Solving Sparse Systems of Linear Equations

America Holloway
Department of Computer Science
University of California, Irvine, Irvine, CA
ahollowa@ics.uci.edu

1 Introduction

Let \( A \in \mathbb{R}^{n \times n} \) be a sparse, square matrix and \( b \in \mathbb{R}^n \) a real vector. (A sparse matrix has few nonzero entries relative to its size). Our goal is to find the vector \( x \in \mathbb{R}^n \) such that \( Ax = b \). One approach is to use a direct method for finding an exact solution. However, when a direct method is infeasible (e.g., due to time or memory constraints) we must settle for finding a good enough approximation to \( x \) instead of the exact solution. Even for smaller matrices where direct methods can be used, the coefficient matrix \( A \) often becomes filled-in during the process of solving for \( x \).

In this paper, we implement four algorithms for finding approximate solutions to sparse systems of equations: Gauss-Seidel, Successive Over Relaxation, the Full Orthogonalization Method and the Generalized Minimum Residual Method. We test each algorithm on a collection of matrices taken from the University of Florida Sparse Matrix Collection [1].

2 Iterative Methods

Given an approximation \( x \) to the exact solution, we define the residual as the difference between the right-hand-side vector \( b \) and \( Ax \). Ideally, we would like the residual to equal zero, or to be as close to zero as possible. Both Gauss-Seidel (GS) and Successive Over Relaxation (SOR) iterate through the components of the current approximation, and modify the component values so as to reduce the residual.

Let \( x^k \) be the approximation at the \( k \)th iteration. For the \( k+1 \)st iteration, we would like the \( i \)th component of the residual \( b - Ax^{k+1} \) to equal zero.

\[
0 = b_i - \sum_{j=1}^{n} a_{i,j} x_i^{k+1}
\]

Solving for \( x_i^{k+1} \) gives,

\[
0 = b_i - \sum_{j=1}^{n} a_{i,j} x_i^{k+1}
\]

\[
a_{i,i} x_i^{k+1} = b_i - \sum_{j=1, j \neq i}^{n} a_{i,j} x_j^k
\]

\[
x_i^{k+1} = \frac{1}{a_{i,i}} \left( b_i - \sum_{j=1, j \neq i}^{n} a_{i,j} x_j^k \right)
\]

Note that each component of \( x^{k+1} \) is computed using the old approximation \( x^k \). This is known as the Jacobi Algorithm. GS is the same algorithm except the new value of the \( i \)th component is used to compute the remaining component values. Note that both Jacobi and GS assume the diagonal elements of \( A \) are nonzero since we must be able to multiply by \( \frac{1}{a_{ii}} \) at each iteration.
For SOR, the new approximation is a convex combination of the $x^{k+1}$ vector computed from GS and the previous approximation $x^k$. The update rule is given below.

$$x^{k+1} = \omega x_{GS}^{k+1} + (1-\omega)x^k$$

The value $\omega$ is a parameter set by the user. When $\omega = 1$, SOR is equivalent to GS.

## 3 Projection Methods

Recall from class that the normal equations allow us to solve a linear least squares problem by finding the vector $x$ that minimizes $||b - Ax||_2$. We derived the normal equations and later showed that the best approximation to $b$ is actually the projection of $b$ onto the subspace spanned by the columns of the matrix $A$.

We can generalize the above process as follows: Given a linear system $Ax = b$, we want to find the vector $x \in \mathcal{L}$ that minimizes the norm of the residual $b - Ax$. The difference is that now our search space is a subspace $\mathcal{L} \subset \mathbb{R}^n$ and we are not necessarily minimizing the two-norm of the residual vector. We can generalize this process even more. Instead of searching for $Ax$ directly we are going to search for the vector $x \in \mathcal{M}$ whose image $Ax$ minimizes the norm of the residual vector. This is the general formulation of a projection method and is summarized below.

Find $x \in \mathcal{M}$ such that $(b - Ax) \perp \mathcal{L}$.

The condition that $(b - Ax) \perp \mathcal{L}$ is equivalent to saying $Ax$ is the projection of $b$ onto $\mathcal{L}$, or in other words, that $Ax$ minimizes the norm of the residual. Thus to define a projection method we only need to choose the subspaces $\mathcal{M}$ and $\mathcal{L}$.

If we have an initial guess $x_0$, the formulation then becomes

Find $x = x_0 + \delta$ where $\delta \in \mathcal{M}$ such that $(b - Ax) \perp \mathcal{L}$.

Once we have chosen $\mathcal{M}$, $\mathcal{L}$, and $x_0$ we need an algorithm for finding $x$. Let $m$ be the dimension of $\mathcal{M}$ and $\mathcal{L}$. Let $V$ be a basis for $\mathcal{M}$ and $W$ be a basis for $\mathcal{L}$ (i.e. $V$ and $W$ are $n \times m$ matrices). $\delta$ can be written as a linear combination of the basis vectors of $V$, i.e., let $\delta = Vy$ where $y$ is an $m \times 1$ column vector. Then

$$x = x_0 + \delta \quad \text{such that} \quad b - Ax \perp \mathcal{L}$$
$$x = x_0 + Vy \quad \text{such that} \quad b - Ax \perp \mathcal{L}$$
$$x = x_0 + Vy \quad \text{such that} \quad b - A(x_0 +Vy) \perp \mathcal{L}$$
$$x = x_0 + Vy \quad \text{such that} \quad b - Ax_0 - AVy \perp \mathcal{L}$$

Denote the residual of the initial guess as $r_0$. Then

$$x = x_0 + Vy \quad \text{such that} \quad r_0 - AVy \perp \mathcal{L}$$

If $r_0 - AVy \perp \mathcal{L}$, then $w_i^T(r_0 - AVy) = 0$ for all basis vectors $w_i$ of $\mathcal{L}$. This can be rewritten as

$$W^T(r_0 - AVy) = 0$$
$$W^T r_0 - W^T AVy = 0$$

$$(1) \quad W^T r_0 = W^T AVy$$

$$y = (W^T AV)^{-1} W^T r_0$$

This provides a method for finding $x$. If we have a basis $V$ for $\mathcal{M}$, a basis $W$ for $\mathcal{L}$ and an initial guess $x_0$, we can find $y$ given the above equation. Once we have $y$, we can solve for $x$. Both projection methods we consider follow this format.
3.1 Krylov Subspaces and Arnoldi’s Method

The above algorithm requires choosing the subspaces $\mathcal{M}$ and $\mathcal{L}$. For both the Full Orthogonalization Method (FOM) and the Generalized Minimum Residual Method (GMRES), the subspace $\mathcal{M}$ is chosen to be the Krylov Subspace $\mathcal{K}(r_0, m)$ where

$$
\mathcal{K}(r_0, m) = \text{span}(r_0, Ar_0, A^2r_0, \cdots, A^{m-1}r_0)
$$

Recall that $r_0$ is the residual of our initial guess $x_0$. These algorithms differ in their choice of $\mathcal{L}$. For FOM, $\mathcal{M} = \mathcal{L}$ and for GMRES $\mathcal{L} = A\mathcal{M}$.

Arnoldi’s Method computes an orthonormal basis for the Krylov subspace $\mathcal{K}(r_0, m)$ using the Gram-Schmidt Algorithm discussed in class. We begin by normalizing $r_0$. This is the first basis vector. Given the first $i$ basis vectors, we compute the $(i+1)$st basis vector by orthogonalizing and normalizing $Av_i$ using Gram-Schmidt (here $v_i$ denotes the $i$th basis vector).

3.2 FOM and GMRES

For FOM, $\mathcal{M} = \mathcal{L}$. Recall from equation 1 that to solve for $y$ we must compute $(W^TAV)^{-1}W^Tr_0$ where $W = V$. Luckily, $W^Tr_0$ and $W^TAV$ can be computed as a side product of Arnoldi’s algorithm.

For GMRES, $\mathcal{L} = A\mathcal{M}$. We can again exploit Arnoldi’s algorithm to compute the vector $y$. However in the process a least squares problem must also be solved. In our implementation, we do this by transforming the coefficient matrix of the least squares problem into a diagonal matrix and using back substitution. For more details, see [2].

(Note that for both FOM and GMRES, the dimension of the Krylov subspace $\mathcal{M}$ is a user-chosen parameter.)

4 Results

We downloaded 5 square matrices from the University of Florida’s Sparse Matrix Collection. The only requirement was that each matrix had nonzero diagonal elements. Each system was solved using the four methods described earlier: GS, SOR, FOM and GMRES. For SOR, $\omega$ was set to .75. For the Krylov subspace methods, we implemented a variation of FOM and GMRES called RestartedFOM and RestartedGMRES. For each iteration, we increase the value of $m$ (the dimension of the Krylov subspace) by one until we reach a maximum value set by the user. This allows us to search smaller subspaces first which may reduce the number of floating-point operations required to converge to a solution.

Each algorithm is run until the norm of the residual is less than $1 \times 10^{-7}$. The time (in seconds) and the number of iterations is recorded for each method.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>No. Rows</th>
<th>Gauss-Seidel</th>
<th>SOR</th>
<th>FOM</th>
<th>GMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Iters</td>
<td>Time</td>
<td>Iters</td>
</tr>
<tr>
<td>CAGE4</td>
<td>9</td>
<td>0.063</td>
<td>36</td>
<td>0.028</td>
<td>51</td>
</tr>
<tr>
<td>LFAT5</td>
<td>14</td>
<td>0.431</td>
<td>825</td>
<td>0.600</td>
<td>1103</td>
</tr>
<tr>
<td>BFWA</td>
<td>62</td>
<td>50.701</td>
<td>12289</td>
<td>67.207</td>
<td>16391</td>
</tr>
<tr>
<td>TUB100</td>
<td>100</td>
<td>50.200</td>
<td>12283</td>
<td>67.189</td>
<td>16382</td>
</tr>
<tr>
<td>STEAM2</td>
<td>600</td>
<td>50.435</td>
<td>12288</td>
<td>68.176</td>
<td>16389</td>
</tr>
</tbody>
</table>

Table 1. Timing Results

Both GS and SOR are dominated by FOM and GMRES. As the size of the matrix increases, GS and SOR require an increasingly large amount of time. The time needed for FOM and GMRES grows more modestly as the size of the matrix increases.
The performance of GMRES and FOM are comparable; it seems to depend on the characteristics of the system. It is interesting to note that for TUB100, GMRES had a longer running time but fewer iterations. (This was observed in many other runs not reported here). A single iteration of GMRES requires solving a linear system of equation. Thus each iteration necessarily takes longer but may also reduce the norm of the residual more dramatically.

References