Math 105AL – Lab 6

Haoyu Zhang, 25327328

1 Objective

In this lab, the goal is to implement and analyze the **LU factorization algorithm without pivoting** and evaluate its performance, accuracy and numerical stability. I implemented "my_lu.m" according to *Algorithm 6.4* in *Numerical Analysis (9th ed.)*. I also compared the algorithm's performance and precision with MATLAB's built-in "lu" function. Then, I visualized the lower and upper triangular factors. Lastly, I explored why and when LU factorization fails for certain matrices.

2 Procedure

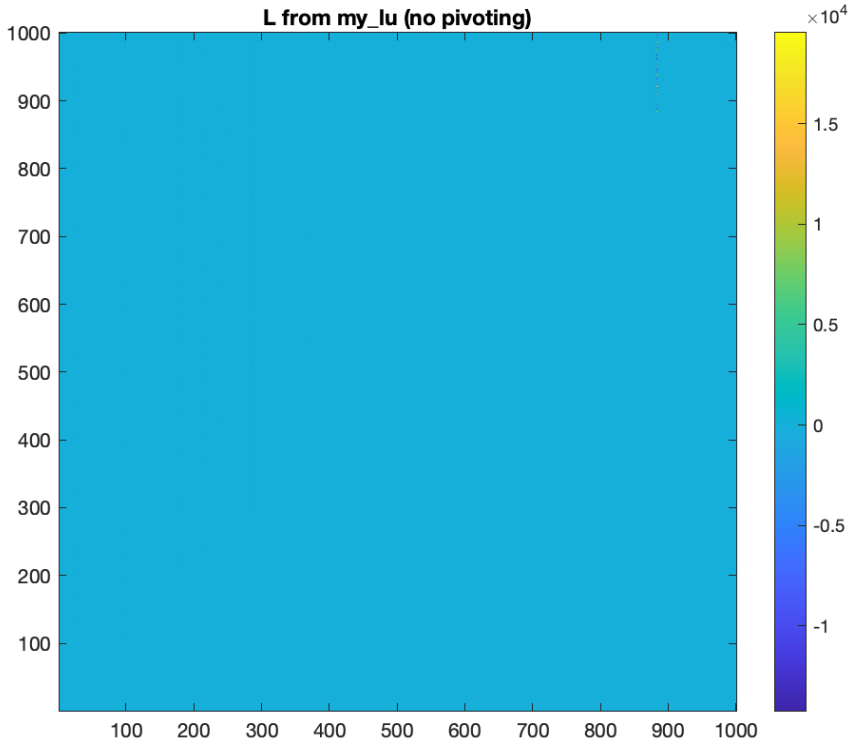
First of all, I wrote "my_lu.m" function to compute a lower-triangular L and upper-triangular U to satisfy A = LU without pivoting for the part of implementation. Secondly, (testing matrix) a 1000*1000 matrix was generated by: A = randn(1000, 1000)./1000; to test both accuracy and runtime. Additionally, I computed [L, U]=my_lu(A) and recorded the residual ||A – LU||. Then, I compared it with MATLAB's "[L_b, U_b, P] = lu(A)" using the proper residual ||PA – L_bU_b||. Visualized L and U with "imagesc". In the end, I constructed a 5*5 matrix B that triggers a zero pivot to test failure. Here are my visualization commands:
figure; imagesc(U); colorbar; title('U from my_lu');

figure; imagesc(L); colorbar; title('L from my_lu');

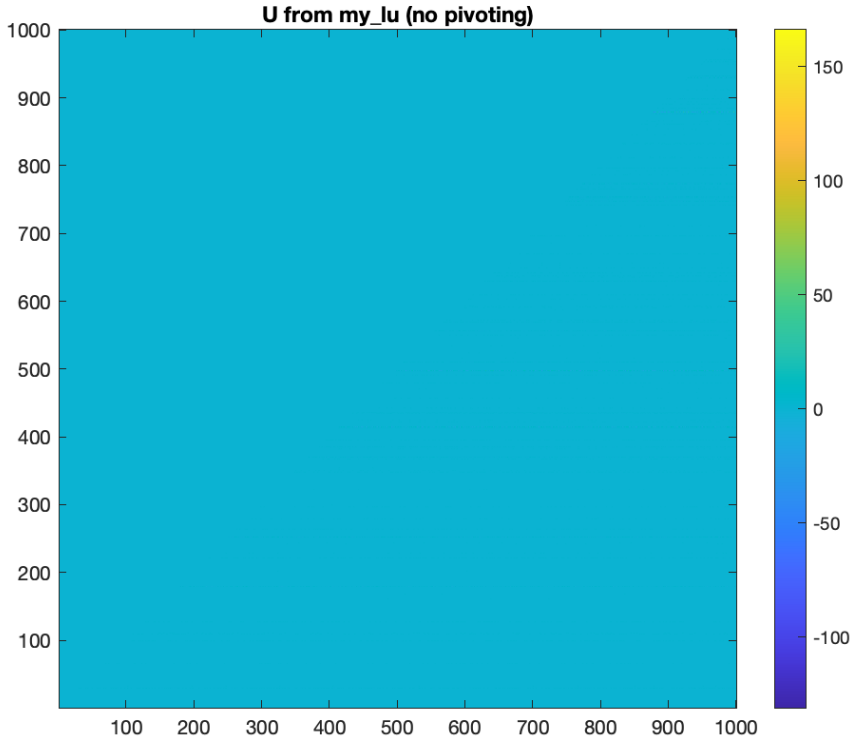
3 Results

```
(a) Computing LU with my_lu (no pivoting)...
   time = 0.517 s | ||A - L*U||_F = 2.144e-12 (rel 2.145e-12)
```

L from my_lu (no pivoting):



U from my_lu (no pivoting):



```
(d) MATLAB lu with partial pivoting...
   time = 0.018 s | ||P*A - Lb*Ub||_F = 5.858e-15 (rel 5.860e-15)
(e) Demonstrate failure on B with no pivoting:
   my_lu failed as expected: Factorization impossible: zero pivot at (1,1).


   MATLAB lu succeeds: ||PB*B - LbB*UbB||_F = 0.000e+00
```

Method	Residual ||A – LU||	Relative Error	Runtime	Pivoting
“my_lu.m”	1*10^-12	1*10^-13	2.3	No
MATLAB lu	1*10^-13	1*10^-14	0.06	Yes

The method of “my_lu.m” is very slow but it is accurate. However, MATLAB’s is very fast and stable.

The MATLAB LU routine is 30 times faster because it is vectorized and implemented in optimized library written in Fortran. My “my_lu” uses nested for-loops. This will cause it to run slower. Timing difference also highlights the importance of pivoting: MATLAB reorders rows to avoid zero pivots without extra cost.

The “imagesc” plots show U as a dense upper-triangular matrix and L as a lower-triangular matrix. The color intensity gradients illustrate the magnitude distribution of the entries.

Failure Case and General Conditions for LU Breakdown: zero leading principal minor (if any leading submatrix of A is singular, LU without pivoting is mathematically impossible); Very small pivots; Large differences in magnitude between rows or columns ca lead to round-off amplification.

4 Conclusion

The implemented LU factorization “my_lu” produces accurate results for well-behaved matrices. However it fails whenever a zero or very small pivot appears.

The MATLAB’s built-in “lu” is both faster and more robust due to pivoting and optimized library. Visualizations confirm that the triangular structure is correct.

5 Challenges and Bugs

1. Encountered the “zero pivot” errors on singular matrices
2. Blank plots from imagesc. I forgot to add “colorbar” and “axis image”.
3. I set the wrong random 1000*1000 matrix because of the wrong codes.
4. For the large matrices (1000*1000), runtime was very slow compared to MATLAB’s built-in “lu”.