Finite difference methods for boundary value problems

In the last several lectures, we considered finite difference methods for initial value problems. We will transition in this lecture to numerical methods for *boundary value problems* (BVPs). We will focus this week on finite difference methods for BVPs. Just as we saw for IVPs, finite difference methods for BVPS are based on local interpolation. This connection makes them a natural segue from our IVP discussion. Next week, we will consider a different class of numerical methods for BVPs referred to as spectral methods. By contrast with the interpolation-based finite difference method, spectral methods are based on least-squares approximation of the solution to the BVP.

The distinction between finite difference methods and spectral methods is analogous to the difference between interpolation and least squares approximation for functions, which we covered earlier this semester.

1 *Boundary value problems*

Before diving into finite difference methods for BVPs, we first take a moment to review what defines this type of differential equation in the first place. The BVPs we will consider are of the form

$$\alpha \frac{d^2u}{dx^2} + \beta \frac{du}{dx} + \gamma u = f, \quad x \in [a, b] \tag{1}$$

for real constants α and β, γ, boundary points a and b ($a < b$), and some prescribed forcing $f(x)$. BVPs are fully defined by augmenting the differential equation (1) with *boundary conditions*. For now, we will consider the so-called Dirichlet boundary conditions

$$u(a) = u_a, \; u(b) = u_b \tag{2}$$

Can you write down another set of admissable boundary conditions for this 1D problem?

Notice the definitional distinction from IVPs: in this case the prescribed conditions for $u(x)$ are not at an initial instance in time, but instead enforce that u satisfy conditions at the *boundary points* $x = a$ and $x = b$. Let us make this distinction clear:

Philosophy behind BVPs

Whereas IVPs describe the dynamical response of a system to stimuli/initial conditions, BVPs describe the steady state (or equilibrium) response of a system to forcing.

It is precisely this distinction from IVPs that makes BVPs amenable to solution strategies besides finite difference methods, which are based on local interpolation. While finite difference methods work well for these types of problems, the presence of boundary conditions that define the solution globally makes global approaches suitable candidates for BVPs. By contrast, the dynamical evolution of an IVP from an initial condition makes the local-interpolation-based finite difference method by far the most common numerical framework for IVPs.

1.1 *The Poisson problem*

In the next two lectures, we will consider the specific case of (1) where $\alpha = 1$ and $\beta = \gamma = 0$. This is called the 1D Poisson problem,

and is given by

$$\frac{d^2u}{dx^2} = f, \quad x \in [a,b] \tag{3}$$

(We will continue to use the boundary conditions (2) to fully define the BVP).

The use of this specific BVP will make it easier to derive and characterize the error of various methods. However, the methods we derive for the Poisson problem extend straightforwardly to the more general differential equation (1).

Notice that we do not need a numerical method to solve this problem: the exact solution can be obtained by integrating (3) twice. However, this will give us an approachable setting within which to develop our numerical tools. We will then consider increasingly complex problems where analytical solutions are harder to come by.

2 *Finite difference methods for BVPs*

2.1 *Discretizing the domain*

The first thing that we need to develop our method is a set of interpolation points. How should we define these points? The simplest approach is to uniformly distribute them over the interval $[a,b]$, and this is what we will do. The discrete points under this uniform distribution are defined by

$$x_j = a + \frac{(b-a)(j-1)}{n}, \quad j = 1,\ldots,n+1 \tag{4}$$

Figure 1: A schematic of the uniform point distribution we are considering.

Nonuniform point spacings are of course possible, and can sometimes be useful (*e.g.*, perhaps there is some fine-scale behavior in a subregion of $[a,b]$ that warrants tightly spaced points, whereas the solution is expected to behave more smoothly in other regions of $[a,b]$). The problems we will consider do not need this added layer of complexity. In any case, the ensuing derivation is applicable to the nonuniform point spacings, though the arithmetic becomes more cumbersome.

2.2 *Approximating the solution with local interpolation*

Finite difference methods use local interpolation. That is, for each interpolation point x_j, we will approximate $u(x)$ using a piecewise-defined p^{th}-order polynomial. We will first develop the finite difference method for a general p^{th}-order polynomial. After this, we will develop a second-order finite difference method as a specific example of the more general case.

We require $p+1$ points to uniquely define this polynomial, so p additional points are needed in addition to x_j. We will use the points $\{x_{j-p/2},\ldots,x_j,\ldots,x_{j+p/2}\}$; *i.e.*, points that are centered about x_j. Finite difference methods constructed from this collection of points are called *centered* difference methods. Other point choices are possible and used in practice. For example, a *one-sided difference*

Notice that this *centered* selection of points requires that $p/2$ is an integer: p must be even. If we desired to use an odd-valued p, we would have to use a non-centered selection of points x_j.

method could be obtained by, *e.g.*, using the points $\{x_j, \ldots, x_{j+p}\}$. Indeed, many combinations of points are possible, and there are advantages and disadvantages to the various choices. We will focus on centered methods in this class, but be aware that other methods exist and may be obtained through non-centered point selections.

This centered definition of points is not necessarily valid near the boundary points a and b. For example, if $p = 2$ then the points used to define the Lagrange polynomials about x_1 should in principle be $\{x_0, x_1, x_2\}$. This is impossible, since x_0 is undefined. This cues us into the important fact that points near boundaries must be handled with care. We will not worry about this detail for now (we will address it later when we consider the 2^{nd} order finite difference method).

Now that we have the $p+1$ necessary points to define our polynomial at a given x_j, how do we write the p^{th} order polynomial? We know using the monomial basis $\{1, x, x^2, \ldots, x^p\}$ is a bad idea because of poor conditioning. We will thus return to our *Lagrange basis polynomials*. We define the k^{th} Lagrange basis polynomial of order p associated with the point x_j as

$$L_k^{(j)}(x) = \prod_{\substack{m=j-p/2 \\ (m\neq k)}}^{j+p/2} \frac{x - x_m}{x_k - x_m} \tag{5}$$

Notice that the definition of the Lagrange polynomials are different from before: they now have the superscript (j) to designate that they are defined *centered with respect to* x_j. Let us consider an example. Say that we want to build a polynomial of order 2 centered at x_5. There are three associated Lagrange basis polynomials: $L_4^{(5)}(x), L_5^{(5)}(x), L_6^{(5)}(x)$. We can write out $L_4^{(5)}(x)$ explicitly as

$$L_4^{(5)}(x) = \frac{(x - x_5)(x - x_6)}{(x_4 - x_5)(x_4 - x_6)}$$

This is indeed a 2^{nd} order polynomial: the leading order term in the numerator is x^2 and the denominator is just a constant equal to $2\Delta x^2$, where Δx is the spacing between two adjacent discrete points. Can you write out $L_5^{(5)}$ and confirm that it is also a second order polynomial?

for $k = j - p/2, \ldots, j, \ldots, j + p/2$.

Each of these Lagrange basis polynomials is a p^{th} order polynomial, so any linear combination of them is also a p^{th} order polynomial. We will therefore approximate $u(x)$ over the interval $x_{j-p/2} \leq x \leq x_{j+p/2}$ as

$$u(x) \approx \sum_{i=j-p/2}^{j+p/2} c_i L_i^{(j)}(x) \tag{6}$$

Notice that in this piecewise representation of our numerical solution, as the index j changes, so too does our approximation interval and expression for $u(x)$!

The aim: determining the unknown coefficients

Now that we have an expression for how we will approximate the exact solution $u(x)$, what remains is for us to solve for the unknown coefficients c_i. This will give us a way to evaluate our approximate solution $u_a(x)$ near (and at) each point x_j, as desired. Before discussing the computation of these coefficients, let us recall an observation: from (5), we see that $L_i^{(j)}(x_k) = 1$ if $i = k$ and is equal to 0 if $i \neq k$. Using this fact and evaluating (5) at x_j, we see that over the interval $x_{j-p/2} \leq x \leq x_{j+p/2}$:

$$\begin{aligned} u(x_j) &\approx \sum_{i=j-p/2}^{j+p/2} c_i L_i^{(j)}(x_j) \\ &= c_j \end{aligned} \tag{7}$$

An example is again useful here: returning to the 2^{nd} order polynomials about x_5, we have that $L_4^{(5)}(x_4) = 1$, $L_4^{(5)}(x_5) = L_4^{(5)}(x_6) = 0$. Can you determine what the analogous answers are for $L_5^{(5)}$ and $L_6^{(5)}$?

That is, c_j is an approximation of $u(x_j)$. Thus, for finite difference methods we may write

$$u(x) \approx \sum_{i=j-p/2}^{j+p/2} u_i L_i^{(j)}(x) \tag{8}$$

where $u_i \approx u(x_i)$. Thus, computing the various coefficients b_i is equivalent to calculating the values u_i that approximate that *exact* solution u at the grid point x_i.

This fact should not surprise us: we found in function interpolation that when using the Lagrange basis, the coefficients in the expansion were the values of function at the interpolation points!

The finite difference equations

So how do we solve for these u_i variables? We approximate $u(x)$ using (8) over the appropriate x-interval, substitute this expression into the governing equation (3), and evaluate it at the interpolation point x_j. Doing this gives

$$\sum_{i=j-p/2}^{j+p/2} u_i \frac{d^2 L_i^{(j)}}{dx^2}\bigg|_{x=x_j} = f(x_j), \quad j = 2, \ldots, n \tag{9}$$

Notice the similarities to and distinctions from function interpolation: in this BVP setting, we are still using interpolation. However, in this case, we are interpolating the BVP rather than some function that we wish to approximate.

Solving this set of equations gives us our approximations u_i to the exact solution at the i^{th} interpolation point, $u(x_i)$.

Again, do not forget that (9) is not the full story. We need to incorporate boundary conditions to get a solvable system of equations.

We have now derived the finite difference method for the 1D Poisson equation (to within appropriate consideration of boundary conditions). We derived this method for polynomials of unspecified order p, and it may not be obvious how to solve for the coefficients from the relatively abstract equations (9). To provide a more concrete example, we will consider the specific case of $p = 2$ next.

2.3 *A second order finite difference method*

For $p = 2$, the x-interval over which the numerical solution is piecewise-defined about a given interpolation point x_j is $x_{j-1} \leq x \leq x_{j+1}$. In addition, the expression (8) reduces to

$$u(x) \approx \sum_{i=j-1}^{j+1} u_i L_i^{(j)}(x), \quad j = 2, \ldots, n \tag{10}$$

Notice that we are now taking care to account for boundary terms: the index j does not include 1 or $n+1$. If j did include 1, then the index i would run from 0 to 2, but there is no x_0! There is an analogous line of reasoning for $j = n+1$.

The equations for $L_i^{(j)}(x)$ are also more manageable. For example,

$$\begin{aligned} L_{j-1}^{(j)}(x) &= \frac{(x - x_j)(x - x_{j+1})}{(x_{j-1} - x_j)(x_{j-1} - x_{j+1})} \\ &= \frac{1}{2\Delta x^2}(x - x_j)(x - x_{j+1}) \end{aligned} \tag{11}$$

The other two Lagrange polynomials associated with the interpolation point x_j are $L_j^{(j)}$ and $L_{j+1}^{(j)}$. What are the expressions for those basis functions?

Recall that we defined $\Delta x = x_j - x_{j-1}$. Note that since we assumed uniform spacing the specific choice of j does not matter. If the spacing between points was nonuniform, we would instead use the notation $(\Delta x)_j$, though apart from this the ensuing analysis applies to nonuniform spacing distributions.

Using the expression (11) (and its analogs for $L_j^{(j)}$ and $L_{j+1}^{(j)}$), the

equation for the unknown coefficients u_i, (9), becomes

$$u_{j-1}\frac{d^2}{dx^2}\left(\frac{1}{2\Delta x^2}(x-x_j)(x-x_{j+1})\right)+u_j\frac{d^2}{dx^2}\left(-\frac{1}{\Delta x^2}(x-x_{j-1})(x-x_{j+1})\right)$$
$$+u_{j+1}\frac{d^2}{dx^2}\left(\frac{1}{2\Delta x^2}(x-x_{j-1})(x-x_j)\right)=f(x_j),\quad j=2,\ldots,n \tag{12}$$

It is straightforward to compute the second derivative of the various second-order polynomials in (12). Doing so lets us simplify (12) to

$$\frac{1}{\Delta x^2}\left(u_{j-1}-2u_j+u_{j+1}\right)=f(x_j),\quad j=2,\ldots,n \tag{13}$$

We almost have all the information we need to assemble the matrix system to solve for the unknowns u_j that approximate $u(x_j)$: we have $n-1$ equations at our disposal. However, we need two more equations to arrive at a solvable linear system. These equations will come from the boundary conditions (2), which we write in our finite difference setting as

$$u_1=u_a, u_{n+1}=u_b \tag{14}$$

We can now write out the matrix system to solve for the unknowns. Combining (13) and (14) gives

$$\frac{1}{\Delta x^2}\begin{bmatrix}\Delta x^2 & & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & 1 & -2 & 1 & & \\ & & & & 1 & -2 & 1 & \\ & & & & & & \Delta x^2\end{bmatrix}\begin{bmatrix}u_1\\u_2\\u_3\\\vdots\\u_{n-1}\\u_n\\u_{n+1}\end{bmatrix}=\begin{bmatrix}u_a\\f(x_2)\\f(x_3)\\\vdots\\f(x_{n-1})\\f(x_n)\\u_b\end{bmatrix} \tag{15}$$

We will use the convention in this class that all unspecified matrix entries are zero.

We can implement (15) in a computer and arrive at a solution for the variables $u_1,\ldots,u_{n+1}$ that approximate the exact solution at the interpolation points $x_1,\ldots,x_n$. Before doing this, we make an observation about the matrix system (15): as written, the first and last rows represent redundant information. These rows represent "equations" at the points x_1 and x_n, respectively, when in fact the equation at these points is trivial. We already know that $u_1 = u_a$, $u_{n+1} = u_b$. Given this, we can remove these rows (along with the first and last columns of the matrix), and arrive at a slightly truncated

variant of (15):

$$\frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}\begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} f(x_2) - \frac{u_a}{\Delta x^2} \\ f(x_3) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) - \frac{u_b}{\Delta x^2} \end{bmatrix} \tag{16}$$

Can you convince yourself that (15) and (16) will give identical answers?

Either of the variants (15) or (16) are valid and will lead to identical numerical solutions (to within machine precision). However, the smaller form (16) can be useful both for analyzing the convergence properties of the method and for developing fast solution approaches.

A coded example and motivation for understanding convergence

Let us put our new method to the test and work through a specific problem. We will consider (3) with

$$\begin{aligned} f(x) = &-4(b-a)\pi \sin(2\pi(b-a)x) \\ &-4(b-a)^2\pi^2 x\cos(2\pi(b-a)x) \end{aligned} \tag{17}$$

How did I arrive at this exact solution? I *chose* it: one can arbitrarily choose any function to be the "exact solution", and determine what $f(x)$ and the boundary conditions must be such that the desired function is indeed a solution. This approach is called the *method of manufactured solutions*, and is immensely important to testing new algorithms and codes.

The exact solution to this problem is $u(x) = x\cos(2\pi(b-a)x)$, and the associated boundary conditions are $u_a = a\cos(2\pi(b-a)a)$ and $u_b = b\cos(2\pi(b-a)b)$. We will use $a = 2$ and $b = 4$, though any values will work.

We can solve for the u_j that approximates $u(x_j)$ at the various interpolation points using (15) or (16). We adopt the following notation: the discrete points and numerical solution are collected into a vector via $\hat{\boldsymbol{x}} = [x_1, x_2, \ldots, x_{n+1}]^T$ and $\hat{\boldsymbol{u}} = [u_1, u_2, \ldots, u_{n+1}]^T$. Notice the distinction between these vector variables and the scalar-valued exact solution $u(x)$.

It is clear from the figure that our numerical solution is getting better as n increases. That is, u_j gets closer to $u(x_j)$ for $j = 1, \ldots, n+1$ as n gets larger. We can make this more precise by looking at the *error* in the numerical solution. Figure 3 shows the quantity

$$||\bar{\boldsymbol{u}} - \hat{\boldsymbol{u}}||_2 \tag{18}$$

where $\bar{\boldsymbol{u}}$ is understood to be the element-wise application of the exact solution u on the interpolation points contained in $\hat{\boldsymbol{x}}$, and $||\cdot||_2$ is the grid function 2-norm defined as

$$||\boldsymbol{g}||_2 := \sqrt{\Delta x \sum_{j=1}^{n} g_j^2} \tag{19}$$

Where did the factor of Δx arise from? The grid function norm aims to approximate the size of a function over the domain $[a, b]$. In a continuous setting, this quantity would be $\sqrt{\int_a^b g^2(x)dx}$. Notice that this means that the norm is induced from the inner product $(f, g) = \int_a^b f(x)g(x)dx$. The grid function norm mimics this integration through the factor Δx. Note that for an r-dimensional problem, the factor becomes Δx^r instead of Δx.

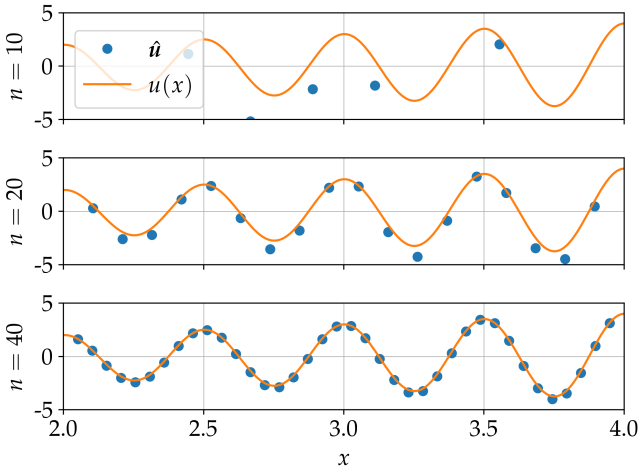

Figure 2: Numerical solution of the 1D Poisson equation along with the exact solution for $n = 10, 20$, and 40.

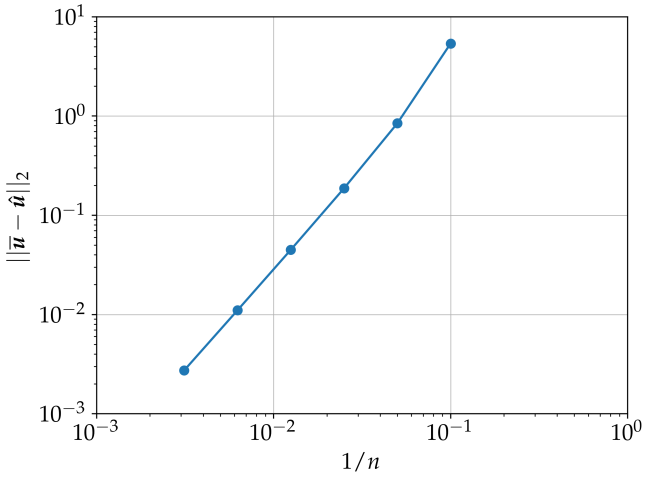

Figure 3: Error in the numerical solution of the 1D Poisson equation for $n = 10$, 20, 40, 80, 160, and 320.

for some n-dimensional vector $\boldsymbol{g}$.

This decay in error is comforting: we would hope that our solution gets better as we use more points! At the same time, this result raises a number of questions. Should we always expect this desirable behavior? Can we say something more specific about how quickly the u_j approach $u(x_j)$, perhaps as a function of the spacing between points (Δx)? Are we solving (16) in the fastest possible way?

We will explore these important questions in the next lecture.