2011

Project: Gaussian Reduction

In MATLAB

—Abstract—

The Gaussian Reduction Project is mainly about first writing a function in MATLAB to solve a system of linear equations using direct Gaussian reduction, along with a second version using partial pivoting for comparison with the first one. Through the project, it will run both programs on correctness, timing and distribution tests to compare on different sizes and find out the relationship between the running time on an $N * N$ system and N^3. It will also run both the original and ill-conditioned systems for perturbation experiment and more to find out the effect of pivoting. The project is going to explore the Gaussian Reduction as well as partial pivoting.

CSC 160
3/20/2011

CSC 160

The Gaussian Reduction Project

03/20/2011

The Gaussian Reduction Project (MATLAB)

Introduction

This project involved writing a function in MATLAB to solve a system of linear equations using Gaussian reduction. There is a second version additionally using partial pivoting for comparison with the first one using direct Gaussian reduction. Through the project, it will run both programs on correctness, timing and distribution tests to compare twp programs (one without partial pivoting and one with partial pivoting) on different sizes. Though the plotting the result from the tests, we can also what percentage overhead adding pivoting entails and the relationship between the running time on an $N * N$ system and N^3 as well. It will also run both the original and ill-conditioned systems on each program for perturbation experiment and more to find out the effect of pivoting. The project will mainly explore the Gaussian Reduction as well as the effect of partial pivoting on Gaussian Reduction.

- Matrix Solutions to Systems of Linear Algebraic Equations (Attaway 11.2)

A linear algebraic equation is an equation of the form

$$a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n = b \qquad \text{(Eq. 1.1)}$$

where the a's are constant coefficients, the x's are the unknowns, and b is a constant.

A system of linear algebraic equations is a set of equations of the form:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots + a_{3n}x_n = b_3$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n = b_m \qquad \text{(Eq. 1.2)}$$

This is called an $m * n$ system of equations; for there are m equations and n unknowns.

It can also be represented in matrix form as

$$Ax = b \qquad \text{(Eq. 1.3)}$$

where A is a matrix of the coefficients, x is a column vector of the unknowns, and b is a column vector of the constants from the right-hand side of the equations. The detailed combination in matrix production form is shown as below:

$$\begin{matrix} & A & & & & & x & = & b \end{matrix}$$
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{12} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix} \quad \text{(Eq. 1.4)}$$

All systems of linear equations have either: no solutions, one solution, or infinitely many solutions. The goal for the solver function is to use the Gaussian elimination (explained later) to find the unique one solution vector for the linear equation which has an $N * N$ system as its coefficient matrix.

- Gauss Elimination (Attaway 11.2.2.1)

The Gauss elimination method to solve a system of equations consists of:

- creating the augmented matrix [A: coefficient matrix (parametric matrix); b: constant vector]
- Applying row reduction operations to this augmented matrix to get an upper triangular form in the coefficient matrix (forward elimination)
- Then use back-substitution to solve

Generally, the process of Gaussian elimination consists of two parts. The first part (forward elimination) reduces a given system to upper-triangular form, or results in a degenerate equation with no solution or infinite solutions, indicating the system is inconsistent or singular. This is accomplished through the use of elementary row operations (EROs). The second part uses back substitution to find the solution of the system above.

- Partial Pivoting (Wikipedia)
- In partial pivoting, the algorithm selects the entry with largest absolute value from the column of the matrix that is currently being considered as the pivot element. Partial pivoting is generally sufficient to adequately reduce round-off error.

Procedure (explanation of the functions used to experiment the methods)

- Program with basic Gaussian elimination: gauss_reduce()

Solving a system of linear equations uses direct Gaussian reduction. This function takes a square matrix of parameters, and a vector of constants, and returns a solution vector. The function is able to take systems of any size (up to practical limits) and it can check that the matrix is square, and that the constant vector has the appropriate number of elements, printing an error message and returning the zero vector if these solvability conditions are violated. The program will also print an error message and return a zero vector if it runs into a zero pivot which might indicate a singular system, though does not

necessarily imply that, (double-check the zero pivot after elementary row operations (EROs) reaching upper-triangular form).

Secondary functions are called by the main function to do each parts of the Gaussian elimination.

Sample for running the program:

```
%Run the programs on examples of 2x2 system:
%Generate example of 2x2 using the 'dessert':
[param_mat_2, const_vec_2] = random_test_case(2)
param_mat_2 =
  -42.3942  -32.6584
  -32.5711  -34.1581
const_vec_2 =
   84.2446
  -78.2955


%Run the programs on example of 2x2 using the 'Main Course 1':
solution_vec_2 = gauss_reduce(param_mat_2, const_vec_2)
solution_vec_2 =
  -14.1385
   15.7738
```

- (Second Version) Program additionally using partial pivoting: gauss_reduce_pp()

This program serves as the second version of the first program, which functions the same but additionally uses partial pivoting (pp) to refine the basic Gaussian elimination. Partial pivoting identifies and swaps in the correct pivot row for a specified working column (modifying a matrix and constant vector).

Experiments:

- Correctness, Timing and Distribution Tests

1. Correctness Tests

Run both programs on examples of 2x2, 3x3, and 4x4 systems, and check that the solutions are correct with a function that takes the coefficient matrix, the solution vector, and the constant vector, multiplies the coefficients by the solution and checks that the result is equal to the constant vector. (Using the help of eps() to check since the values are floating).

2. Timing Tests

Use *tic, toc* to time each program running on 10x10, 100x100, and 1000x1000 systems. Plot the results over a range of sizes to see if it supports the theoretical result that the running time on an NxN system is proportional toN^3. We might also find out what percentage overhead adding pivoting entails.

3. Distribution Tests

Run your solver on 100 random 3x3 examples (generated by random_test_case() function), and compute the mean and standard deviation of each of the 3 solution variables. Compare the characteristics of both mean and standard deviation for each program.

- Perturbation Experiment (Effect of Pivoting)

For each of 100 random 10 x 10 systems generated by random_test_case() (i.e. with coefficient values between -100.0 and 100.0), first find the solution to the system

using each program and record it. Then perturb the system by adding a (different) random real-valued number between -1.0 and 1.0 to each of the matrix coefficients and each constant element (1% of the original range). Again, using each program to solve the perturbed system and then compute the distance between the solution to the perturbed system and the solution to the original system.

Since we might not get a significant result (interesting differences in the mean and standard deviations) with that perturbation in the range [-1.0, 1.0], so we will compare the result in each range of 10^0 to 10^{-9}, and analyze the result.

An error will exit from the program when it encountered a zero pivot (or other reason) and a new random system will be re-run in the experiment, so that it will not contaminate the data. (Note: the probability of a zero pivot occurring should be quite low.)

- Ill-Conditioned Systems and More

A random ill-conditioned 10 x 10 system can be produced by first generating the coefficient matrix (and corresponding constant vector) at random, using values between -100.0 and 100.0 as the above perturbation experiment. Then randomly select 3 of the 10 rows, and make one of them almost the average of the other two by taking the $\frac{\text{sum}}{2}$ (average) plus a random value between -1.0 and 1.0 (1% of the original range). Repeat the perturbation experiment using 100 such ill-conditioned systems. This is for comparing answers to original and perturbed systems using the ill conditioned systems, rather than purely random ones as above. Test both the basic program and the second version using partial pivoting. Analyze the data through plotting and find out how the mean and standard deviation of the error compare to that of the well conditioned random systems. The result from comparison will indicate the effect of partial pivoting. Again, repeat the ill-conditioned systems on smaller ranges, like 0.01%, 0.00001%.

- Extra Credit

Repeat the perturbation experiment using 100 randomly generated systems of size 5, 10, 20, 50, 100, 200. From the plot of the result, explain if the mean error is correlated with the size of the system.

Results (figures and statistical analysis)

- Correctness, Timing and Distribution Tests

1. Correctness Tests (from diary_1)

```
%Run the programs on examples of 2x2, 3x3, and 4x4 systems:
%Generate example of 2x2:
[param_mat_2, const_vec_2] = random_test_case(2);
%Run the programs on example of 2x2:
solution_vec_2 = gauss_reduce(param_mat_2, const_vec_2)
solution_vec_2 =
   -0.3390
   -0.6408

%(Gaussian Reduction with partial pivoting)
solution_vec_2_pp = gauss_reduce_pp(param_mat_2, const_vec_2)
```

```
solution_vec_2_pp =
   -0.3390
   -0.6408
%check if the solutions are correct:
[ checked_result_2 ] = checking_solution(param_mat_2, const_vec_2,
solution_vec_2)
checked_result_2 =
The solutions are correct.
%(the result from Gaussian Reduction with partial pivoting)
[ checked_result_2_pp ] = checking_solution(param_mat_2, const_vec_2,
solution_vec_2_pp)
checked_result_2_pp =
The solutions are correct.

%Generate example of 3x3:
           ⋮
%check if the solutions are correct:
[ checked_result_3 ] = checking_solution(param_mat_3, const_vec_3,
solution_vec_3)
checked_result_3 =
The solutions are correct.
%(the result from Gaussian Reduction with partial pivoting)
[ checked_result_3_pp ] = checking_solution(param_mat_3, const_vec_3,
solution_vec_3_pp)
checked_result_3_pp =
The solutions are correct.

%Generate example of 4x4:
           ⋮
%check if the solutions are correct:
[ checked_result_4 ] = checking_solution(param_mat_4, const_vec_4,
solution_vec_4)
checked_result_4 =
The solutions are correct.
%(the result from Gaussian Reduction with partial pivoting)
[ checked_result_4_pp ] = checking_solution(param_mat_4, const_vec_4,
solution_vec_4_pp)
checked_result_4_pp =
The solutions are correct.
```

Statistical analysis:

The solution vectors from both programs are correct (checked using eps(100)).

2. Timing Tests

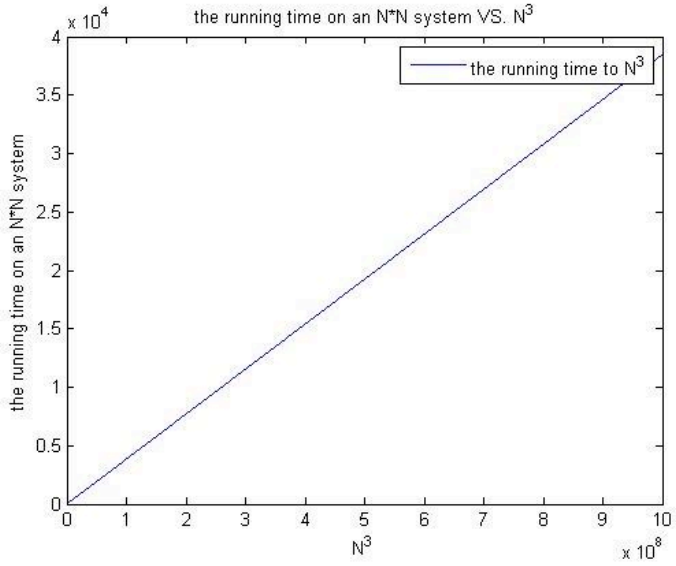

Figure 1.1 the running time on an N*N system vs. N^3

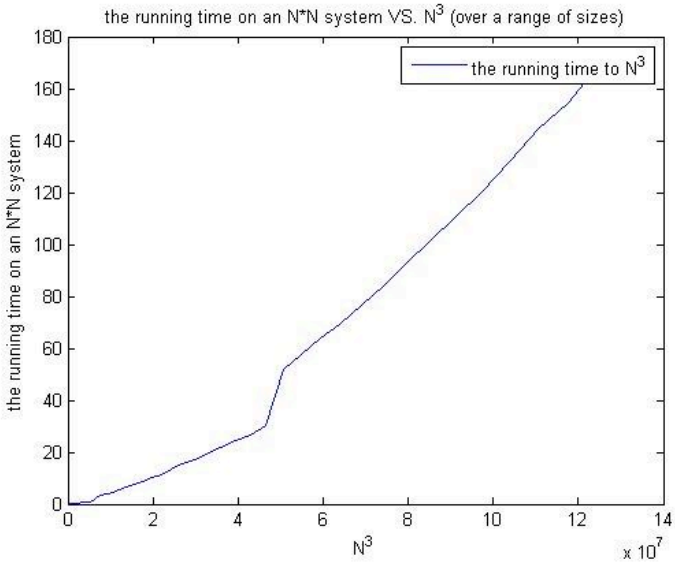

Figure 1.2 the running time on an N*N system vs. N^3 (over a range of sizes)

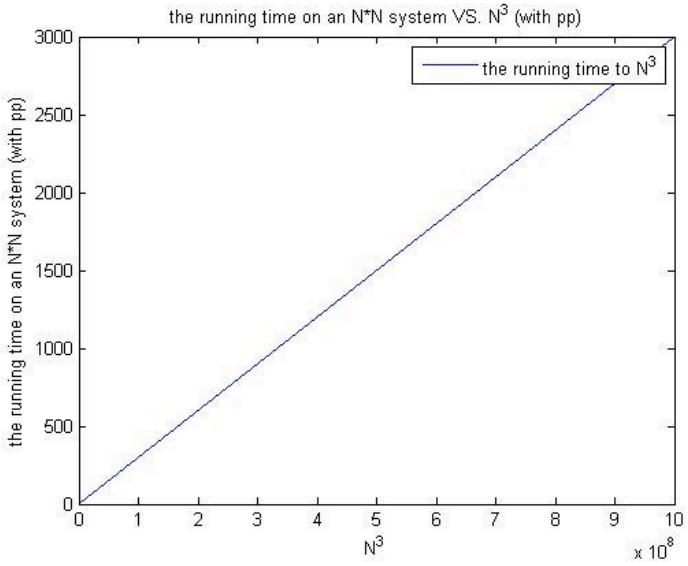

Figure 1.3 the running time on an N*N system vs. N^3 (with pp)

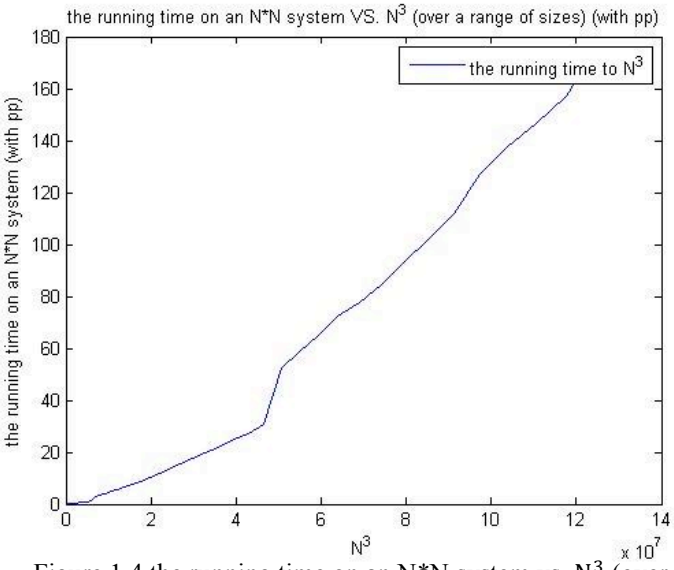

Figure 1.4 the running time on an N*N system vs. N^3 (over a range of sizes) (with pp)

Statistical analysis:

The Figures 1.1 – 1.4 shows that there is approximately a linear relationship between the running time on an N*N system and N^3:

Running time on an N*N system = $k * N^3$, k = constant real number (Eq. 2.1)

The running time on an N*N system is approximately proportional toN^3 either in the basic program and the second version with partial pivoting.

From the Figures 1.1 – 1.4, the percentage overhead adding pivoting entails is quite small, approximately under 7%. The growth on running time in the beginning is slower when $N^3$is smaller than $5 * 10^7$.

3. Distribution Tests (from diary_1)

```
%Run your solver on 100 random 3x3 examples:
The mean for the first variable is -0.349473;
 the mean for the second variable is 0.105242;
 the mean for the third variable -0.448598.
The standard deviation for the first variable is 3.662361;
 the standard deviation for the second variable is 4.727055;
 the standard deviation for the third variable 5.787888.

%Run your solver (with partial pivoting) on 100 random 3x3 examples:
With partial pivoting, the mean for the first variable is -0.484948;
 the mean for the second variable is 0.552242;
 the mean for the third variable -0.288744.
With partial pivoting, the standard deviation for the first variable is
3.409167;
 the standard deviation for the second variable is 4.228195;
 the standard deviation for the third variable 4.180806.
```

Statistical analysis:

The mean for each of the three variables is around zero (mostly within [-1, 1]). The standard deviation of each of the three variables is around 4 (mostly within [3, 6]). The mean and the standard deviation for each of the three solution variables are comparatively quite small comparing with [-100, 100], the range of each element in the 3x3 examples (generated by random_test_case() function).

- Perturbation Experiment (Effect of Pivoting)

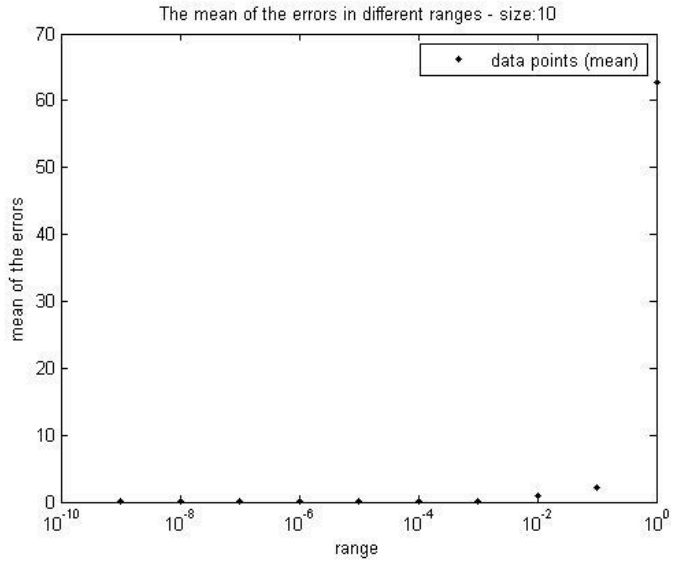

Figure 2.1 the means of errors in different perturbed ranges

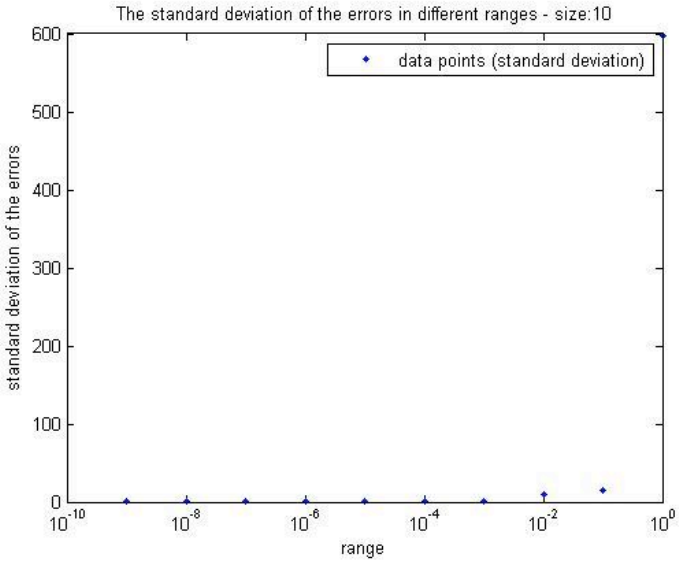

Figure 2.2 the standard deviations of errors in different perturbed ranges

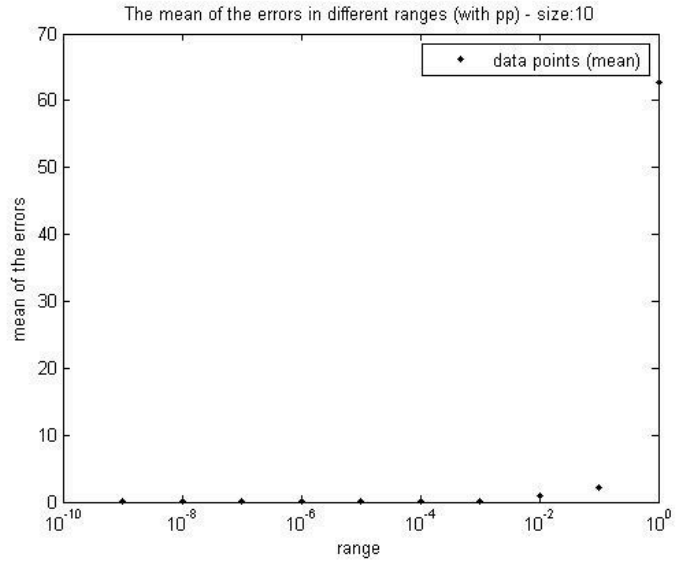

Figure 2.3 the means of errors in different perturbed ranges (with pp)

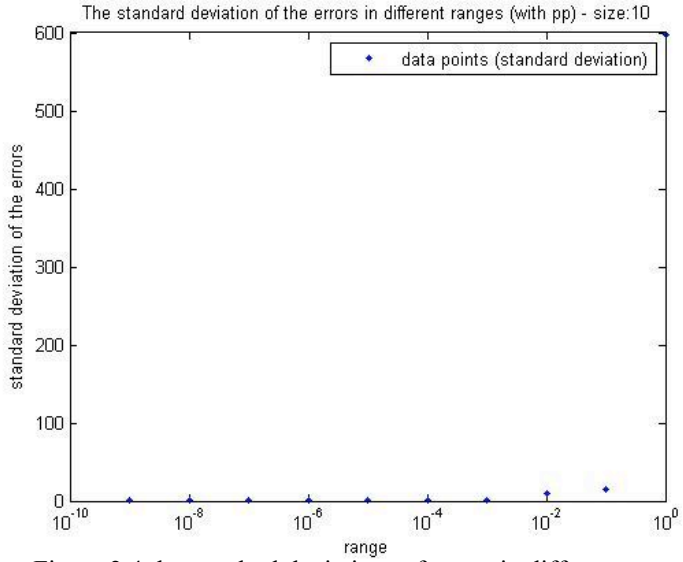

Figure 2.4 the standard deviations of errors in different perturbed ranges (with pp)

Statistical analysis:

Through plotting on Figure 2.1 – 2.4, we compared the result with perturbation in each range of 10^0 to 10^{-9}, and after analyzing the result we still cannot distinguish significant differences in the mean and standard deviations between the basic Gaussian reduction program and the second version with partial pivoting.

Note: the main program has made sure an error will exit from the program when it encountered a zero pivot (or other reason) and a new random system (of 10 * 10) will be re-run in the experiment, so that it will not contaminate the data.

- Ill-Conditioned Systems

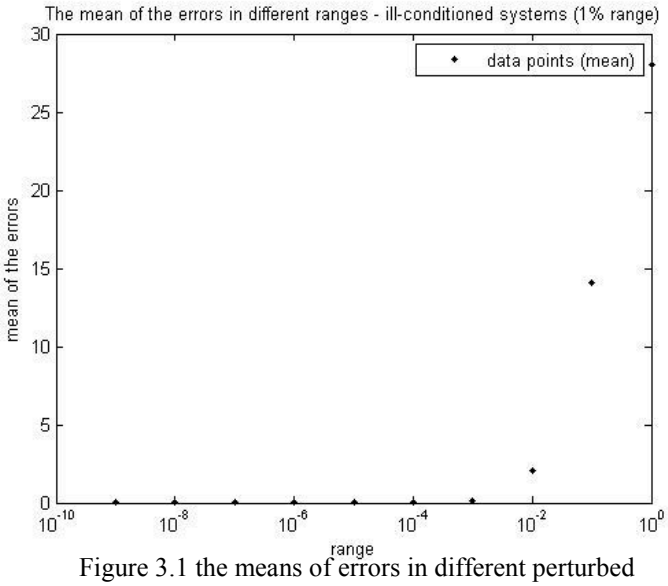

Figure 3.1 the means of errors in different perturbed ranges (1% ill-conditioned systems)

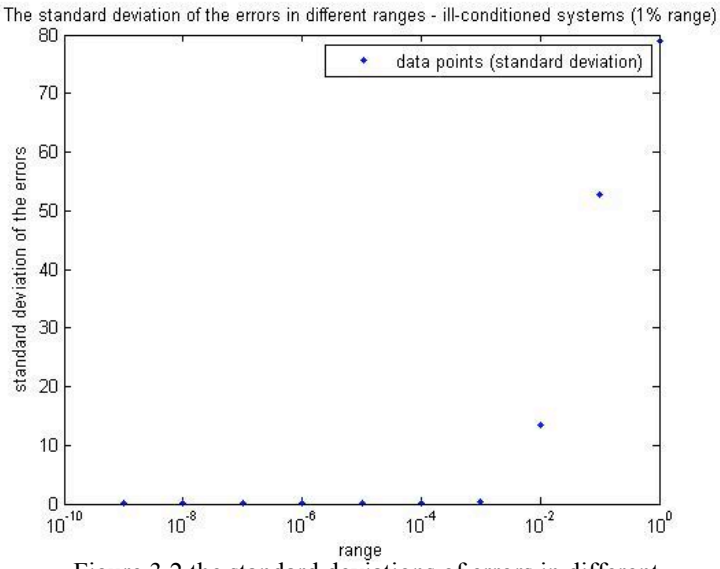

Figure 3.2 the standard deviations of errors in different perturbed ranges (1% ill-conditioned systems)

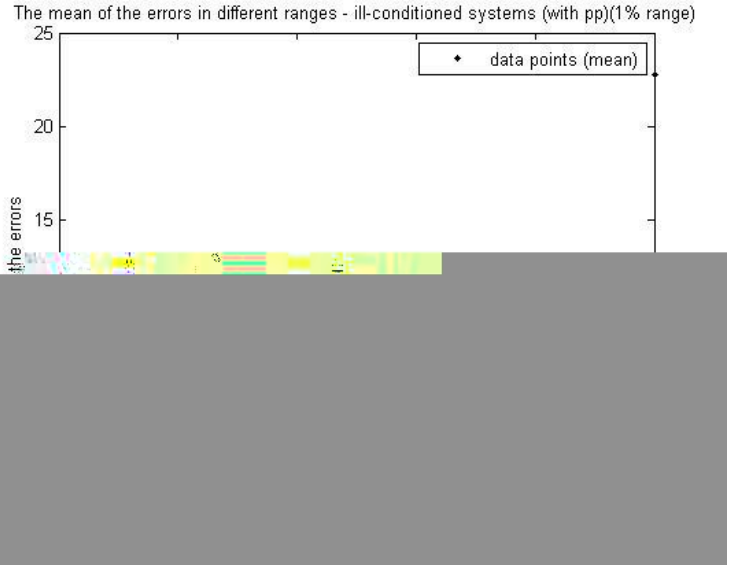

Figure 3.3 the means of errors in different perturbed ranges (1% ill-conditioned systems) (with pp)

Figure 3.4 the standard deviations of errors in different perturbed ranges (1% ill-conditioned systems) (with pp)

Statistical analysis:

From Figures 3.1 – 3.4, we compared the result with perturbation in each range of 10^0 to 10^{-9} in 1% ill-conditioned systems. The mean and standard deviation of the error might be a little smaller for [-1, 1] perturbed range, but are larger for the rest of perturbed ranges comparing to those of the well conditioned random systems.

Comparing Figure 3.1, 3.2 and 3.3, 3.4, the mean and standard deviation of the error in 1% ill-conditioned systems are smaller in the solver program with partial pivoting.

- Extra Credit

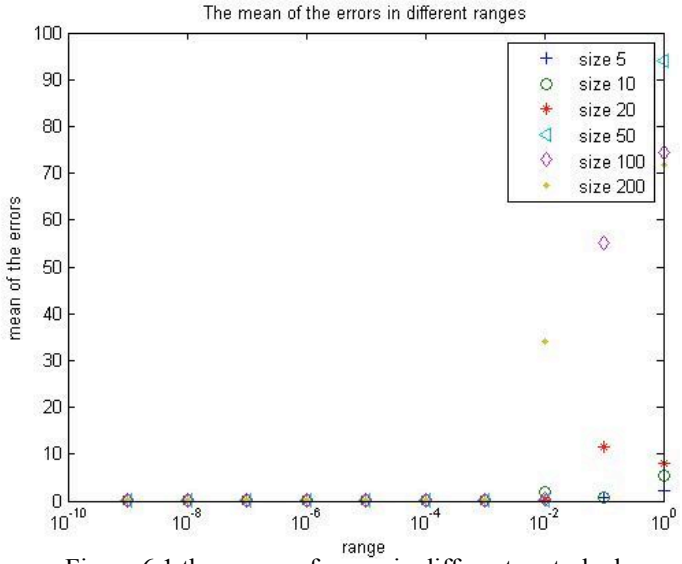

Figure 6.1 the means of errors in different perturbed ranges (in different sizes of systems)

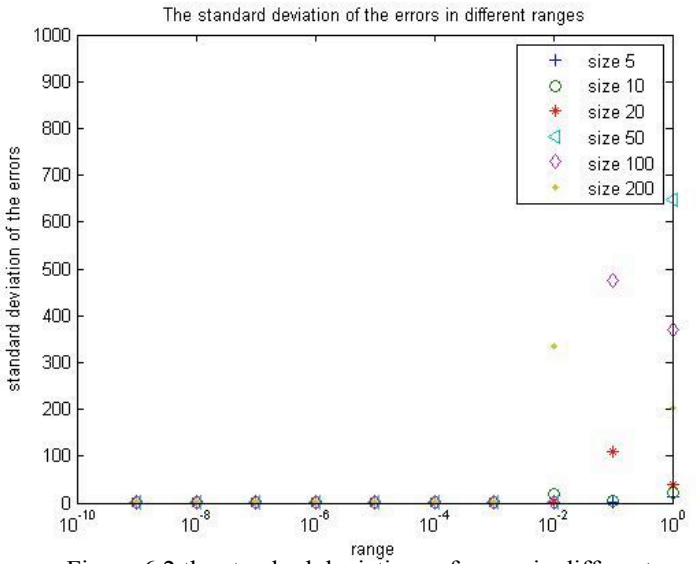

Figure 6.2 the standard deviations of errors in different perturbed ranges (in different sizes of systems)

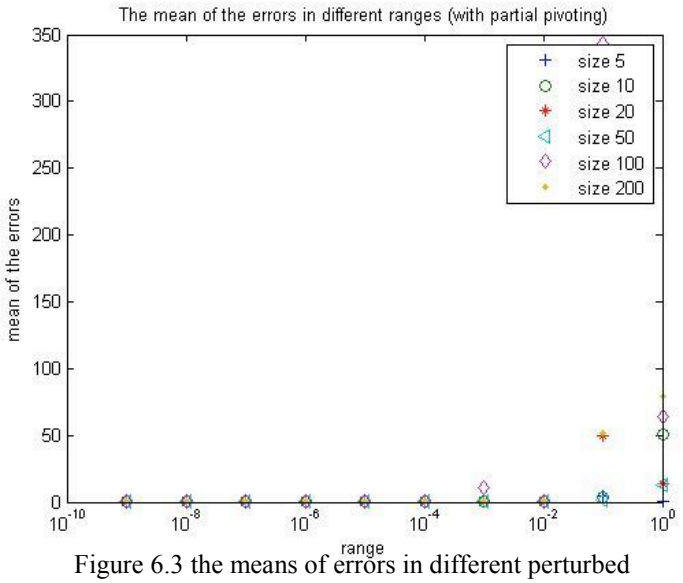

Figure 6.3 the means of errors in different perturbed ranges (in different sizes of systems) (with pp)

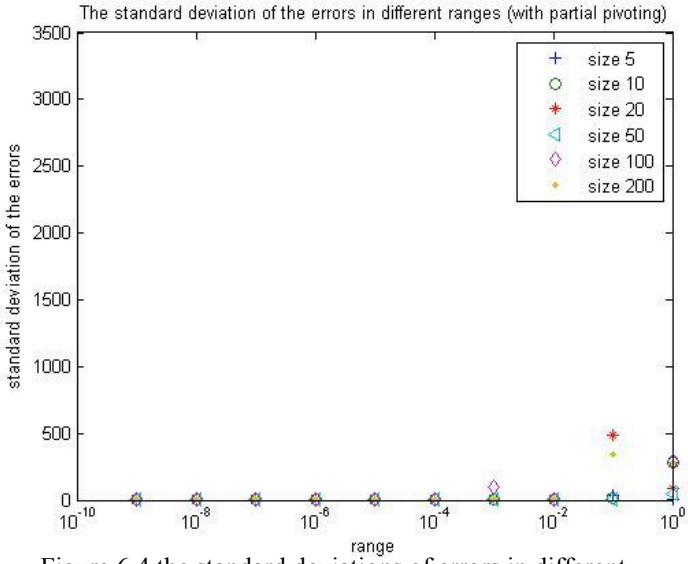

Figure 6.4 the standard deviations of errors in different perturbed ranges (in different sizes of systems) (with pp)

Statistical analysis:

From Figures 6.1 – 6.4, we compared the result of the means and standard deviations of errors in different perturbed ranges in different sizes systems. The plotting indicates that when the perturbed range is larger or equal to 10^{-3} (range $\geq 0.001\%$), the sizes of system seem to have effects on the result of the perturbation experiment, which means the mean error is correlated with the size of the system in comparatively larger perturbed ranges.

Comparing Figure 6.1, 6.2 and 6.3, 6.4, the mean and standard deviation of the error in different perturbed ranges are less affected by different sizes of systems in the solver program with partial pivoting.

Discussion (including conclusions and the recommendations for future work

From the Correctness Tests, it can be concluded that both solver programs are running well judging from the correctness of the result.

From the Timing Tests,

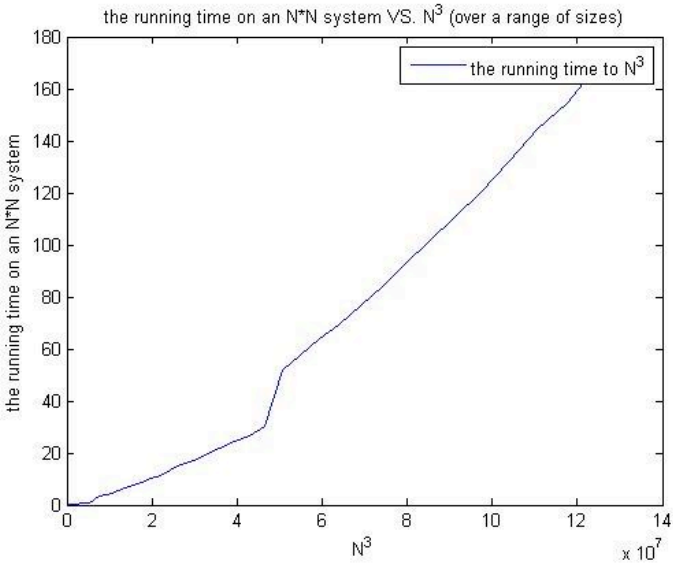

Figure 1.2 the running time on an N*N system vs. N^3 (over a range of sizes)

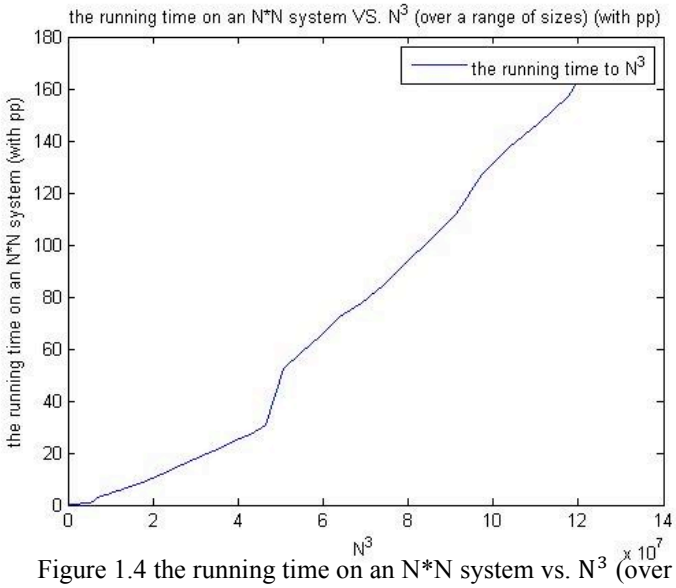

Figure 1.4 the running time on an N*N system vs. N^3 (over a range of sizes) (with pp)

the experiment figures show that there is approximately a linear relationship between the running time on an N*N system and N^3:

Running time on an N*N system = k *N^3, k = constant real number (Eq. 2.1)

The running time on an N*N system is approximately proportional toN^3 either in the basic program and the second version with partial pivoting.

From the Figures 1.1 – 1.4, the percentage overhead adding pivoting entails is quite small, approximately under 7%. In conclusion, partial pivoting can refine the Gaussian elimination but will NOT add much time in running the whole program. The slower growth on running time in the beginning (when $N^3$is smaller than $5 * 10^7$), may indicate that MATLAB may be recognizing the existence of the loop in the function and changing the processing algorithm behind the screen.

From the Distribution Tests, The mean for each of the three variables is around zero (mostly within [-1, 1]). Since each element in the original matrices generated by random_test_case() function is in the range of [-100, 100], the mean for each of the three variables are expected to be around the middle of the range, zero. The standard deviation of each of the three variables is around 4 (mostly within [3, 6]). Therefore, it is within the expectation from the characteristic of the input data, the mean and the standard deviation for each of the three solution variables are around zero and comparatively quite small comparing with [-100, 100] (the range of each element in the 3x3 examples).

- Perturbation Experiment (Effect of Pivoting)

Through the data presented above in the 'Result' part (Figure 2.1 – 2.4), we compared the result with perturbation in each range of 10^0 to 10^{-9}, and after analyzing the result, significant differences still cannot be distinguished from the mean and standard deviations between the basic Gaussian reduction program and the second version with partial pivoting. However, this does not necessarily mean that partial pivoting does not have any effect.

Note: the main program has made sure an error will exit from the program when it encountered a zero pivot (or other reason) and a new random system (of 10 * 10) will be re-run in the experiment, so that it will not contaminate the data (the size of the sample data will always be 100).

- Ill-Conditioned Systems

From Figures 3.1 – 3.4, we compared the result with perturbation in each range of 10^0 to 10^{-9} in 1% ill-conditioned systems. The mean and standard deviation of the error might be a little smaller for [-1, 1] perturbed range, but are larger for the rest of perturbed ranges comparing to those of the well conditioned random systems. From Figures 4.1 – 4.4, we compared the result with perturbation in each range of 10^0 to 10^{-9} in 0.01% ill-conditioned systems. The standard deviation of the error is larger for most of perturbed ranges comparing to those of the well conditioned random systems. From Figures 5.1 – 5.4, we compared the result with perturbation in each range of $10^0$to 10^{-9} in 0.00001% ill-conditioned systems. The standard deviation of the error is much larger for most of perturbed ranges comparing to those of the well conditioned random systems and those of the larger range ill-conditioned systems (shown as above from Figure 3.1 - 4.4). In conclusion, both the mean and the standard deviation of the errors will get larger as the percentage for the ill-condition get smaller.

Comparing Figure 3.1, 3.2 and 3.3, 3.4, the mean and standard deviation of the error in 1% ill-conditioned systems are smaller in the solver program with partial pivoting. Comparing Figure 4.1, 4.2 and 4.3, 4.4, the mean and standard deviation of the error in 0.01% ill-conditioned systems are more consistent and smaller in the solver program with partial pivoting. Comparing Figure 5.1, 5.2 and 5.3, 5.4, the mean and standard deviation of the error in 0.00001% ill-conditioned systems are more consistent but not necessarily smaller in the solver program with partial pivoting. In conclusion, when running on ill-conditioned systems, the mean and standard deviation of the error in are mostly more consistent and smaller in the solver program with partial pivoting (at least under 0.00001% ill-condition range). Partial pivoting can refine the basic Gaussian elimination to reach more accurate solutions in ill-conditioned systems.

- Extra Credit

From Figures 6.1 – 6.4, we compared the result of the means and standard deviations of errors in different perturbed ranges in different sizes systems. The plotting indicates that when the perturbed range is larger or equal to 10^{-3} (range $\geq 0.001\%$), the sizes of system seem to have effects on the result of the perturbation experiment, which means the mean error is correlated with the size of the system in comparatively larger perturbed ranges. In conclusion, when the perturbed range is larger or equal to 10^{-3} (range $\geq 0.001\%$), the mean error is correlated with the size of the system, when the system size is larger, the mean and the standard deviation of the error will both be comparatively larger.

Comparing Figure 6.1, 6.2 and 6.3, 6.4, the mean and standard deviation of the error in different perturbed ranges are less affected by different sizes of systems in the solver program with partial pivoting. In conclusion, with partial pivoting, the basic Gaussian elimination functions will have more consistent and accurate performance in different sizes systems.

Below is the specific comparison (in addition to the overall comparison in 'Result' part above) between size 5 and size 100 in the perturbation experiment. The above conclusion about the mean and the standard deviation of the error will be both rising with the growth of system sizes is visualized and proved in the Figures 7.1 – 7.4 (below).

Comparison in mean:

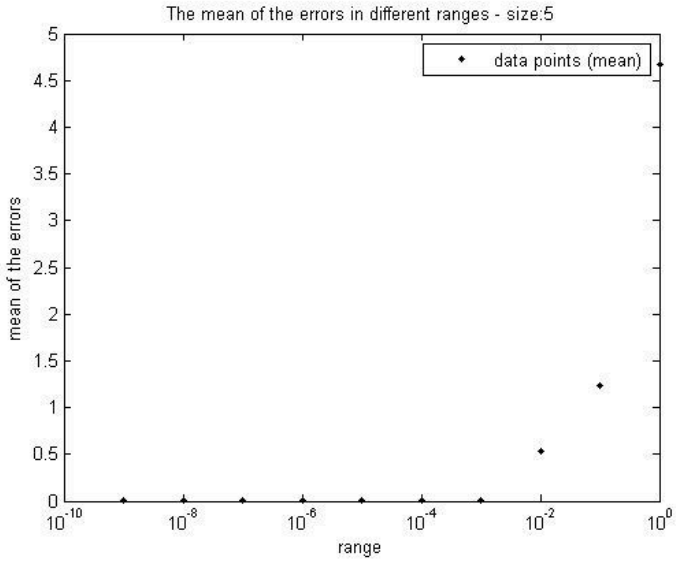

Figure 7.1 the means of errors in different perturbed ranges (system size: 5)

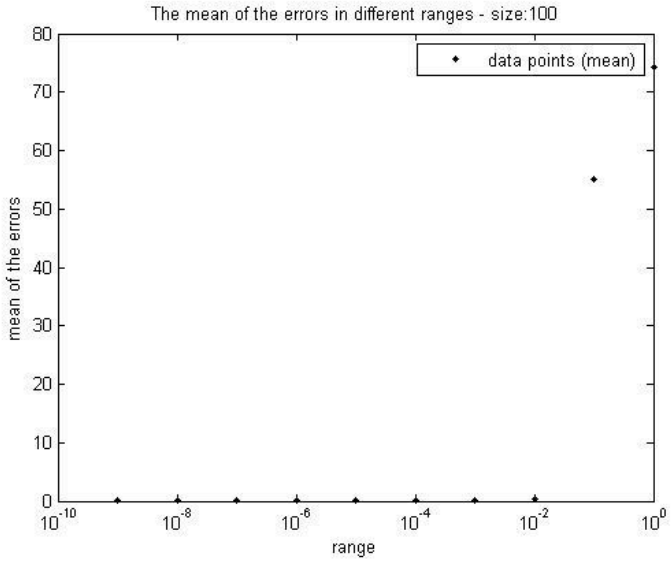

Figure 7.2 the means of errors in different perturbed ranges (system size: 100)

Comparison in standard deviation:

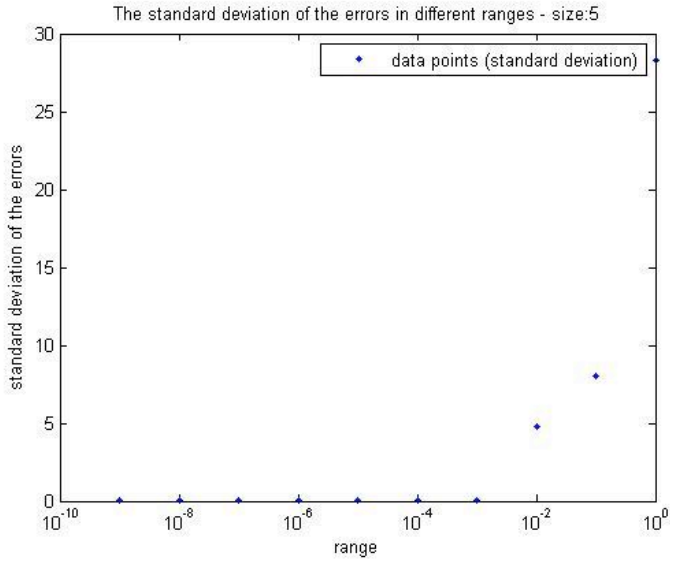

Figure 7.3 the standard deviations of errors in different perturbed ranges (system size: 5)

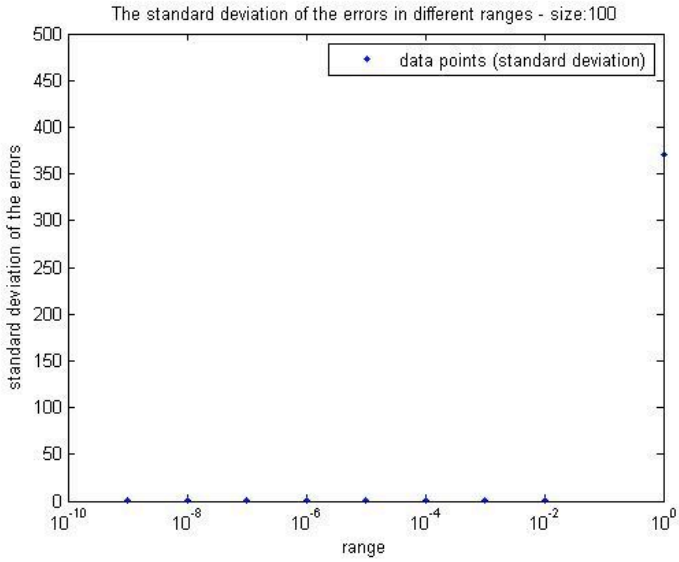

Figure 7.4 the standard deviations of errors in different perturbed ranges (system size: 100)

Appendix

pivot.m

Note: For the rows under the first row, the pivot can be turned to zero during row reduction (after normal partial pivoting). The function 'pivot' I designed can detect and avoid this problem before it happens. It will select another row to switch with the current pivot's row if it detects that the pivot can be turned to zero during row reduction.

References

Attaway, S. 2009 A Practical Introduction to Programming and Problem Solving Chapter 11 Solving Systems of Linear Algebraic Equations

The main ideas of the functions (including the inside computing method) mainly originated from BlackBoard CSC 160 Gaussian Reduction Assignment page.

Author: Chris Brown

Complete URL:

http://my.rochester.edu/webapps/portal/frameset.jsp?tab_tab_group_id=_2_1&url=/webapps/blackboard/execute/courseMain?course_id=_40459_1

Date that accessed the site: 03/08/2011

Author: (source) Wikipedia.org

Complete URL:

http://en.wikipedia.org/wiki/Pivot_element#Partial_and_complete_pivoting

Date that accessed the site: 03/08/2011