Math 4445 Term Project
Iterative Methods for Solving Linear Systems

There are two general categories of approaches to solving a linear system of equations, $Ax = b$. These two categories are labeled as direct methods and iterative methods. Direct methods (typified by Gaussian Elimination, for example) require a relatively high fixed cost in terms of computational resources (time and memory), but are guaranteed to produce an approximation to the exact solution $x^*$, that is maximally accurate in a certain sense. By contrast, iterative methods generate a sequence of vectors $x_0, x_1, x_2, \ldots, x_k, \ldots$ that provides (hopefully) increasingly accurate approximations to $x^*$. A fundamental component of practical iterative methods is that they are able to generate elements of this vector sequence cheaply, generally comparable to the cost of a single matrix-vector multiply for each new vector iterate. If the problem setting brings with it only a moderate requirement of accuracy in the computed solution, there is the prospect of stopping the iteration at an early stage when accuracy requirements have been already met, at a net cost far below what a direct method (such as Gaussian elimination) might have required. Of course, the danger that must be confronted is that convergence of $x_k$ to $x^*$ may be tortuously slow, so that many iterations may be required which may in turn drive the cost up to an unacceptable level.

This project is intended to introduce you to the fundamentals of iterative methods and give you an opportunity to explore some of the basic ideas.

1 Basic Construction of an Iterative Method

We begin by writing $A$ as the difference of two matrices, $A = M - N$ where $M$ is invertible. This is called a splitting of $A$. Obviously there are many ways one could do this and we will have to explore ways of distinguishing useful ways of defining a splitting for $A$.

Once a splitting has been selected, one may then rearrange the equation $Ax = b$ to get $Mx = Nx + b$ or equivalently

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

Now, if an initial approximation $x_0$ is close to the exact solution $x^*$ and $x_0$ is substituted for $x$ on the right-hand side then the resulting “$x$” on the
left-hand side should be close to \( x^* \) as well. Furthermore, if it is possible to insure that the resulting left-hand side actually is closer to the exact solution \( x^* \) than the original approximation \( x_0 \), we could repeat the process to get a still better approximation. This leads to the iteration:

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

But what determines whether or not the iterates do get closer to \( x^* \) as the method proceeds, and then how quickly they get close to \( x^* \)?

Since the exact solution \( x^* \) satisfies \( x^* = M^{-1}Nx^* + M^{-1}b \), subtract this equation from \( x_{k+1} = M^{-1}Nx_k + M^{-1}b \) to get an expression for the error at the \( k \)th iteration:

\[
e_k = x_{k+1} - x^* = M^{-1}N(x_k - x^*) = (M^{-1}N)^2(x_{k-1} - x^*) = (M^{-1}N)^k(x_0 - x^*) = (M^{-1}N)^k e_0
\]

Hence the rate at which the error \( e_k \to 0 \) is governed by the “smallness” of \( M^{-1}N \). To simplify things for the moment, suppose we happen to have an initial error vector, \( e_0 \) that is a scalar multiple of an eigenvector of \( M^{-1}N \):

\[
(M^{-1}N)e_0 = \lambda e_0
\]

Then since \((M^{-1}N)^k e_0 = \lambda^k e_0\), we can see that \( e_k \to 0 \) if and only if \( |\lambda| < 1 \). That is, the iterative method would converge to the exact solution if and only if the eigenvalue \( \lambda \) has magnitude strictly less than 1. Of course, there is no reason to expect that the initial error \( e_0 \) to have lined up nicely with an eigenvector of \( M^{-1}N \), but it isn’t hard to use this special case to guide us into a more general situation. If \( M^{-1}N \) has a basis of eigenvectors \( \{u_1, u_2, u_3, \ldots, u_n\} \) with associated eigenvalues \( \{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n\} \), we can be sure to be able to represent \( e_0 \) as a linear combination of eigenvectors of \( M^{-1}N \):

\[
e_0 = \sum_{i=1}^n \alpha_i u_i
\]

and then

\[
e_k = (M^{-1}N)^k e_0 = \sum_{i=1}^n \alpha_i \lambda_i^k u_i
\]
To be sure that \( e_k \to 0 \), we must be sure that the component of \( e_k \) in the direction of each eigenvector goes to zero, that is, we need \( |\lambda_i| < 1 \) for each \( i = 1, 2, \ldots, n \). The slowest convergence possible would occur if \( e_0 \) were purely in the direction of the eigenvector associated with the largest magnitude eigenvalue and then we would see a decrease in error (for large enough \( k \)) no slower than

\[
\|e_k\| \approx \left( \max_i |\lambda_i| \right) \|e_{k-1}\| \approx \left( \max_i |\lambda_i| \right)^k \|e_0\|
\]

That is, the error at each step decreases by a factor no worse than the largest magnitude eigenvalue of \( M^{-1}N \). (Things are a little more complicated when \( M^{-1}N \) does not have a full basis of eigenvectors, but roughly speaking, the same conclusion is true.)

The quantity \( \max_i |\lambda_i| \) is called the *spectral radius* of \( M^{-1}N \) and is denoted by \( \rho(M^{-1}N) \).

Suppose that we have chosen a splitting of \( A = M - N \) so that

\[
\rho(M^{-1}N) < 1.
\]

Since the largest magnitude eigenvalue is smaller than one (in magnitude), all eigenvalues of \( M^{-1}N \) must have magnitude strictly less than 1, and iteration with this splitting will converge to the exact solution \( x \), regardless of how good (or how bad) the initial approximation \( x_0 \). In order to guarantee that the error at the \( k \)th step is reduced to, say, 1% of its initial value (i.e., \( \|e_k\| \leq 10^{-2} \|e_0\| \)), \( k \) must be large enough to satisfy \( \rho(M^{-1}N)^k \leq 10^{-2} \) implying

\[
k \geq \frac{-2}{\log_{10} (\rho(M^{-1}N))}
\]

At least this many iterations should be done to guarantee a reduction of error to 1% of its initial value. More may be necessary (see below).

**Problem 1.1** Let \( A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \) and consider the splitting \( M = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \) and \( N = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \). Predict (with careful justification) how many iterations will be necessary to guarantee a reduction in the error to 0.1% (i.e., \( 10^{-3} \)) of its initial value.
Problem 1.2 Show that if $A$ is singular then the iteration $x_{k+1} = M^{-1}Nx_k + M^{-1}b$ will not be convergent in general, even if $M$ is non-singular.

Problem 1.3 Consider the matrix $A(\tau) = \begin{bmatrix} 1 + \tau & 1 - \tau \\ 1 - \tau & 1 + \tau \end{bmatrix}$ with the splitting $M(\tau) = \begin{bmatrix} 1 + \tau & 0 \\ 0 & 1 + \tau \end{bmatrix}$ and $N(\tau) = \begin{bmatrix} 0 \\ -(1 - \tau) \end{bmatrix}$. Notice that at $\tau = 0$, $A(\tau)$ is singular. Plot and discuss the values of $\rho(M(\tau)^{-1}N(\tau))$ vs. $\tau$ for $\tau \in [-1, 1]$.

It is not always convenient (or even possible) to calculate the eigenvalues of $M^{-1}N$ in our efforts to get a handle on $\rho(M^{-1}N)$. Sometimes, it may be easier to calculate a matrix norm $\|M^{-1}N\|$, where typically either the $\infty$-norm or the 1-norm is calculated. Notice that for any eigenvalue $\lambda$ of $M^{-1}N$ (with associated eigenvector $u$):

\[ |\lambda|\|u\| = \|\lambda u\| = \|M^{-1}Nu\| \leq \|M^{-1}N\|\|u\| \]

So, in particular, $\rho(M^{-1}N) \leq \|M^{-1}N\|$, and $\|M^{-1}N\|$ will always provide a pessimistic (but cheap) estimate to the rate at which the iteration will converge.

Although the condition $\rho(M^{-1}N) < 1$ is both necessary and sufficient for eventual convergence of the error to zero, it is not sufficient to guarantee monotone convergence of the error. In particular, the error could actually increase for a time in the course of the iteration before eventually diminishing to zero.

Problem 1.4 Consider an iteration matrix $B(\alpha) = M^{-1}N = \begin{bmatrix} \alpha & 4 \\ 0 & \alpha \end{bmatrix}$. $\rho(B(\alpha)) = \alpha$ and the basic iteration is convergent for $|\alpha| < 1$. Determine how many iterations are needed to get an iteration error $\|e_k\|_\infty$ smaller than 10% of the initial error $\|e_0\|_\infty$, if $\alpha = 0.99$ and $e_0 = \{1, 1\}^t$? How does this compare with an estimate based on $\rho(B(\alpha))^k$? Plot $\|e_k\|_\infty$ vs. $k$ and $\rho(B(\alpha))^k$ vs $k$ on the same plot.
2 Two Basic Iterations

How does one select a good splitting?

From a practical point of view, we’d like to choose a splitting \( A = M - N \) so that linear systems having \( M \) as a coefficient matrix are easy to solve. Two nice possibilities should occur naturally to us – when \( M \) is diagonal; and when \( M \) is triangular.

The Jacobi iteration: Take \( M = \text{diag}(A) \) (the diagonal of \( A \) in Matlab notation). The iteration \( x_{k+1} = M^{-1}Nx_k + M^{-1}b \) may then be written componentwise as

\[
x_i^{(k+1)} = \frac{1}{a_{ii}} \sum_{j \neq i} a_{ij}x_j^{(k)} + \frac{b_i}{a_{ii}}
\]

for each \( i = 1, \ldots, n \).

**Problem 2.1** Consider the matrix given by

\[
A = \begin{bmatrix}
3 & 1 & -1 \\
2 & 4 & 1 \\
1 & -1 & 3
\end{bmatrix}
\]

Without computing any eigenvalues, show the Jacobi method is convergent and give an upper bound to the number of iterations necessary to reduce the error to 1% of its initial value.

**Problem 2.2** Suppose \( A \) is an \( n \times n \) matrix that is *diagonally dominant* (which means that \( \sum_{j \neq i} |a_{ij}| < |a_{ii}| \) for each row index \( i \)). Show that Jacobi’s method is convergent. Furthermore, if \( \xi = \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} \) and \( x \) is the exact solution to \( Ax = b \), show that the Jacobi iterates \( \{x_k\} \) satisfy

\[
\|x - x_k\|_\infty \leq \max_i \frac{\xi_i}{1 - \xi_i} \|x_k - x_{k-1}\|_\infty
\]

Notice that the right-hand side contains explicitly computable quantities.

The Gauss-Seidel iteration: Take \( M = \text{tril}(A) \) (the lower triangle of \( A \) in Matlab notation). At each step solve the (lower) triangular system \( Mx_{k+1} = Nx_k + b \). This may be written componentwise as

\[
x_i^{(k+1)} = \frac{1}{a_{ii}} \sum_{j<i} a_{ij}x_j^{(k+1)} - \frac{1}{a_{ii}} \sum_{j>i} a_{ij}x_j^{(k)} + \frac{b_i}{a_{ii}}
\]
This is in the spirit of Jacobi’s method with the added (plausible) refinement that we use the most recent values available in computing the (Jacobi) right-hand side $\sum_{j \neq i} a_{ij} x_j^{(\ell)}$, but assigning $\ell = k + 1$ if it is available and $\ell = k$ otherwise.

**Problem 2.3** Consider the matrix given by

$$A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

If $J$ denotes the Jacobi iteration matrix and $G$ denotes the Gauss-Seidel iteration matrix, show that

$$\rho(G) = \rho(J)^2 < 1$$

and as a result, about twice as many iterations might be necessary when using Jacobi iteration than would be necessary when using Gauss-Seidel iteration to solve $Ax = b$ to the same accuracy.
Problem 2.4 Write a Matlab M-file that applies both the Jacobi and Gauss-Seidel methods to the linear system \( Ax = b \) where

\[
A = \begin{bmatrix}
T & I \\
I & T
\end{bmatrix}, \quad T = \begin{bmatrix}
-4 & 1 & 0 & 0 & 0 \\
1 & -5 & 1 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 \\
0 & 0 & 1 & -5 & 1 \\
0 & 0 & 0 & 1 & -4
\end{bmatrix},
\]

and \( I \) is the 5\( \times \)5 identity matrix; the right-hand side is \( b = [0 \ 6 \ 3 \ 1 \ 1 \ 6 - 1 \ 4 \ 0 \ 6]^T \). Use as an initial approximation \( x_0 = b \) (i.e., take the right-hand side itself as a crude approximation to the solution) and terminate the iterations when the relative change in each component is less than 0.05\% from one iteration to the next. Additionally, your m-file should calculate and record the (true) error at each iteration and plot error vs iteration index on semilog axes at the end (use the matlab function \texttt{semilogy}). For the purposes of calculating the error, you can use the fact that the exact solution for the given right hand side is \( x^* = [-1 \ -2 \ -2 \ -1 \ -1 \ -2 \ -1 \ -2 \ -1 \ -2]^T \). Calculate the spectral radius of both the Jacobi iteration matrix and the Gauss-Seidel iteration matrix and use this to predict how many iterations should be necessary to reduce the error to 1\% of its initial magnitude. Comment on your plot and computational results and compare your predictions with the performance of the m-file you’ve written.

3 SOR Methods

Convergence of the Gauss-Seidel is often monotone in the components so it may be tempting to accelerate the process by extrapolating an improved estimate from a combination of current and previous iterates.

Suppose we know all components of the previous iterate \( x_k \) and the first \( i - 1 \) components of the current iterate \( x_{k+1} \) for some index \( i, \ 1 \leq i \leq n \). To compute the \( i \)th component, \( x^{(k+1)}_i \), begin by computing the Gauss-Seidel
value \( x_i^{(k+1)} \) from

\[
a_{ii} x_i^{(k+1)} + \sum_{j<i} a_{ij} x_j^{(k+1)} + \sum_{j>i} a_{ij} x_j^{(k)} = b_i
\]  

Then extrapolate an improved value from the previous value \( x_i^{(k)} \) and the current Gauss-Seidel value \( x_i^{(k+1)} \) as:

\[
x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \omega x_i^{(k+1)}
\]  

for an appropriate choice of \( \omega \). If \( \omega < 1 \) this strategy is called underrelaxation and corresponds to interpolating a value for \( x_i^{(k+1)} \) between \( x_i^{(k)} \) and \( \hat{x}_i^{(k+1)} \); if \( \omega > 1 \) this strategy is called overrelaxation and corresponds to extrapolating a value for \( x_i^{(k+1)} \) starting from \( x_i^{(k)} \) and shooting beyond \( \hat{x}_i^{(k+1)} \). Since acceleration of convergence is the goal, extrapolation is often the appropriate tactic and so these methods are often labeled “SOR” methods (for Successive OverRelaxation). Notice that for \( \omega = 1 \), this is just Gauss-Seidel method.

Once we decide how to fix the extrapolation parameter, \( \omega \), this clearly would define an iterative method of sorts, but how can we express this iteration in terms of a splitting of \( A \)? Multiply equation (1) by \( \omega \), solve equation (2) for \( \omega \hat{x}_i^{(k+1)} \), and substitute:

\[
a_{ii}(x_i^{(k+1)} - (1 - \omega)x_i^{(k)}) + \omega \sum_{j<i} a_{ij} x_j^{(k+1)} + \omega \sum_{j>i} a_{ij} x_j^{(k)} = \omega b_i. 
\]  

After dividing out \( \omega \), rearrange equation (3) to get:

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

This (component) iteration may be written as a vector iteration \( Mx_{k+1} = Nx_k + b \), by formulating a splitting of \( A \) as follows. Let \( D \) be the diagonal of \( A \); \( L \) be the strict lower triangle of \( A \) (excluding the diagonal); and \( U \) be the strict upper triangle of \( A \) (also excluding the diagonal). Then for any \( \omega \neq 0 \),

\[
A = L + D + U = L + \frac{1}{\omega} D - \frac{1}{\omega} D + D + U = \frac{1}{\omega} (\omega L + D) - \frac{1}{\omega} [(1 - \omega) D - \omega U]
\]

This defines a splitting of \( A = M - N \) with \( M = \frac{1}{\omega} (\omega L + D) \) and \( N = \frac{1}{\omega} [(1 - \omega) D - \omega U] \) corresponding to the iteration defined by (4).
Notice that the convergence rate of this iteration is governed by the size of the iteration matrix \( H(\omega) = M^{-1}N = (I + \omega D^{-1}L)^{-1}(1 - \omega)I - \omega D^{-1}U \).

**Problem 3.1** For the matrix \( A \) given in problem (2.4), graph the spectral radius of \( H(\omega) \) vs. \( \omega \) for all \( \omega \in [0, 2] \). With the help of this plot, determine the optimal relaxation parameter (the \( \omega \) giving the smallest spectral radius). Predict how many iteration steps would be necessary to solve the linear system \( Ax = b \) of problem (2.4) to the tolerances stated in that problem. Modify your m-file of problem (2.4) to perform an SOR iteration with the optimal relaxation parameter that you’ve obtained, and plot the error vs. iteration index as in problem (2.4) together with the Jacobi and Gauss-Seidel error plots. Comment on your results.