Experiments on ILU(0) Preconditioning

Cagri Abis
Kyle Bordeau
Andrew Rice

April 27, 2012

Contents

Abstract

The goal of this paper is to investigate the effect of using preconditioners in solving the linear equation $Ax = b$ with three different types of iterative solvers: conjugate gradient method, biconjugate gradient method, and generalized minimum residual method. Specifically, the incomplete LU decomposition for a sparse matrix, ILU(0), is used to obtain the preconditioning matrices. MATLAB is used to compute the decompositions, solve the equations, and record timings, residuals, and iterations for the testing.

1.0 Introduction & Background

Systems of linear equations are very common in the real world. A set of linear equations can be used to describe physical shapes or the behavior of structures. Because linear equations are so prevalent, it is important that there are efficient methods to solve them. The most direct method of solving a system is to find the inverse of the coefficient matrix. However for large systems of more than 1000 equations, it becomes difficult to compute the matrix inverse. In this case it is often best to use an iterative solving method to obtain an accurate answer.

Iterative solvers yield more accurate and faster results for well-conditioned matrices. The condition number of a matrix A is defined as

$$K_P(A) = ||A||_P||A^{-1}||_P$$

and is an indicator of how accurate a result can be derived from a matrix. For a linear solution of a matrix, the condition number is the number of digits that will be lost in matrix multiplication [2]. Solving with iterative methods requires many iterations to coverage to a solution. Therefore, computing matrix multiplications again and again will cause a considerable loss in precision. A small condition number helps to preserve the precision after numerous iterations.

The lowest possible condition number is 1 and occurs for the identity matrix. If the identity matrix were used as the coefficient matrix, then the solution can be reached in one iteration. A desired method then to improve the efficiency of some iterative solver is to use a preconditioning matrix, P, that will decrease the condition number of the problem such that $P^{-1}Ax = P^{-1}b$ [1]. Ideally $P = A^{-1}$, but as previously mentioned it is not always feasible to compute a matrix inverse. One alternative is to decompose A into upper and lower triangular matrices using the LU decomposition. Then, the preconditioner would be $P = LU$ and the new problem is $U^{-1}L^{-1}Ax = U^{-1}L^{-1}b$. This new equation is much easier to compute because the condition number of the resulting coefficient matrix is small.

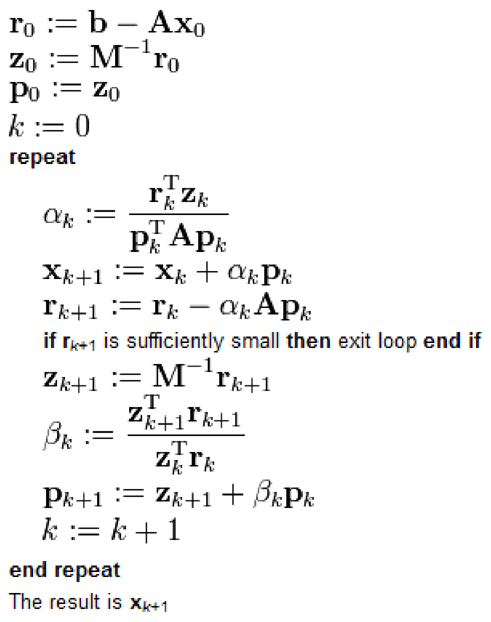
For small problems the LU decomposition will compute the correct matrices such that $LU = A$. For the real world case, the matrix A is often very large and mostly sparse. For these large sparce matrices, the LU decomposition needs to be replaced with the incomplete LU decompositon for sparce matrices with no fill, or ILU(0). This paper investigates the performance of using the ILU(0) as preconditioners for three iterative methods: conjugate gradient (CG), biconjugate gradient (BCG), and generalized minimum residual (GMRES).

2.0 Theory

In this paper the conjugate gradient, biconjugate gradient, and general minimum residual iterative methods are used to solve systems of linear equations and are going to be tested for preconditioning effects. The algorithms for CG, BCG, and GMRES are shown in Figures 1, 2, and 3 respectively. The exact applications for and differences between these algorithms is not the concern of this report, however it is worth noting that the CG method only works for symmetric posititive definite matrices.

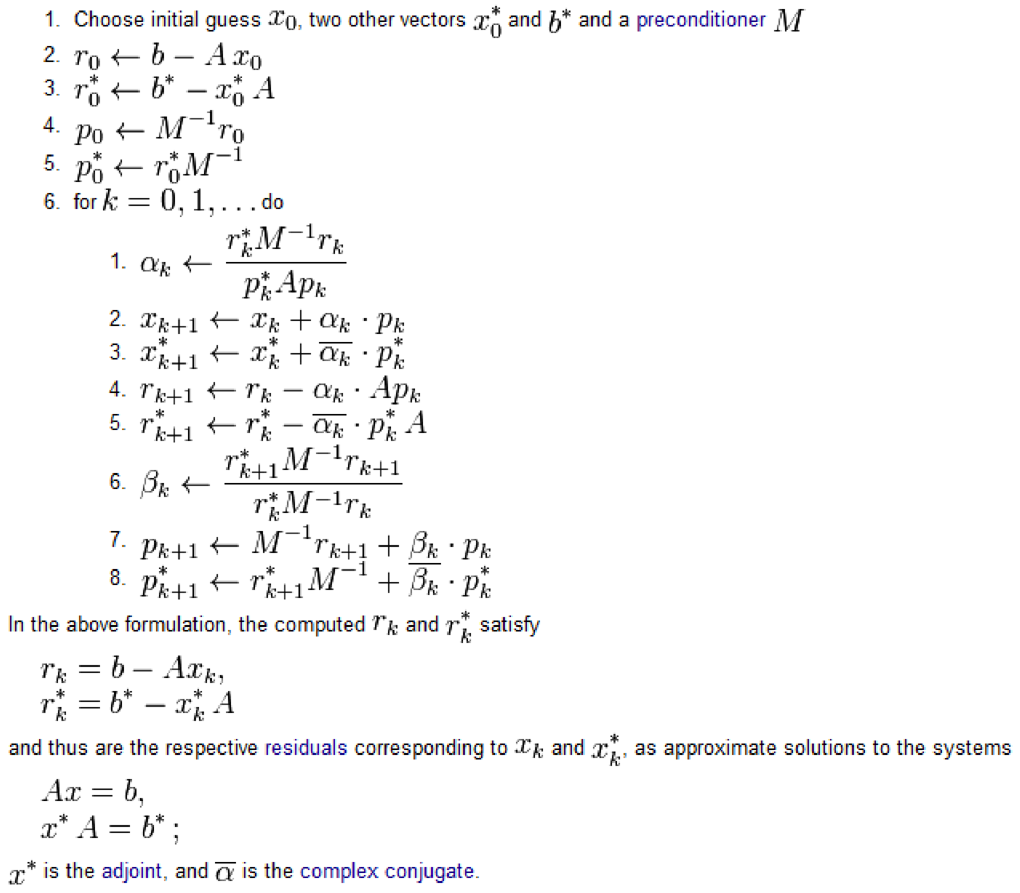
In MATLAB there are three different kinds of incomplete LU decomposition. The first is the crout version, which requires the specification of a drop tolerance value, δ. In the crout version, entries of both the upper and lower triangular matrices are compared to the desired drop tolerance. In U, nonzero entries must satisfy $|U_{i,j}| \geq \delta||A_{*,j}||$. In L, nonzero entries must satisfy $|L_{i,j}| \geq \delta||A_{*,j}/U_{j,j}||$. Any entry that does not meet the requirement will be thrown out, the exception is diagonal entries which are all kept regardless of their value.

The second type of ILU is ILU(0), which is known as “nofill” MATLAB. It does not have a drop tolerance and it is the easiest ILU version to call and use, as long as zero entries in the diagonals are avoided.



```
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$
$\mathbf{p}_0 := \mathbf{z}_0$
$k := 0$
**repeat**
    $\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{z}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$
    $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
    **if** $\mathbf{r}_{k+1}$ is sufficiently small **then** exit loop **end if**
    $\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$
    $\beta_k := \frac{\mathbf{z}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{z}_k^{\mathrm{T}}\mathbf{r}_k}$
    $\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
    $k := k + 1$
**end repeat**
The result is $\mathbf{x}_{k+1}$
```

Figure 1: Conjugate gradient method algorithm [3]



```
1. Choose initial guess $x_0$, two other vectors $x_0^*$ and $b^*$ and a preconditioner $M$
2. $r_0 \leftarrow b - A\,x_0$
3. $r_0^* \leftarrow b^* - x_0^*\,A$
4. $p_0 \leftarrow M^{-1}r_0$
5. $p_0^* \leftarrow r_0^* M^{-1}$
6. for $k = 0, 1, \ldots$ do
    1. $\alpha_k \leftarrow \frac{r_k^* M^{-1} r_k}{p_k^* A p_k}$
    2. $x_{k+1} \leftarrow x_k + \alpha_k \cdot p_k$
    3. $x_{k+1}^* \leftarrow x_k^* + \overline{\alpha_k} \cdot p_k^*$
    4. $r_{k+1} \leftarrow r_k - \alpha_k \cdot A p_k$
    5. $r_{k+1}^* \leftarrow r_k^* - \overline{\alpha_k} \cdot p_k^*\,A$
    6. $\beta_k \leftarrow \frac{r_{k+1}^* M^{-1} r_{k+1}}{r_k^* M^{-1} r_k}$
    7. $p_{k+1} \leftarrow M^{-1} r_{k+1} + \beta_k \cdot p_k$
    8. $p_{k+1}^* \leftarrow r_{k+1}^* M^{-1} + \overline{\beta_k} \cdot p_k^*$
```

In the above formulation, the computed r_k and r_k^* satisfy

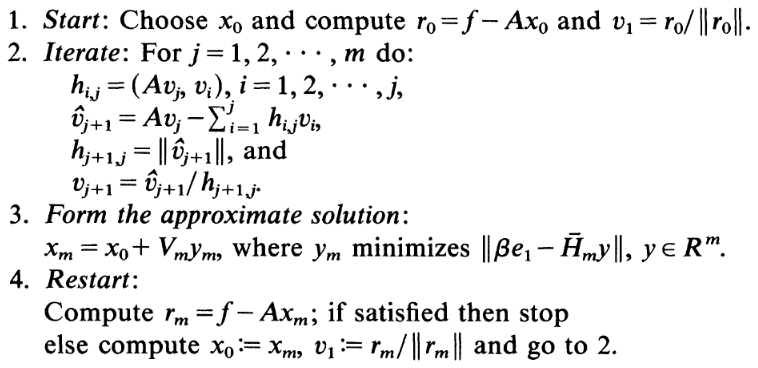
$$r_k = b - Ax_k,$$
$$r_k^* = b^* - x_k^*\,A$$

and thus are the respective residuals corresponding to x_k and x_k^*, as approximate solutions to the systems

$$Ax = b,$$
$$x^*\,A = b^*;$$

x^* is the adjoint, and $\overline{\alpha}$ is the complex conjugate.

Figure 2: Biconjugate gradient method algorithm [4]



```
1. Start: Choose $x_0$ and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. Iterate: For $j = 1, 2, \cdots, m$ do:
      $h_{i,j} = (Av_j, v_i)$, $i = 1, 2, \cdots, j$,
      $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$,
      $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
      $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. Form the approximate solution:
   $x_m = x_0 + V_m y_m$, where $y_m$ minimizes $\|\beta e_1 - \bar{H}_m y\|$, $y \in R^m$.
4. Restart:
   Compute $r_m = f - Ax_m$; if satisfied then stop
   else compute $x_0 := x_m$, $v_1 := r_m/\|r_m\|$ and go to 2.
```

Figure 3: GMRES method algorithm [5]

The last form of ILU includes pivoting and threshold, which is known as "ilutp" in MATLAB. It is not a desired ILU version for preconditioners due to its pivoting. However it is the most accurate version and therefore it is going to be used for comparison purposes in this paper.

3.0 Testing & Results

3.1 ILU speeds on varying size and density matrices

To better understand to workings of the ILU function in MATLAB, some initial examinations of the timings and the various options of the ILU function were examined.

In the first case, the density of the matrix was held fixed at 10^{-5}, where the density is defined as the ratio of nonzero elements to the total number of elements in the matrix. The matrix size was then varied from $n = 1$ to $n = 10^5$. Each matrix was a randomly generated $n \times n$ symmetric positive definite matrix. In Figure 4 it is shown that the number of nonzeros in L and U remain the same as in the original matrix S. Next, the timings for the ILU factorization in Figure 4 were recorded and shown in Figure 5. The time to compute the decomposition is initially minimal, but grows very quickly as it appear quadratic in a log-log plot.

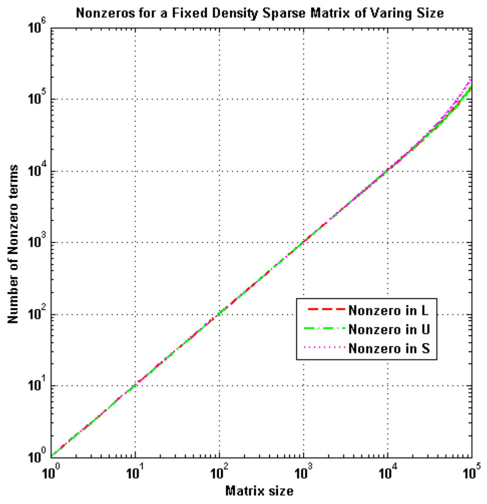


Figure 4: The number of nonzero terms in matrix. S is the original matrix, and L and U are the upper and triangular matrices returned by ILU.

For the next set of tests, the matrix size was held fixed at $n = 1000$ and the density varied from 10^{-5} to 10^{-3}. This range was chosen as it covers the typical range of sparse matrices and even into the density that would no

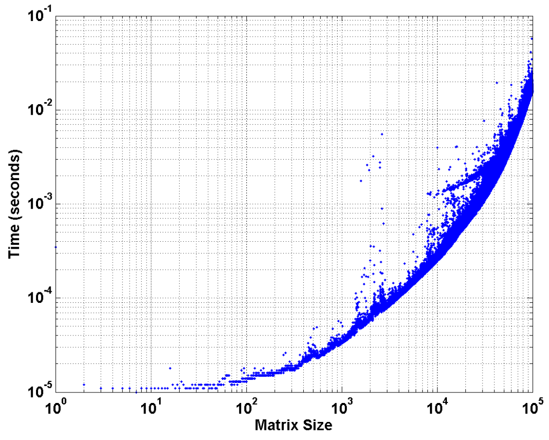


Figure 5: Run times for the default ILU with varying matrix size.

longer be considered sparse. Figure 6 depicts the ratio of nonzeros in $L + U$ to original matrix S. In the case of small densities (10^{-4}), the diagonal values dominate and the ratio approaches 2. As the density increases, the weight of the diagonal becomes less significant and the ratio approaches 1.

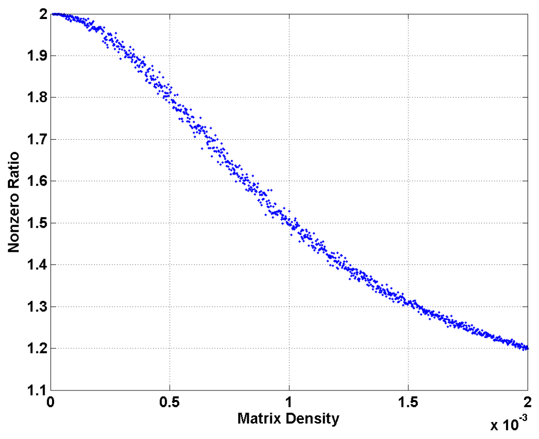


Figure 6: Ratio of number of nonzero entries in L and U to the number of nonzeros in the original matrix S

In the final set of tests, the matrix size is again constant and the density and drop tolerance is varied. The time to compute the decomposition for varied density is shown in Figure 7 and the timing for varied drop tolerance is shown in Figure 8. It is observed that the speed decreases as the drop tolerance increases as shown in Figure 9. As the drop tolerance deceases, the ILU factorization approaches the complete LU factorization.

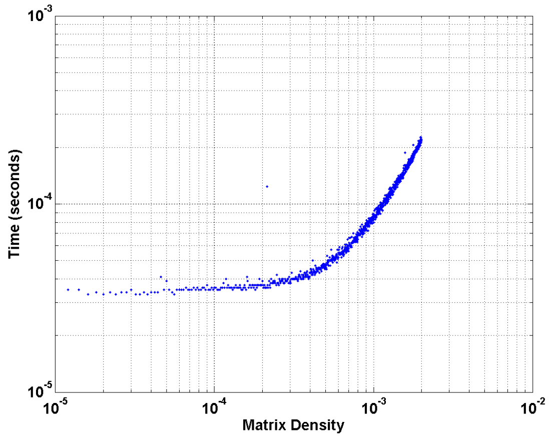


Figure 7: ILU decomposition timing for a 1000×1000 matrix with varying density.

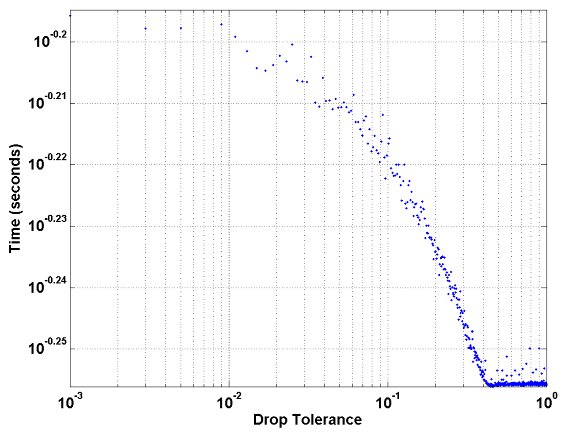


Figure 8: ILU decomposition timing for a 1000×1000 matrix with drop tolerance.

3.2 ILU Residual Testing

The three types of ILU all produce different decompositions. Testing was done by decomposing matrices of varying size and density and then comparing two important residuals. The first residual examined was the relative residual computed using $||S - LU||/||S||$. Figure 9 shows the residuals for each type of ilu for a varying sized symmetric positive definite matrix with density 10^{-2}. The most accurate is the ilutp version followed by crout and then nofill.

The same test was also run on non-symmetric matrices, the results can be seen in Figure 10. The residuals prove to exhibit the same behavior: ilutp is the best, then crout, then no fill.

One more method was used to compare the residuals of ilu, finding the eigenvalues of the resultant matrix of $SU^{-1}L^{-1}$. The resulant matrix should be the identity matrix and its eigen values should be all ones. Figures 11, 12, 13 show a histogram of the eigenvalues plotted on a log scale fpr a sparse matrix with density 10^{-2}. The values should all be near zero in the log scale. Again, ilutp shows the best performance, with crout and no fill not performing as well.

3.3 Preconditioning Results

To test the effect of using L and U as preconditioners, five real world symmetric positive definite matrices of varying size were obtained from matrix market [6]. The solution, x, was set as a vector of ones and used to determine the b vector. Each of the three iterative solvers attempted to find x using no preconditioners and using the ILU solution

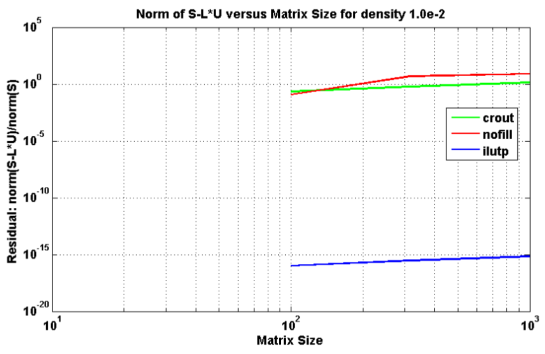


Figure 9: Relative residuals for varying sized symmetric positive definite matrix with density of 10^{-2}.

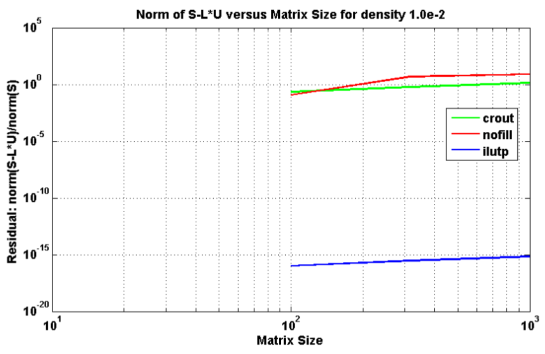


Figure 10: Relative residuals for varying sized non-symmetric matrix with density of 10^{-2}.

as preconditioners, and timings were recorded for each matrix. Results of the timings are shown in Figure 14. For all three solvers, the time to compute was improved with GMRES experiencing the biggest improvement. The time to compute was improved because the solver was able to reach its specified tolerance in less iterations. The number of iterations is shown in Figure 15.

MATLAB only has one option that can be changed which is milu, or modified incomplete LU decomposition. Specifying either row or column for milu compensates the diagonal of U in such a way that the row or column sum is conserved. Further testing was done to examine how changing milu would affect the computation. Initial results showed the using either the row or column option greatly improved the results of the preconditioned computation. However, further testing showed that by using an x vector with all ones could be a special case that allowed for the computation to be very easy. Another test resulted in decreased performance from the solvers. Therefore, at this time the effect of using the modified LU decomposition is unknown.

4.0 Conclusions & Future Work

We have found that using ILU(0) as a preconditioner for an iterative solver is a very effective strategy for producing faster results to a set of linear equations. Other versions of the incomplete LU decomposition do produce better results for the decomposition, but they will not work as preconditioners for all matrices. ILU(0) will work for all matrices and will compute fast for even very large systems which makes it ideal for real world large problems.

For the purpose in this report, only symmetric positive definite matrices were fully studied the conjugate gradient method only works for that type. A continuation of this work would include a study of the matrices that are not positive definite or symmetric. In order to solve these systems, the biconjugate gradient stabilized method will need to be used. An additional point of interest is to also study the hyperpower method to perhaps compute an even better preconditioner.

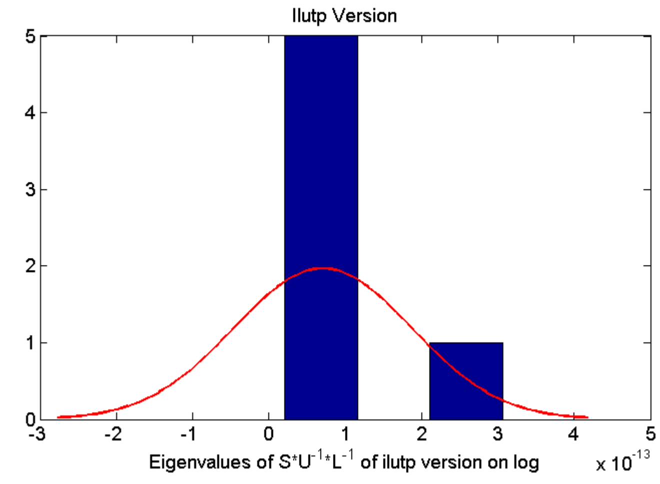


Figure 11: Eigenvalues of $SU^{-1}L^{-1}$ on a log scale for ilutp.

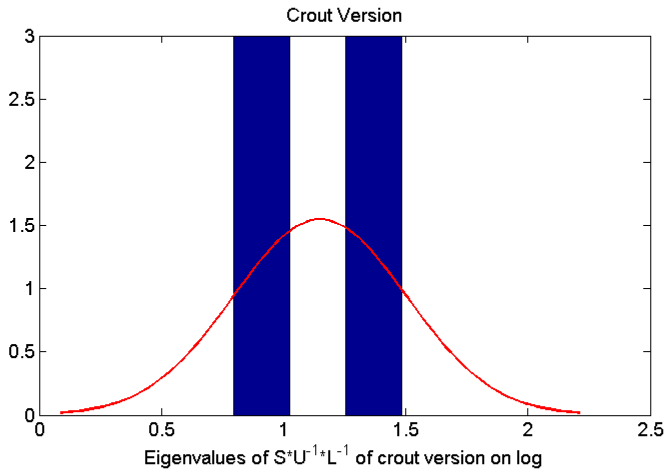


Figure 12: Eigenvalues of $SU^{-1}L^{-1}$ on a log scale for crout.

References

[1] "Preconditioner." Wikipedia. Wikimedia Foundation, 04 Dec. 2012. Web. 22 Apr. 2012.
http://en.wikipedia.org/wiki/Preconditioner.
[2] Watkins, David S. Fundamentals of Matrix Computations. Hoboken, NJ: Wiley, 2010. Print.
[3] "Conjugate Gradient Method." Wikipedia. Wikimedia Foundation, 18 Apr. 2012. Web. 22 Apr. 2012.
http://en.wikipedia.org/wiki/Conjugate_gradient_method.
[4] "Biconjugate Gradient Stabilized Method." Wikipedia. Wikimedia Foundation, 04 Dec. 2012. Web. 22 Apr. 2012.
http://en.wikipedia.org/wiki/Biconjugate_gradient_stabilized_method.
[5] Saad, Youcef, and Martin H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Non-symmetric Linear Systems." SIAM Publications Online. SIAM J. on Scientific Computing, 29 Nov. 1983. Web. 22 Apr. 2012.
http://epubs.siam.org/sisc/resource/1/sjoce3/v7/i3/p856_s1.

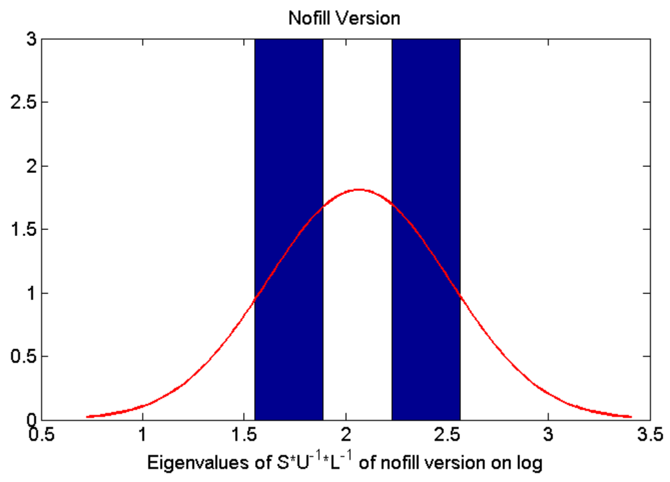


Figure 13: Eigenvalues of $SU^{-1}L^{-1}$ on a log scale for nofill.

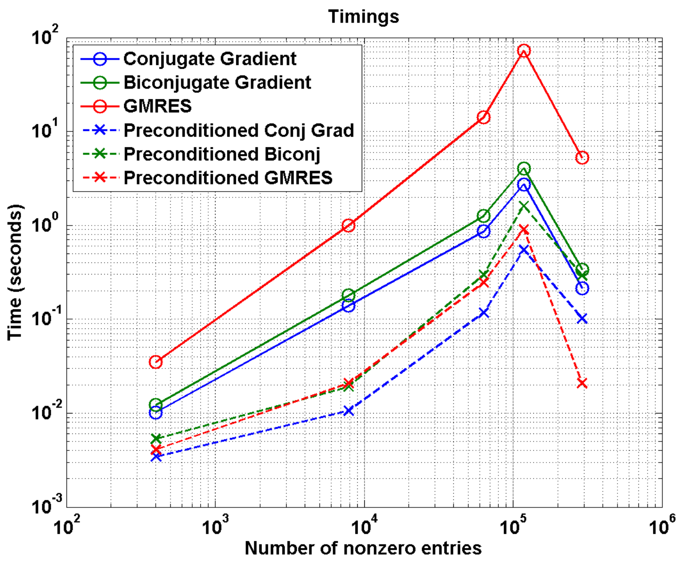


Figure 14: Log-log plot of timing of the iterative solvers with and without preconditioning.

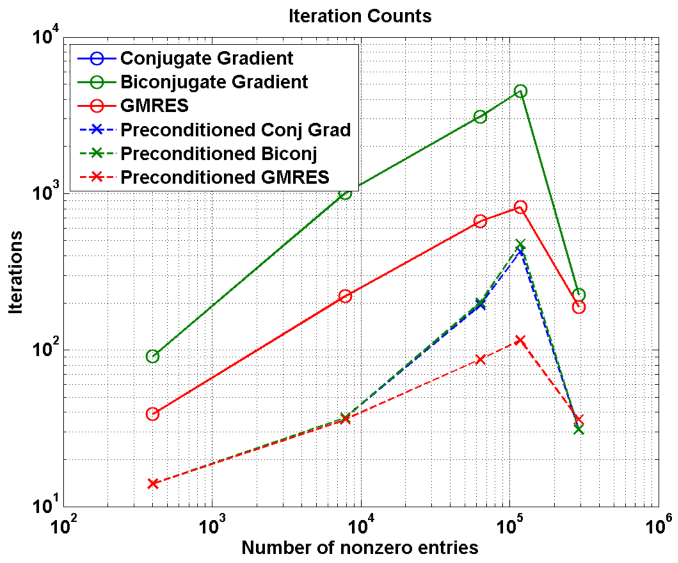


Figure 15: Log-log plot of the iteration counts for the iterative solvers with and witout preconditioning.