CENG 371 - Scientific Computing
Fall 2022
Homework 2

Şengül, Sena Nur
e2310498@ceng.metu.edu.tr

November 30, 2022

Q1)

I applied sherman's march and pickett's charge in Matlab. After I learnt we can use another programming language and I wrote crout's method in Python.

Sherman's march implementation:

```
Lk =

    1.0000         0         0         0
    0.5000    1.0000         0         0
    0.3333    1.0000    1.0000         0
    0.2500    0.9000    1.5000    1.0000


Uk =

    1.0000    0.5000    0.3333    0.2500
         0    0.0833    0.0833    0.0750
         0         0    0.0056    0.0083
         0         0         0    0.0004
```

Pickett's charge implementation:

```
L =

    1.0000         0         0
   -0.3000    1.0000         0
    0.5000   -0.2500    1.0000


U =

   10.0000   -7.0000         0
         0  -10.0000    6.0000
         0         0    0.1538
```

Crout’s method:
L=

```
[[1.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [5.00000000e-01 8.33333333e-02 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.33333333e-01 8.33333333e-02 5.55555556e-03 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [2.50000000e-01 7.50000000e-02 8.33333333e-03 3.57142857e-04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [2.00000000e-01 6.66666667e-02 9.52380952e-03 7.14285714e-04
  2.26757370e-05 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.66666667e-01 5.95238095e-02 9.92063492e-03 9.92063492e-04
  5.66893424e-05 1.43154905e-06 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.42857143e-01 5.35714286e-02 9.92063492e-03 1.19047619e-03
  9.27643785e-05 4.29464715e-06 9.00974924e-08 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.25000000e-01 4.86111111e-02 9.72222222e-03 1.32575758e-03
  1.26262626e-04 8.09375809e-06 3.15341224e-07 5.65997062e-09
  0.00000000e+00 0.00000000e+00]
 [1.11111111e-01 4.44444444e-02 9.42760943e-03 1.41414141e-03
  1.55400155e-04 1.23333457e-05 6.72727944e-07 2.26398825e-08
  3.55135948e-10 0.00000000e+00]
 [1.00000000e-01 4.09090909e-02 9.09090909e-03 1.46853147e-03
  1.79820180e-04 1.66500166e-05 1.13522841e-06 5.39361907e-08
  1.59811125e-09 2.22661056e-11]]
```

U=

```
[[ 1.          0.5         0.33333333  0.25        0.2         0.16666667
   0.14285714  0.125       0.11111111  0.1       ]
 [ 0.          1.          1.          0.9         0.8         0.71428571
   0.64285714  0.58333333  0.53333333  0.49090909]
 [ 0.          0.          1.          1.5         1.71428571  1.78571429
   1.78571429  1.75        1.6969697   1.63636364]
 [ 0.          0.          0.          1.          2.          2.77777778
   3.33333333  3.71212121  3.95959596  4.11188811]
 [ 0.          0.          0.          0.          1.          2.5
   4.09090909  5.56818182  6.85314685  7.93006993]
 [ 0.          0.          0.          0.          0.          1.
   3.          5.65384615  8.61538462 11.63076923]
 [ 0.          0.          0.          0.          0.          0.
   1.          3.5         7.46666668 12.60000002]
 [ 0.          0.          0.          0.          0.          0.
   0.          1.          3.99999993  9.52941166]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          1.          4.49999584]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          0.          1.        ]]
```

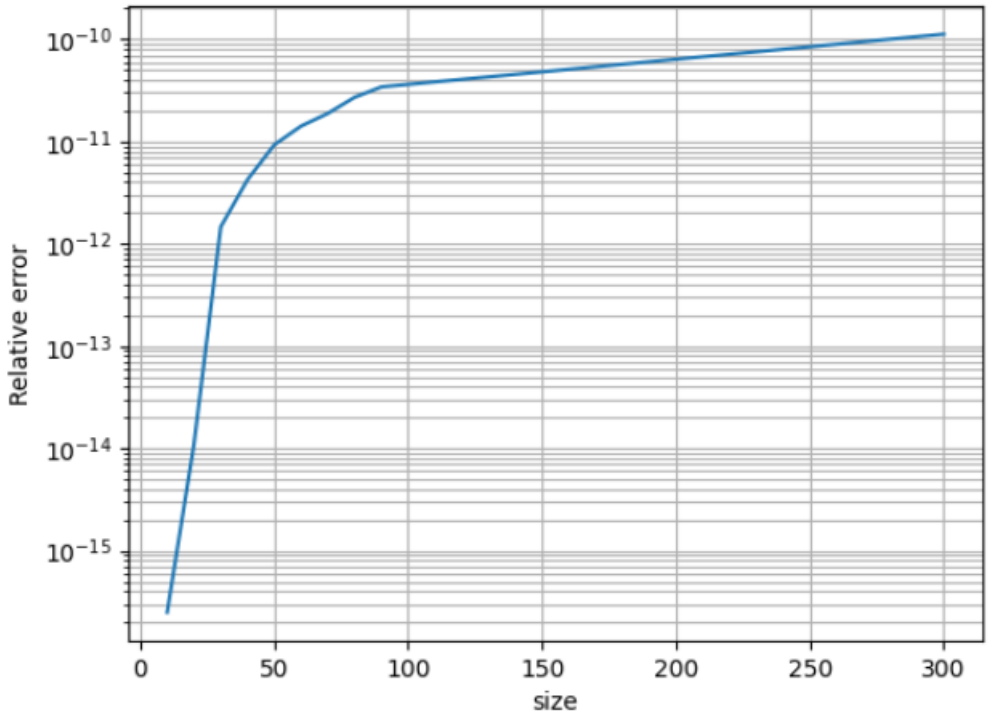
Q2)

Sherman’s elapsed time = Elapsed time is 0.001262 seconds.

Pickett’s Elapsed time = is 0.001045 seconds.

Crout’s= Elapsed time is 0.003138

I found elapsed time with tic/toc in matlab and also time.time() in Python. Crout's method lowest speed and requires forward and backward substituiton and extra memory space. Also in my implementation I used three helper in crouts method for recursion. Maybe this effects run time.
On the other hand, Pickett's charge has small run time.This implementation assumes L11,U11 ,LK1 already computed. Algorithm sweeps across the entire matrix. Sherman's march is arithmetically identical ro Gaussian Elimination,exactly same operations on each element but this algorithm does not allow pivoting for size.



I used plt.semilogy for this graph. Other graphics are similar to that.

n	picketts error	shermans error
10.0000	250.1822e-18	299.2821e-18
50.0000	9.2927e-12	6.0477e-12
100.0000	42.6198e-12	44.5731e-12
150.0000	76.3326e-12	102.3674e-12
200.0000	94.8622e-12	136.6416e-12
250.0000	103.9173e-12	170.6485e-12
300.0000	112.0069e-12	211.0843e-12

I found those with applying relative function which uses np.norm(x,ord=2) and np.dot because this formula gives relative error:

$$\frac{\|\acute{A}_n - \bar{L}_n U_n\|_2}{\|A_n\|_2}.$$

Q2-B)

RuntimeWarning: divide by zero encountered in double_scalars
b[i][j] = (1 / L[i][i]) * (A[i][j] - sum)

In Crout's method, When I give input hilberts(100)(big size) I have warning and Nan values. Lii is sometimes equal the zero. But I dont see any problem in Sherman's march and Pickett's charge. They calculated correctly.