Iterative Methods for Linear Systems I: Fixed Point Relaxation Methods and Steepest Descent

Varun Shankar

February 7, 2016

1 Introduction

In this chapter, we will review a certain class of solution techniques for sparse linear systems. Consider a system of the form

\[ A x = b, \]  

where \( A \) is a sparse matrix, i.e., \( A \) has far fewer non-zero entries than zero entries. While the LU decomposition algorithm can be adapted to sparse matrices (as can the QR decomposition), it may be desirable in practical applications to solve linear systems to a specified tolerance, rather than exactly. This is not possible with the LU decomposition algorithm; intermediate results there are no closer to solving the system than the original system itself.

This gives rise to the idea of iterative methods. By performing some form of iteration on a guess vector using the matrix \( A \) and the right hand side \( b \), it may be possible to converge to the solution \( x \). We will now describe one such class of methods called relaxation methods, which are based on a stationary fixed point iteration.

2 Stationary Relaxation Methods

To understand this, consider splitting the matrix \( A \) so that \( A = M - N \). Then, we can write \( Ax = b \) as \( Mx = Nx + b \). Much like the other fixed point iterations you have learned in earlier courses, this leads to a simple fixed point iteration of the form

\[ x_{k+1} = M^{-1}Nx_k + M^{-1}b, \]  

\[ \Rightarrow x_{k+1} = x_k + M^{-1}(b - Ax_k). \]
Thus, beginning with a guess $x_0$, we iterate until $x_{k+1}$ and $x_k$ are the same (within some tolerance). Define the residual $r_k = b - Ax_k$. Then, we have

$$x_{k+1} = x_k + M^{-1}r_k.$$  \hspace{1cm} (4)

Clearly, the properties of $M$ will dictate whether this iteration converges to $x$, and at what rate. We will now discuss a few choices of the splitting matrix $M$. The following methods are called stationary methods because the choice of $M$ is fixed over all iterations. Further, they are called relaxation methods because they are all equivalent to a smoothing operation on the iterates $x_k$.

How does one pick $M$? Well, clearly, we want $M^{-1}$ to be easy to compute (or apply). At the same time, the best choice of $M$ would be as close to $A^{-1}$ as possible. The tension between these two choices leads to different methods.

### 2.1 Jacobi iterations

In the Jacobi method, we pick $M = D$, where $D$ is the diagonal matrix simply containing the diagonal elements of $A$. With this choice, it is obviously trivially easy to compute $M^{-1}$. Thus, we have

$$x_{k+1} = v x_k + D^{-1}r_k.$$  \hspace{1cm} (5)

Notice that this never needs $A$ supplied in explicit form. To get the residual, we only need a routine that applies $A$ to any vector (in this case, $x_k$). In component form, the Jacobi iteration looks like this:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^k \right], \quad i = 1, \ldots, n.$$  \hspace{1cm} (6)

For each component $i$, this iteration simply zeroes out the $i^{th}$ residual by adjusting $x_k$. The other unknowns are not touched. Thus, each operation can be performed on each unknown completely in parallel. While the Jacobi method is not the fastest converging iterative method, its pleasingly parallel nature makes it extremely popular in several contexts.

### 2.2 Gauss-Seidel

The Gauss-Seidel method involves picking $M$ to be $L$, the lower triangular part of $A$ (including the diagonal). Since inverting a triangular system is in fact fairly straightforward (forward substitution), the Gauss-Seidel method is only slightly more expensive than the Jacobi method. More importantly, since we use $L$ rather than $D$, Gauss-Seidel converges twice as fast as Jacobi, when Jacobi converges on the problem. The iteration is thus

$$x_{k+1} = x_k + L^{-1}r_k.$$  \hspace{1cm} (7)
Remember: you should never explicitly form $L^{-1}$. Rather, you should perform forward substitution to solve the system $Ly = r_k$. In component form, Gauss-Seidel is given by

$$x_i^{k+1} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j<i} a_{ij} x_j^{k+1} - \sum_{j>i} a_{ij} x_j^k \right], i = 1, \ldots, n. \quad (8)$$

Note that we use newly computed values $x_j^{k+1}$ to compute other values $x_i^{k+1}$. Thus, Gauss-Seidel cannot be executed straightforwardly in parallel.

### 2.3 Successive Over-Relaxation

The SOR method is straightforward. The idea is to use a combination of the Jacobi idea and the Gauss-Seidel idea to improve on Gauss-Seidel’s convergence. Thus, we say that

$$x_{k+1} \leftarrow \omega x_{k+1} + (1 - \omega) x_k. \quad (9)$$

It can be shown that if $1 < \omega < 2$, the resulting algorithm (SOR) is much faster than Gauss-Seidel. In matrix form, this is given by

$$x_{k+1} = x_k + \omega ((1 - \omega)D + \omega L)^{-1} r_k. \quad (10)$$

Typically, $\omega$ is chosen based on knowledge of the structure of $A$. For example, if we know the eigenvalues of $A$ in closed form, $\omega$ could be selected to be some function of the eigenvalues of $A$.

### 2.4 Convergence of Stationary Relaxation Methods

We will now analyze the rate at which the above fixed point iterations converge to the true solution. We know that the error at the $k^{th}$ step is

$$e_k = x - x_k = A^{-1} r_k. \quad (11)$$

We want that as $k \to \infty$, $e_k \to 0$. To derive the conditions for this, we must analyze the properties of the iteration. Let us write the $k^{th}$ iterate as

$$x_k = M^{-1} b + (I - M^{-1} A) x_{k-1}. \quad (12)$$

Formally, we also know that

$$x = M^{-1} b + (I - M^{-1} A) x. \quad (13)$$

Let $T = I - M^{-1} A$. Subtracting the two equations, we have

$$e_k = T e_{k-1} = T^2 e_{k-2} = \ldots = T^k e_0. \quad (14)$$
The matrix $T$ is called the iteration matrix. Clearly, $e_k \to 0$ only if $T^k \to 0$.

This will happen if $||T|| < 1$ in any induced matrix norm.

It is also possible to derive a tighter bound. We in fact know that $T^k \to 0$ as $k \to \infty$ if the “magnitude” of $T$ is less than 1. In other words, we simply want the magnitude of the largest eigenvalue to be less than 1. This quantity (the magnitude of the largest eigenvalue) is called the spectral radius of $T$, $\rho(T)$.

Thus, a fixed point iteration converges if $\rho(T) < 1$. The rate of convergence, of course, is determined by how small $\rho(T)$ is; the smaller the spectral radius, the faster the convergence. Clearly, if the matrix $T$ has large eigenvalues that are somehow positive and greater than 1, the stationary iteration will not converge unless the tolerance is chosen carefully.

3 Steepest Descent

In steepest descent, we now seek an iteration of the form

$$x_{k+1} = x_k + \alpha_k p_k,$$

(15)

where $p_k$ is a search direction, and $\alpha_k$ is a carefully-chosen constant (for each iteration). Clearly, this is a generalization of the stationary methods discussed thus far. The goal here is to pick $p_k = r_k$, and find the $\alpha_k$ that helps us “descend” fastest to the solution. Thus, we have an iteration of the form

$$x_{k+1} = x_k + \alpha_k r_k.$$  

(16)

To find $\alpha_k$, we first remark that the solution $x$ to any linear system of the form $Ax = b$ can be viewed as finding a vector $x$ that minimizes

$$\phi(x) = \frac{1}{2} x^T A x - b^T x.$$  

(17)

To find $\alpha_k$, we similarly wish to minimize $\phi(x_k + \alpha r_k)$ over all values of $\alpha$. This amounts to minimizing

$$\phi(x_k + \alpha r_k) = \frac{1}{2} (x_k + \alpha r_k)^T A (x_k + \alpha r_k) - b^T (x_k + \alpha r_k).$$

(18)

This is done by setting the derivative of the above function w.r.t $\alpha$ to 0. This gives

$$\alpha r_k^T Ar_k + r_k^T Ax_k - r_k^T b = 0.$$  

(19)

Using the fact that $b - Ax = r_k$, we get

$$\alpha = \alpha_k = \frac{r_k^T r_k}{r_k^T Ar_k} = \frac{(r_k, r_k)}{(r_k, Ar_k)}.$$  

(20)

Though the steepest descent method is typically faster than the relaxation method, it displays the same order of convergence: the number of iterations is linear in the number of unknowns. To get much faster convergence