

CMSC 661 Lecture 3/29/05 by Dianne P. O'Leary

Notes by Sean Leventhal

The first part of the course was concerned with taking an elliptic PDE of the form $Au = f$ and converting it into the a linear equation of the form $Au = f$, where A is large and sparse. We are now discussing ways of solving these systems, which have been broken into two methods, direct, and iterative. The main direct methods were Cholesky and LU decomposition, which both required reordering to do efficiently. The iterative methods are divided into SIM and Krylov methods.

Review of SIM methods

In SIM we take our original equation, $Ax = b$ and perform the following transformation:

$$Ax = b \tag{1}$$

$$(M - N)x = b \tag{2}$$

$$Mx = Nx + b \tag{3}$$

$$x = M^{-1}Nx + M^{-1}b \tag{4}$$

We then define the following terms:

$$A = M - N \tag{5}$$

$$G = M^{-1}N \tag{6}$$

$$c = M^{-1}b \tag{7}$$

$$\hat{G} = (I - G) \tag{8}$$

This lets us rewrite $Ax = b$ as either $x = Gx + c$ or $\hat{G}x = c$. We use these values to make successive estimations of x recursively:

$$x^{(k+1)} = Gx^{(k)} + c \tag{9}$$

Krylov Methods

Notice above that if $x^{(0)} = 0$ we get the following series:

$$x^{(1)} = c \tag{10}$$

$$x^{(2)} = Gc + c \tag{11}$$

$$x^{(3)} = G(Gc + c) + c = G^2c + Gc + c \tag{12}$$

$$\vdots \quad \vdots \quad \vdots \tag{13}$$

So at each step $x^{(n)}$ is a specific linear combination of $G^{n-1}c, \dots, Gc,$ and c . The span of these vectors is called a Krylov space, and is written $K_n(G, c)$, where the subscript indicates the number of powers of G to include.

The Krylov methods are derived by attempting to find a better linear combination of the given vectors than the SIM methods.

There are two ways to define the "best" vector:

Variational Approach: Given x^* is the true solution, choose x as follows: $\min \|x - x^*\|_Z$, where $\|y\|_Z^2 = y^T Z y$ (Z symmetric and positive definite).

Galerkin Approach: Choose $x^{(k)} \in K_k(G, c)$ such that $x^{(k)}$ is orthogonal to every vector in $K_k(G, c)$.

Whichever of these we choose, the subspace K expands at every iteration, so eventually it will cover the space that the solution lies in, and we will have an exact solution.

Basis functions for Krylov Methods

In addition to choosing one of the two approaches to selecting a new vector, we would like to work with a basis that is computationally convenient. To see the problem with the current basis consider the following: We have some G with eigenvectors v_i , and eigenvalues λ_i , sorted such that the eigenvalues ascend.

$$c = \sum_{i=1}^n \alpha_i v_i \quad (14)$$

$$Gc = \sum_{i=1}^n \lambda_i \alpha_i v_i \quad (15)$$

$$\vdots \quad \vdots \quad \vdots \quad (16)$$

$$G^n c = \sum_{i=1}^n \lambda_i^n \alpha_i v_i \quad (17)$$

But this sequence converges to $\alpha_n \lambda_n^k v_n$, as the largest eigenvalue will grow faster than any other. That means that the differences between successive vectors will get very small (outside of a scaling factor), and there will be issues with round off error. Instead, we would like to have an orthogonal basis. In particular we say that two vectors, u and v are B-orthogonal if $u^T B v = 0$. We then recursively define the basis for $K_k(G, c)$ as follows:

$$p^{(0)} = \frac{c}{\|c\|_B} \quad (18)$$

We then assume we have the first j vectors, and define the next using Gram-Schmidt. We start with $r = \hat{G}p^{(j)}$. We then use Gram-Schmidt, subtracting from r its projection onto each of the other vectors already present, and then normalizing the new result to length one. We call this vector $p^{(j+1)}$. In particular we can write:

$$p^{(j+1)} = \frac{r - h_{0,j}p^{(0)} - \dots - h_{j,j}p^{(j)}}{h_{j+1,j}} \quad (19)$$

$$h_{i,j} = p^{(i)T} B p^{(j+1)} \quad (i \leq j) \quad (20)$$

$h_{j+1,j}$ is chosen so that $\|p^{(j+1)}\|_B = 1$

The above can then be rewritten as a vector equation as follows:

$$r = h_{j+1,j} p^{(j+1)} + h_{0,1} p^{(0)} + \dots + h_{j+1,j} p^{(j)} \quad (21)$$

$$\hat{G} p^{(j)} = \begin{bmatrix} p^{(0)} & p^{(1)} & \dots & p^{(j+1)} \end{bmatrix} \begin{bmatrix} h_{0,j} \\ \vdots \\ h_{j+1,j} \end{bmatrix} \quad (22)$$

We can then group all of these vector operations together as $\hat{G} P_n = P_n \hat{H}_n$. Here H_k is $k+1 \times k$ with entries $h_{i,j}$ (this is zero if it wasn't one of the cases previously defined), and P_k is a matrix containing the vectors $p^{(\bullet)}$. This equation will be used to prove several things later, so you should keep it in mind. Notice that the vectors of P_n are B-orthogonal, so $P_n^T B P_n = I$.

A brief aside: It can be seen above that H must have the same eigenvalues as G. However, H is zero below the first subdiagonal. This means that it takes fewer operations to solve for its eigenvalues. That said, it still takes a long time, and usually a smaller H matrix is used, giving approximate eigenvalues.

This described thus far of finding P and H is called the Arnoldi Algorithm, and it is typically performed with $B = I$, (though the teacher also mentioned $B = G$).

Work For the Arnoldi Algorithm

After k iterations of the Arnoldi algorithm $O(k^2)$ inner products of length n have been formed. This means that an exact solution will take $O(n^3)$ which is higher than desired. However, if $B = I$ and \hat{G} is symmetric H_k must be symmetric, and because we know H_k is zero below the first subdiagonal it must be tridiagonal. This reduces the number of inner products to $O(k)$. The process for solving this tridiagonal system is called Lanczos Tridiagonalization.

Variational Approach

Now that we have a basis to work with we must decide how we will choose the next vector at each iteration. In the variational approach we want to simply minimize the distance from the correct solution. We can write $x^{(k)} = P_k y^{(k)}$, expressing x as a linear combination of the basis functions. Then we can express the quantity we want to minimize as a function of this, take the derivative and set it to zero. This will give us the solution.

$$\|x^{(k)} - x^*\|_Z = \left(P_k y^{(k)} - x^* \right)^T Z \left(P_k y^{(k)} - x^* \right) \quad (23)$$

To minimize this we differentiate it and set it to zero, which gives us:

$$P_k^T Z P_k y^{(k)} = P_k^T Z x^* \quad (24)$$

Unfortunately, this leaves us with two unknowns, $y^{(k)}$ and X^* . So the objective is to choose Z in a clever way, and reduce the number of unknowns.

First Clever Z: Choose Z to be $Z = \hat{G}^T B \hat{G}$, which we know is symmetric positive definite. We can then substitute into both the left and right hand side, and use the property from the basis we derived $\hat{G} P_n = P_n \hat{H}_n$.

$$\text{RHS: } P_k^T Z x^* = P_k^T \hat{G}^T B \hat{G} x^* = P_k^T \hat{G}^T B c = H_k^T P_{k+1}^T B c$$

$$\text{LHS: } P_k^T Z P_k = P_k^T \hat{G}^T B \hat{G} P_k = H_k^T P_{k+1}^T B P_{k+1} H_k = H_k^T H_k$$

$$\text{So with this choice of } Z \text{ we get: } H_k^T H_k y^{(k)} = H_k^T P_{k+1}^T B c$$

This algorithm is called GMRES (generalized minimum residual).

Second Clever Z: Choose $Z = B \hat{G}$. This is only possible if $B \hat{G}$ is symmetric and positive definite, as otherwise we cannot have a Z -norm. We can substitute this into both sides just as in the GMRES algorithm.

$$\text{RHS: } P_k^T Z x^* = P_k^T B \hat{G} x^* = P_k^T B c$$

$$\text{LHS: } P_k^T Z P_k = P_k^T B \hat{G} P_k = P_k^T B P_{k+1} H_k = \bar{H}_k$$

Here \bar{H} is a matrix comprised of the first k rows of H_k . This gives us the Conjugate gradients method:

$$\bar{H}_k y^{(k)} = P_k^T B c \quad (25)$$

Galerkin Approach/Krylov Projection

Rather than minimize the distance to the true solution we can instead make sure that the error is orthogonal to the current Krylov space. In other words, we want $r^{(k)} = c - \hat{G} x^{(k)}$ to be B -orthogonal to the columns of P_k . This can be written as:

$$0 = P_k^T B (c - \hat{G} x^{(k)}) = P_k^T B (c - \hat{G} P_k y^{(k)}) \quad (26)$$

From which we can derive:

$$P_k^T B \hat{G} P_k y^{(k)} = P_k^T B c \quad (27)$$

$$P_k^T B P_{k+1} H_k = P_k^T B c \quad (28)$$

$$\bar{H}_k y^{(k)} = P_k^T B c \quad (29)$$

This is called the Arnoldi Iteration, and it is noticeably very similar to conjugate gradient. The only difference is that it does not require that $B \hat{G}$ be symmetric and positive definite. This means that using conjugate gradient in cases where the above condition does hold performs *both* a projection and a minimization method, at the same time.

Additional Remarks

There is another way to get a good basis, called the nonsymmetric Lanczos algorithm. This leads to a variety of other iterative methods, as well as variations on all of the ones discussed. The new algorithms have the advantage of requiring only a few p vectors from the past, rather than these methods which must store the entire P_k matrix. This greatly reduces storage, and the time required to produce the basis. Unfortunately, it is possible for these algorithms to fail. If we want to reduce storage in the methods we developed, there are certain special cases, such as self-adjoint elliptic PDEs, where we can safely do so with the conjugate gradient methods.