_{M-1} & a_{M-1} \\
 & & & & & 0 & 1
\end{bmatrix}$$

where the diagonal elements for $i=1,2,\ldots,(M-1)$ are found as

$$b_i = \frac{\partial g_i}{\partial \rho_{i-1}} = 1 - \frac{h}{r_i}, \quad d_i = \frac{\partial g_i}{\partial \rho_i} = -2 + 5h^2\rho_i^4, \quad a_i = \frac{\partial g_i}{\partial \rho_{i+1}} = 1 + \frac{h}{r_i}$$

and for $i=0$ we have

$$d_0 = \frac{\partial g_0}{\partial \rho_0} = -1 + h^2\,\frac{5\rho_0^4}{6}, \qquad a_0 = \frac{\partial g_0}{\partial \rho_1} = 1, \quad \text{and} \quad c_i = -g_i(\boldsymbol{\delta}^{(p)}) \quad \text{for all } i$$

Table 10.3: Numerical estimates and absolute true errors for $M = 10$, 20 and 100, and the true solution for specified nodal points.

	True	For $M = 10$		For $M = 20$		For $M = 100$	
x	Solution	Estimate	Abs. Error	Estimate	Abs. Error	Estimate	Abs. Error
0	1	0.998140	1.86×10^{-3}	0.999549	4.51×10^{-4}	0.999982	1.79×10^{-5}
0.3	0.985329	0.983279	2.05×10^{-3}	0.984791	5.38×10^{-4}	0.985308	2.18×10^{-5}
0.6	0.944911	0.941917	2.99×10^{-3}	0.944158	7.54×10^{-4}	0.944881	3.02×10^{-5}
0.9	0.887357	0.883645	3.71×10^{-3}	0.886436	9.20×10^{-4}	0.887320	3.67×10^{-5}
1.2	0.821995	0.818143	3.85×10^{-3}	0.821042	9.53×10^{-4}	0.821957	3.80×10^{-5}
1.5	0.755929	0.752450	3.48×10^{-3}	0.755067	8.62×10^{-4}	0.755895	3.44×10^{-5}
1.8	0.693375	0.690563	2.81×10^{-3}	0.692678	6.98×10^{-4}	0.693347	2.78×10^{-5}
2.1	0.636285	0.634244	2.04×10^{-3}	0.635778	5.07×10^{-4}	0.636265	2.03×10^{-5}
2.4	0.585206	0.583923	1.28×10^{-3}	0.584886	3.19×10^{-4}	0.585193	1.28×10^{-5}
2.7	0.539949	0.539353	5.96×10^{-4}	0.539801	1.49×10^{-4}	0.539943	5.94×10^{-6}
3	0.5	0.5		0.5		0.5	

The iteration procedure is repeated until the norm (ℓ_2 or ℓ_∞) of the displacement vector is smaller than a predetermined tolerance value, i.e., $\|\boldsymbol{\delta}^{(p)}\| < \varepsilon = 10^{-6}$. This example problem is highly nonlinear due to the y_i^5 terms. Thus, when solving such highly nonlinear systems of equations, some convergence difficulties can arise. To avoid such problems, a smart initial guess of $\rho_i^{(0)} = 1 - r_i/6$ was adopted, which allowed the nonlinear system to converge to a solution very quickly within 4-6 iterations. It is worth noting here that for flat initial guess values ($\rho_i^{(0)}$=constant), the situations where the numerical solution either did not converge or converged to an unwanted solution were encountered.

The true solution along with the numerical estimates and absolute true errors for $M=10$, 20, and 100 were obtained and tabulated in **Table 10.3**. The mean absolute errors are found to be 2.47×10^{-3}, 6.15×10^{-4}, and 2.46×10^{-5} for $M = 10$, 20, and 100, respectively. The finite difference method employed here is a second-order accurate method; therefore, when reducing the interval size by 10, the global errors are reduced by a factor of 100 on average. This is consistent with second-order convergence expected from central finite differences.

Discussion: The FDM discretization of a two-point BVP results in a system of nonlinear equations. The next step involves using Newton's method to numerically solve the system of nonlinear equations, a process that necessitates the Jacobian to produce an improved estimate. We can encounter the following difficulties in solving two-point BVPs using the Newton's method:

- Since it introduces truncation errors, the FDM may require a finer grid structure to obtain high-accuracy solutions, which in turn increases the computational cost.
- Incorrect or imprecise implementation of the boundary conditions at the nodal boundary points may reduce accuracy or cause instability. In this regard, implementation of Neumann or Robin

BC must be handled with care. Particularly, in this example problem, pay attention to how the indeterminacy at $r=0$ was removed.

- Newton's method requires evaluating the Jacobian matrix at each iteration. For a simple ODE (i.e., $f(x, y, y')$) the Jacobian matrix can be easily constructed analytically and solved efficiently, as in this example. Numerical differentiation may be required for functions containing more complex expressions, which can also contribute to numerical errors. Overall, for a fine grid structure (i.e., large systems), computing the Jacobian matrix is computationally expensive and memory-intensive.
- Divergence or non-convergence may be observed when the initial guess is too far from a solution or the function has regions with steep gradients, discontinuities, or sharp turns. So, for a problem that is sensitive to initial guesses, start the iteration process with a smart initial guess, if possible.
- Slow convergence may be observed due to poor conditioning, weak Jacobian matrix properties, or flat regions in the function where the update steps become very small.
- The system may have multiple solutions, and the system may converge to an unwanted or incorrect solution. In other words, the converged solution may be physically meaningless or outside desired bounds.
- The method may fail if the Jacobian is singular or near-singular (highly stiff). In which case, use an alternative method to obtain a numerical solution.

EXAMPLE 10.4: Solving a Two-point Nonlinear BVP as an IVP.

The following two-point nonlinear BVP is encountered in several important fields of science and engineering. It is known as the steady-state form of the Fisher–KPP equation reaction-diffusion model. A chemical species is diffusing through a porous catalyst slab of thickness $L = \pi/8$ m, with a nonlinear reaction following logistic kinetics.

$$D\,\frac{d^2U}{dx^2} + k\,U(1-U) = 0$$

where $D = 10^{-6}$ m^2/s is the diffusion coefficient, $k = 1.6\times 10^{-5}$ (mol·s)$^{-1}$ is the reaction rate constant, and U (mol/m^3) is the concentration of reactants. The BCs are specified as

$$U'(0) = 0, \qquad U(\frac{\pi}{8}) = 3$$

Convert the given BVP to an IVP, and then apply the RK4 scheme to find a numerical solution. (a) Divide the physical domain into 20 subintervals and solve the resulting IVP to within $\varepsilon = 10^{-6}$ tolerance. (b) Compare your estimates with the true solution given by $U(x) = (3/2)\sec^2 2x$.

SOLUTION:

Setting $Y_1(x) = U(x)$ and $Y_2(x) = U'(x)$, the two-point BVP is reduced to the following set of coupled first-order IVP.

$$\frac{dY_1}{d\tau} = Y_2, \quad \frac{dY_2}{d\tau} = -\frac{k}{D}U(1-U), \quad Y_1(0) = U(0), \quad Y_2(0) = U'(0) = 0, \quad Y_1(\frac{\pi}{8}) \to 3$$

Since $U(0)$ is unknown, we cannot obtain the numerical solution of the IVP in question until we find the value of $U(0)$. Therefore, we assign $Y_1(0) = A$, where A is an arbitrary value, which is advised to be compatible with the physics of the problem. Then, we will obtain the numerical solution of the IVP using the RK4 scheme and record the estimated solution at $x = \pi/8$; i.e., $Y_1(\pi/8) = c$.

For $M = 20$ subintervals, we have $h = \Delta x = \pi/80$ and $x_i = ih$ for i=0,1,...,20. Then, the system of coupled nonlinear equations is solved as an IVP using the RK4 method. We find the numerical estimates for two independent initial guesses: $A_1 = 0.5$ and $A_2 = 2$.

$$\begin{aligned} &\text{For} \quad A_1 = 0.5 \quad \Longrightarrow \quad c_1 = Y_1(\pi/8) = 0.219927 \\ &\text{For} \quad A_2 = 2.0 \quad \Longrightarrow \quad c_2 = Y_1(\pi/8) = 7.720881 \end{aligned}$$

Note that these starting guesses give lower and upper bound estimates for $Y(\pi/8)$. We will know that the right answer, $Y(\pi/8) = 3$, lies between them. Then, we can use the secant method to find an improved guess as follows:

$$A_{p+1} = A_p + (A_{p-1} - A_p)\frac{3 - c_p}{c_{p-1} - c_p}, \qquad p \geqslant 2$$

Using the starting values, (A_1, c_1) and (A_2, c_2), a new prediction (A_3) is found as

$$A_3 = 2 + (0.5 - 2)\frac{3 - 7.720881}{0.219927 - 7.720881} = 1.055944$$

The corresponding numerical solution (c_3) of A_3 is obtained using the RK4 method. The subsequent predictions and the target estimates are obtained iteratively until the target BC value converges, i.e., $|c_p - c_{p-1}| < \varepsilon$. The iteration history of the initial slope and the target estimates is summarized in Table 10.4. The converged value of $U(0)$ was found to be 1.5 after seven iterations.

Table 10.4

p	A_p	c_p
1	0.5	0.219927
2	2	7.720881
3	1.055944	1.147673
4	1.321979	2.086410
5	1.580887	3.515480
6	1.487496	2.926435
7	1.499159	2.994993
8	1.500011	2.999356

Next, we find the final numerical solution for the BVP on $[0,\pi/8]$ using the RK4 scheme with the initial slope $U(0)=1.5$. The true solution and the true (absolute) errors are presented in **Table 10.5**. It is observed that Y_1 is smoother and grows more slowly than Y_2 up to $x\cong 0.6$, where Y_2 has a sharper curvature than Y_1. That is why the absolute true errors beyond $15h$ show a large jump, requiring a smaller h to capture the curvature more accurately. Since the estimated Y_1 and Y_2 for $M=20$ are not smaller than $\varepsilon=10^{-6}$ throughout the solution interval, the number of grid points should be increased.

Table 10.5: The corresponding numerical solution, absolute true error as well as the true solution.

x_i	$Y_{1,true}(x_i)$	Abs. Error in Y_1	$Y_{2,true}(x_i)$	Abs. Error in Y_2
0	1.5	0	0	0
h	1.502316	7.05×10^{-10}	2.361046	2.16×10^{-7}
$2h$	1.509291	1.27×10^{-8}	4.751351	4.35×10^{-7}
$3h$	1.521013	3.65×10^{-8}	7.200949	6.65×10^{-7}
$4h$	1.537628	7.31×10^{-8}	9.741457	9.17×10^{-7}
$5h$	1.559349	1.24×10^{-7}	12.40695	1.20×10^{-6}
$6h$	1.586457	1.92×10^{-7}	15.23498	1.53×10^{-6}
$7h$	1.619311	2.79×10^{-7}	18.26771	1.91×10^{-6}
$8h$	1.658359	3.90×10^{-7}	21.55334	2.37×10^{-6}
$9h$	1.704152	5.29×10^{-7}	25.14780	2.94×10^{-6}
$10h$	1.757359	7.06×10^{-7}	29.11688	3.64×10^{-6}
$11h$	1.818790	9.28×10^{-7}	33.53895	4.51×10^{-6}
$12h$	1.889424	1.21×10^{-6}	38.50839	5.62×10^{-6}
$13h$	1.970445	1.57×10^{-6}	44.14009	7.05×10^{-6}
$14h$	2.063287	2.03×10^{-6}	50.57536	8.89×10^{-6}
$15h$	2.169694	2.62×10^{-6}	57.98973	1.13×10^{-4}
$16h$	2.291796	3.40×10^{-6}	66.60349	1.46×10^{-4}
$17h$	2.432212	4.42×10^{-6}	76.69604	1.89×10^{-4}
$18h$	2.594181	5.79×10^{-6}	88.62559	2.49×10^{-4}
$19h$	2.781747	7.65×10^{-6}	102.85681	3.32×10^{-4}
$20h$	3	1.02×10^{-4}	120	4.49×10^{-4}

Discussion: The shooting method, which also aims to find the approximate solution over a specified interval, is conceptually and computationally different from the FDM. The shooting method takes advantage of this by guessing any missing initial conditions, integrating the ODE forward, and checking whether the computed solution satisfies the BC at the other end. If it does not, the guess is iteratively refined using a root-finding technique like the secant method. The process repeats until the solution matches the target boundary condition within a specified tolerance.

The benefits of solving a BVP as an IVP are presented below:

Simplicity of IVP Solvers: The shooting method transforms a BVP into an IVP. IVP solvers (like Runge-Kutta methods) are well-developed, easy to implement, and widely available in scientific libraries. This is a convenient way of solving BVPs *via* IVP methods, often more straightforward than using direct BVP solvers.

Computational Efficiency: FDM leads to solving a (possibly large) system of algebraic equations, while the shooting method requires solving an IVP repeatedly with varying parameters, which avoids the need for solving large systems of equations. Although the computational cost and complexity can differ significantly depending on the nature of the problem, IVPs tend to require less memory and computational effort per iteration.

High Accuracy Solutions: FDM or FVM methods usually have a second-order global error, $\mathcal{O}(h^2)$. Since a system of algebraic equations is solved, local errors at each node do not propagate sequentially, i.e., the errors are balanced across the domain. This helps control the global error more uniformly. On the other hand, in the shooting method, a third- or higher-order global (smaller) error can be easily achieved with RK or similar marching algorithms when the IVP is well-behaved; otherwise, errors can blow up due to sensitivity or poor initial guesses.

Adaptive Step Size Control: FDMs do not have the flexibility to automatically adjust the step size. The user must manually design the meshing if a variable-step-size grid is necessary. However, IVP solvers typically include adaptive step size control algorithms that improve accuracy and efficiency without extra programming effort.

Stability Characteristics: FDM is generally more robust and stable, especially for applied linear problems that are not too stiff or chaotic. Stability is closely related to the properties of the resulting linear system (usually a banded or tridiagonal matrix). For well-posed BVPs with smooth coefficients and proper discretization, FDM tends to be numerically stable. For nonlinear problems, convergence and stability depend on how the system of nonlinear equations is solved (e.g., Newton-Raphson). This can cause issues unless the grid is sufficiently fine or tailored adaptively. If a system is stiff or poorly conditioned, numerical instability or amplification of round-off errors can be encountered.

The stability of shooting methods depends on the IVP solver scheme. If a stable method (e.g., implicit Runge-Kutta for stiff IVPs) is used, one will inherit stability. However, the shooting method can be numerically unstable due to sensitivity in the initial condition, especially for nonlinear or stiff problems. Shooting methods can also suffer from accumulated numerical errors, particularly over long intervals or with rapidly growing solutions.

EXAMPLE 10.5: Solving Blasius equation as an IVP.

The *Blasius equation* is a well-known third-order nonlinear ordinary differential equation that arises in boundary layer theory for steady, two-dimensional incompressible laminar flow over a flat plate. The equation is

$$f''' + \frac{1}{2} f f'' = 0, \quad f(0) = f'(0) = 0, \quad f'(\infty) = 1$$

where $\eta = y\sqrt{U/x\nu}$ is the similarity variable, $\psi = \sqrt{\nu U x} f(\eta)$ is the so-called stream function, $f = f(\eta)$ is the similarity function, U is the free stream velocity, (x, y) denotes the Cartesian coordinate system, and (u, v) denotes the velocity components in (x, y) directions.

The boundary conditions $f(0)$, $f'(0)$, and $f'(\infty)$ refer to no displacement, no slip, and free stream velocity, respectively. Convert the Blasius's equation to an IVP, and then apply the RK4 scheme to find a numerical solution. Integrate from $\eta = 0$ to $\eta_{max} = 10$ and determine $f''(0)$.

SOLUTION:

The Blasius equation is converted to a system of coupled first-order ODEs by setting $y_1(\eta) = f(\eta)$, $y_2(\eta) = f'(\eta)$, and $y_3(\eta) = f''(\eta)$, which yields

$$y_1' = y_2, \qquad y_2' = y_3, \qquad y_3' = -\frac{1}{2} y_1 y_3$$

where the initial conditions are expressed as

$$y_1(0) = 0, \qquad y_2 = 0, \qquad y_3 = c$$

where $c = f''(0)$ is an unknown, but we will adjust it such that it gives $f'(\eta_{\text{max}}) = 1$, where $\eta_{\text{max}} \equiv 10$.

We set $\Delta\eta = 0.05$ and integrate the IVP from $\eta = 0$ to η_{max} using a RK4 solver. We run the solver for two independent predictions to determine two predictions for $B_p = f'(10)$. For $c_1 = 0.3$ and $c_2 = 0.5$, we find

$$\begin{aligned} &\text{For} \quad c_1 = 0.3 \quad \Longrightarrow \quad B_1 = y_2(10) = 0.934556 \\ &\text{For} \quad c_2 = 0.5 \quad \Longrightarrow \quad B_2 = y_2(10) = 1.313725 \end{aligned}$$

which means that the true value of c is in between 0.3 and 0.5.

Using the most recent two predictions, we find an improved prediction for c_{p+1} with the *Secant method* as follows:

$$c_{p+1} = c_p + (c_{p-1} - c_p)\frac{1 - B_p}{B_{p-1} - B_p}, \qquad p \geqslant 2$$

For $p = 2$, we obtain $c_3 = 0.334520$. We then resolve the IVP to find the target value B_3 to be 1.00494. (Note that the target value B_p should yield $B = 1$ as $p \to \infty$.) Since the system is nonlinear, we repeat this iteration procedure a few times, eventually yielding the converged value of $c = 0.33206$. Finally, the IVP is solved for the last time for $c = 0.33206$.

The numerical solutions for $f(\eta)$, $f'(\eta)$, and $f''(\eta)$ are plotted against η (*see* **Figure 10.5**). Note that the similarity function $f(\eta)$ represents a scaled stream function of the flow. It does not directly give us velocity, but its derivatives do. At $\eta = 0$, which is right at the plate, $f(0) = 0$ (no flow into the flat wall). Physically, $f(\eta)$ increases monotonically from 0 at the plate to some upper limit ($f(8) \cong 6.28$) as you move away from the surface.

The $f'(\eta)$ is the dimensionless velocity profile in the direction along the plate (tangential velocity, u/U). At the wall ($\eta = 0$), the $f'(0) = 0$ is the *no-slip condition* (fluid sticks to the wall). As $\eta \to \infty$, $f'(\eta) \to 1$ means that the flow speed approaches the free-stream velocity far from the plate. The shape of $f'(\eta)$ smoothly rises from 0 to 1 and tells us how quickly the velocity ramps up from the wall to the outer flow. We observe $f'(\eta) = 1$ for $\eta > 5$, corresponding to the free stream region above the boundary layer.

The $f''(\eta)$ represents the dimensionless shear rate or velocity gradient at each point, and it is directly related to the wall shear stress. At $\eta = 0$, we obtained $f''(0) = 0.33206$, which is a crucial number as it determines the *skin friction coefficient*. As $\eta \to \infty$, $f''(\eta) \to 0$ means that the shear rate drops off far from the wall, where the fluid is moving at constant free stream velocity.

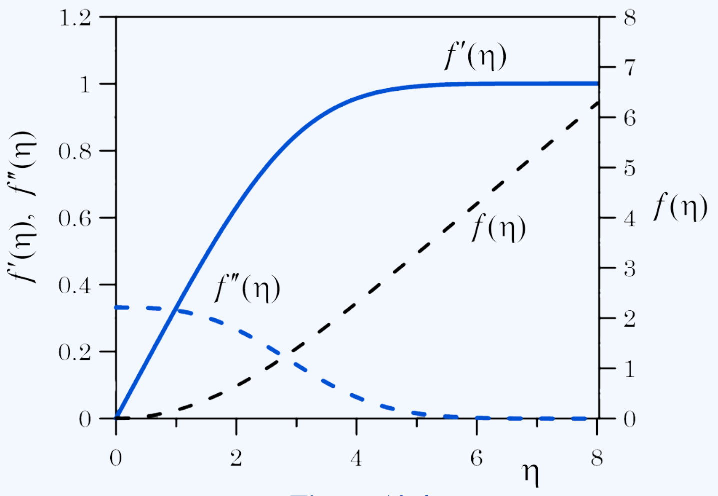


Figure 10.4

The numerical results of the solution in the range $4.8 \leqslant \eta \leqslant 5.1$ are tabulated in **Table 10.7**. The boundary layer edge is defined as $u/U = 0.99 = f'(\eta)$. Because, in theory, the velocity approaches U asymptotically, it never exactly reaches it at any finite distance from the wall. So there is no sharp boundary where the viscous effects just suddenly stop. To make the problem manageable and define a practical boundary, a threshold value of 99% of the free-stream velocity is a commonly accepted engineering standard. We note that $f'(\eta) = 0.99$ is realized at $\eta \cong 4.95$, which can be further refined by reducing $\Delta\eta$. In practical engineering problems, however, $\eta = 5$ is commonly used.

Table 10.6: The true as well as absolute true solutions.

η	$f(\eta)$	$f'(\eta)$	$f''(\eta)$
4.80	3.08534	0.987795	0.0218709
4.85	3.13476	0.988847	0.0202349
4.90	3.18422	0.989820	0.0186981
4.95	3.23374	0.990719	0.0172566
5.00	3.28330	0.991547	0.0159066
5.05	3.33289	0.992311	0.0146440
5.10	3.38253	0.993013	0.0134649

Discussion: Solving BVPs with a BC at infinity requires practical approximations and numerical methods such as truncating the domain at a sufficiently large size or using the shooting method to iterate until the *infinite* condition is satisfied within a tolerance. These techniques work very well, especially for problems like the Blasius boundary layer, where the solution stabilizes quickly.

EXAMPLE 10.6: Solving a Fourth-Order Linear BVP.

Consider the following fourth-order linear ODE:

$$y^{(\text{iv})} + 2y'' + y(x) = 0, \qquad 0 \leqslant x \leqslant \pi/2$$

subjected to the following BCs:

$$y(0) = 1, \quad y'(0) = 3, \quad y(\pi/2) = 1 + \pi, \quad y'(\pi/2) = 1 - \pi$$

Apply the finite difference method (FDM) to find an approximate solution for the given BVP using uniformly spaced 10 intervals and compare the approximate solution with the true solution given as $y = (1 + 2x)(\sin x + \cos x)$.

SOLUTION:

The given problem is a fourth-order linear BVP with four BCs: two Dirichlet and two Neumann BCs. We follow the numerical solution steps one-by-one as illustrated below:

Step 1: Gridding. A one-dimensional uniform grid, shown in **Figure 10.5**, is generated by dividing the solution interval ($0 \leqslant x \leqslant \pi/2$) into 10 uniformly spaced subintervals, giving $h = \pi/20$. The abscissas of the nodal points are found from $x_i = ih$ for $i = 0, 1, 2, \ldots, 10$. Due to the Dirichlet BCs on both sides, y_0 and y_{10} are nodal points whose solutions are known (shown with check marks).

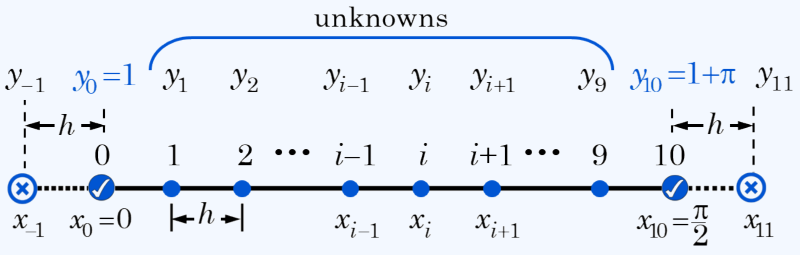


Figure 10.5

Step 2: Discretizing. The ODE is satisfied for all $x_i \in (0, \pi/2)$, thus we write

$$y_i^{(\text{iv})} + 2y_i'' + y_i = 0$$

Next, we approximate $y^{(\text{iv})}$ and y'' by the CDFs (*see* Table 5.3) resulting in

$$\frac{y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}}{h^4} + 2\,\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + y_i = 0$$

Multiplying both sides by h^2 and collecting like terms gives a *five-point stencil*:

$$y_{i-2} + (-4 + 2h^2)y_{i-1} + (6 - 4h^2 + h^4)y_i + (-4 + 2h^2)y_{i+1} + y_{i+2} = 0 \qquad \text{(A)}$$

for $i = 1, 2, \ldots, 9$. Note that the difference equations for $i = 0$ and $i = 10$ are excluded since Dirichlet BCs were imposed on both boundary nodes.

Step 3: Implementing BCs. Substituting the numerical values in Eq. (A), we obtain

$$\begin{aligned} y_0 &= 1 && \text{for } i = 0 \\ y_{i-2} + b\,y_{i-1} + c\,y_i + b\,y_{i+1} + y_{i+2} &= 0, && \text{for } i = 1, 2, \ldots, 9 \\ y_{10} &= 1 + \pi && \text{for } i = 10 \end{aligned}$$

where $b = -3.950652$ and $c = 5.901913$ and we deduced y_0 and y_{10} from the Dirichlet BCs.

For $i = 1$ and $i = 9$, Eq. (A) yields a couple of fictitious nodes (y_{-1} and y_{11}) (*see* **Figure 10.5**). These fictitious nodes are eliminated using the prescribed Neumann BCs in the same manner as implemented in the two-point BVP.

The left boundary BC is discretized using the CDF as $y'_0 \cong (y_1 - y_{-1})/2h = 1$, which leads to $y_{-1} = y_1 - 6h$. Substituting this approximation into Eq. (A) along with $i = 1$, we obtain

$$(c+1)\,y_1 + b\,y_2 + y_3 = 6h - b\,y_0$$

Similarly, for the right boundary, we approximate the Neumann BC as $y'_{10} \cong (y_{11} - y_9)/2h = 1 - \pi$, leading to $y_{11} = y_9 + 2(1-\pi)h$. Likewise, substituting this approximation into Eq. (A) with $i = 9$, we find

$$y_7 + b\,y_8 + (c+1)\,y_9 = 2(\pi - 1)h - b\,y_{10}$$

Step 4: Constructing a Linear System. When all the difference equations are put together, we obtain a system of linear equations (**Ay**=**r**), whose coefficient matrix is pentadiagonal.

$$\mathbf{A} = \begin{bmatrix} c+1 & b & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b & c & b & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & b & c & b & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & b & c & b & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & b & c & b & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & b & c & b & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & b & c & b & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & b & c & b \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & b & c+1 \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} 6h - b\,y_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2(\pi-1) - b\,y_{10} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{bmatrix}$$

Step 5: Solving the system. A difference equation with a five-point stencil leads to a linear system of a banded matrix (specifically, pentadiagonal). For a moderate number of grid points, this linear system of equations can be solved using a general Gauss-elimination solver or specialized solvers such as the Thomas algorithm adopted for pentadiagonal systems. *See* Computer Assignment, **CA2.4** for the general structure of a pentadiagonal system of equations. A pentadiagonal system of equations can also be regarded as a banded linear system with a bandwidth of 2 on each side of the main diagonal with a total bandwidth of 5 and can be solved as a banded system too (*see* Pseudocode 2.14).

Table 10.7: The numerical estimates, true solution as well as absolute error.

		True Solution	$M = 10$		$M = 40$	
i	x_i	$y(x_i)$	Estimate	Abs. Error	Estimate	Abs. Error
0	0	1	1			
1	h	1.5035596	1.5071881	3.63×10^{-3}	1.50379	2.28×10^{-4}
2	$2h$	2.0518010	2.0574479	5.65×10^{-3}	2.05216	3.54×10^{-4}
3	$3h$	2.6126269	2.6189090	6.28×10^{-3}	2.61302	3.93×10^{-4}
4	$4h$	3.1520757	3.1578975	5.82×10^{-3}	3.15244	3.63×10^{-4}
5	$5h$	3.6356550	3.6402510	4.60×10^{-3}	3.63594	2.86×10^{-4}
6	$6h$	4.0297125	4.0326820	2.97×10^{-3}	4.0299	1.83×10^{-4}
7	$7h$	4.3027999	4.3041233	1.32×10^{-3}	4.30288	7.93×10^{-5}
8	$8h$	4.4269837	4.4270230	3.93×10^{-5}	4.42698	8.47×10^{-7}
9	$9h$	4.3790538	4.3785393	5.15×10^{-4}	4.37902	3.44×10^{-5}
10	$10h$	4.1415927	4.1415927		4.14159	

The numerical solutions for $M = 10$ and 40 uniform grid spacings and the true solution are presented in **Table 10.7** together with the absolute true errors. For the $M = 10$ case, we generally observe at least two-decimal-place accuracy along the solution domain with an average absolute error of 3.41×10^{-3}. For $M = 40$, the average absolute error drops to 2.11×10^{-4}, corresponding to one-sixteenth of that for the $M = 10$ case. Notice that the discretization scheme employed here is second-order accurate with a global error $\mathcal{O}(h^2)$. Thus, the grid spacing is divided by four, and the errors are reduced by a factor of sixteen.

Discussion: Solving a linear system resulting from the numerical treatment of a fourth-order ordinary differential equation (ODE) often involves a pentadiagonal system due to the nature of the discretization. Depending on the size of the system (small to medium problems), the resulting linear pentadiagonal system can be solved using *Direct Solvers* (pentadiagonal or banded solvers).

For large systems, iterative methods such as Gauss-Seidel or SOR can be employed if the coefficient matrix is diagonally dominant, which is usually *not the case*. However, Krylov subspace methods (GMRES, BiCGSTAB, Conjugate Gradient) may be used if matrix properties (positive definiteness, symmetry, etc.) are satisfied.

The CDFs for the 2nd and 4th derivatives are numerically stable for most reasonable boundary conditions. When solving a linear fourth-order ODE with smooth coefficients and good boundary conditions, FDM is typically stable.

The condition number of the resulting pentadiagonal matrix can grow as you refine the mesh (i.e., smaller h), potentially leading to numerical instability. However, ill-conditioning does not mean the method is unstable per se, but it can make solutions *sensitive* to round-off errors. Preconditioning or higher precision arithmetic can help in these cases.