# Numerical Optimizations and Linear Algebra

# Assignment 1: Gaussian Elimination

## Introduction

In the current document the outcomes of 1st Assignment are reported. Algorithm of Gaussian Elimination is implemented using Python Programming Language and taking advantage of any available libraries.

## Gaussian Elimination

Gaussian Elimination is an algorithm used for solving systems of linear equations of the form . In the following, consider that a matrix of dimensions is given to be analyzed.

### Partial Pivoting

For Partial Pivoting matrix is decomposed to matrices and , having

is a lower triangular matrix, is an upper triangular matrix and is a permutation matrix (consisting of ones and zeros). restores the order of the linear equations, that was disturbed because of pivoting. The latter is needed since in partial pivoting the rows of matrix are exchanged in order to bring the element with the largest absolute value to the upper left corner of the matrix that is examined in a specific instance.

For purposes of memory saving, the Partial Pivoting function creates a copy of matrix that then its lower and upper part are iteratively transformed to matrices and respectively. For the same reason, permutation matrix is replaced by a vector with row numbers as its elements. Function returns the three matrices: and .

### Complete Pivoting

For Complete Pivoting matrix is decomposed to matrices and , having

is a lower triangular matrix, is an upper triangular matrix and and are permutation matrices (consisting of ones and zeros). and respectively restore the order of the linear equations and the order of coefficients in the equations, that were disturbed because of pivoting. The latter is needed since in complete pivoting the rows and columns of matrix *A* are exchanged in order to bring the element with the largest absolute value to the upper left corner of the matrix that is examined in a specific instance.

For purposes of memory saving, the Complete Pivoting function creates a copy of matrix that then its lower and upper part are iteratively transformed to matrices and respectively. For the same reason, permutation matrices are replaced by two vectors with row and column numbers as their elements. Function returns the four matrices: , and .

## Experiment 1: Toeplitz matrix

In order to test the implemented functions, Toeplitz matrices , of dimensions are created, for . A Toeplitz matrix is one in which each descending diagonal from left to right is constant.

Main diagonal’s values are equal to .

Values in other diagonals come from expression:

For those matrices a vector, , is chosen at random using NumPy’s method random.randn(). Then is calculated as the dot product of and , giving the wanted systems.

Now, it is time to pretend that is unknown and try to get an estimation of it usingand the algorithms of Partial and Complete Pivoting. The estimation of is .

* Partial Pivoting

For permutation matrices we have that , thus .

Set hen . By solving the two systems is acquired.

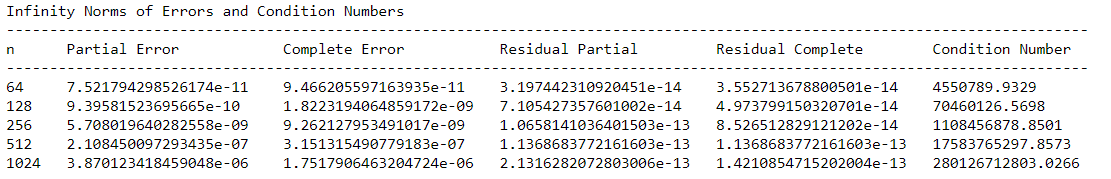
The systems mentioned are solved using SciPy’s linalg.solve() method.

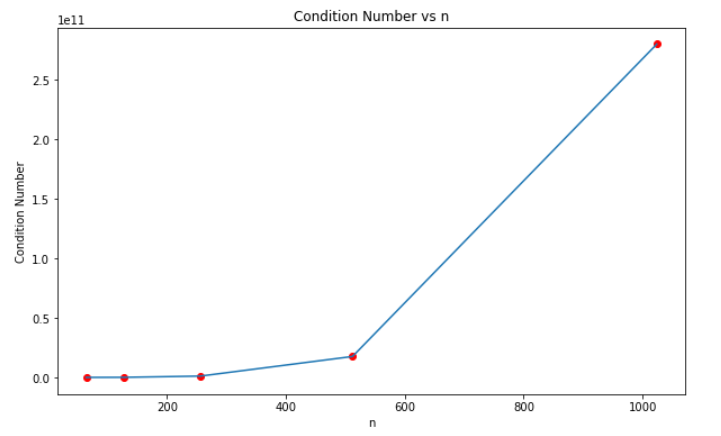
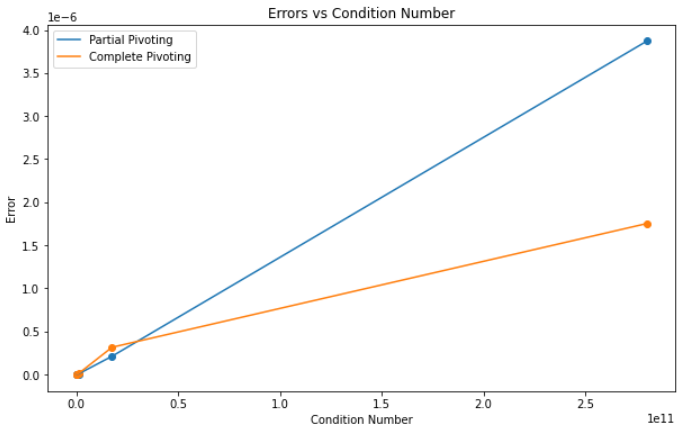
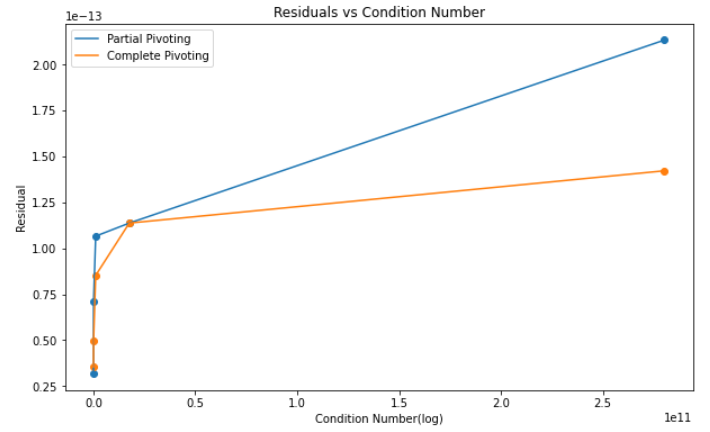
* Complete Pivoting

For permutation matrices we have that , , thus

Set , and . By solving the three systems is acquired.

The systems mentioned are solved using SciPy’s linalg.solve() method.

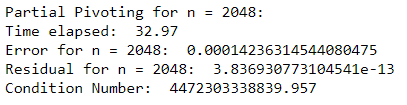
In order to compare the methods, the infinity norms of error and residual are computed. Condition numbers of matrices are also computed. Outcomes are presented below:

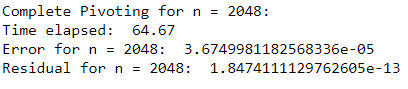


As it can be observed, as n increases, so does the condition number of the matrix. And as for errors and residuals, they do also increase as condition number increases. Also, as expected, complete pivoting errors and residuals are generally smaller than partial pivoting ones, although as stated in the next section partial pivoting does better in terms of execution time.

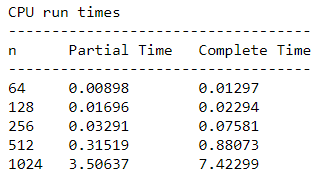
For n = 2048, it is expected that both errors and residuals as well as condition number will increase. Condition number is expected to go as high as 10 times the condition number of n = 1024. Residuals seem to stay relatively stable or increase slightly. Thus, for n = 2048 it is expected to get residual in the same order as for n = 1024. Errors, on the other hand, increase about 10 times every time n increases 2 times or, as stated, every time condition number increases 10 times.

Assumptions are validated by results shown below:





## CPU Time

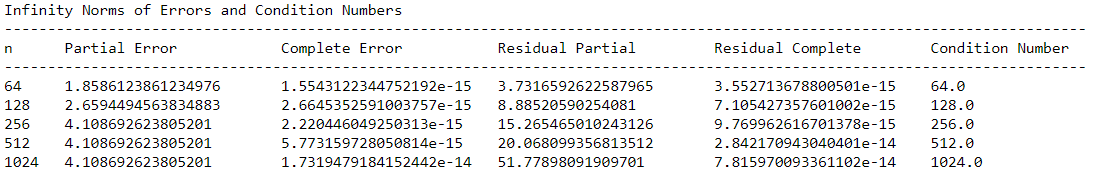
In this section, the CPU run times for the previous calculations, are presented, for the different values of .

It is obvious that Partial Pivoting runs faster than Complete. This is more intense as increases to larger values. For , the implementation that utilizes Partial Pivoting needs 32.97 seconds, whereas the implementation that utilizes Complete Pivoting needs 64.67 seconds.

In conclusion, as gets larger values, using Complete Pivoting is not recommended.

## Experiment 2

As a second experiment, there are created matrices that have 1 in the diagonal and last column and -1 in the lower triangular part (zeros everywhere else). Systems are obtained as in section 2, decomposition with Partial and Complete Pivoting is applied and an estimation of x is calculated.

In order to compare the methods, the infinity norm of residual is computed. Condition numbers of matrices are also computed. Outcomes are presented below:

It can be observed that errors when using Partial Pivoting for decomposition are very large. Moreover, Python provides with messages that the outcome may not be accurate. That is because there is no pivoting when using Partial Pivoting. How the algorithm tries to solve the problem is by adding the first row to the others to generate zeros in the lower triangular of matrix. That leads to the last element increasing 2 times for every iteration.

By the end of the algorithm, the last column of matrix will have values equal to for . When attempting to solve the system, back substitution starts by the element which is relatively a lot larger than the rest, especially as increases. That value is used for the calculation of last value of which takes a very small value.

Iteratively, the problem migrates to the other computations and because of rounding errors significant digits of values of are lost. Finally, many values of are set to 0 giving the large residuals shown to the table above.

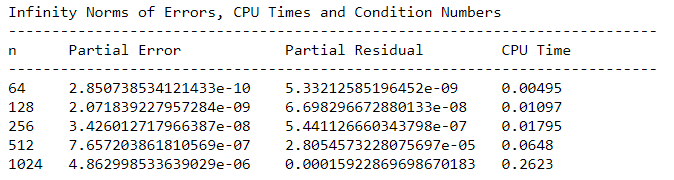
On the other hand, when using Complete Pivoting for the decomposition, errors are very small. The exchange of columns prevents the elements of to get such values and thus rounding errors remain relatively low. There is no question that choosing Complete Pivoting for this problem is the best amongst the two.

## Sherman – Morrison

For this section, two random vectors were generated and then divided by their norm, giving vectors whose norm is 1. Then their outer product was added to matrices of section 2, transforming the problem of linear equations to . It is known that with the use of Sherman – Morrison formula it holds that:

With

Using the decompositions of Partial Pivoting performed before and that formula for the matrix inversion, it is possible to solve the problem in time.

Results obtained are found in the following table: 

Indeed, CPU run times are very low and additionally norms of residuals and errors are kept low too. The matrix comes from triangular solutions and is calculated in time. Then, inversion of new matrix comes only from calculation of dot products and is also calculated in time. The final estimation of is given by one dot product and thus, time is of order .