CENG 371

Scientific Computing

Homework 2

Bartu Kılıçkaya 2380640 / CENG

1)

a) Call Sherman with sherman(“Matrix”,0,”matrix_size”) if needed.

b) Call picketts with pickets(“Matrix”,-1,”matrix_size”,0,0) if needed.

c) Call crout with crout(“matrix”,-1,”matrix_size”,0,0) if needed.

You can print upper and lower decompositions either in main or in function itself.

2)

a)

These are the time and error values of each function starting with 1 and increasing 25(i+=25) in every iteration up to 300. You can change i values in main.

```
i =  1  Sherman Error:  1.0 e-12 Sherman Time :  0.00010429999999999461

i =  1  Pickett's Error:  0.0 Pickett's Time :  0.00011349999999998861

i =  1  Crout Error:  0.0 Crout Time :  6.739999999999524e-05 e-3

i =  26  Sherman Error:  1.0 e-12 Sherman Time :  0.012068399999999998

i =  26  Pickett's Error:  2.437453323701953e-16 Pickett's Time :  0.005343699999999951

i =  26  Crout Error:  0.9999675001554308 Crout Time :  0.00010820000000000274e-3

i =  51  Sherman Error:  1.0 e-12 Sherman Time :  0.050866199999999997

i =  51  Pickett's Error:  4.3955097264795864e-16 Pickett's Time :  0.0267926

i =  51  Crout Error:  0.9999973366487959 Crout Time :  0.00012220000000001674e-3

i =  76  Sherman Error:  1.0 e-12 Sherman Time :  0.19046949999999996
```

```
i =  76  Pickett's Error:  4.774855619335335e-16 Pickett's Time :  0.057030000000000025

i =  76  Crout Error:  0.9999989514631081 Crout Time :  0.00015750000000003261e-3

i =  101  Sherman Error:  1.0 e-12 Sherman Time :  0.24658409999999997

i =  101  Pickett's Error:  8.122199069312421e-16 Pickett's Time :  0.2163155

i =  101  Crout Error:  0.9999993848580608 Crout Time :  0.00024129999999988883e-3

i =  126  Sherman Error:  1.0 e-12 Sherman Time :  0.6032668999999999

i =  126  Pickett's Error:  1.0879135587751078e-15 Pickett's Time :  0.3082746999999999

i =  126  Crout Error:  0.9999995689201212 Crout Time :  0.00029769999999995633e-3

i =  151  Sherman Error:  1.0 e-12 Sherman Time :  0.8763620999999997

i =  151  Pickett's Error:  1.3395398289842662e-15 Pickett's Time :  0.5058286000000001

i =  151  Crout Error:  0.9999996667068568 Crout Time :  0.00040149999999997215e-3

i =  176  Sherman Error:  1.0 e-12 Sherman Time :  1.3448049000000002

i =  176  Pickett's Error:  1.5173033284303133e-15 Pickett's Time :  0.5652156999999995

i =  176  Crout Error:  0.9999997262565985 Crout Time :  0.00055029999999995595e-3

i =  201  Sherman Error:  1.0 e-12 Sherman Time :  3.300882200000001

i =  201  Pickett's Error:  1.559213809234935e-15 Pickett's Time :  1.2206311999999997

i =  201  Crout Error:  0.9999997660485218 Crout Time :  0.00068390000000002926e-3

i =  226  Sherman Error:  1.0 e-12 Sherman Time :  2.273875200000001

i =  226  Pickett's Error:  2.349272571380946e-15 Pickett's Time :  1.3200403000000005

i =  226  Crout Error:  0.9999997944933823 Crout Time :  0.00100909999999992079e-3

i =  251  Sherman Error:  1.0 e-12 Sherman Time :  2.7576555999999997

i =  251  Pickett's Error:  2.4122297070341676e-15 Pickett's Time :  1.684515900000001

i =  251  Crout Error:  0.9999998159277449 Crout Time :  0.00105260000000012914e-3
```

```
i =  276  Sherman Error:  1.0 e-12 Sherman Time :  3.1577146999999997

i =  276  Pickett's Error:  2.77531112504714e-15 Pickett's Time :  1.9817628999999997

i =  276  Crout Error:  0.9999998328357835 Crout Time :  0.0012852999999992676e-3


Process finished with exit code 0
```

Starting with errors, Sherman errors are always computed to one which is the worst one between each other. Crout is almost one which is really bad since it is so close to one. However; pickett's relative error is really low which is approximately zero, so best algorithm in terms of relative error average is by far pickett's algorithm.

For runtime comparison, as the size of the matrix increased, the elapsed time in all three of the algorithms increased as expected. However, the best algorithm in terms of efficiency is by far Crout's algorithm. The worst one is Sherman.

b)

All three of the algorithms worked up to n=300, but in python environment, I couldn't display every single element in each lower and upper matrices especially for large n values. All methods was recursively implemented and returned the Lower and Upper pairs correctly. To sum up, we managed to factorize all square matrices up to n successfully.