Gaussian Elimination Precision Across Different Techniques

Jacob Margolis and Gabe Werman

Objective

The goal of this project is to determine the precision of Gaussian Elimination with and without pivoting.

Hypothesis

Since the two algorithms are so closely related, there is no reason to think that the two algorithms will ever differ in solutions.

Gaussian Elimination can be used to solve a system of equations. Given Ax = B, or:

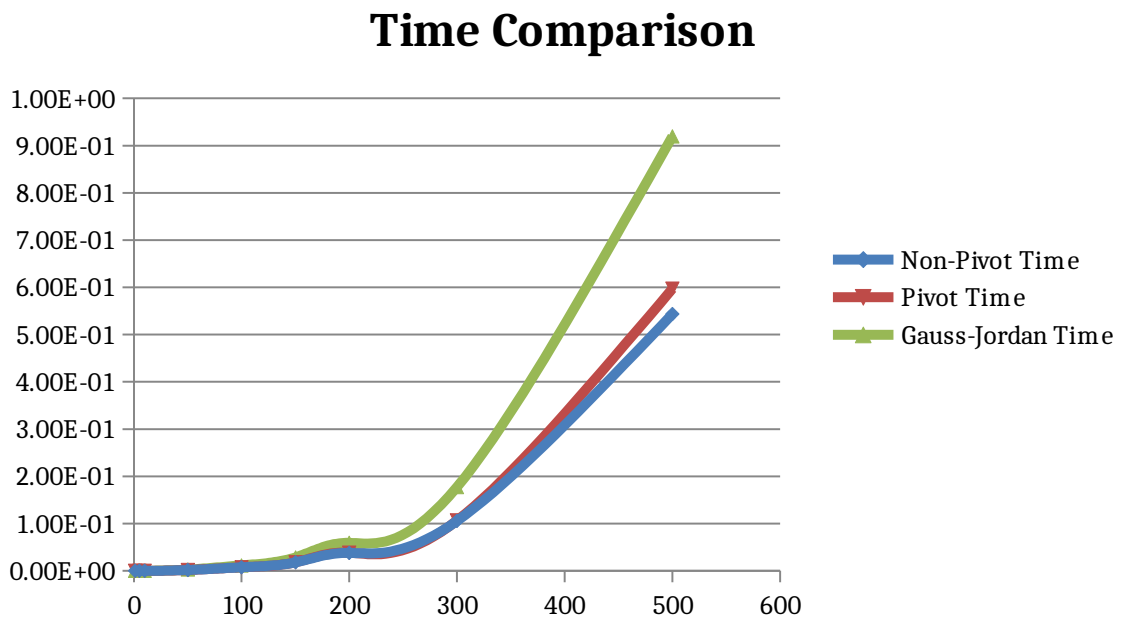
$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\
\ldots \qquad\qquad\qquad\qquad &\quad \ldots \\
\ldots \qquad\qquad\qquad\qquad &\quad \ldots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n
\end{aligned}
$$

we can solve x_i by adding and subtracting rows from each other. Gaussian Elimination subtract the right amount of the top row from each row as to eliminate the variable x_1 from the remainder of equations. It then moves on to the next row, and repeats the process. This will reduce A into an upper triangular matrix. Then the last equation will be solvable, and can be substituted into the other equations. This process is then repeated until every variable has a value. The only time this will run into problems is when an a_{ii} is zero. If this is the case, we can swap rows and be able to continue the process. The algorithm without the ability to pivot rows is Gaussian Elimination without pivoting, the other with pivoting.

We now research the question of “is one algorithm more precise than the other?” If the answer is yes, then hopefully we will be able to tease out a distinction between two answers in a certain case to demonstrate that one algorithm is more precise.

Runtime

This analysis actually would not be able to show that pivoting is more or less precise, but is telling nonetheless. Running pivot and non-pivot on various matrix sizes yielded the following:



These results show that not pivoting is quickest, then pivoting, and lastly Gauss-Jordan. Gauss-Jordan elimination is yet another technique we implemented to solve Ax + B. Gauss-Jordan reduces the matrix A to upper triangular just like the previous two, but instead of substituting, reduces the columns upward starting at the bottom, until A looks like the identity matrix, and B will be the solution vector.

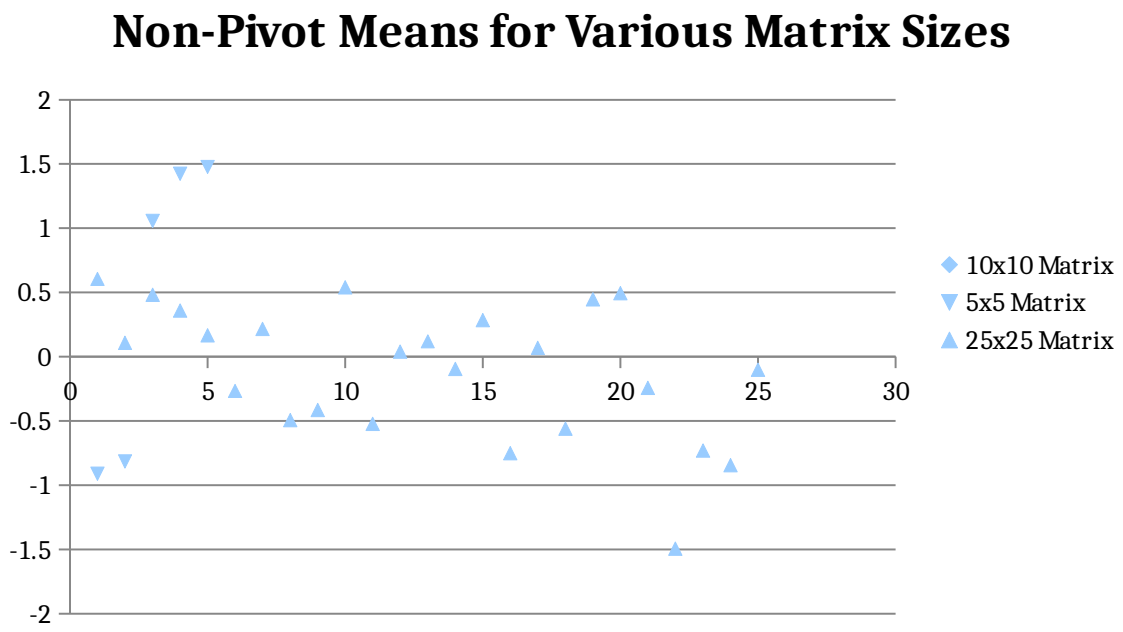
The order of speed makes sense, because the elimination with pivot is the same as non-pivot, but with an added algorithm in the case of zeros. The extra time appears to be insignificant, perhaps only increasing proportional to n. Gauss-Jordan is slower than the others, but has the advantage of being easier to check by hand.

It can also be seen through inspection that these curves are proportional to n^3. Further, the code starts at one row, subtracts from each of the remaining rows. That itself is O(n^2), but each row operation itself has n elements, so each row operation takes O(n) time. Combined, Gaussian Elimination should take O(n^3) time, and that is confirmed in our data.

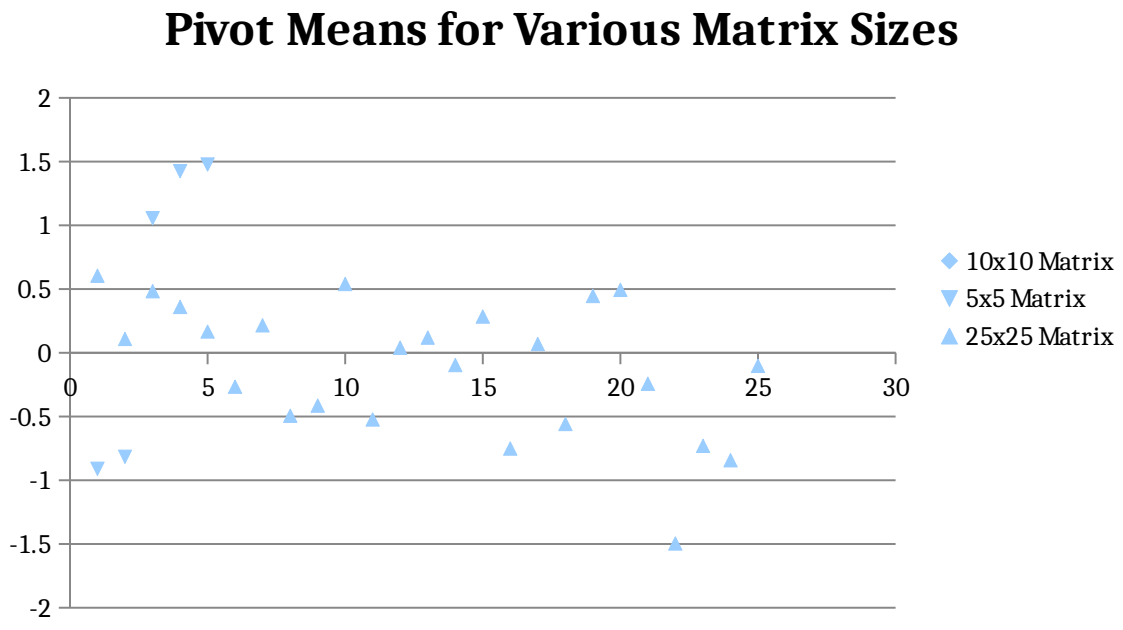
Descriptive Analytics

Upon testing individual cases, the pivot and non-pivot algorithms consistently gave exactly the same result. A more efficient way to check for discrepancies is to examine the mean and standard deviation over many tests, (100, unless otherwise specified). For the most part, this is the tactic used to compare the

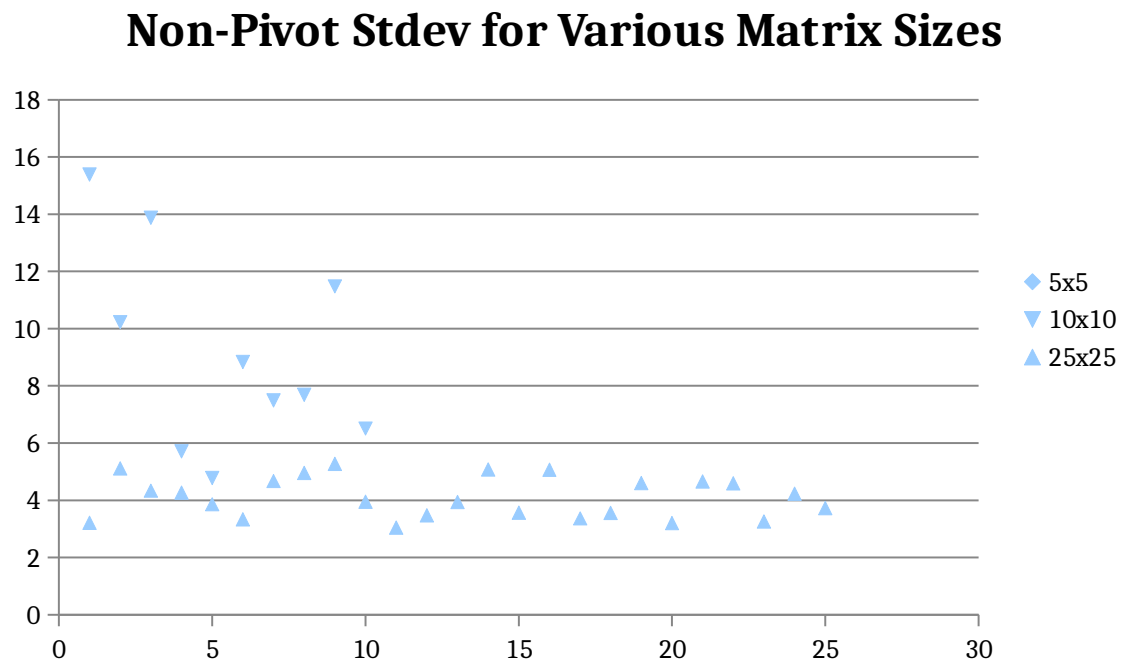
results of the algorithms. For instance, we can quickly check to see that pivot and no pivot run the same for matrices of different size:

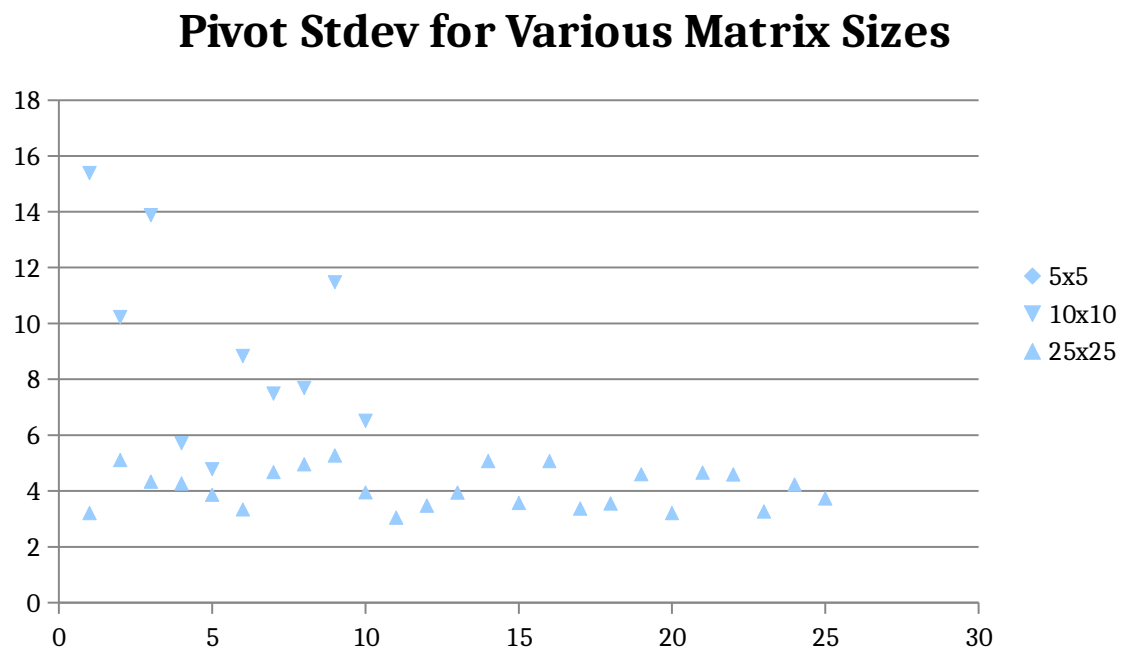


Is exactly the same as:



And the standard deviations:





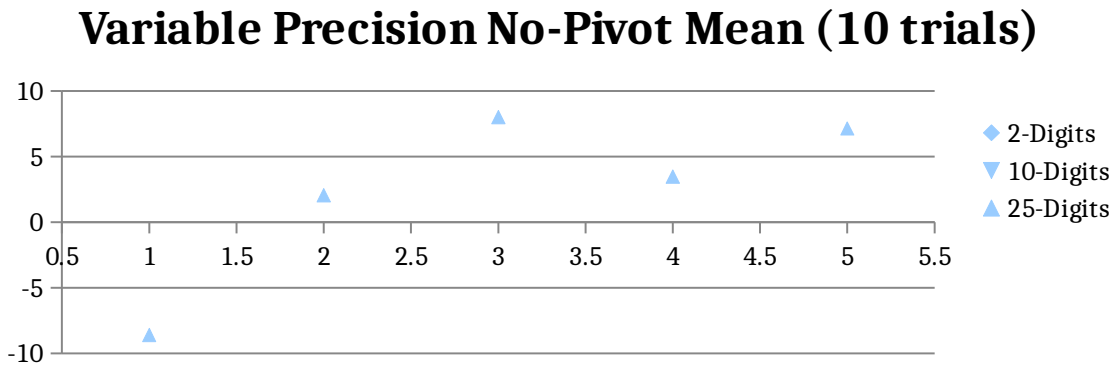
And yet again, we have perfect equality between pivot and non-pivot. More can be learned from these descriptive tools than just equality. For example, it is apparent that the standard deviation of solutions decreases as the size of the matrix (n) increases. The graphs depicting mean also show the means of larger matrices

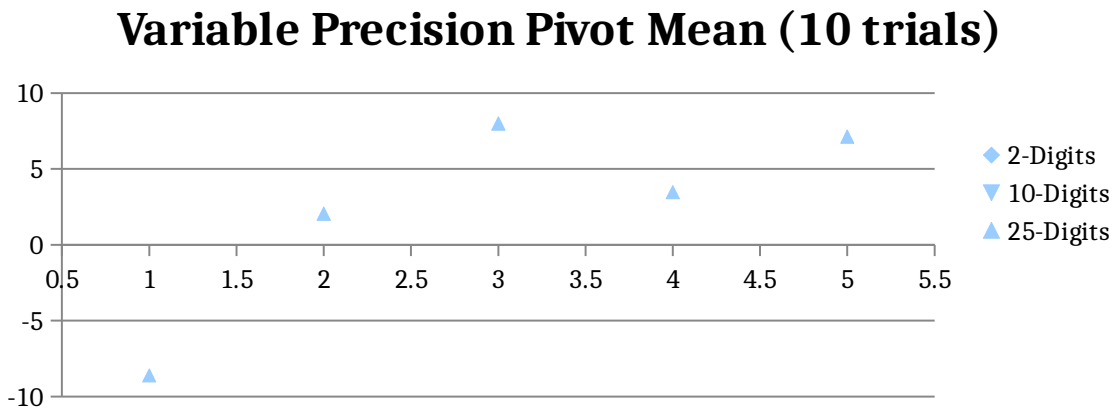
generally being closer to zero. Perhaps this phenomenon can be explained in the following way: With more variables in each equation, it is less likely for any one variable to be so important in satisfying the equation, because there are so many other variables that will have similar sign and magnitude coefficients. This being the case, each variable is more likely to contribute less, meaning that their values will be smaller. This would explain both the mean and standard deviations. But now back to the original question: distinction in the precision of each algorithm.

Precision

One way in which we can try to get a distinction between algorithms is to see that they show the same answer for various levels of precision inputs. The reason this might be able to attain different results is from rounding error. If one algorithm magnifies the error more than the other, a distinction will be had.

First we made sure that we had each algorithm work for single precision, double precision, and variable precision, (for which we used values of 2-digits, 10-digits, and 25-digits). The variable precision used a lot of built in language, and took a while to complete, so it was mostly tested on 5x5 matrices. Attached in the excel file are lists of results that perfectly match results from pivot and non-pivot, so I will only show a couple here. For instance, the mean in variable precision matches perfectly:

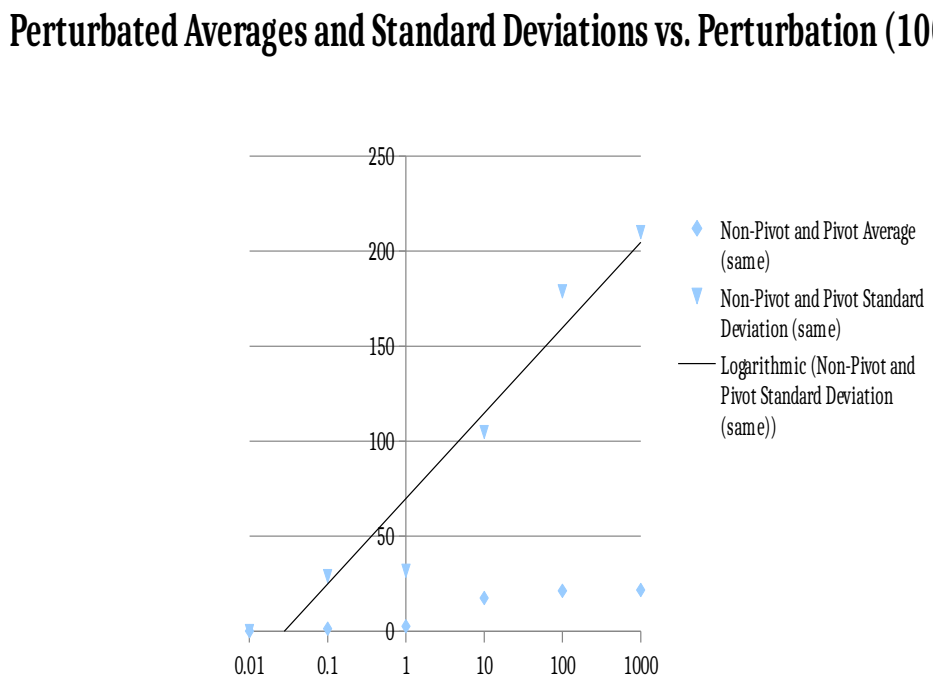




Alas, across all precisions, even with changing the size of the matrix, the data suggests that there is no difference in the pivot and no pivot results.

Perturbations

Perhaps the reason that pivot and no pivot seem the same is that we have not been checking minute values. The goal of perturbation therefore is to create a very similar matrix, and subtract the results of the very similar matrix to that of the original. By checking the distance between the answers, we might find a difference between pivot and no pivot. The data, below and attached in the excel spreadsheet, shows no such difference:



The data was exactly the same, so we started plotting it just once, despite it representing two cases. This graph represents just one of many attempts, which included using different matrix sizes, different perturbation factors, and different precisions. As per usual, no distinction was to be had.

Ill-Conditioned Systems

The last permutation of tests we will run to find a difference in precision of pivot and non-pivot is on Ill-Conditioned Matrices, defined in the assignment. The basis of what makes these matrices interesting is that they are very similar to underdetermined matrices.

Consider the following system of equations:

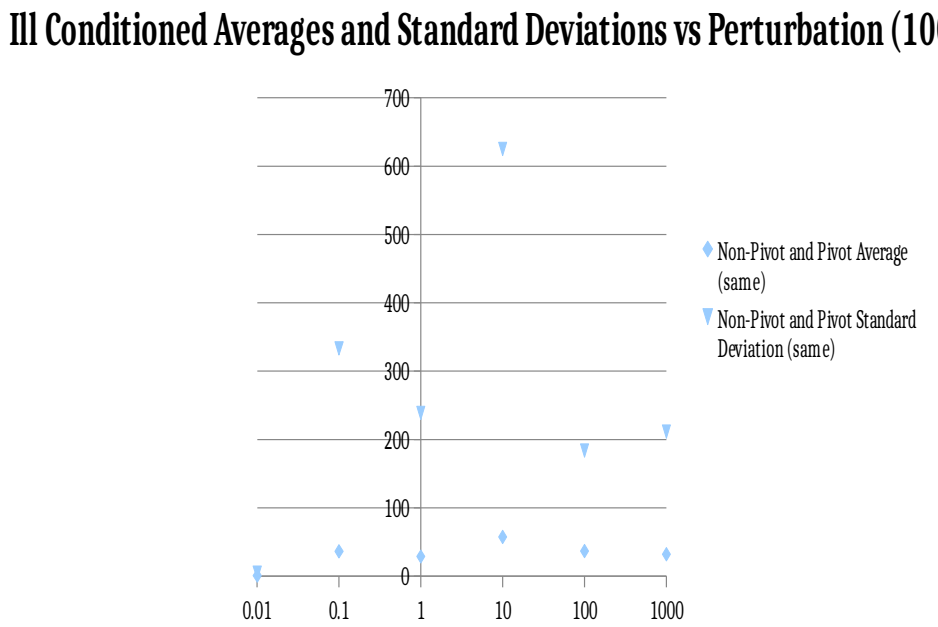
$x + y = 2$

$2x + 2y = 4$

This is a square matrix (same amount of variable as equations), but it is impossible to solve for x and y. In general, not all variable can be solved for if one

row is a linear combination of other rows. Without any perturbation, the Ill-Conditioned Matrix would run into this 'underdetermined' issue. Gauss with pivoting and Gauss-Jordan can detect this, and return the error. No-pivot treats this type of matrix as having a zero pivot error, which is inevitable in these types of matrices.

Regardless, the perturbation will allow the algorithms to complete, now only different than a regular matrix in the sense that they are close to being underdetermined. At this point unsurprisingly, the data suggests no distinction between pivot and no-pivot.



Again, the values are the same so it is combined into one graph. This graph represents one of many test cases, changes like matrix size is recorded in the attached Excel spreadsheet.

Conclusion

As expected, there is nothing discernible in the precision of pivot and no pivot. This supports the hypothesis that the algorithms function in the same manner. When examining the differences in the code, it further supports the hypothesis. The only difference in the codes is that pivot has more functionality when a diagonal value is equal to zero- something that never occurs in testing! Even if it did occur, no pivot would just error, so nothing useful can be learned from comparing no-pivot to pivot. Gaussian Elimination with and without pivoting have the same precision.

(If any data referenced in this write-up is not shown from the graphs, it is available in the attached spreadsheet)

Extra Credit

-Implementing Gauss-Jordan elimination, and running a couple tests to see how its precision compares. (the same)

-Single and Double precision testing for comparisons of pivoting and non-pivoting when using descriptive analytics (mean and stdev)

-Variable precision for each case:

1. "Go the whole shot and use variable precision arithmetic, e.g. ***vpa(), digits()...***. This is unexplored territory for CB as of 2012: definitely Extra Credit, definitely a way to FORCE precision-related effect to the surface."

-Varying matrix sizes on perturbation and ill-conditioned tests (CB also said was extra credit) as well as on all other functions tested.

-Varying the perturbation factor for perturbation and ill-conditioned tests.