CS 542G: EigenPDEs, the Finite Element Method

Robert Bridson

November 17, 2008

1 Eigenproblems in PDEs

We mentioned last time that the Finite Element Method has a strong connection to the idea behind Rayleigh-Ritz: approximately solve a large problem by restricting it to a well-chosen small subspace. Before going further into FEM, it's worth illuminating this connection even more by looking at a related problem: finding eigenvalues and eigenfunctions for PDEs. FEM applied to the PDE eigenproblem is *exactly* Rayleigh-Ritz applied to an infinite dimensional original problem.

PDE eigenproblems arise in many areas, including the solution and analysis of regular PDEs (as we in fact were doing last time with Fourier analysis; the eigenfunctions of the Laplacian on a rectangle are Fourier modes). In some applications the connection with eigenvalues and eigenfunctions is clearer however. For example, many electrical engineering problems such as antenna design revolve around natural radio frequencies of an object, which are eigenvalues of a particular form of Maxwell's partial differential equations. An easier to visualize example is vibration analysis: how does a structure vibrate and at which natural frequencies? This is immensely important not just for the obvious acoustic problems (e.g. how loud is the interior of a car or plane going to be?) but many other engineering problems (e.g. how strong is a bridge or other structure when loaded with time-varying wind forces or traffic forces?). We won't cover all the relevant physics, but just work through the simplest 1D model problem: studying the vibration of a taut string, as one might find on a viola or other instrument, or in structures involving cables or wires such as power lines.

While there are many ways a string can deform—it can stretch along its length, it was twist around its main axis, it can displace to the side in any directional orthogonal to the main axis, or perform even more complicated motions—we'll make the simplifying assumption that the only important one is displacement orthogonal to the main axis, and moreover assume it only displaces in one direction. We'll also assume the displacement is small enough that we can ignore the small amount of extra stretching (and higher tension) it produces. We also will assume that the ends of the string are perfectly stationary, so their displacement is fixed at zero. These modeling assumptions are fine to get a basic understanding of the real problem, but we should always have in the back of our head the memory that there could be many smaller effects we won't see in this model.[1]

With these assumptions in place, being with the rest state of the string, with no displacement. We'll line it up on the x-axis, from endpoint $x = 0$ to endpoint $x = L$. The point at a given x in the rest position is only allowed to move vertically, $y = u(x)$, with the displacement given as a function $u(x)$. The endpoints are fixed, so $u(0) = u(L) = 0$. We now want to determine how $u(x)$ varies as a function of time, $u(x,t)$: it boils down to getting acceleration from Newton's law $F = ma$ for which we need to understand the forces and mass in the string.

Assume the tension of the string is a (roughly) constant T: that is, if you replaced a tiny part of the string with a force meter, it would give the value T. Consider the part of the string going from x to $x + \Delta x$. The net vertical component of force on this substring is the sum of vertical forces at the endpoints x and $x + \Delta x$, which are well approximated through the use of the slopes there: $-Tu'(x)$ and $Tu'(x + \Delta x)$. (A more complete derivation would base this on the angle θ of the string, and then argue that $T\sin\theta \approx T\tan\theta = Tu'$.) At the same time, the total mass of this substring is its length Δx times the linear density ρ, measured in kg/m. The vertical component of Newton's law $F = ma$ for the substring is:

$$-T\frac{\partial u(x)}{\partial x} + T\frac{\partial u(x+\Delta x)}{\partial x} = \rho\Delta x\frac{\partial^2 u}{\partial t^2}$$

If you want, you can interpret the vertical acceleration $\partial^2 u/\partial t^2$ as some sort of average over the substring, but to make this exact, we can instead divide through by Δx to look at the force-per-unit-length and mass-per-unit-length:

$$T\frac{\frac{\partial u(x+\Delta x)}{\partial x} - \frac{\partial u(x)}{\partial x}}{\Delta x} = \rho\frac{\partial^2 u}{\partial t^2}$$

and then take the limit as $\Delta x \to 0$ to get a PDE at x:

$$T\frac{\partial^2 u}{\partial x^2} = \rho\frac{\partial^2 u}{\partial t^2}$$

Along with the boundary conditions $u(0) = u(L) = 0$, this is a good first approximation describes the motion of a taut string, called the "wave equation".

To get a picture of how any possible solution evolves in time—and with the idea that vibrations are going to be important—we take the Fourier transform in time. Or rather, exploiting linearity, we look

[1]One example we saw, or rather heard, in class is the phenomenon of "subharmonic" notes on the violin or viola, which are not predicted by this simplified model and in fact haven't yet been fully explained.

for a particular solution where the time dependence is sinusoidal of a given frequency:

$$u(x,t) = U(x)\sin(\omega t)$$

Plugging this into the PDE gives:

$$T\frac{\partial^2 U(x)}{\partial x^2}\sin(\omega t) = \rho U(x)(-\omega^2)\sin(\omega t)$$

Taking out the common factor of $\sin(\omega t)$ gives a PDE eigenproblem for $U(x)$:

$$\begin{aligned}
\frac{\partial^2 U(x)}{\partial x^2} &= \lambda U(x) \qquad \text{for } 0 < x < L \\
U(0) &= 0 \\
U(L) &= 0
\end{aligned}$$

where the eigenvalue is related to the time frequency by

$$\lambda = \frac{-\rho\omega^2}{T}$$

or turning this around,

$$\omega = \sqrt{\frac{-T\lambda}{\rho}}$$

It's not too hard to go and solve this analytically, since it's just 1D: the k'th eigenfunction is

$$U_k(x) = \sin\left(\frac{2k\pi x}{L}\right)$$

for $k = 1, 2, \ldots$, with associated eigenvalue

$$\lambda_k = -\left(\frac{2k\pi}{L}\right)^2$$

and hence time frequency

$$\omega_k = \sqrt{\frac{T}{\rho}}\frac{2k\pi}{L}$$

The full general solution of the PDE is just a linear combination of these eigenmodes. (As an aside, the musically pleasant result apparent above is that the string naturally vibrates at frequencies which are integer multiples of the "fundamental" $k = 1$ case; the higher frequencies are sometimes called "harmonics", and go nicely with the fundamental, being different by octaves or other tonal intervals.)

More complicated vibration problems which can't be simplified to 1D can still be tackled this way. For example, the motion of a taut membrane (such as the one stretched over the head of a drum) can be approximated from the following eigenproblem:

$$\begin{aligned}
\nabla \cdot \nabla u &= \lambda u \qquad \text{for } x \in \Omega \\
u &= 0 \qquad \text{for } x \in \partial\Omega
\end{aligned}$$

Here the Laplacian generalizes the second derivative we used in 1D. (Incidentally, the reason drums have some sort of pitch, but don't ring with as clear a tone as a string, is that the time frequencies accompanying the eigenvalues are not just simple integer multiples of a fundamental; the same story goes for thick bells which typically have some dissonant harmonics.)

FEM for the Poisson problem means selecting a finite dimensional subspace of functions and minimizing the variational form over just this subspace. FEM for the above eigenproblem is the same thing, selecting a finite dimensional subspace of functions and solving the projected eigenproblem on that subspace exactly the same way as we used Rayleigh-Ritz in finite dimensions—in fact, for PDE eigenproblems, FEM is more or less a synonym for Rayleigh-Ritz.

2 The Finite Element Method

2.1 Derivation

Return now to FEM, looking at it in more detail. Last time we took the Poisson problem:

$$\begin{aligned} \nabla \cdot \nabla u(x) &= f(x) \qquad \text{for } x \in \Omega \\ u(x) &= 0 \qquad \text{for } x \in \partial\Omega \end{aligned}$$

and re-expressed it in variational form:

$$\min_{u : u|_{\partial\Omega}=0} \int_\Omega \|\nabla u(x)\|^2 + 2f(x)u(x)\, dx$$

We introduced FEM as a numerical discretization where we find the minimum of this objective in a carefully chosen finite dimensional space. More precisely, this is known as the **Galerkin** method.

Other FEM approaches exist (including generalizing Galerkin to handle equations which don't have a variational form), but the defining feature of all FEM approaches is the finite dimensional subspace from which approximate solutions are taken. We'll focus on Galerkin as being the most popular and successful[2] of the bunch, and the one with the simplest and strongest theoretical underpinnings.

To turn this idea into an actual algorithm, we first must choose a basis for the finite dimensional subspace. Leaving aside the question of what space to pick for now, let's represent this abstractly as n functions $\phi_1(x)$, $\phi_2(x)$, ..., $\phi_n(x)$ which satisfy the Dirichlet boundary condition $\phi_i(x) = 0$ on the boundary $\partial\Omega$. Every element in the subspace is a unique linear combination of these basis functions (and natu-

[2]We mean success in terms of the number of problems where it works really well: there are certainly other classes of problems we won't touch on where Galerkin FEM fails miserably, and where other finite element approaches are superior.

rally satisfies the required boundary condition). In particular, the solution we're after can be represented as

$$u(x) = \sum_{i=1}^{n} u_i \phi_i(x)$$

Here the u_i are just constant coefficients, which can be wrapped up in a vector u.

Plug this representation of the solution into the variational form:

$$\begin{aligned}
\int_\Omega \|\nabla u(x)\|^2 + 2f(x)u(x) &= \int_\Omega \left\| \sum_{i=1}^{n} \nabla u_i \phi_i(x) \right\|^2 + 2f(x) \sum_{i=1}^{n} u_i \phi_i(x)\, dx \\
&= \int_\Omega \left(\sum_{i=1}^{n} u_i \nabla \phi_i(x) \right) \cdot \left(\sum_{j=1}^{n} u_j \nabla \phi_j(x) \right) + 2 \sum_{i=1}^{n} u_i \phi_i(x) f(x)\, dx \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} u_i \left(\int_\Omega \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx \right) u_j + 2 \sum_{i=1}^{n} \left(\int_\Omega \phi_i(x) f(x)\, dx \right) u_i
\end{aligned}$$

Define the square $n \times n$ matrix A with

$$A_{ij} = \int_\Omega \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx$$

and the n-vector f with

$$f_i = \int_\Omega \phi_i(x) f(x)\, dx$$

and then the finite dimensional problem simplifies to just

$$\min_u (u^T A u + 2u^T f)$$

Differentiating this quadratic objective with respect to u gives $2Au + 2f$; setting that to zero to find the minimum gives the following linear system:

$$Au = -f$$

Once we've solved this, we have our approximate solution.

2.2 The Linear System

Just from inspection of the definition of A, the matrix is clearly symmetric. In fact we can say more—since it is the Hessian of a quadratic function with an obvious lower bound it must in fact be SPD. Essentially, since the original minimization was well-posed, the minimization restricted to a smaller subspace must be at least as well-posed. Contrast the simplicity of this argument with our proof that the finite difference system was also SPD—in fact, for more general finite difference schemes, one of the common ways to prove the discrete problem is well-posed is to show it is equivalent to a particular finite element method which has a stronger theoretical infrastructure.

Also from the definition of A, as long as we can arrange for most pairs of basis functions $\phi_i(x)$ and $\phi_j(x)$ to have non-overlapping support—i.e. one is zero wherever the other one is nonzero—then the matrix A will be sparse, which also helps speed up solution.

2.3 Details

Next time we will look more closely at two details of the method:

- Choosing the basis functions, and proving the method works
- Calculating A and f to set up the linear system

The final two lectures of the course will then look in more detail at other practical aspects of FEM: generating a mesh on which suitable basis functions can be defined, and solving large and sparse linear systems. Particularly the last topic has wide application, of course, in many other scientific computing problems.