2019-11-13

1 The need for model problems

Direct methods for solving linear systems and eigenvalue problems are (mostly) "black box." We design algorithms that work well for a broad category of problems with given structural properties; once we understand the structure, there is often a reasonably routine choice of solvers. Of course, even for direct methods, it is not entirely true that we get "black box" performance — for example, the fill in sparse direct factorization methods is highly dependent on the sparsity structure of the matrix at hand. Nonetheless, users of sparse solvers can largely leave the details to specialists once they understand the basic lay of the land.

For the remainder of the semester, we will focus on iterative solvers, which are a different beast altogether. Iterative solvers produce a sequence of approximate solutions that (ideally) converge to the true solution to a linear system or eigenvalue problem. However, the rate of convergence is highly dependent on both the iterative method and the details of the problem. Even when we are able to take advantage of a good library of iterative solvers, there are often a wide variety of methods to choose from and a large number of parameters that we need to understand and tune to get good performance.

Because iterative methods are more problem-dependent than direct methods, we will focus our presentation on a set of model problems that exhibit characteristics common in many problems drawn from physical models. We will also comment on other types of problem structures as we go along, but will mostly leave the details to select homework problems.

2 The 1D model problem

It is difficult to say many useful things about the convergence of iterative methods without looking at a concrete problem. Therefore, we will set the stage with a very specific model problem: a discretization of the Poisson equation. We start with the one-dimensional case.

The continuous version of our model problem is a one-dimensional Poisson equation with homogeneous Dirichlet boundary conditions:

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f \text{ for } x \in (0,1) \\ u(0) &= 0 \\ u(1) &= 0 \end{aligned}$$

Let $x_j = j/(n+1)$ for $j = 0, 1, \ldots, n+1$ be a set of mesh points. We can approximate the second derivative of u at a point by a finite difference method:

$$-\frac{d^2u}{dx^2}(x_j) \approx \frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2}$$

where $h = 1/(n+1)$ is the mesh spacing. If we replace the second derivative in the Poisson equation with this finite-difference approximation, we have a scheme for computing $u_j \approx u(x_j)$:

$$\begin{aligned} -u_{j-1} + 2u_j - u_{j-1} &= h^2 f_j \text{ for } 1 \leq j \leq n \\ u_0 &= 0 \\ u_{n+1} &= 0 \end{aligned}$$

We can write this approximation as a matrix equation $Tu = h^2 f$, where

$$T = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

Part of what makes this simple Poisson discretization so appealing as a model problem is that we can compute the eigenvalues and eigenvectors directly. This is because solving the $(T - \lambda)\psi = 0$ is equivalent to considering the constant coefficient difference equation

$$\psi_{k+1} - (2 - \lambda)\psi_k + \psi_{k-1} = 0$$

subject to the boundary conditions $\psi_0 = \psi_{n+1} = 0$. Solutions to this difference equation must have the form

$$\psi_k = \alpha \xi^k + \beta \bar{\xi}^k,$$

where ξ and $\bar{\xi}$ are the roots of the characteristic polynomial $p(z) = z^2 - (2 - \lambda)z + 1$. For $0 \leq \lambda \leq 4$, these roots form a complex conjugate pair, each with unit magnitude; that is, we can write $\xi = \exp(i\theta)$ for some θ, and so

$$\xi^k = \exp(ik\theta) = \cos(k\theta) + i \sin(k\theta).$$

Thus, any solution to the difference equation must have the form

$$\psi_k = \gamma \cos(k\theta) + \mu \sin(k\theta).$$

Plugging in the boundary conditions, we find that $\gamma = 0$, and $\theta = l\pi/(n+1)$ for some l. Thus, the normalized eigenvectors of T are z_j with entries

$$\begin{aligned} z_j(k) &= \sqrt{\frac{2}{n+1}} \sin\left(\frac{jk\pi}{n+1}\right) \\ &= \sqrt{\frac{2}{n+1}} \sin((j\pi)x_k) \end{aligned}$$

and the corresponding eigenvalues are

$$\lambda_j = 2\left(1 - \cos\frac{\pi j}{n+1}\right).$$

For $j \ll n$, Taylor expansion gives that

$$\lambda_j = h^2(\pi j)^2 + O\left(h^4(\pi j)^4\right).$$

By way of comparison, the continuous Dirichlet eigenvalue problem

$$-\frac{d^2w}{dx^2} = \mu w, \quad w(0) = w(1) = 0$$

has eigenfunctions of the form

$$w_j = \sin(j\pi x), \quad \mu_j = (j\pi)^2.$$

Thus, the eigenvectors of $h^{-2}T$ are *exactly* the sampled eigenfunctions of $-d^2/dx^2$ on $[0, 1]$ with Dirichlet boundary conditions, while the extremal eigenvalues of $h^{-2}T$ satisfy

$$h^{-2}\lambda_j = \mu_j + O(\mu_j^2 h^2).$$

3 The 2D model problem

The problem with the 1D Poisson equation is that it doesn't make a terribly convincing challenge – since it is a symmetric positive definite tridiagonal, we can solve it in linear time with Gaussian elimination! So let us turn to a slightly more complicated example: the Poisson equation in 2D. Before discussing the 2D Poisson equation, though, let us digress to introduce two useful notations: the vec operator and the Kronecker product.

The vec operator simply lists the entries of a matrix (or an array with more than two indices) in column-major order; for example,

$$\operatorname{vec} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}.$$

The Kronecker product $A \otimes B$ of two matrices is a block matrix where each block is a scalar multiple of B:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots \\ a_{21}B & a_{22}B & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

The Kronecker product and the vec operation interact with each other as follows:

$$(B \otimes A) \operatorname{vec}(C) = \operatorname{vec}(ACB^T).$$

The Kronecker product also satisfies the identities

$$(A \otimes B)^T = A^T \otimes B^T$$
$$(A \otimes B)(C \times D) = (AB) \otimes (CD)$$

which implies, for example, that the Schur form of a Kronecker product is a Kronecker product of Schur forms:

$$(U_A \otimes U_B)^* (A \otimes B)(U_A \otimes U_B) = T_A \otimes T_B.$$

As one illustrative application of Kronecker products, consider the Sylvester operator $X \mapsto AX - XB$. Using Kronecker products, we can write this as

$$\operatorname{vec}(AX - XB) = (A \otimes I - I \otimes B) \operatorname{vec}(X).$$

Note that if $A = U_A T_A U_A^*$ and $B = U_B T_B U_B^*$ are Schur forms, then

$$A \otimes I - I \otimes B = (U_A \otimes U_B)(T_A \otimes I - I \otimes T_B)(U_A \otimes U_B)^*,$$

and $T_A \otimes I - T_B \otimes I$ is an upper triangular matrix. This transformation, followed by a triangular solve, is essentially what you did in problem 3 of your last homework.

Now let us return to the model 2D Poisson discretization. This is an approximation to the equation

$$-\nabla^2 u = -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f$$

for $(x, y) \in (0, 1)^2$, with Dirichlet boundary conditions $u(x, y) = 0$ for $|x| = 1$ or $|y| = 1$. If we discretize on a regular mesh with interior points indexed by $1 \leq i \leq n$ and $1 \leq j \leq n$, we can write the solution as a matrix U. When we discretize, we have a partial derivative in x corresponding to acting across columns of U, and a partial derivative in y corresponding to acting across rows of U. We can write this operation as

$$TU + UT = h^2 F,$$

or as an ordinary matrix equation of dimension $N = n^2$

$$(T \otimes I + I \otimes T) \operatorname{vec}(U) = h^2 \operatorname{vec}(F).$$

What properties do we have for $T_{n \times n} = T \otimes I + I \otimes T$?

1. $T_{n\times n}$ is symmetric and positive definite.
2. $T_{n\times n}$ is (non-strictly) diagonally dominant.
3. If (z_j, λ_j) are the eigenpairs for T, those for $T_{n\times n}$ are $(z_j \otimes z_l, \lambda_j + \lambda_l)$.
4. The condition number of $T_{n\times n}$ scales like $O(h^{-2})$.

4 Methods for solving the 2D model problem

Suppose we wanted to solve the 2D model problem in practice. What methods do we have at our disposal so far? Of course, we have several direct methods

1. We could run Gaussian elimination on $T_{n \times n}$. This takes time $O(N^3)$, where $N = n^2$.
2. The matrix $T_{n \times n}$ is also a banded matrix with bandwidth n so we could do band Gaussian elimination at a cost of $O(N^2 n) = O(N^{2.5})$.
3. A sparse direct solve using nested dissection ordering runs in $O(N^{1.5})$.
4. Treating the problem as a Sylvester equation and running Bartels-Stewart requires $O(n^3)$ time to find the eigensystem of T and to transform U and F using the eigenvector matrix; and $O(n^2)$ time for the subsequent (diagonal) linear solve.
5. The eigenvector matrix for T corresponds to a *discrete sine transform*, which is closely related to the FFT; and we know the eigenvalues in closed form. This allows us to reduce the time for Bartels-Stewart to $O(n^2 \log n) = O(N \log N)$.

In the coming lectures, we turn to a variety of *iterative methods*. These methods do not produce an exact answer, but rather produce a sequence of ever-better approximations to the truth. With appropriate parameter choices, the time to reduce the error by a constant factor scales like[1]

Jacobi	N^2
Gauss-Seidel	N^2
CG	$N^{3/2}$
SOR	$N^{3/2}$
SSOR with Chebyshev acceleration	$N^{5/4}$
Multigrid	N

For both the direct and iterative methods, the more structure we use, the faster we can go.

[1] See Table 6.1 of *Applied Numerical Linear Algebra* by J. Demmel.