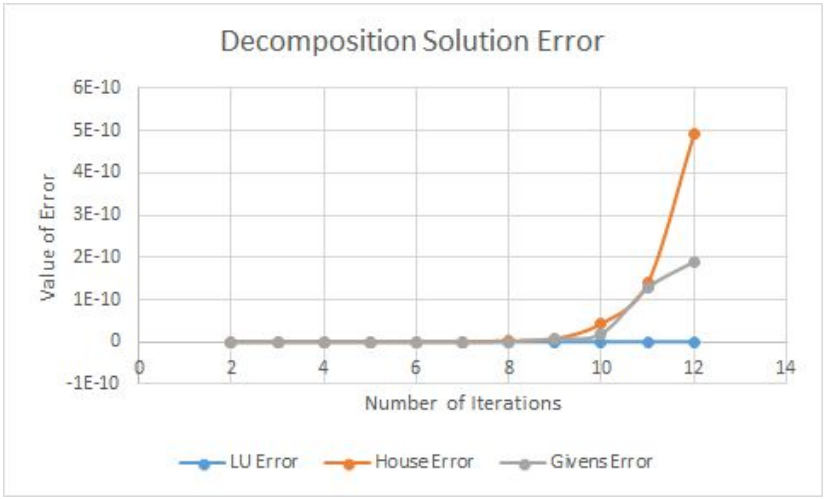
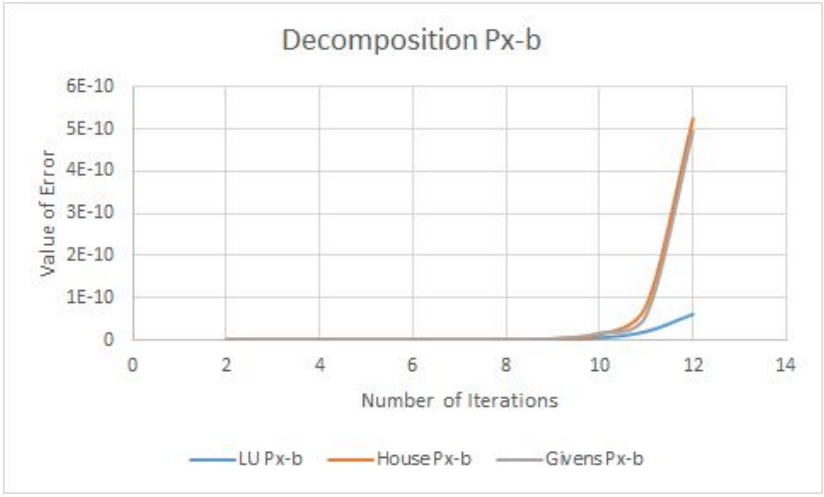
**MATH 2605**
**Project Discussion**
Samuel Isang
Bruck Woldeyes
Lily Lau

## Part 1: The Symmetric Pascal Matrix




This a graph of the errors calculated for the Px-b for the pascal matrix for the LU factorization QR factorization Householder and QR factorization Givens. For the decomposition of Px-b, it is evident from the graph that errors do not begin until at about n = 10 iterations. The errors are not very large, as seen for their exponential value. The method with the largest error is the Householder method, followed closely by the Givens method while LU factorization has the least amount of error.

**a) Why is it justified to use the LU or QR–factorizations as opposed of calculating an inverse matrix?**




The idea behind implementing implementing Ax = b using forward and backward substitution is simply ease of implementation and general reduction in overall error. Due to the fact that calculating the inverse of matrix would require many multiplication and division operations, the final error due to the precision of the floating points would be higher than using forward and backward algebraic expressions to which are proportional to the size of the matrix.

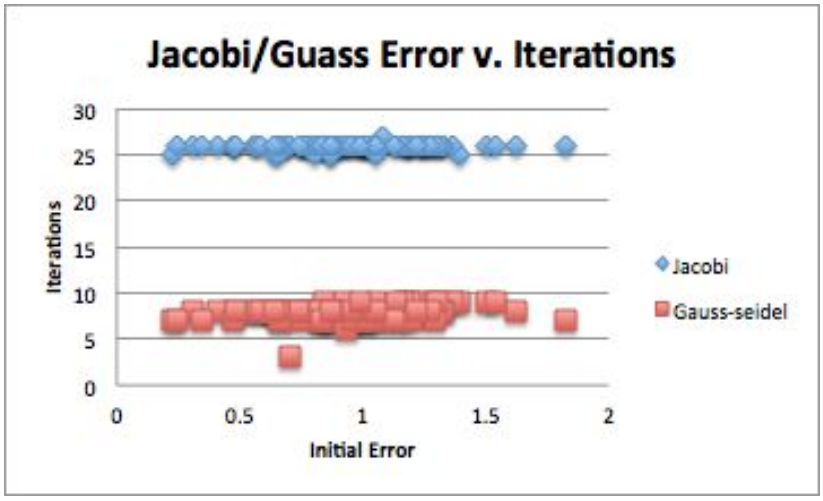
**b) What is the benefit of using LU or QR−factorizations in this way?**

From the graph using LU factorization and QR decomposition does not give a lot of error until the number of iterations significantly increases. There is no change in the LU Px-b or House Holder Px-b which shows that these factorizations can be used effectively when they are scaled to higher n values. The increase in error for Px-b was no larger than 5E^-12 this was due to a definite decrease in runtime. The overall benefit however for using LU and QR factorization this way is because of their conditioning property (simply evaluated by the matrix condition number). Conditioning property of a particular matrix is what determines if it makes sense for an algorithm to provide a numerical solution to a linear system involving that matrix. Since linear systems on computers have errors, the goal is to minimize the conditioning number as much as possible. LU factorization and QR decomposition help us meet those goals while giving a good enough speed during computation and calculation.

## Part 2: Convergence of the Iterative Methods

**a) We randomly generated 100 3x1 initial vectors x. Discuss the effect of different initial vector positions.**

When doing iterative methods, Jacobi and gauss-seidel, the accuracy of the initial guess vector is important into determining how many iterations the code will have to make to converge. The more accurate your initial guess is to the exact solution the less iterations it will have to make to converge.




**b) Discuss the steps needed to carry out both Jacobi and Gauss Seidel methods.**

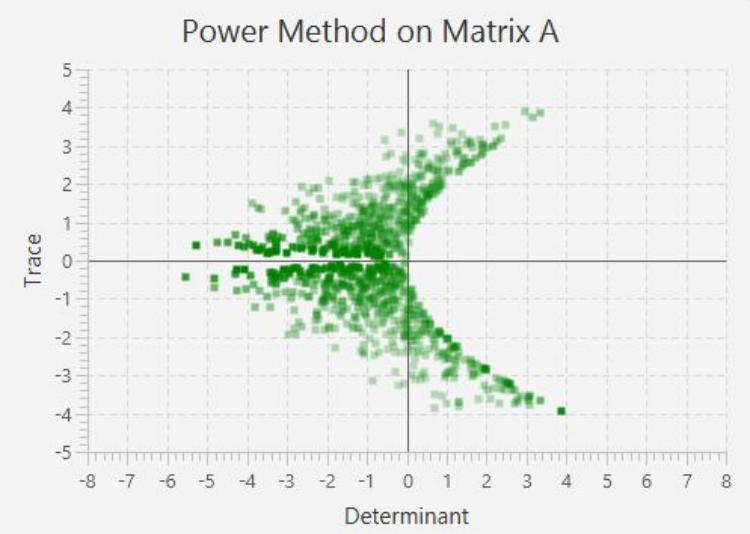
When we randomly generated 100 different initial vectors with values between -1 and 1 and calculated the Jacobi and Gauss-seidel for each one, the average number of Jacobi iterations to Gauss-seidel iterations had a ratio of about 3.3. Typically it took the jacobi method around 26 iterations each time to converge and reach a solution to a certain precision, while it took only 7 to 10 iterations for the gauss-seidel to converge. This means that the gauss-seidel is a much quicker method into converging and finding a solution Xn given an initial guess vector. The probable reasons for this is that gauss seidel takes

advantage of calculating the eigenvalues of a triangular matrix rather than the diagonal like jacobi. Also  gauss seidel can update values simultaneously while Jacobi has to recalculate.

**c) Interpret the results displayed on the graph.**

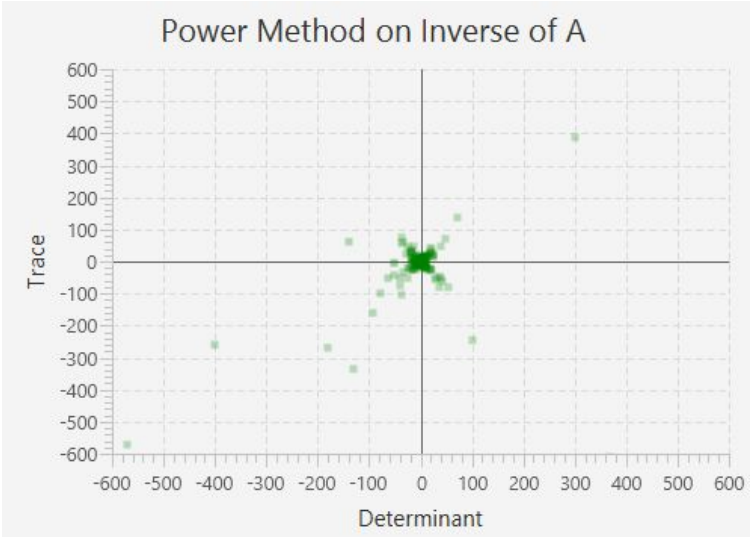
Looking at the graph of the initial error vs. number of iterations for Jacobi and gauss-seidel, it shows that the rate of convergence of the gauss-seidel is much faster than Jacobi. For gauss-seidel the number of iterations ranged from 7 to 10 iterations based on the random initial vector, and the number of iterations consistently stayed around 26 for Jacobi. This shows the ratio calculated in part b, that the number of iterations is on average 3.3 times more using the Jacobi method over the gauss-seidel. Also based on the initial error, it showed that generally, the higher the initial error is the more iterations it took for each method to converge.

## Part 3: Convergence of the Power Method




**a) Discuss the general shape of the scatterplots and how the two plots are related.**

Our graphs depict the relationship between the determinant and trace of a matrix after it has been run by the Power Method. The first graph shows the data gathered by 1000 matrices with entries randomly generated from [-2, 2] while the second graph produces the results on the inverse of those matrices. As you can see, the graphs produce very different results. The graph of Matrices A take on an almost parabolic shape towards the right end and then tapers out towards the x axis as the determinant becomes more negative. As for the graph of the Inverses of A, the points seem to be condenses in a small ball around the origin. From there, some points taper off in extremely far from the origin (ex. point (-600, -600) and point (400, 400)).




The reason for these differences have to do with the fact that the trace of Inverse

A equals to the trace of A divided by the determinant of A **[tr(A-) = tr(A)/det(A)]** and that the determinant of Inverse A is equals to **[1 / det(A)]**. Many points on the graph of A have distant x and y points. For example, the far left point appears to be close to (-5.5, 0.5), meaning its determinant is -5.5 and its trace is 0.5. By the formula mentioned above, you can see that the trace of the inverse of the point will be a very small number (.5 / 5.5) as will its inverse (1 / 5.5). Most points on graph A have the same relation as the one just illustrated, so therefore it makes sense that the points on the graph of Inverse A tend to cluster towards the origin.

**b) The relationship between the position of a matrix on the plot and the number of iterations needed in the power method.**

The opacity of the points represents the number of iterations undergone by the Power Method before the matrix converged. The more opaque points (darker green, opacity close to 1.0) tells us that the matrix was close to the max number of iterations required to converge while the less opaque points converged relatively quickly (lighter green, opacity closer to 0.0). It appears that on graph A, the matrices that converged very slowly (dark green points) tend to have varying determinant values and trace values VERY close to zero (both positive and negative. Towards the edges of the graph, the matrices converge quickly and have either very large or very small traces. The graph on Inverse A also agrees with this trend. Towards the origin (most of the points), the matrices converged slowly while the matrices converged quickly when the trace was very large or very small. So, it appears that when the ratio **|| lambda(1)/lambda(2) ||** is very large, the matrices converge more quickly than when the ratio is small.