Numerical Linear Algebra for PDEs

March 8, 2019

$$Ax = f$$

1 Thomas Algorithm

Let's take a little bit of a dive into how this would be concretely solved. In our case, A isn't just any old matrix, it's a tridiagonal matrix (with maybe a few points on the corners if the BCs are periodic). Wilson's algorithm is an efficient method ($\mathcal{O}(n)$) for $Au = f$ when A is tridiagonal. Let's see how this works.

Look at the equation as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

Now re-write this in terms of x_{i-1}. At the start you have

$$b_1 x_1 + c_1 x_2 = d_1$$

so

$$x_1 = \frac{d_1 - c_1 x_2}{b_1}$$

Now use this equation to eliminate x_1 from equation two, and then solve for x_2. You can do this all the way down to the end, and equation $n-1$ gives you an equation for x_{n-1} in terms of x_n. But now the last equation is

$$b_n x_{n-1} + c_n x_n = d_n$$

so you plug in your equation for x_{n-1} to eliminate x_{n-1} from the equation, and solve for x_n. Once you have this value, you recursively work back down the chain to calculate each of the x_i.

2 Factorization Techniques

Gaussian elimination can instead be thought of as generating two matrices, such that

$$A = LU$$

have L as lower triangular and U is upper triangular. Then backsubstitution is simple: first equation has one unknown, second has two, etc. and you iterate downwards to solve a lower or upper triangular system. Thus

$$LUx = f$$

can be easily solved by "inverting" L and then "inverting" U. This is a way of writing down Gaussian elimination without ever explicitly creating the inverse matrix A^{-1}. There are varients of this (sparse LU) which get the lower and upper triangular matrices when A is sparse, but this is a nice formulation since A^{-1} would be dense.

3 Classical Iterative Methods

The basic idea is

1. Form the residual $r = f - Au_k$
2. Compute the correction $e = Br$ with $B \approx A^{-1}$
3. Obtain the next approximation $u_{k+1} = u_k + e$

The operator B is the iterator and is assumed to be nonsingular. If we write this out, this means that

$$u_{k+1} = u_k + B\left(f - Au_k\right)$$

or

$$u_{k+1} = \left(I - BA\right)u_k + Bf$$

That means that if $B = A^{-1}$,

$$u_{k+1} = Bf = A^{-1}f$$

is the true solution! So if we instead check against the true solution u, we can get that

$$u - u_{k+1} = \left(I - BA\right)\left(u - u_k\right)$$

so $(I - BA)$ is the amplification matrix for the error. If we choose B sufficiently close to A, then this scheme will make $u_{k+1} \to u$.

How do we choose B? There are many different possible choices. One way of thinking about it is to split

$$A = D + L + U$$

where D is the diagonal, L is the lower triangular matrix, and U is the upper triangular matrix. This is done because each part is trivial to invert. Popular methods are:

1. Richardson: $B = \omega I$

2. Jacobi: $B = \omega D^{-1}$
3. Gauss-Seidel: $B = (D + L)^{-1}$
4. Successive Over-Relaxation: $B = \omega (D + \omega L)^{-1}$ (with a specific ω it's very optimized!)

Using Julia or MATLAB, writing Gauss-Seidel is simple:

$$u = u + tril(A) \backslash (f - A * u)$$

3.1 Properties of the method

1. Compatible with matrix-free operators
2. Does not require $\mathcal{O}\left(n^3\right)$ inversions! (But cost is somewhere else?)

3.2 Convergence

Let $e_k = u - u_k$ be the error in the kth step. Notice that we can write this as:

$$e_{k+1} = (I - BA)\, e_k = (I - BA)^{k+1}\, e_0$$

and so this converges iff $(I - BA)$ causes a shrinkage in norm. Whether this occurs can happen by looking at the eigenspectrum. Then we require that

$$|1 - \lambda| < 1$$

for all eigenvalues, which gives that

$$\|e_{k+1}\| = \|I - BA\| \|e_k\| < \|e_k\|$$

With this idea, you can prove some theorems.

3.2.1 Theorem 1

Richardson method with $B = \omega I$ converges iff $0 < \omega < \frac{2}{\lambda_{max}}$, and the optimal convergence rate is achieved with

$$\omega^* = \frac{2}{\lambda_{min} + \lambda_{max}}$$

The optimal convergence rate is

$$\rho_{\omega^*} = \frac{\kappa(A) - 1}{\kappa(A) + 1}$$

where κ is the condition number of the matrix A:

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}}.$$

3.2.2 Theorem 2

Jacobi method converges iff $2D - A = D - L - U$ is a symmetric positive definite matrix.

Note that a matrix is diagonally dominated if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i and is strictly if for at least one i it's $>$. One can prove that symmetric and strictly diagonally dominant is SPD.

3.2.3 Corollary

If A is strictly diagonally dominated, then Jacobi iteration always converges. Proof is simple. $A = D + L + U$, so $2D - A = D - L - U$ is diagonally dominated.

3.2.4 Theorem 3

Gauss-Seidel always converges.

3.2.5 Convergence rate of Poisson

The convergence rate with Δx is

$$\rho \leq 1 - C\Delta x^2$$

so convergence slows as $\Delta x \to 0$.

4 Conjugate Gradient and GMRES

4.1 Krylov Subspaces

A Krylov subspace is a sequence of nested subspaces

$$K_k = \text{span}\left\{b, Ab, \ldots, A^{k-1}b\right\}$$

Key property: $A^{-1}b \in K_n$ (even when $K_n \neq \mathbb{R}^n$!). Why is this? Take the Cayley-Hamilton Theorem:

$$\rho(A) = A^n + a_1 A^{n-1} + \ldots + a_n I = 0$$

where $\rho(\lambda) = \det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + \ldots + a_{n-1}\lambda + a_n$. Therefore you move a_n over and multiply by $A^{-1}b$ to get

$$A^{-1}b = \frac{1}{-a_n}\left(A^{n-1}b + a_1 A^{n-2}b + \ldots + a_{n-1}b\right) \in K_n$$

4.2 Conjugate Gradient (CG)

Assume A is symmetric positive definite. While this property won't be directly used in the justification here, it is required for convergence of cg, but the general ideas of how cg works can then be extended to not require SPD. Conjugate gradient starts from the idea of gradient decent, where we want to move in the direction that decreases the error the fastest. To do gradient decent, we could walk in the direction defined by the residual equation $Ae_k = r_k$, and then

$$\alpha_k = \frac{(r_k, r_k)}{(Ar_k, r_k)} = \frac{(u - u_k, r_k)_A}{(r_k, r_k)_A}$$

would be the norm of the gradient and so we'd walk

$$u_{k+1} = u_k + \alpha_k r_k.$$

Notice this method is nonlinear! But since we want to solve a linear equation, we can do something slightly smarter. Tracing this method back, we can approximate the gradiant method as

$$u_{k+1} = u_0 + \alpha_1 r_1 + \ldots + \alpha_k r_k.$$

Now let $\mathbb{V}_k = \text{span}\{r_0, r_1, \ldots, r_k\}$. Instead of just walking downhill, let's choose to walk in the direction that is orthogonal to all of the residual vectors we had before. Math ensues to find a basis for $\mathbb{V}_k$, orthogonalize via Gram-Schmidt, and then get a recursive process. The result is

$$\begin{aligned} u_{k+1} &= u_k + \alpha_k \rho_k \\ r_{k+1} &= r_k - \alpha_k A \rho_k \\ \rho_{k+1} &= r_{k+1} + \beta_k \rho_k \end{aligned}$$

where

$$\alpha_k = \frac{(u - u_0, \rho_k)_A}{(\rho_k, \rho_k)_A}$$

and

$$\beta_k = -\frac{(r_{k+1}, \rho_k)_A}{(\rho_k, \rho_k)_A}$$

This is called the conjugate gradient method. You can then get more efficient methods for calculating α_k and β_k as well.

4.2.1 Theorem

The convergence rate for a symmetric positive definite matrix is

$$2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k$$

4.2.2 Theorem

Conjugate gradient is the optimal Krylov subspace method, that is

$$\{\alpha_k\} = \arg\min \|u - u_k\|.$$

You can't choose any better update mechanism!

4.2.3 Corollary

Since $A^{-1}b \in K_n$ and $\{\alpha_k\}$ achieves the minimum distance from the true solution in the K_k, it follows that CG converges in n iterations. In fact, CG converges in the number of steps that matches the dimension of the Krylov subspace, assuming no numerical issues.

4.3 Preconditioners

Let $B \approx A^{-1}$. Now use the subspace $\mathbb{V}_k = \text{span}\{Br_0, Br_1, \ldots, Br_k\}$. Do the same analysis and you get the algorithm with

$$\rho_{k+1} = Br_{k+1} + \beta_k \rho_k$$

and a convergence rate of

$$2\left(\frac{\sqrt{\kappa(BA)} - 1}{\sqrt{\kappa(BA)} + 1}\right)^k$$

B is called a preconditioner. If you have a good inverse approximation that is easy to calculate, you can decrease the condition number and speedup the convergence. Notice that the classical methods are possible preconditioners!

4.4 Interesting Result

Nonlinear CG can be interpreted as L-BFGS with $m = 1$ memory.

4.5 GMRES

All of this required symmetric positive definite. An example of how this can fail is on the following problem:

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, f = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x_0 = 0.$$

Applying CG here gives $\alpha_0 = 0$, $x_1 = x_0$, $r_1 = r_0$, and then division by zero. So GMRES recovers the structure of CG but on more general matrices by directly aiming for its "best" properties. The goal is to make your residual vector orthogonal to your previous residuals and satisfy $\arg\min \|f - Au_k\|$ (least square problem). CG did this before automatically with a few (hidden) tricks. To do

this now, we first compute an orthonormal basis of our Krylov subspace by using Arnoldi iteration. Actually computing this orthogonal basis can get costly when the Krylov subspace gets large, so in practice you only keep m of the history. You then get a method that can be applied to any matrix. Preconditioning is still a thing to do which increases convergence rates "similarly".

5 Conclusion

Iterative solvers allow for solving $Au = f$ without even requiring a matrix A. Classical linear methods can be slow to converge based on the condition number of the matrix, but preconditioned GMRES is a general method (where preconditioners can be the classical iteration pieces) for solving such an equation. In many cases, it can be much faster than doing direct solving via LU or QR factorization if the matrix is large. Sparse LU factorizations can be faster, but require you know your sparsity pattern! GMRES is an easy solution for the general case, or when you get lazy.