LECTURE 5: *Finite Difference Methods for Differential Equations*

1.7.3 Application: Boundary Value Problems

Example 1.6 (Dirichlet boundary conditions)**.** Suppose we want to solve the differential equation

$$-u''(x) = g(x), \qquad x \in [0,1]$$

for the unknown function u, subject to the *Dirichlet* boundary conditions

$$u(0) = u(1) = 0.$$

One common approach to such problems is to approximate the solution u on a uniform grid of points

$$0 = x_0 < x_1 < \cdots < x_n = 1$$

with $x_j = j/N$.

We seek to approximate the solution $u(x)$ at each of the grid points $x_0, \ldots, x_n$. The Dirichlet boundary conditions give the end values immediately:

$$u(x_0) = 0, \qquad u(x_n) = 0.$$

At each of the interior grid points, we require a local approximation of the equation

$$-u''(x_j) = g(x_j), \qquad j = 1, \ldots, n-1.$$

For each, we will (implicitly) construct the quadratic interpolant $p_{2,j}$ to $u(x)$ at the points x_{j-1}, x_j, and x_{j+1}, and then approximate

$$-p''_{2,j}(x_j) \approx -u''(x_j) = g(x_j).$$

Aside from some index shifting, we have already constructed $p''_{2,j}$ in equation (1.21):

$$p''_{2,j}(x) = \frac{u(x_{j-1}) - 2u(x_j) + u(x_{j+1})}{h^2}.$$

Just one small caveat remains: we cannot construct $p''_{2,j}(x)$, *because we do not know the values of* $u(x_{j-1})$*,* $u(x_j)$*, and* $u(x_{j+1})$: finding those values is the point of our entire endeavor. Thus we define approximate values

$$u_j \approx u(x_j), \qquad j = 1, \ldots, n-1.$$

and will instead use the polynomial $p_{2,j}$ that interpolates u_{j-1}, u_j, and u_{j+1}, giving

$$p''_{2,j}(x) = \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2}. \tag{1.22}$$

Let us accumulate all our equations for $j = 0, \ldots, n$:

$$\begin{aligned} u_0 &= 0 \\ u_0 - 2u_1 + u_2 &= -h^2 g(x_1) \\ u_1 - 2u_2 + u_3 &= -h^2 g(x_2) \\ &\vdots \\ u_{n-3} - 2u_{n-2} + u_{n-1} &= -h^2 g(x_{n-2}) \\ u_{n-2} - 2u_{n-1} + u_n &= -h^2 g(x_{n-1}) \\ u_n &= 0. \end{aligned}$$

Notice that this is a system of $n+1$ linear equations in $n+1$ variables $u_0, \ldots, u_{n+1}$. Thus we can arrange this in matrix form as

$$\begin{bmatrix} 1 & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} 0 \\ -h^2 g(x_1) \\ -h^2 g(x_2) \\ \vdots \\ -h^2 g(x_{n-2}) \\ -h^2 g(x_{n-1}) \\ 0 \end{bmatrix}, \tag{1.23}$$

where the blank entries are zero. Notice that the first and last entries are trivial: $u_0 = u_n = 0$, and so we can trim them off to yield the slightly simpler matrix

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} -h^2 g(x_1) \\ -h^2 g(x_2) \\ \vdots \\ -h^2 g(x_{n-2}) \\ -h^2 g(x_{n-1}) \end{bmatrix}. \tag{1.24}$$

Solve this $(n-1) \times (n-1)$ linear system of equations using Gaussian elimination. One can show that the solution to the differential equation inherits the accuracy of the interpolant: the error $|u(x_j) - u_j|$ behaves like $O(h^2)$ as $h \to 0$.

Ideally, use an efficient version of Gaussian elimination that exploits the banded structure of this matrix to give a solution in $O(n)$ operations.

Example 1.7 (Mixed boundary conditions)**.** Modify the last example to keep the same differential equation

$$-u''(x) = g(x), \qquad x \in [0, 1]$$

but now use *mixed* boundary conditions,

$$u'(0) = u(1) = 0.$$

The derivative condition on the left is the only modification; we must change the first row of equation (1.23) to encode this condition. One might be tempted to use a simple linear interpolant to approximate the boundary condition on the left side, adapting formula (1.14) to give:

$$\frac{u_1 - u_0}{h} = 0. \tag{1.25}$$

This equation makes intuitive sense: it forces $u_1 = u_0$, so the approximate solution will have zero slope on its left end. This gives the equation

$$\begin{bmatrix} -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ -h^2 g(x_1) \\ -h^2 g(x_2) \\ \vdots \\ -h^2 g(x_{n-2}) \\ -h^2 g(x_{n-1}) \end{bmatrix}, \tag{1.26}$$

where we have trimmed off the elements associated with the $u_n = 0$.

The approximation of the second derivative (1.22) is accurate up to $\mathcal{O}(h^2)$, whereas the estimate (1.25) of $u'(0) = 0$ is only $\mathcal{O}(h)$ accurate. Will it matter if we compromise accuracy that little bit, if only in one of the n equations in (1.26)? What if instead we approximate $u'(0) = 0$ to second-order accuracy?

Equations (1.17)–(1.19) provide three formulas that approximate the first derivative to second order. Which one is appropriate in this setting? The *right-looking* formula (1.17) gives the approximation

$$u'(0) \approx \frac{-3u_0 + 4u_1 - u_2}{2h}, \tag{1.27}$$

which involves the variables u_0, u_1, and u_2 that we are already considering. In contrast, the centered formula (1.18) needs an estimate of $u(-h)$, and the left-looking formula (1.19) needs $u(-h)$ and $u(-2h)$. Since these values of u fall outside the domain $[0,1]$ of u, the centered and left-looking formulas would not work.

Combining the right-looking formula (1.27) with the boundary condition $u'(0) = 0$ gives

$$\frac{-3u_0 + 4u_1 - u_2}{2h} = 0,$$

with which we replace the first row of (1.26) to obtain

$$\begin{bmatrix} -3 & 4 & -1 & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ -h^2 g(x_1) \\ -h^2 g(x_2) \\ \vdots \\ -h^2 g(x_{n-2}) \\ -h^2 g(x_{n-1}) \end{bmatrix}. \tag{1.28}$$

Is this $\mathcal{O}(h^2)$ accurate approach at the boundary worth the (rather minimal) extra effort? Let us investigate with an example. Set the right-hand side of the differential equation to

$$g(x) = \cos(\pi x/2),$$

which corresponds to the exact solution

$$u(x) = \frac{4}{\pi^2}\cos(\pi x/2).$$

Verify that u satisfies the boundary conditions $u'(0) = 0$ and $u(1) = 0$.

Figure 1.10 compares the solutions obtained by solving (1.26) and (1.28) with $n = 4$. Clearly, the simple adjustment that gave the $\mathcal{O}(h^2)$ approximation to $u'(0) = 0$ makes quite a difference! This figures shows that the solutions from (1.26) and (1.28) differ, but plots like this are not the best way to understand how the approximations compare as $n \to \infty$. Instead, compute maximum error at the interpolation points,

Indeed, we used this small value of n because it is difficult to see the difference between the exact solution and the approximation from (1.28) for larger n.

$$\max_{0\le j\le n} |u(x_j) - u_j|$$

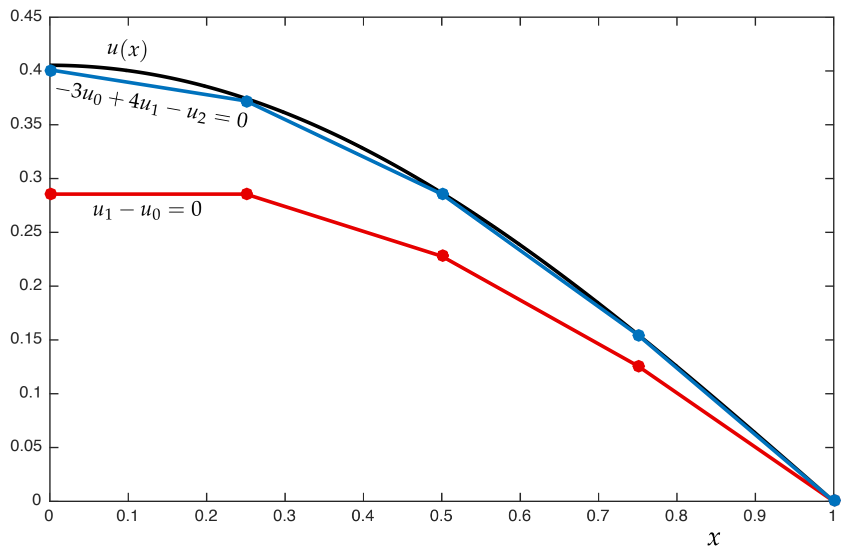

Figure 1.10: Approximate solutions to $-u''(x) = \cos(\pi x/2)$ with $u'(0) = u(1) = 0$. The black curve shows $u(x)$. The red approximation is obtained by solving (1.26), which uses the $\mathcal{O}(h)$ approximation $u'(0) = 0$; the blue approximation is from (1.28) with the $\mathcal{O}(h^2)$ approximation of $u'(0) = 0$. Both approximations use $n = 4$.

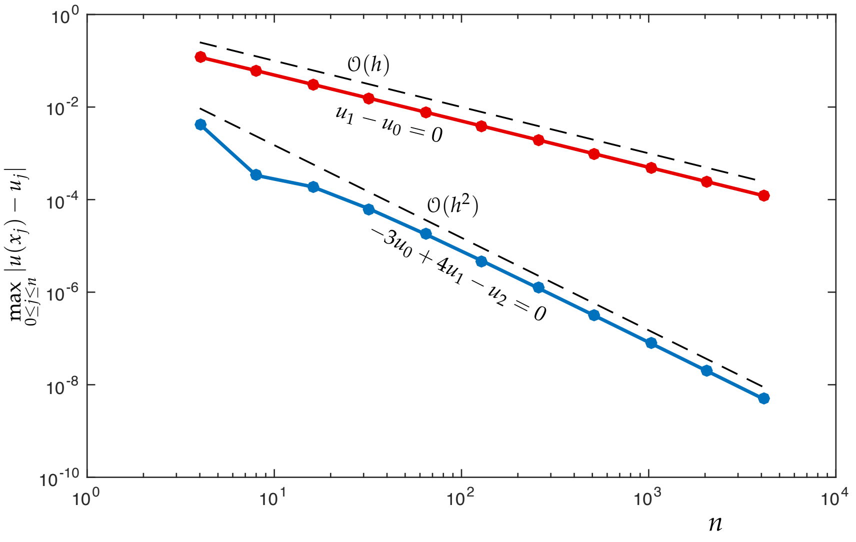

Figure 1.11: Convergence of approximate solutions to $-u''(x) = \cos(\pi x/2)$ with $u'(0) = u(1) = 0$. The red line shows the approximation from (1.26); it converges like $\mathcal{O}(h)$ as $h \to 0$. The blue line shows the approximation from (1.28), which converges like $\mathcal{O}(h^2)$.

for various values of n. Figure 1.11 shows the results of such experiments for $n = 2^2, 2^3, \ldots, 2^{12}$. Notice that this figure is a 'log-log' plot; on such a scale, the errors fall on straight lines, and from the slope of these lines one can determine the order of convergence. The slope of the red curve is -1, so the accuracy of the approximations from (1.26) is $\mathcal{O}(n^{-1}) = \mathcal{O}(h)$ accurate. The slope of the blue curve is -2, so (1.28) gives an $\mathcal{O}(n^{-2}) = \mathcal{O}(h^2)$ accurate approximation.

How large would n need to be, to get the same accuracy from the $\mathcal{O}(h)$ approximation that was produced by the $\mathcal{O}(h^2)$ approximation with $n = 2^{12} = 4096$? Extrapolation of the red curve suggests we would need roughly $n = 10^8$.

This example illustrates a general lesson: when constructing finite difference approximations to differential equations, one must ensure that the approximations to the boundary conditions have the same order of accuracy as the approximation of the differential equation itself. These formulas can be nicely constructed by from derivatives of polynomial interpolants of appropriate degree.