LU and PLU Solver

QuirkyCroissant, 2014

2014

Part I - LU Decomposition

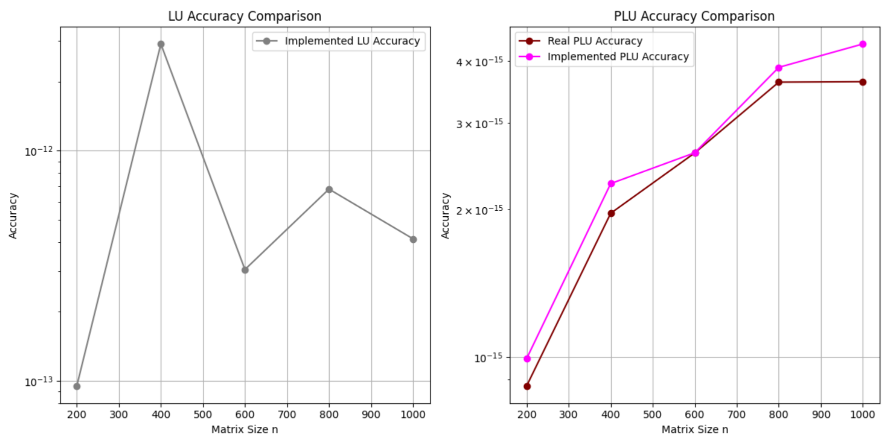

Figure 1: relative error of LU and PLU implementation

If both implemented algorithms get applied on the problems, represented by the test matrices S, it seems that both algorithms perform rather well. The LU decomposition without pivoting achieves about 10^{-12} forward error, whereas the algorithm with pivoting achieves a noticable better performance with a 10^{-15} relative error. To measure how many digits are correct with those we need to look at the power of 10, so with the former algorithm ensures that 12 digits and the later promises 15 digits to be correct. An interesting observation is that for the S matrices the implemented algorithm seems to work just as well as the "real" plu function provided by the scipy.linalg.lu library("Real PLU Accuracy"), and differs at most only maginaly from it.

Part II - Solving Triangular Linear Systems

Input data

We generate the test data matrices by making a list that generates us random generated matrices. To get this we use a list comprehension that returns the sought after list of matrices, inside of it we use a variable ”n” that iterates from 1 to 5 and gets passed into a lambda function by getting multiplied by 200(which garanties that we have the given problemsizes 200, 400, 600, 800, and 1000).

```
lower_test_matrices = [(lambda n: np.tril(np.random.rand(n,n), -1) + np.eye(n))(200*n) for n in range(1, 6)]
upper_test_matrices = [(lambda n: np.triu(np.random.rand(n,n)))(200*n) for n in range(1, 6)]
```

Inside of our list comprehension we generate random numbers between 0 and 1 with np.random.rand(n,n) and insert them eiter into an upper or lower triangular matrix(np.tril / np.triu). With np.eye(n) we also add a identity matrix of dimension n to our lower test matricies, because we still need our ones on the diagonal.

Relative residuals

The provided plot shows that the residuals for both the upper triangular and lower triangular solver(that use forward and backward substitution respectively) solve the linear systems of equations in a rather precise manner. Both metrics place themselves in between 10^{-16} to 10^{-18} (for the solveL()-function) and 10^{-18} to 10^{-19} (for the solveU()-function) which means that our algorithm solves the test equations in a precise manner. It can be observed that our algorithm gets better the larger the problem sizes seem to get, that could be the result of multiple things, such as well conditioned test sets, numerical stability etc.

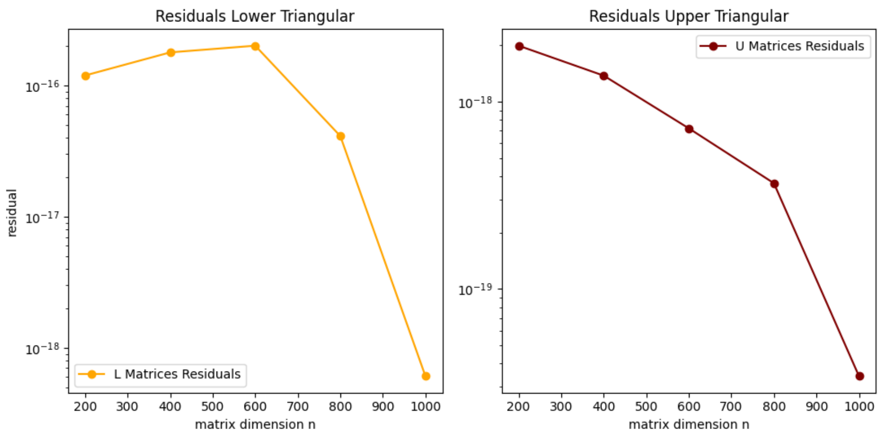

Figure 2: resulting residuals that occur on random generated test data

Part III - Numerical Accuracy of LU-Based Linear Solver

Relative residuals and forward error

In the following graph we can see, that both the LU factorization with and without pivoting(LU and PLU) perform a good result, shown by their respective residuals. It also becomes apparent that PLU always performes better by a magnitude than the variant without pivoting. What is also interesting is that the LU algorithm for the V matricies is a lot less accurate than when performing on the S matricies.

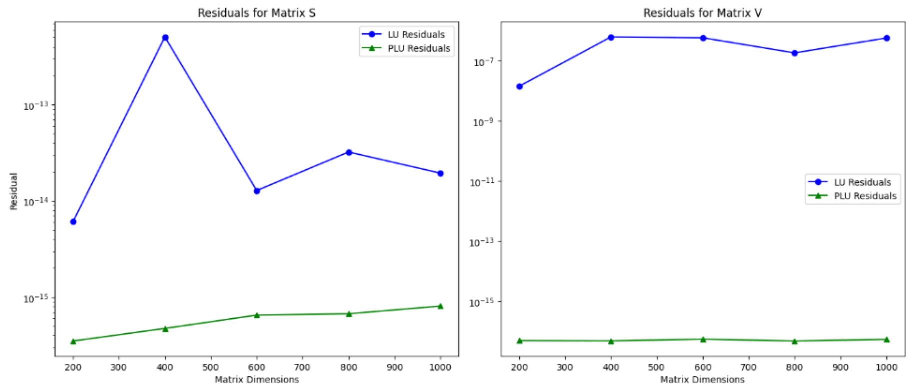

Figure 3: residuals that occur when solving the linear system equations

When looking at the respective forward errors of the operations, the contrast of dealing with the two test sets becomes even more apparent. When running our algorithms on the S matricies than we can see that PLU outperforms LU again - with the former moving at aroung 10^{-10} while the later moves at 10^{-12}. But when we apply those ontop of the V matrices we can see that we get a drastic forward error by both LU algorithms(relative errors between 10^{15} to 10^{18}).

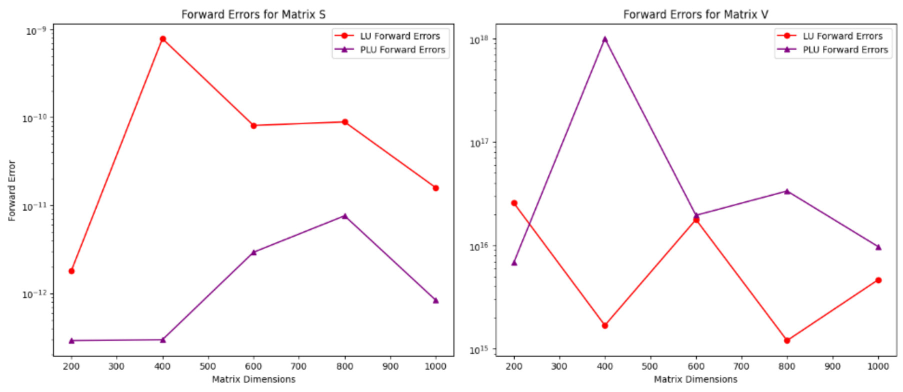

Figure 4: relative error that occurs while solving the linear system equations

Interpretation of the accuracy of the linear solver

By reducing the possibility for large multipliers during the elimination processes, PLU improves the numerical stability of our computations. PLU is also consistently outperforming our variant without partial pivoting by having a smaller residuals metrics when applied on the S matrix sets. This proves that partial pivoting improves our accuracy by minimizing round-off errors. On contrary, when applying said algorithm on the V matrices, PLU has similar performance than LU, possibly because of certain negative properties of the test set that reduce our effectiveness.

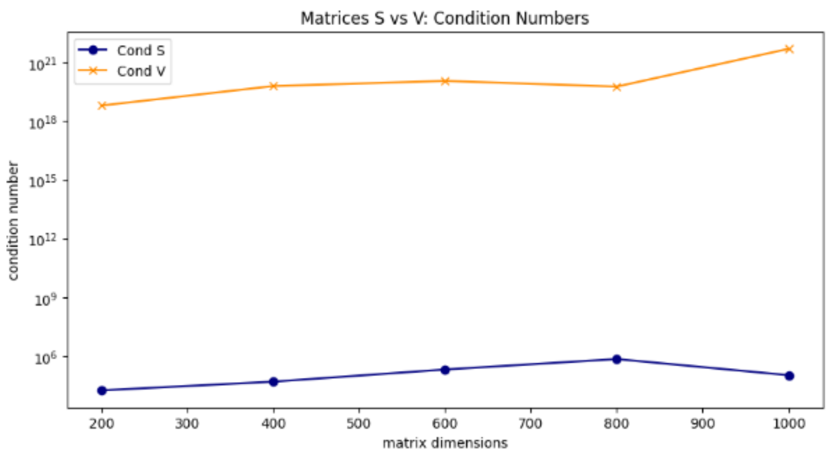

Figure 5: lower condition numbers give us numerical accuracy

The answer likely stems from the fact that S Matrices seem to be better conditioned than the V matrices. S Matrices display a drastically lower condition number, which indicates more accurate results for both LU and PLU. On the flip side, we can make out very large condition numbers beeing shown with V matrices, which make them ill-conditioned which makes them prone to akkumulating round-off errors.
In conclution, the plot proves us, why our accuracy seemingly drops drastically when applying out solve algorithms on certain matrices. Very large condition numbers indicate that numerical accuracy is being compromised.