Dan Scarafoni

Matlab

CSC173

Gaussian Elimination in Matlab

Introduction This project involved utilizing Matlab to create a an algorithm for Gaussian Elimination, a formula used to solve matricies. Matricies are data structures with diverse applications and uses in mathematics, physics, computer science, and many other fields. Many different problems can be represented with matrices. For example, multivariate equations can be represented and solved with matricies. For instance, one way of solving the following set of equations

$$Ax + By + Cz = Q$$
$$Dx + Ey + Fz = R$$
$$Gx + Hy + Iz = S$$

would involve placing the left hand size of the three equations into a 3x3 matrix of the coefficients of x, y, and z, and placing the right hand solutions of the equations into its own 3x1 matrix like so.

$$\begin{bmatrix} ABC \\ DEF \\ GHI \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} Q \\ R \\ S \end{bmatrix}$$

Using Gaussian Elimination, the coefficient matrix can be reduced to the identity matrix, while converting the right hand solution matrix to different values

$$\begin{bmatrix} 100 \\ 010 \\ 001 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} M \\ N \\ O \end{bmatrix}$$

thus,

$$x = M$$
$$y = O$$
$$z = P$$

this means of solving all systems for which a unique solution exists. There are nearly endless applications to Gaussian Elimination and matrices in their use for solving mathematical lemmas,

however, because this project only focused on their application with regards to equation systems with unique solutions, those will be the only ones discussed. For more information about matrices and linear algebra, please see *Differential Equations and Linear Algebra*, third edition, by Spehen Goode and Scott Annin.

Gaussian Elimination algorithm

Two primary algorithms were implemented for this project- standard Gaussian Elimination, and Gaussian Elimination with partial pivoting. The two are very similar in terms formulaic steps, and produce identical results.

Standard Gaussian Elimination begins with an system of equations translated into a coefficient matrix and solution matrix, of the form

$$Ax = B$$

where

A- coefficient matrix

X- variable matrix

B- solution matrix

take, for example, the following system of equations seen previously in the introduction.

$$\begin{aligned} Ax + By + Cz &= Q \\ Dx + Ey + Fz &= R \\ Gx + Hy + Iz &= S \end{aligned}$$

which can be represented as

$$\begin{bmatrix} ABC \\ DEF \\ GHI \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} Q \\ R \\ S \end{bmatrix}$$

Note that in the original three equations, any change done algebraically to one side of the equation must be done to the other. Thus, since any elementary row operation we perform one will will have to be done on the other, the two can be combined into an augmented matrix as seen below

$$\begin{bmatrix} ABCQ \\ DEFR \\ GHIS \end{bmatrix}$$

with the augmented matrix, we now begin the first of two parts of solving the system, called “foward elimination.” Start by selection a “pivot point”. This is the nonzero entry that is furthest left and up in the matrix. In this case, it is “A”. We then divide A's row by A, reducing A to one.

$$\begin{bmatrix} 1 \frac{B}{A} \frac{C}{A} \frac{Q}{A} \\ DEFR \\ GHIS \end{bmatrix}$$

We reduce all the entries in the pivot point's column to 0 by multiplying the pivot point's first entry by the number, and adding it to the row.

$$\begin{bmatrix} 1 & \frac{B}{A} & \frac{C}{A} & \frac{Q}{A} \\ 0 & E-D & F-D & R-D \\ 0 & H-G & I-G & S-G \end{bmatrix}$$

Having successfully reduced the first column, we repeat the process by selecting a pivot point from the the submatrix made by all entries down and to the right of the old pivot points row and column. The process is repeated until the lower right half of the coefficient matrix is reduced to zeros, and the diagonal is reduced to ones, as such.

$$\begin{bmatrix} 1 & \frac{B}{A} & \frac{C}{A} & \frac{Q}{A} \\ 0 & 1 & j & k \\ 0 & 0 & 1 & S-G \end{bmatrix}$$

This finishes the last step of forward elimination. The next step is backwards substitution. This step is very similar to forward elimination, in that it attempts to convert the top and right half of the coefficient matrix into zeros, thus resulting in the identity matrix. We begin by selecting the topmost row's first entry to the right of the leading one, and multiplying the row below it by a certain constant and adding it to this row such that this number will be eliminated. We then repeat the process for the rows below the first selected, so that every element in the top row of the coefficient matrix is converted into a 0. We

repeat this for all rows. The end result is display below.

$$\begin{bmatrix} 1 & 0 & 0 & T \\ 0 & 1 & 0 & U \\ 0 & 0 & 1 & V \end{bmatrix}$$

This gives us the solution

$$x = T$$
$$y = U$$
$$z = V$$

The second algorithm implemented was partial pivoting. This algorithm was almost identical to the standard one, with one difference. In the event that the number 0 was encountered as a pivot point, it would swap that row with another row. This is because it would be impossible to reduce 0 to 1 via elementary row operations, the standard program would return an error. The pivoting added extra stability to the solution. It proved, however, to be mostly unneeded do to the extremely low probability of generating a zero.

Programming

Using Matlab functions, a series of programs were created for testing the various parameters and perturbations recommended by professor Brown. The following were tested

1. Average time of execution of 10 trials (both with and without pivoting)

2. the mean variable values of a three variable system (both with and without pivoting)

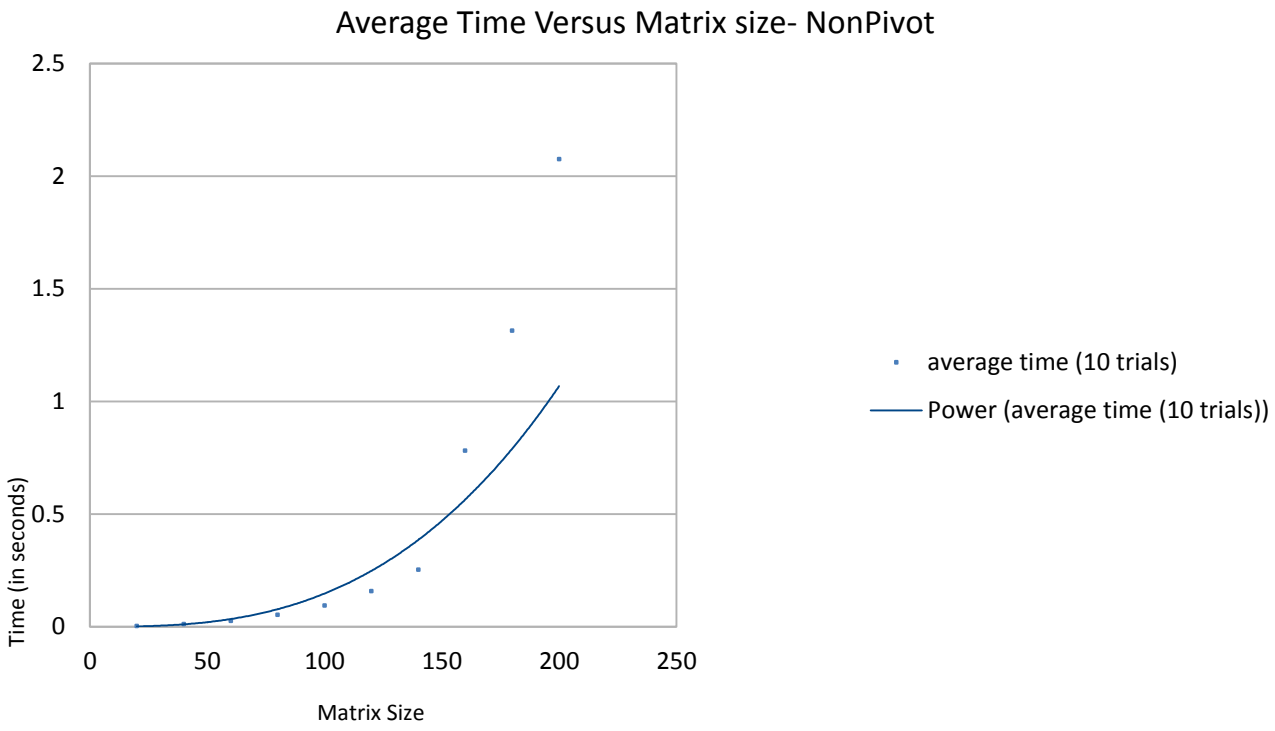
3. the mean variable value in relation to number of variables in the system (both with and without pivoting)

4. the mean variable value in relation to the number of trials run (both with and without pivoting)

5. the mean difference between the answers of a matrix and the same matrix perturbed by certain quantities

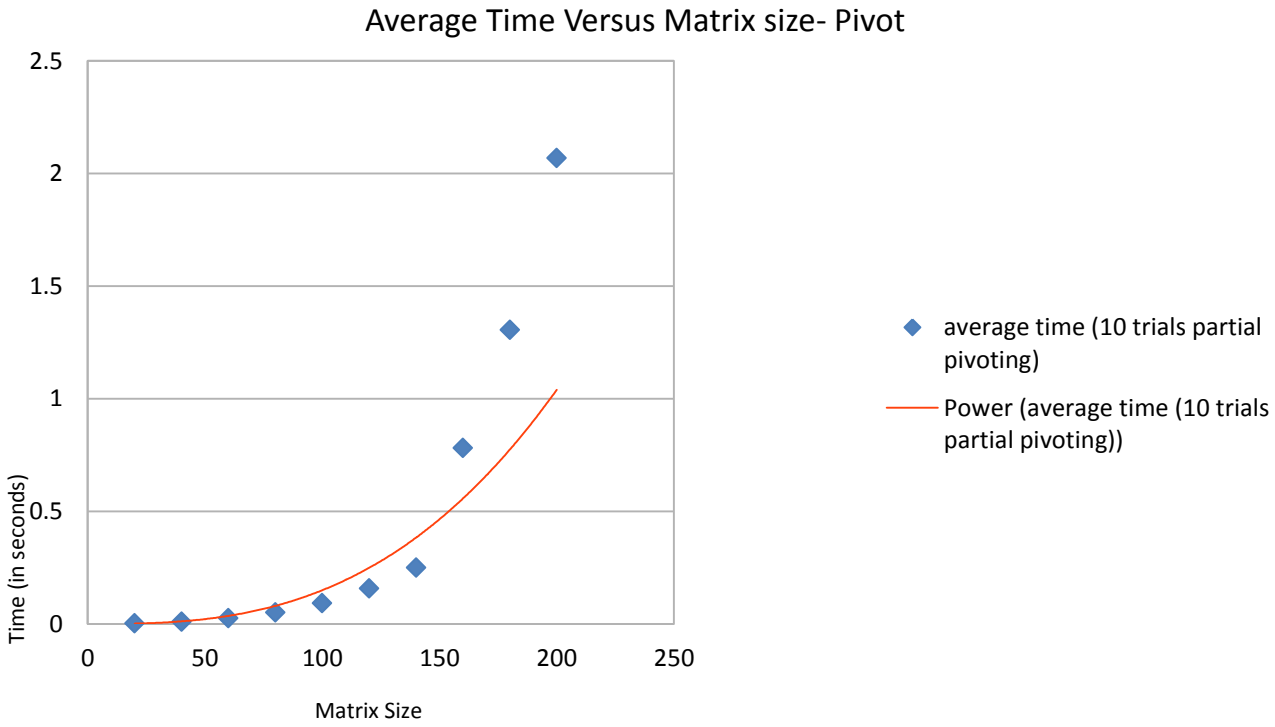
Results

First, the Gaussian Elimination program was run on matrices of sizes 20-.200, and the time it took was measured and recorded. The following graph represents the data. Please note that the times shown are averages for ten runs of each matrix size

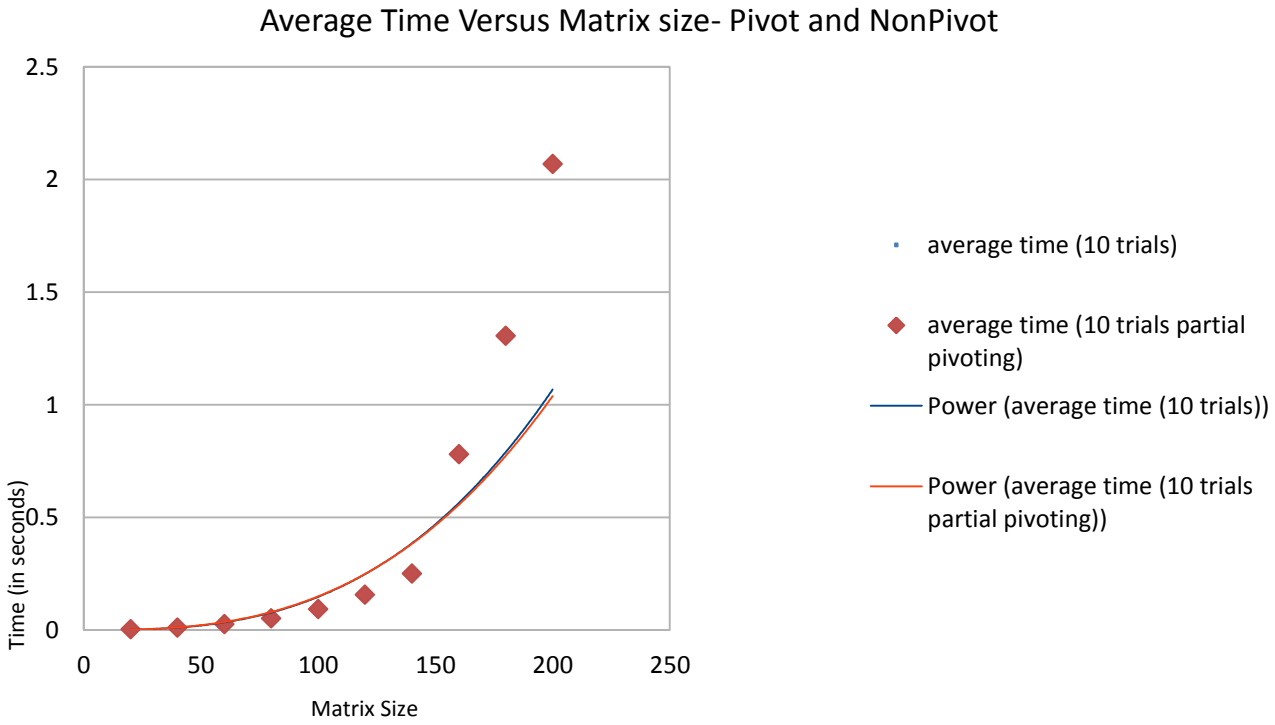


As is shown, this algorithm does seem to run in polynomial time degree three. It certainly does run in bigO(x^3). The R^2 correlation if .95 indicates that this equation accurately depicts the behavior of this function.

The next set of experiments was nearly identical to the first, however, the partial pivoting algorithm was used instead of the non-pivoting one. The results are in the following graph.



Again, this data suggests that the algorithm runs in bigO(n^3). When the two are compared side-by-side, the differences (or lack thereof) become more obvious.

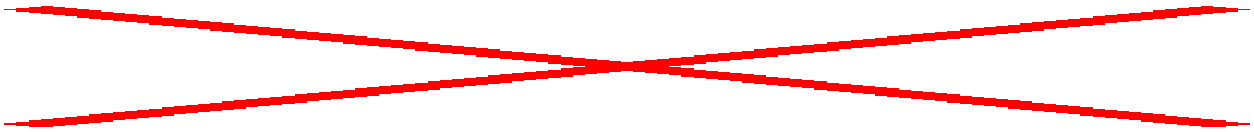


The two algorithms run in nearly identical time. There doesn't seem to be any noticeable difference in run-time with regards to pivoting and non-pivoting algorithms. This is concurrent with theory. The partial pivoting algorithm was developed to add additional stability to the program, enabling it to deal with zero-pivot situations without returning an error.

This prevention, in itself, it not extremely useful for this algorithm. Because partial pivoting only becomes used when certain cells in a matrix are filled with the number 0, it is almost never implemented. The numbers in the matrices' cell range from -100 to 100 and are accurate to no less than six significant figures. This gives, total, a less than 1/2000000 chance of producing a 0, which makes it an event that almost never happens. Thus, the partial pivoting (on standard runs) does not seem to provide any additional time lessening (indeed, the data would suggest that it runs in slightly less efficient time), and only nominal robustness to the code.

The next set of experiments focused on the mean variable value for system of equations. 100 trials were run on 3x3 matrices, and the mean value for all three variables were collected, along with

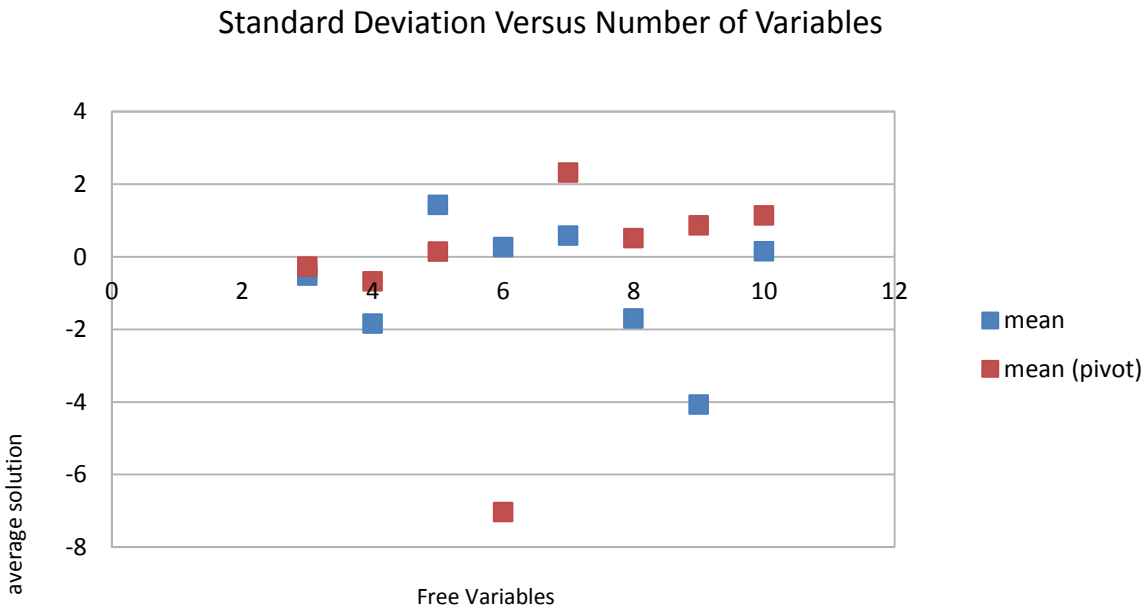
the standard deviation. The following chart illustrates this data.



As the data shows, the average values for both the pivot method and the standard method closely approach zero. This makes sense, as given the opportunity to choose a random number between -100 and 100, the average number for any arbitrary number of choices would be 0. The standard deviation (calculated as the average difference of every data point to the mean) was notedly higher than the average, which also makes sense for this data. Because there is no rhythm or reason as to which value is picked (and the value differences can be very different. It would make sense that the standard deviation would be quite large.

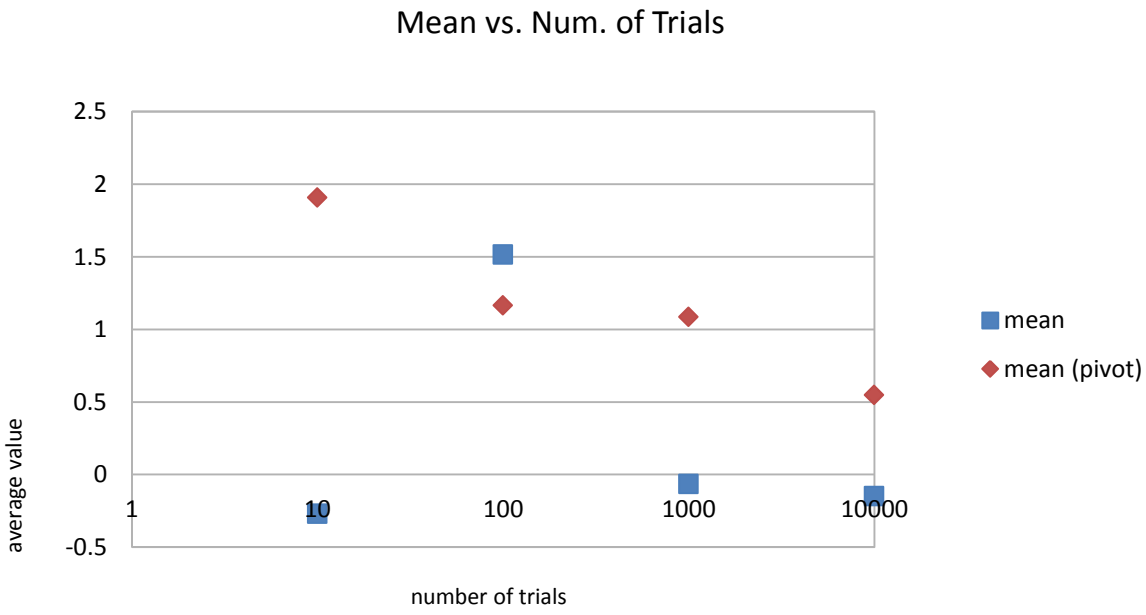
The mean for the partial pivoting algorithm seems to be further from 0 than the non-pivoting formula; further, the standard deviation for these variables is also larger. At present, it is unknown what would cause this difference, however, considering that all the means for the pivoting algorithm are within one standard deviation of the means for the non-pivoting one, it is unlikely to be statistically significant.

The mean and standard deviation of variables were also measured for matricies of sizes greater than three. The following shows the results for sizes three through 10. Please note that each data point represents the average mean and standard deviation of all variables in each matrix, it does not represent each variable separately as above.



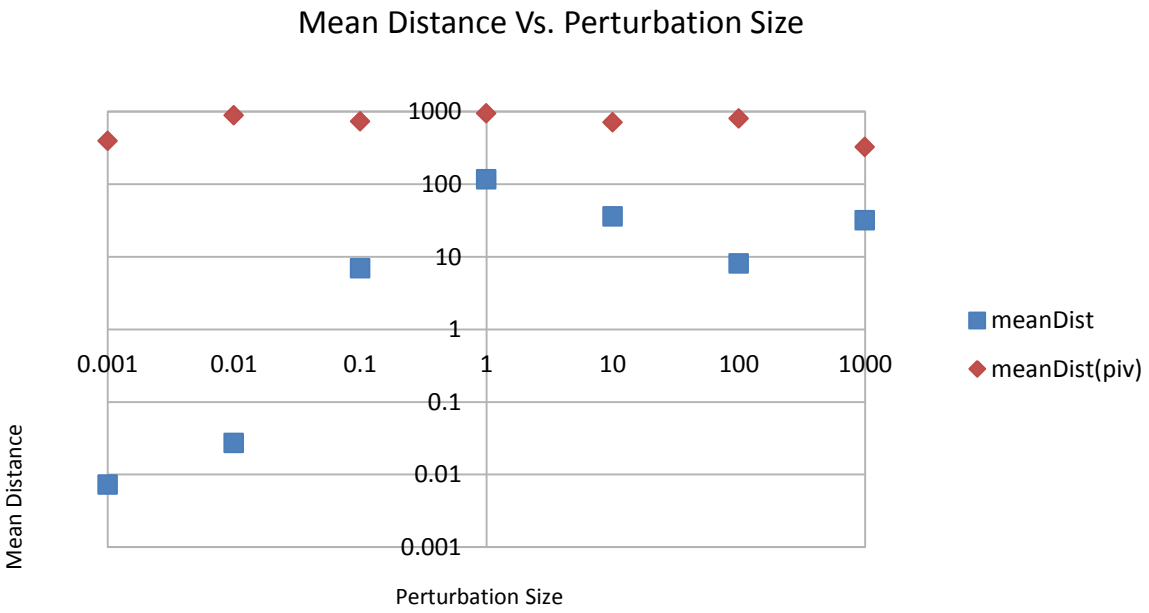
The following graph shows similar data that corroborates the conclusions of the previous data. It would seem that no matter how many variables there are in the matrix, they all have an average near 0, and have significant amounts of standard deviation.

The next experiment tested the mean and standard deviation of the solution variables with regards to number of trials run. Data was gathered in the same way as the previous experiment, and the following graph represents the data. Please note the logarithmic scale (in terms of powers of 10) on the x axis.



This scattered data seems to show that evidence that backs up previous observations. The standard deviation apparently does not change at all with increased trial runs, as expected. The average, however, more and more closely approaches 0. As with the very first graph, we notice that the mean value for the partial pivoting algorithms is at many times higher than the non-pivoting algorithm. However, because both these values are still well within one standard deviation of one another, it is likely that this is not a statistically significant note.

The final experiment focused on perturbation of the coefficient matrices' values and the impact on the solution. For this test, two sets of identical matrices (both coefficient and solution) were created, and on had its coefficient matrix's entries' values changed by a random number between the negative perturbation and the positive perturbation. The resulting answers of these two systems of matrices were compared and there answers are as follows. Please note the logarithmic scale on both the x and y axis



This data would seem to indicate that for the non-pivoting algorithm, the mean distance between the matrix and the perturbed matrix increases with the size of the perturbation. This is consistent with theory, as the perturbation increases in possible size, it increases the percent of the original value that it can change, thus allowing it to create a larger difference in the measured means.

The partial pivoting algorithm, however, did not increase or decrease in mean notably with increased perturbation.

Conclusions

The results found by this experiment seem to indicate that the hypothesis that Gaussian Elimination runs in (n^3) time is true. Although it was only shown it the algorithms run in only bigO(n^3), further experimentation would be able to produce a more specific run time. The data also indicated that there is little difference in terms of run time between the partial pivoting and standard algorithms. We were also able to conclude that he average value of an answer (given the present experiments constraints, would be zero, and that the standard deviation would be very large, due to the

inherently random nature of numbers generated by the program. Lastly, we were able to conclude that in terms of mean values (both for perturbations and non perturbations) the partial pivoting algorithm produced results that may be statistically significant. Further testing is needed to confirm or deny whether this was simply an anomaly or relevant data.