CHAPTER 14

ODE: Boundary-Value Problems

For the higher-order ordinary differential equations considered in the previous chapter, all of the required information about the solution is specified at the same point, say, $x = a$, and the solution function is sought on an interval $a \leq x \leq b$. In many important applications, the information that is known about the desired solution is given at the end points of the interval. Such problems are called ordinary differential equation boundary-value problems (ODE-BVP).

In this chapter, we consider two standard approaches to solving ODE-BVP, the shooting method and the finite-difference method. The shooting method is motivated by the example of trying to hit a target at a specified distance. In the initial-value problem for the motion of an object, both the initial position and the initial velocity are given. From the solution, the location at which the object lands can be determined. In an experimental setting, one could try different velocities, observe the landing points, and eventually make the necessary adjustments to hit the target. Fortunately, for a linear problem, this basic idea leads to a numerical method that does not rely on repeated trial-and-error corrections. For a nonlinear ODE, we obtain an iterative technique analogous to the secant method or Newton's method from Chapter 2.

The finite-difference method is based on dividing the interval of interest into a number of subintervals and replacing the derivatives by the appropriate finite-difference approximations, as discussed in Chapter 11. If the differential equation is linear, it is transformed into a system of linear algebraic equations, which can be solved by the techniques investigated in previous chapters. For a nonlinear ODE, the system of algebraic equations is nonlinear, and the methods of Chapter 7 may be used. The finite-difference approach is also very important for partial differential equations, which are the subject of the next chapter.

Application 14-A Deflection of a Beam

Several boundary-value problems arise in the study of the deflection of a horizontal beam. For example, we consider a beam (see Figure 14.1) that is freely hinged at its ends (i.e., at $x = 0$ and $x = L$), with a uniform transverse load w and axial tension T. In this case, the deflection, $y(x)$, is described by the ODE boundary-value problem

$$y'' - \frac{T}{EI}y = \frac{wx(x-L)}{2EI}, \quad 0 \le x \le L$$

The physical parameters of the beam are the modulus of elasticity, E, and the central moment of inertia, I. We assume that the beam is of uniform thickness, so that the product EI is a constant. For convenience, the downward direction is taken as positive.

The boundary conditions state that the beam is supported, and therefore has no deflection, at $x = 0$ and $x = L$; i.e.,

$$y(0) = y(L) = 0$$

The general solution of the ODE is

$$y(x) = A\sinh(\alpha x) + B\sinh(\alpha(L-x)) - \frac{w}{\alpha^2 T} + \frac{wLx}{2T} - \frac{wx^2}{2T}$$

with $\alpha^2 = T/EI$.

The boundary condition that $y(0) = 0$ gives $B\sinh(\alpha L) = w/\alpha^2 T$, which determines the coefficient B.

Similarly, the condition that $y(L) = 0$ gives $A\sinh(\alpha L) = w/\alpha^2 T$, which determines A.

The ODE above is based on the assumption that y' is small, so that $(y')^2$ is negligible and $1/R \approx y''$. In general, the radius of curvature, R, is related to the deflection by

$$\frac{1}{R} = \frac{y''}{[1+(y')^2]^{3/2}}$$

which leads to the differential equation

$$\frac{y''}{[1+(y')^2]^{3/2}} - \frac{T}{EI}y = \frac{wx(x-L)}{2EI}, \quad 0 \le x \le L$$

for which numerical methods are important. (See Jaeger, 1951, for discussion.)

FIGURE 14.1: Bending of a beam.

Application 14-B A Well-Hit Ball

Many factors influence the path of a baseball after it is hit. As an example, we consider a ball hit so that it lands 300 feet from home plate after 3 seconds. If we assume that air resistance acts only against the horizontal component of the flight and is proportional to the horizontal velocity, the motion of the ball is given by

$$\frac{d^2x}{dt^2} = -c\frac{dx}{dt}$$

and

$$\frac{d^2y}{dt^2} = -g$$

The initial position is $x(0) = 0, y(0) = 3$, and the desired landing time is $t_f = 3$, so that $x(3) = 300, y(3) = 0$. We take the drag coefficient $c = 0.5$. The vertical component of the motion can be found directly by

$$y(t) = -16t^2 + 47t + 3$$

The solution for $x(t)$ can be found by the methods of this chapter. Plotting y versus x gives the path of the ball, as illustrated in Figure 14.2.

The boundary conditions imposed above, giving the vertical position at two specified values of the independent variable (t), are the simplest form of boundary conditions for a two-point ODE-BVP. It is also possible to have boundary conditions that involve the first derivative of the unknown function (either alone or in combination with the value of the function). In the "flight of a baseball" setting, one might wish to stipulate that the horizontal velocity of the ball had a specified value at a certain time.

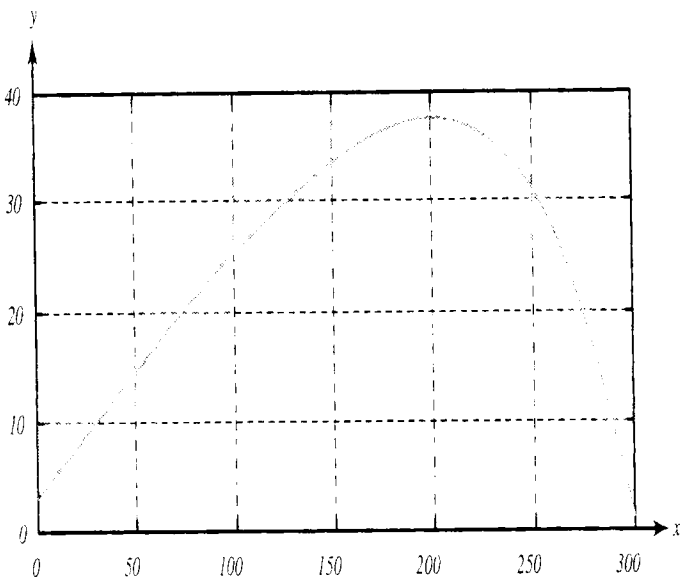

FIGURE 14.2: Flight of a baseball subject to some air resistance.

INTRODUCTION

Higher-order ODE do not necessarily have all of the information about the solution given at the same (initial) point. Problems in which the value of the unknown function or its derivative is given at two different points are known as boundary-value problems. In particular, we investigate the second-order, two-point boundary-value problem of the form

$$y'' = f(x, y, y'), \quad a \le x \le b$$

with Dirichlet boundary conditions

$$y(a) = a, \quad y(b) = b$$

or Neuman boundary conditions

$$y'(a) = a, \quad y'(b) = b$$

or mixed boundary conditions

$$y'(a) + c_1 y(a) = a, \quad y'(b) + c_2 y(b) = b$$

The first method we consider, the shooting method, is based on the techniques presented in Chapter 13 for IVP. The idea is to assume an initial value for $y'(a)$, generate a solution, and then adjust the solution so that it matches the specified value for $y(b)$. If the ODE is linear, we can solve two IVPs and form a linear combination of the solutions that will solve the BVP. This is based on the fact that the sum of two solutions to a linear ODE is also a solution. If the ODE is nonlinear, an iterative process can be used.

The second approach to ODE-BVP, the finite-difference method, is based on introducing a set of equally spaced node points and replacing the derivatives by difference quotients at these points. This gives a system of algebraic equations (linear or nonlinear depending on the nature of the ODE) which can be solved to obtain an approximate solution to the ODE-BVP at the node points.

The function-space methods seek to approximate the solution to the ODE-BVP as a linear combination of certain basis functions. Collocation seeks to satisfy the ODE at the node points by a linear combination of functions that are chosen to satisfy the boundary conditions. Rayleigh-Ritz methods seek to minimize a linear functional whose minimum corresponds to the solution of the ODE, again using basis functions that satisfy the boundary conditions.

Finally, we will present a little of the theory behind the existence and uniqueness of solutions to ODE-BVP. The well-known example problems $y'' = y$ with boundary conditions $y(0) = 0$, $y(\pi) = 1$, and $y'' = y$ with boundary conditions $y(0) = y(\pi) = 0$ illustrate the fact that even very simple ODE-BVP may fail to have a nontrivial solution, or may fail to have a unique solution.

14.1 SHOOTING METHOD

The shooting method uses solution techniques for initial-value problems to solve an ODE boundary-value problem. For a linear ODE, the shooting method takes a linear combination of solutions to two ODE-IVP. The specific form of the ODE-IVP and the combination is specified by the the type of boundary conditions given. The shooting method for nonlinear ODE leads to an iterative process, for which the secant method or Newton's method can be used.

14.1.1 Linear ODE

A linear two-point boundary-value problem can be solved by forming a linear combination of the solutions to two initial-value problems. The form of the IVP depends on the form of the boundary conditions. We begin with the simplest case, Dirichlet boundary conditions, in which the value of the function is given at each end of the interval. We then consider some more general boundary conditions.

Simple Boundary Conditions

Suppose the two-point boundary-value problem is linear, i.e., of the form

$$y'' = p(x)y' + q(x)y + r(x), \quad a \le x \le b$$

with boundary conditions $y(a) = y_a, y(b) = y_b$. If $r(x) \neq 0$ the ODE is called nonhomogeneous. The companion homogeneous ODE is $y'' = p(x)y' + q(x)y$.

The approach is to solve two IVP. The first is the given ODE, with the given initial condition for the function, but with the initial condition for the derivative taken to be 0. We denote this problem as

$$u'' = p(x)u' + q(x)u + r(x), \quad u(a) = y_a, \quad u'(a) = 0$$

The second ODE-IVP is the companion homogeneous ODE, with initial value of 0, and initial condition of the derivative taken to be 1. This problem is

$$v'' = p(x)v' + q(x)v, \quad v(a) = 0, \quad v'(a) = 1$$

From the basic theory of linear ODE, we know that a linear combination of the solutions of these two problems will also be a solution of the nonhomogeneous ODE. In fact, it is easy to verify that any function of the form

$$y(x) = u(x) + cv(x)$$

not only satisfies the nonhomogeneous ODE, but also satisfies the initial condition $y(a) = y_a$. If $v(b) \neq 0$, we find $c = (y_b - u(b))/v(b)$, so the solution of the original two-point BVP is given by

$$y(x) = u(x) + \frac{y_b - u(b)}{v(b)} v(x)$$

On the other hand, if $v(b) = 0$, the solution $u(x)$ satisfies both of the boundary conditions.

The system of two second-order ODE-IVP can be solved by the methods of the previous chapters.

EXAMPLE 14.1 A Simple Linear Shooting Problem

Consider the ODE

$$y'' = \frac{2x}{x^2+1}y' - \frac{2}{x^2+1}y + x^2 + 1$$

with boundary conditions $y(0) = 2, y(1) = 5/3$.

The nonhomogeneous ODE-IVP is

$$u'' = \frac{2x}{x^2+1}u' - \frac{2}{x^2+1}u + x^2 + 1, \quad u(0) = 2, \quad u'(0) = 0$$

The companion homogeneous ODE-IVP is

$$v'' = \frac{2x}{x^2+1}v' - \frac{2}{x^2+1}v, \quad v(0) = 0, \quad v'(0) = 1$$

Converting each of these second-order ODE to two first-order ODE, we define $z_1 = u$, $z_2 = u'$, $z_3 = v$, $z_4 = v'$, so we have

$$\begin{aligned} z_1' &= z_2 \\ z_2' &= \frac{2x}{x^2+1}z_2 - \frac{2}{x^2+1}z_1 + x^2 + 1 \\ z_3' &= z_4 \\ z_4' &= \frac{2x}{x^2+1}z_4 - \frac{2}{x^2+1}z_3 \end{aligned}$$

with initial conditions

$$\mathbf{z}(0) = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

After solving for $\mathbf{z}(x)$, we form the solution to the original problem as

$$y(x) = z_1(x) + \frac{y(1) - z_1(1)}{z_3(1)} z_3(x)$$

For reference, we note that the general solution of the ODE is

$$y(x) = d_1 x + d_2(x^2 - 1) + \frac{x^4}{6} + \frac{x^2}{2}$$

The exact solution of this BVP (shown in Figure 14.3) is

$$y(x) = \frac{x^4}{6} - \frac{3x^2}{2} + x + 2$$

The following MATLAB script and function generate the solution to this problem.

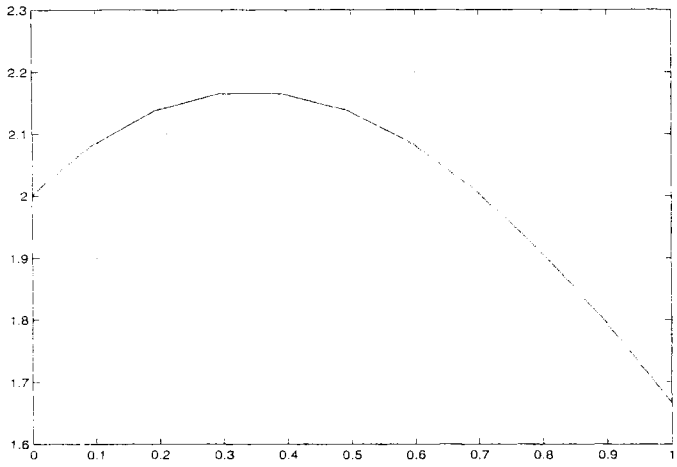

FIGURE 14.3: Solution to a simple ODE-BVP.

We illustrate the shooting method for the linear ODE with simple boundary conditions given in the previous example in the following MATLAB script. We use an m-file function `f_14_1` (rather than an inline function) to define the right-hand side of the system of four first-order ODE, and solve the ODE-IVP using MATLAB's built-in function `ode23`.

Linear Shooting Method

```
% S_14_1
clear all
ya = 2;     yb = 5/3;
z0 = [ ya    0    0    1];
a = 0;      b = 1;
tspan = [ a  b ];
[ x  z ] = ode23('f_14_1', tspan, z0)
[ n  m ] = size(z)
y(1:n, 1) = z(1:n, 1) + (yb - z(n, 1))*z(1:n, 3)./z(n, 3)
plot(x, y);   grid on
```

ODE for Example 14.1

```
function dz = f_14_1(x, z)
dz = [z(2)
      z(2)*2*x./(x.^2 + 1) - z(1)*2/(x.^2 + 1) + x.^2 + 1
      z(4)
      z(4)*2*x./(x.^2 + 1) -  z(3)*2/(x.^2 + 1)];
```

EXAMPLE 14.2 Deflection of a Simply Supported Beam

As described in Application 14-A, the deflection of a beam supported at both ends, subject to uniform loading along its length, is described by the ODE-BVP

$$y'' = \frac{T}{EI}y + \frac{wx(x-L)}{2EI}, \quad 0 \le x \le L$$
$$y(0) = y(L) = 0$$

We illustrate the problem for the following parameter values:

$$L = 100, \quad w = 100, \quad E = 10^7, \quad T = 500, \quad I = 500$$

The problem is linear, of the form

$$y'' = p(x)y' + q(x)y + r(x), \quad a \le x \le b$$

with $a = 0, b = L = 100, p(x) = 0, q(x) = T/EI = 10^{-7}$, and $r(x) = 10^{-8}[x(x-L)]$; the boundary conditions are $y(a) = 0$ and $y(b) = 0$. The solution can be obtained by solving the following IVP over $a \le x \le b$:

$$u'' = q(x)u + r(x), \quad u(a) = 0, \quad u'(a) = 0$$
$$v'' = q(x)v, \quad v(a) = 0, \quad v'(a) = 1$$

Since $y(L) = 0$, the solution of the two-point BVP is given by

$$y(x) = u(x) - \frac{u(b)}{v(b)}v(x)$$

The two second-order IVP are converted to a system of four first-order IVP by the change of variables

$$z_1 = u, \quad z_2 = u', \quad z_3 = v, \quad z_4 = v'$$

The differential equations relating these variables are

$$z_1' = z_2, \quad z_2' = 10^{-7}z_1 + 10^{-8}[x(x-100)]$$
$$z_3' = z_4, \quad z_4' = 10^{-7}z_3$$

with the initial conditions $z_1(0) = 0, z_2(0) = 0, z_3(0) = 0, z_4(0) = 1$. The computed solution is illustrated in Figure 14.4.

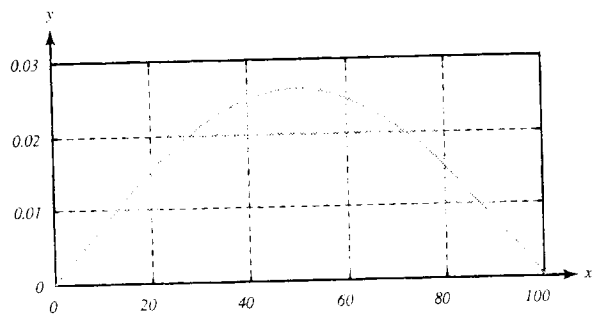

FIGURE 14.4: Deflection of a simply supported beam with uniform loading.

Mixed Boundary Condition at x = b

Suppose that the linear ODE

$$y'' = p(x)y' + q(x)y + r(x)$$

has boundary conditions consisting of the value of y given at $x = a$, but the condition at $x = b$ involves a linear combination of $y(b)$ and $y'(b)$:

$$y(a) = y_a, \qquad y'(b) + cy(b) = y_b$$

As in the previous discussion, the approach is to solve the two IVP:

$$\begin{aligned} u'' &= p(x)u' + q(x)u + r(x), \quad u(a) = y_a, \quad u'(a) = 0 \\ v'' &= p(x)v' + q(x)v, \qquad\qquad v(a) = 0, \quad v'(a) = 1 \end{aligned}$$

The linear combination, $y = u + dv$, satisfies the conditions at $x = a$, since $y(a) = y_a$. We need to find d (if possible) so that y satisfies

$$y'(b) + cy(b) = y_b$$

If $v'(b) + cv(b) \neq 0$, there is a unique solution, given by

$$y(x) = u(x) + \frac{y_b - u'(b) - cu(b)}{v'(b) + cv(b)} v(x)$$

On the other hand, if $v'(b) + cv(b) = 0$, it is easy to see that $y(x) = u(x)$ satisfies both boundary conditions.

General Separated Boundary Conditions

Suppose that the linear ODE

$$y'' = p(x)y' + q(x)y + r(x)$$

has mixed boundary conditions at both $x = a$ and $x = b$; i.e.,

$$y'(a) + c_1 y(a) = y_a, \quad y'(b) + c_2 y(b) = y_b$$

As in the previous discussion, the approach is to solve two IVP; however, the appropriate forms are now

$$\begin{aligned} u'' &= p(x)u' + q(x)u + r(x), \quad u(a) = 0, \quad u'(a) = y_a \\ v'' &= p(x)v' + q(x)v, \qquad\qquad v(a) = 1, \quad v'(a) = -c_1 \end{aligned}$$

The linear combination $y = u + dv$ satisfies $y'(a) + c_1 y(a) = y_a$; we need to find d (if possible) such that y satisfies $y'(b) + c_2 y(b) = y_b$. If $v'(b) + c_2 v(b) \neq 0$, there is a unique solution, given by

$$y(x) = u(x) + \frac{y_b - u'(b) - c_2 u(b)}{v'(b) + c_2 v(b)} v(x)$$

14.1.2 Nonlinear ODE

We now consider the shooting method for nonlinear problems of the form $y'' = f(x, y, y')$ on the interval $[a, b]$. We assume that $y(a)$ is given and that some condition on the solution is also given at $x = b$. The idea is the same as for linear problems, namely, to solve the appropriate initial-value problems and use the results to find a solution to the nonlinear problem. However, for a nonlinear BVP, we have an iterative procedure rather than a simple formula for combining the solutions of two IVP. In both the linear and the nonlinear case, we need to find a zero of the function representing the error—that is, the amount by which the solution to the IVP fails to satisfy the boundary condition at $x = b$. We assume the continuity of f, f_x, and f_y on an appropriate domain, to ensure that the initial-value problems have unique solutions.

We begin by solving the initial-value problem

$$u'' = f(x, u, u'), \quad u(a) = y_a, \quad u'(a) = t \tag{14.1}$$

for some particular value of t. We then find the error associated with this solution; that is, we evaluate the boundary condition at $x = b$ using $u(b)$ and $u'(b)$. Unless it happens that $u(x)$ satisfies the boundary condition at $x = b$, we take a different initial value for $u'(a)$ and solve the resulting IVP. Thus, the error (the amount by which our shot misses its mark) is a function of our choice for the initial slope. We denote this function as $m(t)$.

The first approach we consider uses the secant method to find the zero of the error function. This allows us to treat a fairly general boundary condition at $x = b$. The second approach is based on Newton's method.

Using the Secant Method

To solve a nonlinear BVP of the form

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad h(y(b), y'(b)) = 0$$

we may use an iterative process based on the secant method presented in Chapter 2. We need to find a value of t, the initial slope, so that solving Eq. (14.1) gives a solution that is within a specified tolerance of the boundary condition at $x = b$.

We begin by solving the equation with $u'(a) = t(1) = 0$; the corresponding error is $m(1)$. Unless the absolute value of $m(1)$ is less than the tolerance, we continue by solving Eq. (14.1) with $u'(a) = t(2) = 1$.

If this solution does not happen to satisfy the boundary condition (at $x = b$) either, we continue by updating our initial slopes according to the secant rule (until our stopping condition is satisfied), i.e.,

$$t(i) = t(i-1) - \frac{t(i-1) - t(i-2)}{m(i-1) - m(i-2)} m(i-1)$$

EXAMPLE 14.3 Shooting Method for Nonlinear ODE

We illustrate the nonlinear shooting method in the MATLAB script that follows. The BVP is

$$y'' = -2yy', \quad y(0) = 1, \quad y(1) + y'(1) - 0.25 = 0$$

The exact solution is $y = 1/(x+1)$. The computed solution, $u(x)$, and $u'(x)$ are shown in Figure 14.5. For the given tolerance, tol $= 0.00001$, the process converges in eight iterations.

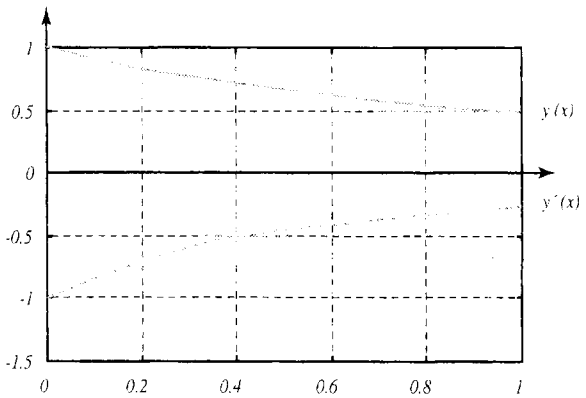

FIGURE 14.5: Solution $y(x)$ and $y'(x)$ for nonlinear shooting problem.

Nonlinear Shooting Using Secant Method

```
% BVP  y'' = f(x, y, y');  y(a) = ya;  h(y(b),y'(b)) = 0
% use secant rule for the error function: m(t) = h(u(b),u'(b)) = 0
clear all
ya = 1;  a = 0;  b = 1;  max_it = 10;  tol = 0.00001;
f = inline('[z(2); -2*z(1).*z(2)]', 'x','z')      % ODE
h = 'z1 + z2 - 0.25';                      % boundary condition
t(1) = 0;   t(2) = 1;  test = 1;   i = 1;   tspan = [ a   b];
while (test>tol)&(i<=max_it)
    if i > 2
       t(i) = t(i-1) - (t(i-1) - t(i-2))*m(i-1)/(m(i-1) - m(i-2))
    end
   z0 = [ya   t(i)];
   [ x , z ] = ode23( f, tspan, z0);
   [ n  nn]  = size(z);   z1 = z(n,1);        z2 = z(n,2);
   m(i) = eval(h);        test = abs(m(i));   i = i+1;
end
[x, z],   plot(x, z(:,1), x, z(:,2))
```

Using Newton's Method

We next illustrate how Newton's method can be used to find the value of $y'(a) = t$ in the initial-value problem for nonlinear shooting. We consider the following nonlinear BVP with simple boundary conditions at $x = a$ and $x = b$:

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad y(b) = y_b$$

We begin by solving the initial-value problem

$$u'' = f(x, u, u'), \quad u(a) = y_a, \quad u'(a) = t$$

The error in this solution is the amount by which $y(b)$ misses the desired value, y_b. For different choices of t, we get different errors, so we define

$$m(t) = u(b, t) - y_b$$

We need to find t such that $m(t) = 0$ (or $m(t)$ is as close to zero as we wish to continue the process). In the previous section, we found a sequence of t using linear interpolation between the two previous solutions.

In order to use Newton's method, we need to have the derivative of the function whose zero is required, namely, $m(t)$. Although we do not have an explicit formula for $m(t)$, we can construct an auxiliary differential equation whose solution allows us to update t at each iteration. This equation is

$$v'' = v f_u(x, u, u') + v' f_{u'}(x, u, u'), \quad v(a) = 0, \quad v'(a) = 1$$

The derivation of the auxiliary ODE is given in the discussion at the end of the section.

We begin by converting the two second-order initial-value problems

$$\begin{aligned} u'' &= f(x, u, u'), & u(a) &= y_a, \quad u'(a) = t_k \\ v'' &= v f_u(x, u, u') + v' f_{u'}(x, u, u'), & v(a) &= 0, \quad v'(a) = 1 \end{aligned}$$

into a system of four first-order ODE, by defining $z_1 = u, z_2 = u', z_3 = v, z_4 = v'$, so we have

$$\begin{aligned} z_1' &= z_2 \\ z_2' &= f(x, z_1, z_2) \\ z_3' &= z_4 \\ z_4' &= z_3 f_u(x, z_1, z_2) + z_4 f_{u'}(x, z_1, z_2) \end{aligned}$$

with initial condition

$$[y_a, t_k, 0, 1]'$$

Given t_k, an estimate for t, we solve the system given above.

To check for convergence, compute $m = z_1(b, t_k) - y_b$.

If $|m| <$tol, stop; otherwise, update t and continue for another iteration.

$$t_{k+1} = t_k - \frac{m}{z_3(b, t_k)}$$

EXAMPLE 14.4 Nonlinear Shooting with Newton's Method

$$y'' = -\frac{(y')^2}{y}, \quad y(0) = 1, \quad y(1) = 2$$

The exact solution of this ODE is $y = \sqrt{3x + 1}$.

The initial-value problem consists of two second-order ODE: the first corresponds to the original problem, but with the boundary condition replaced by an initial condition on the first derivative, $u'(0) = t_k$.

The first ODE is

$$u'' = -(u')^2 u^{-1}, \quad u(0) = 1, \quad u'(0) = t_k$$

The second ODE is an auxiliary equation that allows us to update the value of the parameter t_k. The auxiliary ODE is $v'' = v f_u + v' f_{u'}$, so to construct the auxiliary ODE, we need the following partial derivatives:

$$f_u(x, u, u') = (u')^2 u^{-2}$$
$$f_{u'}(x, u, u') = -2u' u^{-1}$$

Thus, the auxiliary ODE is

$$v'' = v(u')^2 u^{-2} - v' 2u' u^{-1}, \quad v(0) = 0, \quad v'(0) = 1$$

Writing this as a system of four first-order ODE, we define the components of our unknown function z as follows: $z_1 = u, z_2 = u', z_3 = v, z_4 = v'$.

The ODE are

$$z_1' = z_2, \quad z_2' = -\frac{z_2^2}{z_1}$$
$$z_3' = z_4, \quad z_4' = z_3\left(\frac{z_2}{z_1}\right)^2 - \frac{2z_4 z_2}{z_1}$$

The graph of the solution found using the MATLAB script that follows is shown in Figure 14.6.

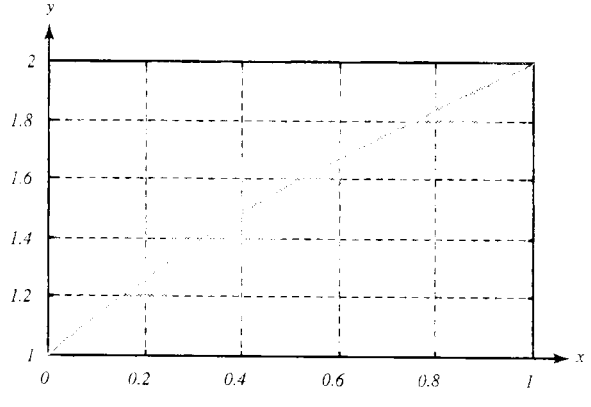

FIGURE 14.6: Solution of nonlinear shooting problem using Newton's method.

The following MATLAB script illustrates the use of the shooting method for a nonlinear ODE-BVP.

Nonlinear Shooting Using Newton's Method

```
% the ODE-BVP
%     y'' = f(x, y, y');
%              y(a) = ya;   y(b) = yb
% becomes the ODE-IVP
%     u'' =  f(x, u, u');
%              u(a) = ya;  u'(a) = t
%     v'' = v*f_u(x, u, u') + v'*f_u'(x, u, u')
%              v(a) = 0;  v'(a) = 1
clear all
%*****************
ya = 1;      yb = 2;
a = 0;       b = 1;
max_it = 5; tol = 0.00001;
%*****************
t(1) = 0;    test = 1;
i = 1;       tspan = [ a   b];
while (test>tol)&(i<=max_it)
   z0 = [ya   t(i)  0  1 ];
   [ x , z ] = ode23( 'f_ns_newt',tspan,z0);
   [ n  nn] = size(z);
   m(i) = z(n,1)-yb;
   test = abs(m(i));
   t(i+1) = t(i) - m(i) / z(n,3);
   i = i+1;
end
plot(x, z(:,1))
```

Function for Example 14.4

```
function dz = f_ns_newt(x, z)
dz = [ z(2)
      -(z(2)^2)/z(1)
       z(4)
       z(3)*(z(2)/z(1))^2 - 2*z(4)*z(2)/z(1) ];
```

Derivation of Auxiliary ODE

To solve a nonlinear ODE-BVP using the shooting method, we replace the original problem

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad y(b) = y_b$$

with an initial-value problem

$$u'' = f(x, u, u'), \quad u(a) = y_a, \quad u'(a) = t$$

The error (the amount by which $u(b, t)$, the solution of this ODE-IVP, differs from the boundary condition, y_b) is $m(t) = u(b, t) - y_b$. We need to find t so that the error is zero (or sufficiently close to zero).

Newton's method updates t as

$$t_i = t_{i-1} - \frac{m(t_{i-1})}{m_t(t_{i-1})}$$

so we need to find an expression for m_t. To do that, we make use of the fact that, for the given form of boundary condition, we have $m_t(t) = u_t(b, t)$.

To find u_t, we differentiate

$$u'' = f(x, u, u')$$

using the chain rule for partial derivatives; this gives

$$(u'')_t = f_t(x, u, u') = f_x x_t + f_u u_t + f_{u'}(u')_t$$

Since x and t are independent, $x_t = 0$, so we have

$$(u'')_t = f_u u_t + f_{u'}(u')_t$$

We introduce a new variable $v = u_t$ and assume sufficient continuity that we can interchange the order of differentiation with respect to x and t, so that $(u'')_t = (u_t)''$ and $(u')_t = (u_t)'$. This gives the linear ODE for v:

$$v'' = f_u v + f_{u'} v'$$

The initial conditions $u(a, t) = y_a$ and $u'(a) = t$ yield the initial conditions for v, namely, $v(a) = 0$ and $v'(a) = 1$.

Thus, solving the ODE-IVP for v allows us to use the fact that $v = u_t = m_t$ to update t in the formula for Newton's method.

Observations

Shooting methods can suffer from instabilities in the IVP; however, for a nonlinear second-order ODE-BVP, the resulting nonlinear zero-finding problem depends on only one variable.

Newton's method generalizes to systems more easily than the secant method and may converge more rapidly, but requires solving twice as many ODE. In the sections that follow, we investigate an alternative approach to solving ODE-BVP.

14.2 FINITE-DIFFERENCE METHOD

The second type of solution technique we consider for ODE-BVP is based on replacing the derivatives in the differential equation by finite-difference approximations (discussed in Chapter 11). The interval of interest, $[a, b]$, is divided into n subintervals by specifying evenly spaced values of the independent variable, $x_0, x_1, x_2, \ldots, x_n$, with $x_0 = a$ and $x_n = b$. Each subinterval is of length $h = x_{i+1} - x_i$. The approximate solution at x_i is denoted y_i.

14.2.1 Linear ODE

We first illustrate the finite-difference method with a simple example.

EXAMPLE 14.5 A Finite-Difference Problem

Use the finite-difference method to solve the problem

$$y'' = y + x(x-4), \quad 0 \le x \le 4$$

with $y(0) = y(4) = 0$ and $n = 4$ subintervals. The finite-difference method will find an approximate solution at the points $x_1 = 1, x_2 = 2$, and $x_3 = 3$. Using the central difference formula for the second derivative, we find that the differential equation becomes the system

$$y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = y_i + x_i(x_i - 4), \quad i = 1, 2, 3$$

For this example, $h = 1$. In writing out the system of algebraic equations, we make use of the fact that at $i = 1, y_0 = 0$ (from the boundary condition at $x = 0$), and similarly, at $i = 3, y_4 = 0$. Substituting in the values of x_1, x_2, and x_3 we obtain

$$\begin{aligned} y_2 - 2y_1 + 0 &= y_1 + 1(1-4) \\ y_3 - 2y_2 + y_1 &= y_2 + 2(2-4) \\ 0 - 2y_3 + y_2 &= y_3 + 3(3-4) \end{aligned}$$

Combining like terms and simplifying gives

$$\begin{aligned} -3y_1 + y_2 &= -3 \\ y_1 - 3y_2 + y_3 &= -4 \\ y_2 - 3y_3 &= -3 \end{aligned}$$

Solving, we find that $y_1 = 13/7, y_2 = 18/7$, and $y_3 = 13/7$.

We note for comparison that the exact solution of this problem is

$$y = \frac{2(1-e^4)}{e^{-4} - e^4} e^{-x} - \frac{2(1-e^{-4})}{e^{-4} - e^4} e^x - x^2 + 4x - 2$$

At $x = 1$, the exact solution is 1.8341 (to four decimal places); the finite-difference solution is $y_1 = 13/7 = 1.8571$.

The General Linear BVP

We now consider the general linear two-point boundary-value problem

$$y'' = p(x)y' + q(x)y + r(x), \quad a \le x \le b$$

with boundary conditions $y(a) = \alpha$, $y(b) = \beta$. To solve this problem using finite differences, we divide the interval $[a, b]$ into n subintervals, so that $h = (b - a)/n$. To approximate the function $y(x)$ at the points $x_1 = a + h, \ldots, x_{n-1} = a + (n-1)h$, we use the central difference formulas from Chapter 11:

$$y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad y'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2h}$$

Substituting these expressions into the BVP and writing $p(x_i)$ as p_i, $q(x_i)$ as q_i, and $r(x_i)$ as r_i gives

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = p_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i + r_i$$

Further algebraic simplification leads to a tridiagonal system for the unknowns $y_1, \ldots, y_{n-1}$ — that is, for $i = 1, \ldots, n-1$,

$$(2 + h\,p_i)y_{i-1} - (4 + 2h^2 q_i)y_i + (2 - h\,p_i)y_{i+1} = 2h^2 r_i$$

where $y_0 = y(a) = \alpha$ and $y_n = y(b) = \beta$.

Using the boundary conditions and simplifying gives:
first (for $i = 1$),

$$-(4 + 2h^2 q_1)y_1 + (2 - hp_1)y_2 = 2h^2 r_1 - \alpha(2 + hp_1)$$

then (for $i = 2, \ldots, n-2$),

$$(2 + hp_i)y_{i-1} - (4 + 2h^2 q_i)y_i + (2 - hp_i)y_{i+1} = 2h^2 r_i$$

and, finally (for $i = n - 1$),

$$(2 + hp_{n-1})y_{n-2} - (4 + 2h^2 q_{n-1})y_{n-1} = 2h^2 r_{n-1} - \beta(2 - hp_{n-1})$$

The system in matrix form is $\mathbf{Ay} = \mathbf{b}$, with $\mathbf{A}$ and $\mathbf{b}$ as follows:

$$\begin{bmatrix}
-(4 + 2h^2 q_1) & (2 - hp_1) & & & & \\
(2 + hp_2) & -(4 + 2h^2 q_2) & (2 - hp_2) & & & \\
\ddots & \ddots & & & & \\
 & (2 + hp_i) & -(4 + 2h^2 q_i) & (2 - hp_i) & & \\
 & \ddots & \ddots & & & \\
 & & (2 + hp_{n-2}) & -(4 + 2h^2 q_{n-2}) & (2 - hp_{n-2}) & \\
 & & & (2 + hp_{n-1}) & -(4 + 2h^2 q_{n-1})
\end{bmatrix}$$

$$\begin{bmatrix} 2h^2 r_1 - (2 + hp_1)\alpha & 2h^2 r_2 & \ldots & 2h^2 r_i & \ldots & 2h^2 r_{n-2} & 2h^2 r_{n-1} - (2 - hp_{n-1})\beta \end{bmatrix}^T$$

The following MATLAB script solves the problem given in Example 14.6. This script includes the Gauss-Thomas method for solving the tridiagonal system.

Finite-Difference Method for Linear ODE

```
%   S_linear_FD
%   y_xx = p(x) y_x  +  q(x) y  +  r(x),
%   y(aa) = ya;      y(bb) = yb
%************************ begin problem definition
%   this example is y_xx = 2 y_x  -2 y  +  0
aa = 0;              bb = 3;              n = 300;
p = 2*ones(1,n-1);   q = -2*ones(1,n-1);   r = zeros(1,n-1);
ya = 0.1;            yb = 0.1*exp(3)*cos(3);   % boundary conditions
%*************************** end problem definition
h = (bb-aa)/n;       h2 = h/2;       hh = h*h;   % define parameters
x = linspace(aa+h, bb, n);           % grid points, x(1), ... x(n-1)
%  upper diagonal (a), diagonal (d), lower diagonal (b)
a = zeros(1,n-1);            a(1:n-2) = 1 - p(1,1:n-2)*h2;
d = - (2 + hh*q);
b = zeros(1,n-1);            b(2:n-1) = 1 + p(1,2:n-1)*h2;
% right-hand side (c)
c(1) = hh*r(1) - ( 1 + p(1)*h2 )*ya;
c(2:n-2) = hh*r(2:n-2);
c(n-1) =  hh*r(n-1) - ( 1- p(n-1)*h2 )*yb;
%%%% begin Gauss-Thomas
m = length(d);       a(1) = a(1)/d(1);     c(1) = c(1)/d(1);
for i = 2 : m-1
    denom = d(i) - b(i)*a(i-1);
    if (denom ==0), error('zero in denominator'), end
    a(i) = a(i)/denom;
    c(i) = (c(i) - b(i)*c(i-1))/denom;
end
c(m) = (c(m) - b(m)*c(m-1))/(d(m) - b(m)*a(m-1));
y(m) = c(m);
for i = m-1: -1 : 1
    y(i) = c(i) - a(i)*y(i+1);
end
%%%% end Gauss Thomas
xx = [aa x];                 yy = [ya  y  yb];
out=[xx'  yy'];              disp(out)
plot(xx,  yy),               grid on
```

EXAMPLE 14.6 Using a MATLAB Script

The preceding MATLAB script solves the BVP

$$y_{xx} = 2y_x - 2y, \quad y(0) = 0.1, \quad y(3) = 0.1e^3 \cos(3)$$

The solution is shown in Figure 14.7.

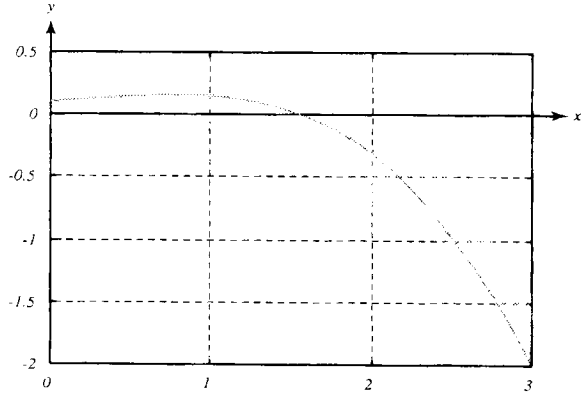

FIGURE 14.7: Linear finite-differences problem.

EXAMPLE 14.7 Deflection of a Beam, Using Finite Differences

Consider again the deflection of a simply supported beam, as described in Application 14-A and Example 14.2:

$$y'' = 10^{-7}y + 10^{-8}x(x - L), \quad 0 \le x \le 100$$

with

$$y(0) = y(100) = 0, \quad p(x) = 0, \quad q(x) = 10^{-7}, \quad r(x) = 10^{-8}[x(x - L)]$$

We let $n = 20$ be the number of subintervals.

The only portion of the script `S_linear_FD` that must be changed for this example is the section on the definition of the problem, given here.

```
%********************** begin problem definition
%   this example is y_xx = 10^(-7)*y  +  10^(-8)*x.*(x-100)
aa = 0;      bb = 100;        n = 20;      h = (bb-aa)/n
x = h:h:bb
p = zeros(1,n-1); q = 10^(-7)*ones(1,n-1);
r = 10^(-8)*x.*(x-100);
ya = 0;  yb = 0;                  % boundary conditions
%************************** end problem definition
```

The solution is the same as that shown in Figure 14.4.

14.2.2 Nonlinear ODE

We next discuss briefly the use of finite differences in nonlinear boundary-value problems. As has been remarked earlier, nonlinear problems are, in general, significantly more difficult than linear problems. We consider the nonlinear ODE-BVP of the form

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad y(b) = y_b$$

We assume that $f(x, y, y')$ has continuous derivatives that satisfy

$$0 < Q_* \leq f_y(x, y, y') \leq Q^* \quad \text{and} \quad |f_{y'}(x, y, y')| \leq P^*$$

for some constants Q_*, Q^*, and P^*.

We use a finite-difference grid with spacing $h \leq 2/P^*$. Let us apply the central difference formula for y' and y'' to obtain a nonlinear system of equations. We denote the result of evaluating f at x_i using $(y_{i+1} - y_{i-1})/2h$ for y' as f_i. The ODE then becomes the system

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

An explicit iteration scheme, analogous to the SOR method, is given by

$$y_i = \frac{1}{2(1+w)}[y_{i-1} + 2wy_i + y_{i+1} - h^2 f_i]$$

where $y_0 = y_a$ and $y_n = y_b$.

If we define the parameters $z = \dfrac{1}{2(1+w)}$ and $k = h^2$, the system of equations becomes

$$\begin{aligned} y(1) &= z(y_a + 2w\ y(1) + y(2) - k\ f(1)) \\ y(j) &= z(y(j-1) + 2w\ y(j) + y(j+1) - k\ f(j)) \quad (j = 2 : n-2) \\ y(n-1) &= z(y(n-2) + 2w\ y(n-1) + y_b - k\ f(n-1)) \end{aligned}$$

The remarkable result is that, for $w \geq h^2 Q^*/2$, the process will converge. (See Keller, 1968, Sec. 3.2.)

The next example illustrates the process for the nonlinear BVP introduced in Example 14.4.

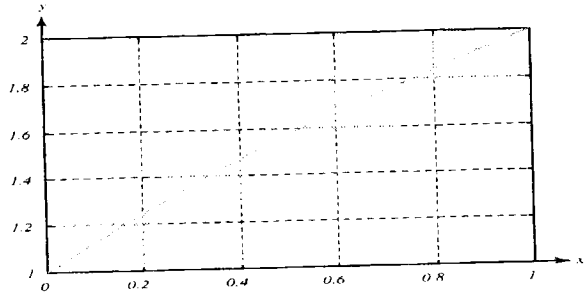

FIGURE 14.8: Solution of a nonlinear ODE-BVP using finite differences.

EXAMPLE 14.8 Solving a Nonlinear BVP by Using Finite Differences

Consider again the nonlinear BVP

$$y'' = -\frac{[y']^2}{y}, \quad y(0) = 1, \quad y(1) = 2$$

We illustrate the use of the iterative procedure just outlined, by taking a grid with $h = 1/4$, so that $x_0 = 0, x_1 = 0.25, x_2 = 0.5, x_3 = 0.75$, and $x_4 = 1$. The general form of the difference equation is

$$y_i = \frac{1}{2(1+w)}[y_{i-1} + 2wy_i + y_{i+1} - h^2 f_i]$$

where

$$f_i = -\frac{[(y_{i+1} - y_{i-1})/2h]^2}{y_i} = -\frac{y_{i+1}^2 - 2y_{i+1}y_{i-1} + y_{i-1}^2}{4h^2 y_i}$$

Substituting the rightmost expression for f_i into the equation for y_i , we obtain

$$y_i = \frac{1}{2(1+w)}[y_{i-1} + 2wy_i + y_{i+1} + \frac{y_{i+1}^2 - 2y_{i+1}y_{i-1} + y_{i-1}^2}{4y_i}]$$

The computed solution after 10 iterations agrees very closely with the exact solution, as is shown in Figure 14.8.

The following MATLAB script uses a rather coarse grid. To use a finer grid, a loop can be added to generate the equations for $i = 2, ..., n-1$, since only the first and last equations have a special form to accommodate the boundary conditions.

```
ya = 1;          yb = 2;
a = 0;           b = 1;
max_it = 10;     n = 4;
w = 0.1;         ww = 1/(2*(1+w));  h = (b-a)/n
y(1:n-1) = 1
for k = 1:max_it
   y(1) = ww*(ya+2*w*y(1)+y(2)+(ya^2-2*ya*y(2)+y(2)^2)/(4*y(1)));
   y(2) = ww*(y(1)+2*w*y(2)+y(3)+(y(1)^2-2*y(1)*y(3)+ y(3)^2)/(4*y(2)));
   y(3) = ww*(y(2)+2*w*y(3)+ yb+(y(2)^2-2*y(2)*yb + yb^2)/(4*y(3)));
end
x = [ a  a+h  a+2*h  a+3*h  b];
z = [ ya    y      yb];
% plot the computed approximate solution
plot(x, z), hold on,
% generate the exact solution, and add it to the graph.
zz = sqrt(3*x+1);   plot(x,zz), hold off
```

14.3 FUNCTION SPACE METHODS

We now consider two methods based on finding a solution function which is the best linear combination of a set of basis functions. The first method, called collocation, uses a set of basis functions which satisfy the boundary conditions and have continuous first and second derivatives. It seeks a linear combination of these basis functions that will satisfy the ODE at a selected set of x values (usually called nodes). The number of basis functions used is the same as the number of node points where the ODE must be satisfied.

The second method, Rayleigh-Ritz, also seeks a solution as a linear combination of basis functions. However, in this case the ODE-BVP is first converted to an equivalent integral functional. This functional depends on the unknown linear combination of basis functions in such a way that the minimum of the functional corresponds to the solution of the ODE-BVP. The advantage of the integral formulation is that the basis functions no longer need to be differentiable.

14.3.1 Collocation

We begin by considering the ODE

$$w''(x) + q(x)w(x) = f(x)$$

with boundary conditions $w(0) = 0$ and $w(1) = 0$.

We seek a solution of the form

$$w(x) = \sum_{i=1}^{n} c_i \phi_i(x)$$

where the ϕ_i are linearly independent functions with continuous first and second derivatives that satisfy the boundary conditions.

We partition the interval $[0, 1]$ by a set of node points

$$0 = x_0 < x_1 < x_2 < \ldots < x_n < x_{n+1} = 1$$

The coefficients $c_1, \ldots, c_n$ will be determined so that the function $w(x)$ satisfies the ODE at the node points.

In order to determine the coefficients, we compute $w''(x)$, and form the linear system of equations

$$w''(x_i) + q(x_i)w(x_i) = f(x_i), \quad \text{for} \quad i = 1, \ldots, n$$

In terms of the basis functions, we have

$$\sum_{j=1}^{n} c_j[\phi_j''(x_i) + q(x_i)\phi_j(x_i)] = f(x_i), \quad \text{for} \quad i = 1, \ldots, n$$

The linear system to be solved is $\mathbf{Ac} = \mathbf{f}$, where $\mathbf{c} = [c_1, \ldots, c_n]$, $\mathbf{f} = [f(x_1), \ldots, f(x_n)]$, and $\mathbf{A} = [a_{ij}]$

$$a_{ij} = \phi_j''(x_i) + q(x_i)\phi_j(x_i)$$

Polynomial Basis Functions

There are a number of possibilities for the basis functions. For example, we could take the functions

$$\phi_1(x) = x(1-x),\ \phi_2(x) = x^2(1-x), \ldots, \phi_n(x) = x^n(1-x)$$

With this choice,

$$w(x) = \sum_{i=1}^{n} c_i x^i (1-x) = x(1-x)(c_1 + c_2 x + \cdots + c_n x^{n-1})$$

The derivatives of the basis functions are

$$\phi_j'(x) = j x^{j-1} - (j+1)x^j, \quad \text{and} \quad \phi_j''(x) = j(j-1)x^{j-2} - (j+1)j x^{j-1}$$

Thus,

$$a_{ij} = j(j-1)(x_i)^{j-2} - (j+1)j(x_i)^{j-1} + q(x_i)(x_i)^j(1-(x_i))$$

EXAMPLE 14.9 Collocation with Polynomial Basis

Consider the simple ODE-BVP

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

with nodes at $x_1 = 0.2, x_2 = 0.4, x_3 = 0.6, x_4 = 0.8$.

```
% script for collocation example, 14_9
% basis functions
q1 = inline('x.*(1-x)'),          q2 = inline('x.^2.*(1-x)')
q3 = inline('x.^3.*(1-x)'),       q4 = inline('x.^4.*(1-x)')
% the functions q'' - q
f1 = inline('-2-x.*(1-x)'),       f2 = inline('2-6*x -x.^2.*(1-x)')
f3 = inline('6*x -12*x.^2 -x.^3.*(1-x)')
f4 = inline('12*x.^2 -20*x.^3 - x.^4.*(1-x)')
x = 0.2:0.2:0.8;
% form the matrix A, column by column
A(:, 1) = f1(x)';                 A(:, 2) = f2(x)';
A(:, 3) = f3(x)';                 A(:, 4) = f4(x)';
% form the right-hand side of the linear system,
% by evaluating the right-hand side of the ODE at the node points
g(:, 1) = (x.*(x-1))'
% solve the linear system to find the coefficients
c = A\g,                          % plot the solution
xx = linspace(0, 1);
yy = c(1)*q1(xx) + c(2)*q2(xx) + c(3)*q3(xx) +c(4)*q4(xx);
plot(xx,yy),  hold on,            % compute and plot exact solution
A = 2/(exp(1) + 1);               B = 2*exp(1)/(exp(1) + 1)
zz = A*exp(xx) + B*exp(-xx) +xx.^2 - xx + 2;
plot(xx, zz),          hold off
```

Spline Basis Functions

Another convenient choice for basis functions are the functions known as B-splines (i.e., basis splines). These are cubic splines defined to be nonzero only on the subinterval $[x_{i-2}, x_{i+2}]$. We assume for convenience that the node points are evenly spaced.

The general form of a suitable cubic spline B_i is

$$\begin{aligned}
&\frac{1}{4h^3}(x - x_{i-2})^3, && x_{i-2} \le x \le x_{i-1} \\
&\frac{1}{4} + \frac{3}{4h}(x - x_{i-1}) + \frac{3}{4h^2}(x - x_{i-1})^2 - \frac{3}{4h^3}(x - x_{i-1})^3, && x_{i-1} \le x \le x_i \\
&\frac{1}{4} + \frac{3}{4h}(x_{i+1} - x) + \frac{3}{4h^2}(x_{i+1} - x)^2 - \frac{3}{4h^3}(x_{i+1} - x)^3, && x_i \le x \le x_{i+1} \\
&\frac{1}{4h^3}(x_{i+2} - x)^3, && x_{i+1} \le x \le x_{i+2}
\end{aligned}$$

However, the splines B_1 and B_n must be modified so that they satisfy the boundary conditions at $x_0 = 0$ and $x_{n+1} = 1$. We define

$$P_1 = B_1 - 0.25B_0 \quad \text{and} \quad P_n = B_n - 0.25B_{n+1}$$

Because the spline B_i has support (i.e., is nonzero) only on the interval $[x_{i-2}, x_{i+2}]$, the coefficient matrix for the linear system $\mathbf{Ac} = \mathbf{f}$ is tridiagonal. The coefficient matrix is closely related to that found using finite differences.

The following MATLAB function `BsplineS` evaluates the k^{th} `Bspline` at points x within the interval $[0, 1]$ at n evenly spaced node points.

Evaluate Bsplines

```
function y = BsplineS(k, x, n)
h = 1/n;    m = length(x);   % the node points are k*h
for j = 1:m
   if  (x(j) <= (k-2)*h),     y(j) = 0;
   elseif  ( (k-2)*h < x(j)  & x(j) <= (k-1)*h)
      y(j) = 0.25/h^3*(x(j)-(k-2)*h)^3;
   elseif ( (k-1)*h < x(j)  & x(j) <= k*h)
      y(j) = 0.25*(1 +(3/h)*(x(j)-(k-1)*h) +(3/h^2)*(x(j)-(k-1)*h)^2 ...
                -(3/h^3)*(x(j)-(k-1)*h)^3);
   elseif (k*h < x(j)  & x(j) <= (k+1)*h)
      y(j) = 0.25*(1 +(3/h)*((k+1)*h-x(j)) +(3/h^2)*((k+1)*h-x(j))^2 ...
                -(3/h^3)*((k+1)*h-x(j))^3);
   elseif ((k+1)*h < x(j)  & x(j) <= (k+2)*h)
      y(j) = 0.25/h^3*((k+2)*h-x(j))^3;
   else,    y(j) = 0;      end
end
```

The function D2_BSplineS evaluates the second derivative of the Bspline functions.

Evaluate Second Derivative of Bsplines

```
function y = D2_BSplineS(k, x, n)
h = 1/n;   m = length(x);    % the node points are k*h
for j = 1:m
    if (x(j) <= (k-2)*h),  y(j) = 0;
    elseif((k-2)*h < x(j) & x(j) <= (k-1)*h)
        y(j) = 1.5/h^3*(x(j) -(k-2)*h);
    elseif ((k-1)*h < x(j) & x(j) <= k*h)
        y(j) = 0.25*( 6/h^2 - (18/h^3)*(x(j) -(k-1)*h));
    elseif (k*h < x(j) & x(j) <= (k+1)*h)
        y(j) = 0.25*(6/h^2 -(18/h^3)*((k+1)*h - x(j)));
    elseif((k+1)*h < x(j) & x(j) <= (k+2)*h)
        y(j) = 1.5/h^3*((k+2)*h - x(j) );
    else,  y(j) = 0;   end
end
```

The following script illustrates the use bsplines to solve the ODE-BVP from Example 14.9; the script forms the linear system A c = g, solves the system, evaluates the solution (q = c1 B1 + ... + c9 B9), and plots the result.

Solve Example 14.9 Using Bsplines

```
x = 0.1:0.1:0.9
for i = 1:9
    a(i, 1) = D2_BSplineS(1,x(i),10) -0.25*D2_BSplineS(0,x(i),10)...
        -(BSplineS(1, x(i),10) -0.25*BSplineS(0,x(i),10));
    for k = 2:8
        a(i,k) = D2_BSplineS(k,x(i),10)- BSplineS(k,x(i),10);
    end
    a(i, 9)= D2_BSplineS(9,x(i),10) -0.25*D2_BSplineS(10,x(i),10)...
        -(BSplineS(9,x(i),10) -0.25*BSplineS(10,x(i),10));
end
a;              g(:, 1) = (x.*(x-1))';   c = a\g;
xx = 0:0.01:1;  m = length(xx);          n = length(c);
for i = 1:m
    q(i, 1) = c(1)*(BSplineS(1,xx(i),10) -0.25*BSplineS(0,xx(i),10));
    for k = 2:8
        q(i,k) = c(k)*BSplineS(k,xx(i),10);
    end
    q(i, 9)= c(9)*(BSplineS(9,xx(i),10) -0.25*BSplineS(10,xx(i),10));
end
zz = sum(q, 2);   plot(xx,zz)
```

Evaluate First Derivative of Bsplines

```
function y = D1_BSplineS(k, x, n)
h = 1/n; m = length(x); % the node points are k*h
for j = 1:m
    if (x(j) <= (k-2)*h),  y(j) = 0;
    elseif((k-2)*h < x(j) & x(j) <= (k-1)*h), y(j) = 0.75/h^3*(x(j)-(k-2)*h)^2;
    elseif ((k-1)*h < x(j) & x(j) <= k*h)
        y(j) = 0.25*(3/h+(6/h^2)*(x(j)-(k-1)*h)-(9/h^3)*(x(j)-(k-1)*h)^2);
    elseif (k*h < x(j) & x(j) <= (k+1)*h)
        y(j) = 0.25*(-3/h-(6/h^2)*((k+1)*h-x(j))+(9/h^3)*((k+1)*h-x(j))^2);
    elseif((k+1)*h < x(j) & x(j) <= (k+2)*h), y(j) = -0.75/h^3*((k+2)*h-x(j))^2;
    else, y(j) = 0;
    end
end
```

14.3.2 Rayleigh-Ritz

We begin by considering the ODE

$$w''(x) + q(x)w(x) = f(x)$$

with boundary conditions $w(0) = 0$ and $w(1) = 0$.

The ODE is known as the Euler equation for the variational problem, to minimize

$$\int_0^1 [w'(x)]^2 - q(x)[w(x)]^2 - 2w(x)f(x)dx$$

over a set of suitably smooth functions that vanish at the ends of the interval.

The minimum is sought over the set of all functions of the form

$$w = \sum_{i=1}^{n} c_i \phi_i$$

We partition the interval $[0, 1]$ by a set of node points

$$0 = x_0 < x_1 < x_2 < \ldots < x_n < x_{n+1} = 1$$

A possible set of basis functions are the piecewise linear hat functions:

$$\phi_j = \begin{cases} 0 & \text{at} \quad x = 0, \ldots, x_{j-1} \\ 1 & \text{at} \quad x = x_j \\ 0 & \text{at} \quad x = x_{j+1}, \ldots, x_n \end{cases}$$

Each function is zero, except on the interval $[x_{j-1}, x_{j+1}]$.

To minimize

$$I(w) = \int_0^1 [\sum_{i=1}^{n} c_i \phi_i']^2 - q(x)[\sum_{i=1}^{n} c_i \phi_i]^2 - 2f(x) \sum_{i=1}^{n} c_i \phi_i \, dx$$

as a function of the coefficients c_j we must have $\partial I/\partial c_j = 0$ for $j = 1, \ldots, n$. Thus

$$\frac{\partial I}{\partial c_j} = \int_0^1 2\sum_{i=1}^{n} \phi_j' c_i \phi_i' - 2q(x)\sum_{i=1}^{n} c_i \phi_i \phi_j - 2f(x)\phi_j \, dx = 0$$

Rearranging, we have

$$\sum_{i=1}^{n} [\int_0^1 \phi_j' \phi_i' - q(x)\phi_i \phi_j] c_i dx = \int_0^1 f(x)\phi_j \, dx$$

This gives a linear system of equations for the unknown coefficients $\mathbf{Ac} = \mathbf{b}$, where the matrix $\mathbf{A}$ has elements

$$a_{i,j} = \int_0^1 \phi_j' \phi_i' - q(x)\phi_i \phi_j \, dx$$

and the vector $\mathbf{b}$ is given by

$$b_j = \int_0^1 f(x)\phi_j \, dx$$

Note that for the piecewise linear hat functions defined above, most of the products of functions that occur in the integrals defining the matrix $\mathbf{A}$ are zero.

The integrals that must be computed involve

$$\int_0^1 q(x)\,dx, \quad \int_0^1 xq(x)\,dx, \quad \int_0^1 x^2 q(x)\,dx, \quad \int_0^1 f(x)\,dx, \quad \text{and} \quad \int_0^1 xf(x)\,dx$$

Depending on the form of $q(x)$ and $f(x)$, these may be computed analytically, or numerically, or by using piecewise linear interpolation of $q(x)$ and $f(x)$ at the node points, and then integrating the approximation.

More General Form

We consider the ODE

$$-(p(x)u'(x))' + q(x)u(x) = g(x, u(x))$$

with boundary conditions $u(a) = \alpha$ and $u(b) = \beta$.

The transformation

$$v(x) = \alpha\frac{b-x}{b-a} + \beta\frac{a-x}{a-b}$$

allows us to convert the problem above to the more convenient problem with vanishing boundary conditions; if $u(x)$ is the solution of the ODE-BVP above, then $y(x) = u(x) - v(x)$ is the solution of

$$-(p(x)y'(x))' + q(x)y(x) = f(x, y(x))$$

with $y(a) = 0,\ y(b) = 0$.

This problem corresponds to finding the minimum of the quadratic functional

$$\int_a^b p(x)[u'(x)]^2 + q(x)[u(x)]^2 - 2u(x)f(x)\,dx$$

over the set of all suitable functions u, i.e., u is a linear combination of a set of basis functions that are integrable and satisfy the boundary conditions.

14.4 CHAPTER WRAP-UP

SUMMARY

Linear Shooting

$$y'' = p(x)y' + q(x)y + r(x), \quad y(a) = y_a, \quad y(b) = y_b$$

Solve the IVP

$$u'' = p(x)u' + q(x)u + r(x), \quad u(a) = y_a, \quad u'(a) = 0$$

and (assuming $u(b) \neq y_b$)

$$v'' = p(x)v' + q(x)v, \quad v(a) = 0, \quad v'(a) = 1$$

If $v(b) \neq 0$, the solution of the original two-point BVP is given by

$$y(x) = u(x) + \frac{y_b - u(b)}{v(b)}v(x)$$

Nonlinear Shooting

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad h(y(b), y'(b)) = y_b$$

Initialize $t(1) = 0$, $t(2) = 1$.
Begin iterations, starting with $j = 2$ and continuing until $|m(j) <$ tol.

Solve: $u'' = f(x, u, u'), \quad u(a) = y_a, \quad u'(a) = t(j-1)$

Compute: $m(j) = y_b - h(u(b), u'(b))$

Update:

$$t(j+1) = t(j) - \frac{[t(j) - t(j-1)]m(j)}{m(j) - m(j-1)}$$

End iteration loop.

Finite-Difference Method for Linear BVP

$$y'' = p(x)y' + q(x)y + r(x)$$

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

Finite-Difference Method for Nonlinear BVP

$$y'' = f(x, y, y'), \quad y(a) = y_a, \quad y(b) = y_b$$

Assuming that f_y and $f_{y'}$ are suitably bounded, the ODE then becomes the system

$$y_{i+1} - 2y_i + y_{i-1}h^2 - f_i = 0$$

An explicit iteration scheme, analogous to the SOR method, is given by

$$y_i = \frac{1}{2(1+w)}[y_{i-1} + 2wy_i + y_{i+1} - h^2 f_i]$$

where $y_0 = y_a$ and $y_n = y_b$. The process will converge for $w \geq h^2 Q^*/2$.

SUGGESTIONS FOR FURTHER READING

For the basic theory, see suggestions from Chapter 12. See also the following:

- Ascher, U. M., R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM (reprint of Prentice Hall, 1988).
- Fox, L., *Numerical Solution of Two-Point Boundary Value Problems in Ordinary Differential Equations*, Dover, New York, 1990.
- Keller, H. B., *Numerical Methods for Two-point Boundary-value Problems*, Blaisdell, Waltham, MA, 1968.
- Troutman, J. L., and M. Bautista, *Boundary Value Problems of Applied Mathematics*, Prindle, Weber & Schmidt Publishing, 1994.

For further discussion of applications of two-point boundary-value problems, see

- Haberman, R., *Elementary Applied Partial Differential Equations, with Fourier Series and Boundary Value Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- Hanna, O. T., and O. C. Sandall, *Computational Methods in Chemical Engineering*, Prentice Hall, Upper Saddle River, NJ, 1995.
- Hornbeck, R. W., *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
- Inman, D. J., *Engineering Vibration*, Prentice Hall, Upper Saddle River, NJ, 1996.
- Jaeger, J. C., *An Introduction to Applied Mathematics*, Clarendon Press, Oxford, 1951.
- Roberts, C. E., *Ordinary Differential Equations: A Computational Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

For discussion of Rayleigh-Ritz, see

- Golub, G. H., and J. M. Ortega, *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*, 1992, pp. 179–186; they give the reference for Rayleigh-Ritz as Strang and Fix [1973].
- Schultz, M., *Spline Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
- Stoer, J., and R. Bulirsch, *Introduction to Numerical Analysis*, pp. 540–545, 1980.

See the Bibliography for full details.