# CENG 371 SCIENTIFIC COMPUTING

# HW2

Name: Andaç Berkay Seval

ID: 2235521

Q1)

The functions Sherman's March, Pickett's Charge are implemented. The implementation can be seen from the corresponding files "shermans.m" and "picketts.m".

The implementation of Sherman's March is pretty straightforward. I chose the partition in the matrix A, such that Ak is the first (n-1)x(n-1) portion of the matrix A, a1k is the (n-1)x1 portion of the matrix A for the last column, $(ak1)^T$ is the 1x(n-1) portion of the matrix for the last row and akk is the 1x1 last element of the matrix A. Therefore, L11 and U11 are (n-1)x(n-1) matrices, lk1 and u1k are (n-1)x1 vectors, ukk is a 1x1 element. Thus, I created L and U matrices with the corresponding values and places of L11, U11, lk1, u1k and ukk.

The implementation of Pickett's Charge is a little bit trickier. Since there are 2 block matrices Ak and Ak1 in the A, and Ak = L11*U11 and Ak1 = Lk1*U11, I thought the dimensions of matrices L11, Lk1 and U11 should be the same in order to be multipliable with each other. Thus, I partitioned Ak, Ak1 and L11,Lk1 such a way that they are interleaved, they have intersections. Thus, the dimension of Ak, Ak1, L11, Lk1 and U11 is (n-1)x(n-1). For example, Ak is the first (n-1)x(n-1) portion of the A, Ak1 is the (n-1)x(n-1) portion of the A for the rows and columns A(2:end, 1:end-1), L11 is the first (n-1)x(n-1) portion of the L, Lk1 is the (n-1)x(n-1) portion of the L for the rows and columns L(2:end, 1:end-1), U11 is the first (n-1)x(n-1) portion of the U. u1k is a vector of size (n-1)x1 for the last column of U. lkk is the vector of size (n-1)x1 for the rows and columns of L(2:end, end) and ukk is the last 1x1 element of U. Thus, I created L and U matrices with the corresponding values and places of L11, Lk1, lkk, U11, u1k, ukk.

Q2)

First of all, I did my homework on Octave Online. Thus, there are some constraints in terms of testing. I got errors like "PAYLOAD TOO LARGE!!!" and although I am pushing it to continue computing, it eventually terminate the program. Therefore, I could not test it with n = 1:300, instead, I could test it with n = 1:10. Unfortunately, it terminated program even n = 1:15.

a) Comparing in terms of run-time: Of course, run-time depends on the implementation. For my case, for the function:

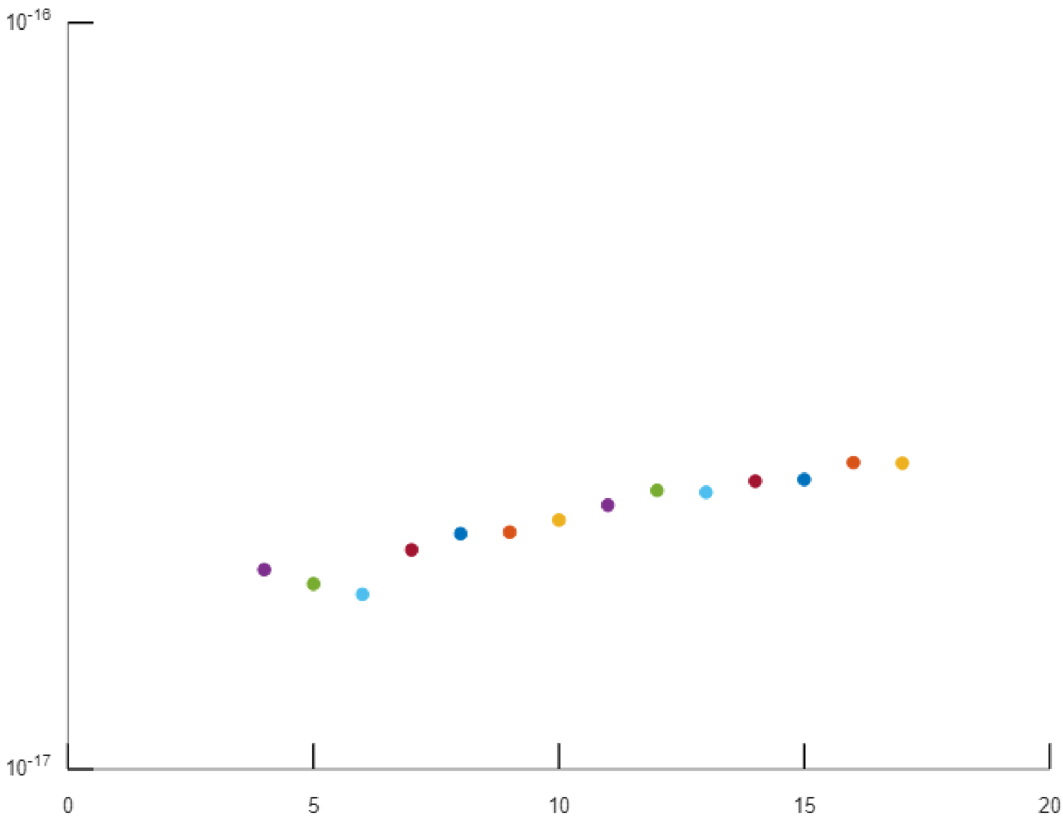
```
tic
for x = 1:10
        h = hilb(x)
        [L,U] = lu_decomposer(h)
end
toc
```

it gives 4.00696 seconds for Sherman's March and 4.3938 seconds for Pickett's Charge. In my Pickett's Charge implementation, matrix blocks have intersections and this might not be accurate to the expected result. However, in an ideal world, total-run time can be found with the function:

```
tic
for x = 1:300
h = hilb(x)
[L, U] = lu_decomposer(h)
end
toc
```

b) Comparing in terms of the plots of their relative errors:

*!!! PAYLOAD TOO LARGE !!!*




*NOTICE: Execution paused due to large payload*

Figure 1: Sherman's March relative error graph

*!!! PAYLOAD TOO LARGE !!!*

*NOTICE: Execution paused due to large payload*

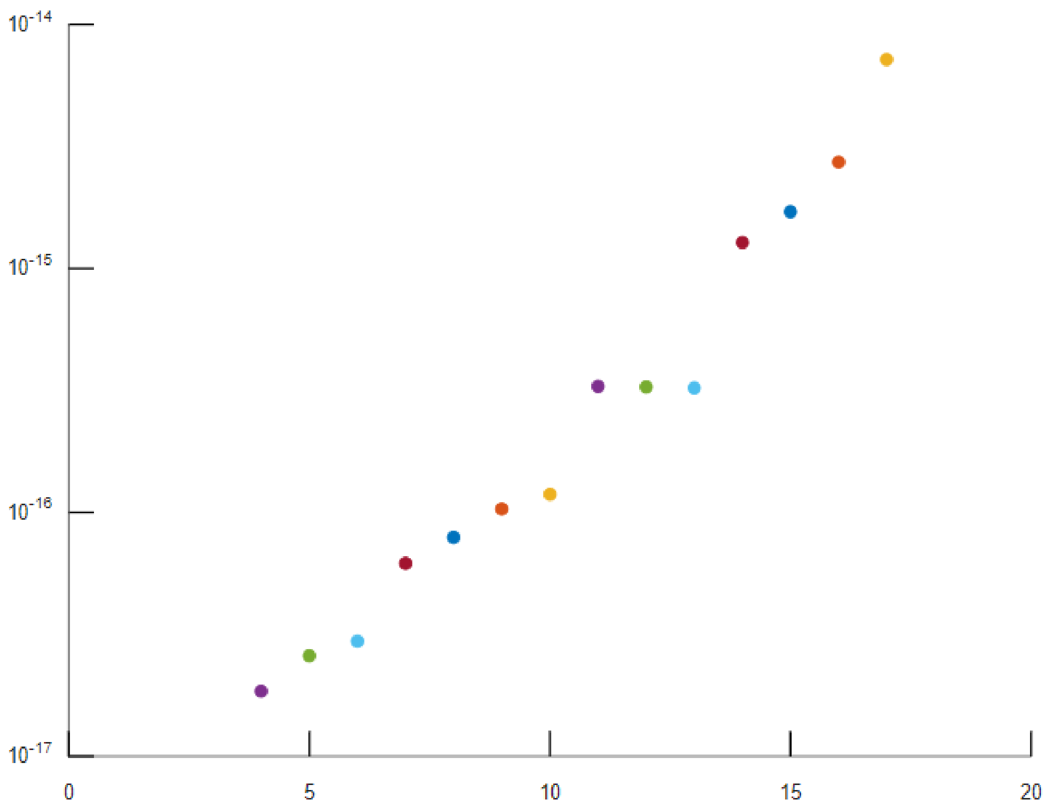



Figure 2: Pickett's Charge relative error graph

Because of Octave Online, I could not test it with n = 1:300, instead, I could test it with n = 1:20. Therefore, for my implementations, while n is increasing, the relative error of Pickett's Charge is increasing more than relative error of Sherman's March. I used the function for graphs:

```
hold on
for x = 1:20 (*)
    h = hilb(x)
    [L, U] = lu_decomposer(h)
    y = norm(h – L*U)/norm(h)
    semilogy(x, y)
end
```

However, in an ideal world, instead of the line (*), for x = 1:300 could be written.

c) No, it can not factorize any square matrix. If a matrix A cannot be reduced to row echelon form by a Gaussian elimination method without interchanging two rows, then the matrix A does not have a LU decomposition. Thus, the functions that I implemented could not factorize these matrices. However, for this matrices, there exists a permutation matrix P such that PA = LU has always a LU decomposition.