Department of Mathematics, University of Utah

Analysis of Numerical Methods I
MTH6610 – Section 001 – Fall 2017

Lecture notes – Iterative methods for linear systems
Monday November 6, 2017

These notes are not a substitute for class attendance. Their main purpose is to provide a lecture overview summarizing the topics covered.

Reading: Trefethen & Bau III, Lecture 32

Many problems in applied mathematics and engineering rely on computing the solution $x \in \mathbb{R}^n$ to the linear system

$$Ax = b,$$

where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ are given. With modern computing power, when $n \lesssim 5000$, this can be accomplished with standard software packages and minimal human effort. These software packages solve this system *direct* methods, that is with optimized versions of Gaussian elimination (LU factorization), or perhaps with the Cholesky decomposition if A is known to be symmetric and positive definite.
However, the cost of this direct approach can be infeasible for larger n. The cost of most implemented direct solver scales like n^3. (Some optimizations can bring the exponent down to around 2.5, but this often requires special coding.) When $n > 50,000$, direct methods with cubic cost in n are not tractable. For example, simply storing a $50,000 \times 50,000$ dense matrix in double-precision format requires 20 gigabytes of memory. Direct solvers often run into computational bottlenecks for large n.
In contrast, *iterative* methods have been developed to compute solutions to linear systems for very large values of n, even exceeding $n = 10^6$. The downside is that such methods sometimes require significant knowledge about the structure or linear algebraic properties of A, and almost all of them are only efficient when A is a *sparse* matrix, i.e., when most of the entries of A are 0. A comprehensive guide to iterative methods for computing solutions to linear systems would require many hundreds of pages – we settle here for a brief, specialized, and largely deficient presentation of some of the more popular strategies and ideas.
The following bullet points distill some of the philosophical ideas and differences between direct and iterative methods:

- Iterative methods build a sequence of approximate solution $x_0, x_1, \ldots$, and frequently terminate when $\|Ax_k - b\|$ is small enough.
- The update step $x_k \mapsto x_{k+1}$ of an iterative method frequently uses only matrix-vector multiplications. Iterative methods are most efficient when these multiplications involve sparse matrices.
- The sequence of approximate solutions usually gradually converges to the solution x (say in the Euclidean norm).

- One can also view direct methods as producing a sequence of vectors that converges to the solution x after the method completes. However, this sequence has $O(1)$ error until the very final step when the error is 0 (in exact arithmetic).

We will discuss two general classes of iterative methods that are relatively popular.

Relaxation methods

Relaxation methods are iterative methods where the update step usually proceeds in a fashion like

$$x_{k+1} = B^{-1}Cx_k + c$$

for a vector c, a matrix C, and an "easily" invertible matrix B. Frequently, B is diagonal. Three popular relaxation approaches are

- Jacobi iteration
- Gauss-Seidel iteration
- Successive over-relaxation

For example, in the Jacobi method, the matrix B is the diagonal of A, and hence easily invertible. Relaxation methods are generally applied when the linear system $Ax = b$ is derived from a discretization of Laplace's equation. (In fact, one frequently needs matrix properties that are satisfied by discretizations of the Laplacian in order to prove that relaxation methods converge.)
Relaxation methods are usually not used by themselves since they converge relatively slowly. However, they are integral ingredients in one of the most popular and effective class of iterative solver techniques known today: multigrid methods.

Krylov subspace methods

A Krylov subspace is the span of the vectors

$$b, Ab, A^2b, \ldots,$$

Krylov iterative methods generate a solution x_k such that

$$r = (Ax_k - b) \quad \perp \quad \text{span}\left\{b, Ab, \ldots, A^{k-1}b\right\}$$

Such methods are in principle similar to the ideas of power iteration for finding eigenvalues. The most popular Krylov subspace methods are

- CG – the conjugate gradient method (for symmetric, positive definite systems)
- BiCG – the biconjugate gradient method
- GMRES – the method of generalized minimum residuals

Because the Krylov subspace will eventually span the entire space, after n iterations one is usually guaranteed a zero residual (i.e., an exact solution).

Preconditioning

It has been observed that the convergence of iterative methods can be significantly better when A is well-conditioned. Preconditioning is a technique that replaces $Ax = b$ by the system

$$P^{-1}Ax = P^{-1}b,$$

or even

$$AP^{-1}(Px) = b.$$

The goal of either such replacement is to generate a matrix $P^{-1}A$ or AP^{-1} that is more well-conditioned than A. Using iterative methods on this new preconditioned system usually converges more quickly. The matrix P^{-1} is called the preconditioner, and it is usually never generated or computed explicitly.