CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Two-point boundary value problems

We now consider a second order ODE in the form

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

Here $y(x)$ is the unknown function defined on the interval $a \leq x \leq b$. The values of y at the boundary points $x = a, x = b$ are given.

Such differential equations arise in many physical models.

For example, the model for an elastic string:

$$y'' = ky + mx(x - L), \qquad y(0) = 0, \quad y(L) = 0.$$

Note that this equation is linear.
One can also have non-linear equations. For example:

$$-(y')^2 - 2b(x)y + 2yy'' = 0, \qquad y(0) = 1, \quad y(1) = a.$$

We study two numerical methods for this two-point boundary value problem:

- Shooting method: based on ODE solvers;
- Finite Difference Method (FDM).

Shooting method

Given some two-point boundary value problem on $a \le x \le b$.

Main algorithm:

- Solve two-point boundary value problem as an initial value problem, with initial data given at $x = a$ (a guess).
- Compute the solution and the value in the solution at $x = b$.
- Compare this with the given boundary condition at $x = b$. Then adjust your guess at $x = a$ and iterate if needed.

It makes a difference if the differential equation is linear and nonlinear. The linear case is simpler.

Linear Shooting.

Let's consider the linear problem in the general form:

$$y''(x) = u(x) + v(x)y(x) + w(x)y'(x), \qquad y(a) = \alpha, \quad y(b) = \beta. \quad (1)$$

Let $\bar{y}$ solve the same equation, but with initial conditions:

$$\bar{y}''(x) = u(x) + v(x)\bar{y}(x) + w(x)\bar{y}'(x), \qquad \bar{y}(a) = \alpha, \quad \bar{y}'(a) = 0. \quad (2)$$

Note that $\bar{y}'(a) = 0$ is the "guess" we make.

Let $\tilde{y}$ solve the same equation, but with different initial conditions:

$$\tilde{y}''(x) = u(x) + v(x)\tilde{y}(x) + w(x)\tilde{y}'(x), \qquad \tilde{y}(a) = \alpha, \quad \tilde{y}'(a) = 1. \quad (3)$$

Note that $\tilde{y}'(a) = 1$ is the other "guess" we make.

As we will see later, it doesn't matter with guesses we make here. Any numbers will work, as long as they are different for $\bar{y}$ and $\tilde{y}$.

Note that both equations (2) and (3) can be written into a system of first order ODEs, and can be solved by Matlab ODE solver.

Assume now both equations (2) and (3) are now solved on the interval $x \in [a, b]$, (say, by some ODE Solver in Matlab), such that the values $\bar{y}(b)$ and $\tilde{y}(b)$ are computed.

Now let

$$y(x) = \lambda \cdot \bar{y}(x) + (1-\lambda) \cdot \tilde{y}(x) \tag{4}$$

where λ is a constant to be determined, such that $y(x)$ in (4) becomes the solution for (1).

We now check which equation the y in (4) solves. We have

$$\begin{aligned} y'' &= \lambda \cdot \bar{y}''(x) + (1-\lambda) \cdot \tilde{y}''(x) \\ &= \lambda(u + v\bar{y} + w\bar{y}') + (1-\lambda)(u + v\tilde{y} + w\tilde{y}') \\ &= u + v(\lambda\bar{y} + (1-\lambda)\tilde{y}) + w(\lambda\bar{y}' + (1-\lambda)\tilde{y}') \\ &= u + vy + wy'. \end{aligned}$$

We see that this y solve the equation (1) for any choices of λ.

We now check the boundary conditions. At $x = a$, we have

$$y(a) \;=\; \lambda\bar{y}(a) + (1-\lambda)\tilde{y}(a) = \lambda\alpha + (1-\lambda)\alpha = \alpha.$$

The boundary condition is satisfied for any choices of λ.

At $x = b$, we have

$$y(b) \;=\; \lambda \bar{y}(b) + (1-\lambda)\tilde{y}(b).$$

Since we must require $y(b) = \beta$, this gives us a equation to find λ,

$$\lambda \bar{y}(b) + (1-\lambda)\tilde{y}(b) = \beta, \qquad \Rightarrow \quad \lambda = \frac{\beta - \tilde{y}(b)}{\bar{y}(b) - \tilde{y}(b)}. \tag{5}$$

Conclusion. The $y(x)$ given as

$$y(x) = \lambda \cdot \bar{y}(x) + (1-\lambda) \cdot \tilde{y}(x)$$

with λ given as

$$\lambda = \frac{\beta - \tilde{y}(b)}{\bar{y}(b) - \tilde{y}(b)}$$

is the solution of the BVP in (1).

Practical issues.

One can solve for $\bar{y}$ and $\tilde{y}$ as initial value problems, even simultaneously. Let

$$y_1 = \bar{y}, \quad y_2 = \bar{y}', \quad y_3 = \tilde{y}, \quad y_4 = \tilde{y}',$$

then

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \end{pmatrix} = \begin{pmatrix} y_2 \\ u + vy_1 + wy_2 \\ y_4 \\ u + vy_3 + wy_4 \end{pmatrix}, \qquad \text{IC: } \begin{pmatrix} y_1(a) \\ y_2(a) \\ y_3(a) \\ y_4(a) \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \alpha \\ 1 \end{pmatrix}.$$

This is a 4×4 system of first order ODEs, which could be solved in Matlab with efficient ODE solvers. The values $\bar{y}(b), \tilde{y}(b)$ will be the last element in the vector y_1, y_3, respectively.

Some extensions of Linear Shooting

Case 1. We now consider the effect of different boundary conditions.

$$y''(x) = u(x) + v(x)y(x) + w(x)y'(x), \qquad y(a) = \alpha, \quad y'(b) = \beta.$$

A same shooting method can be designed, with minimum adjustment for the boundary condition at $x = b$.
The function y in the general algorithm will satisfy the differential equation as well as the boundary condition at $x = a$.
For the boundary condition at $x = b$, we must require

$$y'(b) = \lambda\bar{y}'(b) + (1-\lambda)\tilde{y}'(b) = \beta, \qquad \Rightarrow \quad \lambda = \frac{\beta - \tilde{y}'(b)}{\bar{y}'(b) - \tilde{y}'(b)}.$$

Case 2. Consider a higher order linear equation

$$y''' = f(x, y, y', y''), \qquad y(a) = \alpha, \quad y'(a) = \gamma, \quad y(b) = \beta. \tag{1}$$

Here $f(x, y, y', y'')$ is an affine function in y, y', y''.
A shooting method can be designed as follows. Let $\bar{y}$ and $\tilde{y}$ solve the same equation (1), but with initial conditions:

$$\bar{y}(a) = \alpha, \quad \bar{y}'(a) = \gamma, \quad \bar{y}''(a) = 0. \tag{2}$$

$$\tilde{y}(a) = \alpha, \quad \tilde{y}'(a) = \gamma, \quad \tilde{y}''(a) = 1. \tag{3}$$

Assume now we solved both equations (2) and (3), and the values $\bar{y}(b)$ and $\tilde{y}(b)$ are computed. Let

$$y(x) = \lambda \cdot \bar{y}(x) + (1 - \lambda) \cdot \tilde{y}(x) \tag{4}$$

where λ is a constant to be determined, such that $y(x)$ in (4) becomes the solution for (1).

It is easy to check that y solves the equation in (1), and satisfies the boundary conditions $y(a) = \alpha, y'(a) = \gamma$, due to the linear properties.
It remains to check the last boundary condition at $x = b$.
At $x = b$, we have

$$y(b) \quad = \quad \lambda \bar{y}(b) + (1 - \lambda)\tilde{y}(b) = \beta.$$

which give the same formula to compute λ, i.e,

$$\lambda = \frac{\beta - \tilde{y}(b)}{\bar{y}(b) - \tilde{y}(b)}.$$

Non-linear Shooting.

We now consider the general nonlinear equation

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

Let $\tilde{y}$ solve the IVP

$$\tilde{y}'' = f(x, \tilde{y}, \tilde{y}'), \qquad \tilde{y}(a) = \alpha, \quad \tilde{y}'(a) = z. \tag{1}$$

Note that the condition $\tilde{y}'(a) = z$ is our guess.
The solution of (1) depends on z. Denote

$$\tilde{y}(b) \doteq \phi(z),$$

where ϕ is a non-linear function denoting the relation on how the value $\tilde{y}(b)$ depend on z. We need to find the value z such that

$$\phi(z) = \beta, \qquad \Rightarrow \quad \phi(z) - \beta = 0.$$

Since $\phi(z)$ is a non-linear function, we need to find a root for the above nonlinear equation. One can use secant method!

The algorithm goes as follows.

(1). Choose some initial guess z_1, z_2, and compute the values

$$\phi_1 = \phi(z_1), \qquad \phi_2 = \phi(z_2)$$

(2) Then, the next value z_3 could be computed by a secant step:

$$z_3 = z_2 + (\beta - \phi_2) \cdot \frac{z_2 - z_1}{\phi_2 - \phi_1}.$$

(3). One can then iterate and get values $z_4, z_5, \cdots$ until converges, for example, until $|\phi(z_n) - \beta| \leq \text{tol}$.

Remark:
The nonlinear shooting could be used on linear problem. In that case, one iteration is enough.

Extensions to other types of boundary conditions, as well as higher order nonlinear equations, can be made in a similar way as we did for the linear case. One needs to adopt a secant iteration around the shooting. Students are encouraged to working out the details on their own.

FDM: Finite Difference Methods

We consider the linear problem, in the general form, with Dirichlet boundary condition

$$y''(x) = u(x) + v(x)y(x) + w(x)y'(x), \qquad y(a) = \alpha, \quad y(b) = \beta. \quad (1)$$

Discretize the domain: Choose n, make a uniform grid:

$$h = \frac{b-a}{n}, \quad x_i = a + ih, \quad i = 0, 1, 2, \cdots, n, \quad x_0 = a, \quad x_n = b$$

Goal: Find approximations $y_i \approx y(x_i)$.

Tool: finite difference approximation to the derivatives:

$$\begin{aligned} y'(x_i) &\approx \frac{y(x_{i+1}) - y(x_{i-1})}{x_{i+1} - x_{i-1}} = \frac{y_{i+1} - y_{i-1}}{2h}, \\ y''(x_i) &\approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}. \end{aligned}$$

Plug these into the ODE $y''(x) = u(x) + v(x)y(x) + w(x)y'(x)$, we get

$$\frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right) = u_i + v_i y_i + \frac{w_i}{2h}\left(y_{i+1} - y_{i-1}\right),$$

for $i = 1, 2, \cdots n-1$, where we used the notation

$$u_i = u(x_i), \quad v_i = v(x_i), \quad w_i = w(x_i)$$

We can clean up a bit, and get

$$-(1+\frac{h}{2}w_i)y_{i-1}+(2+h^2v_i)y_i-(1-\frac{h}{2}w_i)y_{i+1}=-h^2u_i. \qquad (2)$$

Calling

$$a_i=-(1+\frac{h}{2}w_i), \quad d_i=(2+h^2v_i), \quad c_i=-(1-\frac{h}{2}w_i), \quad b_i=-h^2u_i$$

discrete equations (2) can be written in a simpler way

$$a_iy_{i-1}+d_iy_i+c_iy_{i+1}=b_i, \qquad i=1,2,\cdots,n-1. \qquad (3)$$

By the boundary conditions $y_0=\alpha, y_n=\beta$, the first and last equation in (3) become

$$\begin{aligned} d_1y_1+c_1y_2 &= b_1-a_1\alpha, \\ a_{n-1}y_{n-2}+d_{n-1}y_{n-1} &= b_{n-1}-c_{n-1}\beta. \end{aligned}$$

The discrete equations

$$a_i y_{i-1} + d_i y_i + c_i y_{i+1} = b_i, \qquad i = 1, 2, \cdots, n-1,$$

lead to a tri-diagonal system of linear equations.

$$A\vec{y} = \vec{b}$$

with

$$\begin{pmatrix} d_1 & c_1 & & & \\ a_2 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-2} & d_{n-2} & c_{n-2} \\ & & & a_{n-1} & d_{n-1} \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} b_1 - a_1\alpha \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} - c_{n-1}\beta \end{pmatrix}$$

One desirable property to have is that the coefficient matrix A being diagonal dominant. In this case, Gaussian elimination without pivoting can be used to solve the linear system. Also, any of the iterative solvers we have learned will converge with any initial guess.

We see that A is strictly diagonal dominant if $|d_i| > |a_i| + |c_i|$, i.e., if

$$\left|2 + h^2 v_i\right| > |1 + hw_i/2| + |1 - hw_i/2|$$

which holds if we assume that every term in the absolute value sign above is positive.

We may now require

$$v(x) > 0, \qquad h \leq \frac{2}{\max_x |w(x)|}.$$

Example

Set up the FDM for the problem

$$y'' = -4(y - x), \qquad y(0) = 0, \quad y(1) = 2.$$

Note that the exact solution is $y(x) = (1/\sin 2)\sin 2x + x$.

Answer. Fix an n, we make a uniform grid:

$$h = \frac{1}{n}, \quad x_i = ih, \quad i = 0, 1, 2, \cdots n.$$

Central Finite Difference for the second derivative $y''(x_i)$ gives us

$$y''(x_i) \approx \frac{1}{h^2}\left(y_{i-1} - 2y_i + y_{i+1}\right) = -4y_i + 4x_i.$$

After some cleaning up, we get

$$y_{i-1} - (2 - 4h^2)y_i + y_{i+1} = 4h^2 x_i, \qquad i = 1, 2, \cdots, n-1,$$

with boundary conditions

$$y_0 = 0, \quad y_n = 2.$$

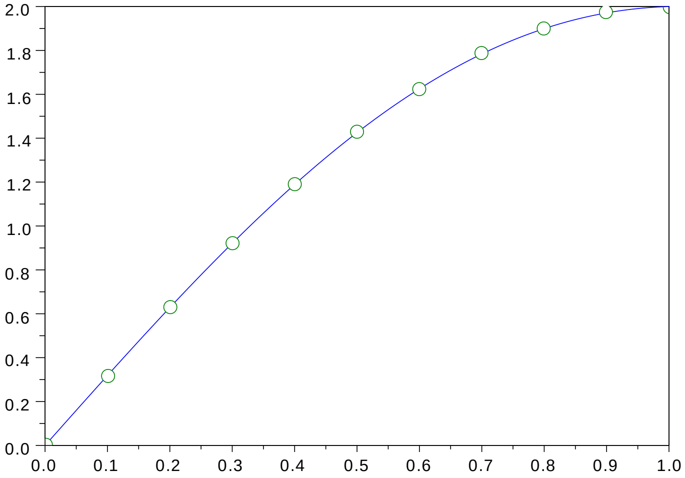
We end up with the tri-diagonal system $A\vec{y} = \vec{b}$:

$$A = \begin{pmatrix} -2+4h^2 & 1 & & & \\ 1 & -2+4h^2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2+4h^2 & 1 \\ & & & 1 & -2+4h^2 \end{pmatrix}$$

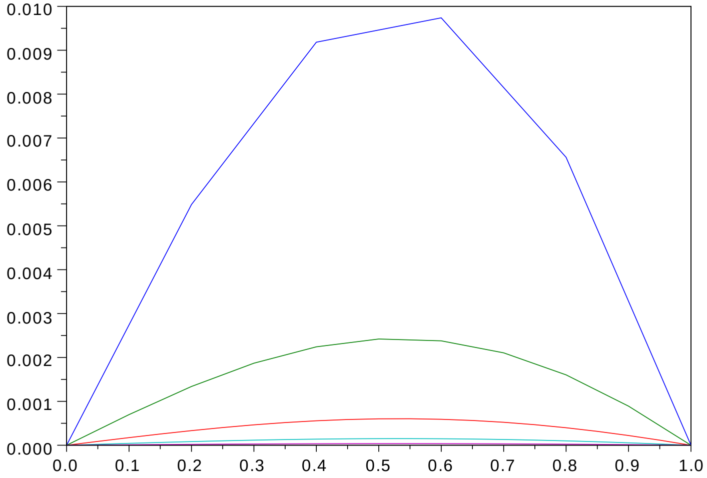
$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}, \qquad \vec{b} = \begin{pmatrix} 4h^2 x_1 - a \\ 4h^2 x_2 \\ \vdots \\ 4h^2 x_{n-2} \\ 4h^2 x_{n-1} - b \end{pmatrix}.$$

The system is "almost" diagonal dominant. It can be solved by Gaussian Elimination efficiently. It could also be solved by any of our iterative methods, and they will all converge.

The graph of the approximate solution with $n = 10$, together with the exact solution:

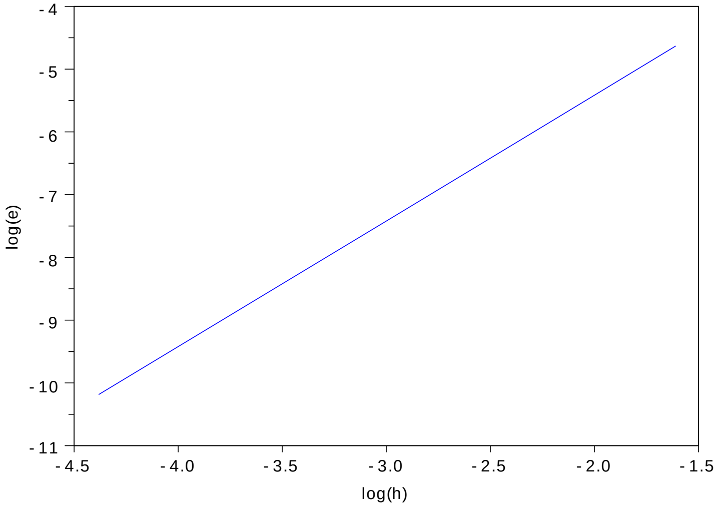


Plot of the error $e_i = y_i - y(x_i)$, for $n = 5, 10, 20, 40, 80$:



We see clearly that error decreases as n increases.

Plot for the $\log(\max(e_i))$ against $\log(h)$:



We see that we get a straight line with slope 2.

A straight line with slope 2 in a $\log(e) - \log(h)$ plot indicates a second order method.

Indeed, assume the method is of order m, then the error is approximately

$$e = Ch^m$$

for some constant C.
Taking log on both sides, we get

$$\log(e) = \log(C) + m \log(h),$$

where m is exactly the slope in the $\log(e)$ vs $\log(h)$ plot.

Neumann Boundary conditions.

Neumann Boundary condition is when the derivative of the unknown is given at the boundary. For example, we consider the Poisson equation in 1D:

$$u''(x) = f(x), \qquad u'(0) = a, \quad u(1) = b. \tag{1}$$

Note the condition at $x = 0$ is given as the derivative of the unknown $u(x)$.

Uniform grid:
Fix an N, let $h = 1/N$ and $x_i = ih$ for $i = 0, 1, 2, \cdots, N$, and let $u_i \approx u(x_i)$ be the approximation.

We now have N unknowns, namely $u_0, u_1, \cdots, u_{N-1}$. We also have $u_N = b$ which is the Dirichlet boundary condition.

We set up the finite difference scheme

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i), \quad \Rightarrow \quad u_{i-1} - 2u_i + u_{i+1} = h^2 f(x_i), \tag{2}$$

which holds for $i = 1, 2 \cdots, N-1$.

Discussion:
Since the central finite difference approximation to $u''(x)$ is second order, we want also to approximate the boundary condition $u'(0) = a$ with a second order finite difference.

The central finite difference for $u'(0)$ is second, but it requires information at $x = -h$.

To handle this, we add an additional grid point outside the domain, $x_{-1} = x_0 - h = -h$.

This point is called *ghost boundary*.

Writing $u_{-1} \approx u(x_{-1})$, we now write out the central finite different for the boundary condition:

$$\frac{u_1 - u_{-1}}{2h} = a, \quad \Rightarrow \quad u_{-1} = u_1 - 2ha. \tag{3}$$

We also write out (2) at $i = 0$:

$$u_{-1} - 2u_0 + u_1 = h^2 f(x_0). \tag{4}$$

Plugging (3) into (4), we get the discrete equation for $i = 0$

$$u_1 - 2ha - 2u_0 + u_1 = h^2 f(x_0), \quad \Rightarrow \quad -2u_0 + 2u_1 = h^2 f(x_0) + 2ha. \tag{5}$$

The equation $i = N - 1$ is slightly different due to the boundary condition $u_N = b$:

$$u_{N-2} - 2u_{N-1} = h^2 f(x_{N-1}) - b. \tag{6}$$

Collecting all the equation with $i = 0, 1, 2, \cdots, N-1$, we obtain the following tri-diagonal system of linear equations:

$$\begin{pmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} h^2 f(x_0) + 2ha \\ h^2 f(x_1) \\ \vdots \\ h^2 f(x_{N-2}) \\ h^2 f(x_{N-1}) - b \end{pmatrix}.$$

Remark.
A Neumann boundary condition at $x = 1$ would be treated in a completely similar way, by adding a ghost boundary point $x_{N+1} = x_N + h$. We omit the details. Students are encouraged to work out the details.

Robin Boundary conditions

For some problem, one might have boundary conditions involving both the unknown and its derivative. These are called **Robin boundary conditions**.

Consider again the 1D Poisson, but with Robin boundary condition at $x = 0$:

$$u''(x) = f(x), \qquad u'(0) - u(0) = \gamma, \quad u(1) = b. \tag{1}$$

Using again the ghost boundary x_{-1}, we can approximate the Robin condition by a second order central finite difference

$$\frac{u_1 - u_{-1}}{2h} - u_0 = \gamma, \quad \Rightarrow \quad u_{-1} = u_1 - u_0 - 2h\gamma.$$

Plugging this into the finite difference scheme at $i = 0$, we get the discrete equation for $i = 0$

$$(u_1 - u_0 - 2h\gamma) - 2u_0 + u_1 = h^2 f(x_0), \quad \Rightarrow \quad -3u_0 + 2u_1 = h^2 f(x_0) + 2h\gamma.$$

The rest of the equations remain unchanged.

We end up with the following tri-diagonal system of linear equations.

$$\begin{pmatrix} -3 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} h^2 f(x_0) + 2h\gamma \\ h^2 f(x_1) \\ \vdots \\ h^2 f(x_{N-2}) \\ h^2 f(x_{N-1}) - b \end{pmatrix}.$$