5633A - Numerical Methods for High-performance Computing

Programming Assignment 4

Min Zhang

QUESTION

For this assignment, we are going to explore Gaussian elimination and fixed-point iterations. Please implement the function

$$[\mathtt{L}, \mathtt{U}, \mathtt{P}] = \mathtt{mylu(A)},$$

which should execute Gaussian elimination with partial pivoting as described in the lecture. For the following, develop MATLAB scripts to carry out the requested tasks.

1. Recall that even LU with partial pivoting can behave poorly for certain pathological matrices, such as ones with the structure

$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

with the behavior becoming more pronounced as the dimension n gets larger. For each $n = 5, 10, 20, 30, 40, 50, 60$, perform the following experiments:

- Generate matrices like the one above but replace the -1's with random uniform negative numbers in $[-1, 0]$ using `-rand()`. For each n generate 1000 such matrices, compute the pivoted LU factorization, and compute the growth factor.

Answer:

The part result of growth factor are shown in this picture:

growth_factor

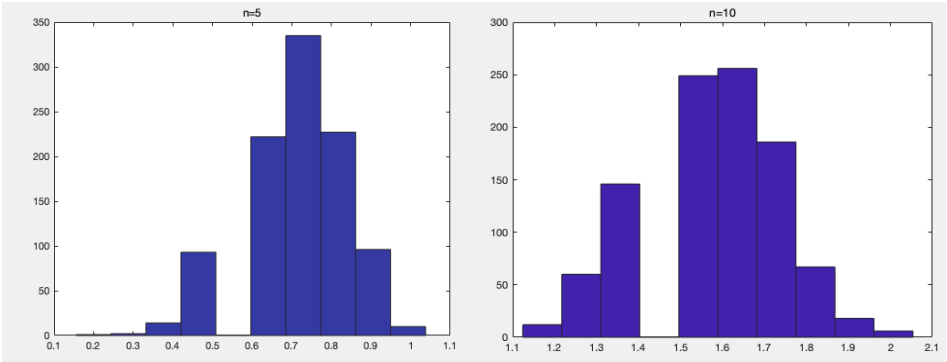
7x1000 double

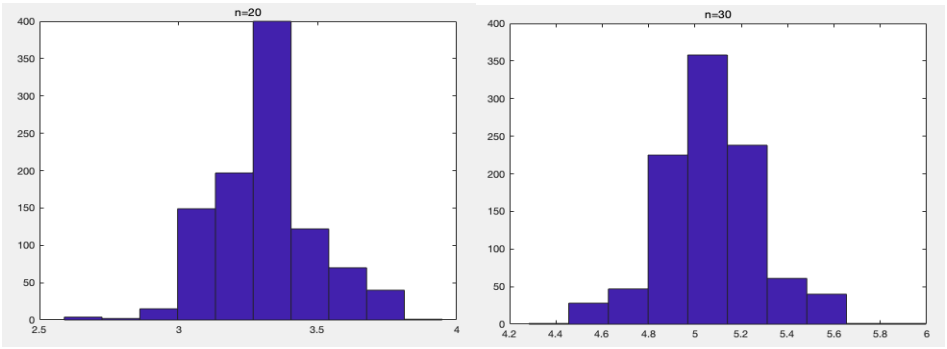
	1	2	3	4	5	6	7	8
1	5.7402	4.0589	3.8937	3.5243	6.8815	5.5093	4.8082	7.2103
2	20.3918	41.8757	32.8616	64.9571	41.6600	50.0141	21.4797	41.3670
3	3.2645e...	1.8220e...	2.1864e...	1.9553e...	2.0090e...	3.3849e...	2.2374e...	1.6138e...
4	9.0752e...	3.4303e...	1.3784e...	8.2283e...	5.7946e...	2.5515e...	1.5947e...	7.0791e...
5	1.4193e...	1.1231e...	2.9614e...	1.5185e...	9.7215e...	1.2509e...	6.8438e...	6.4248e...
6	2.5642e...	7.9803e...	5.4162e...	1.5182e...	3.4580e...	7.8547e...	1.1179e...	1.6965e...
7	4.7355e...	1.4460e...	4.4357e...	4.6601e...	3.3026e...	1.7339e...	1.6959e...	2.3714e...

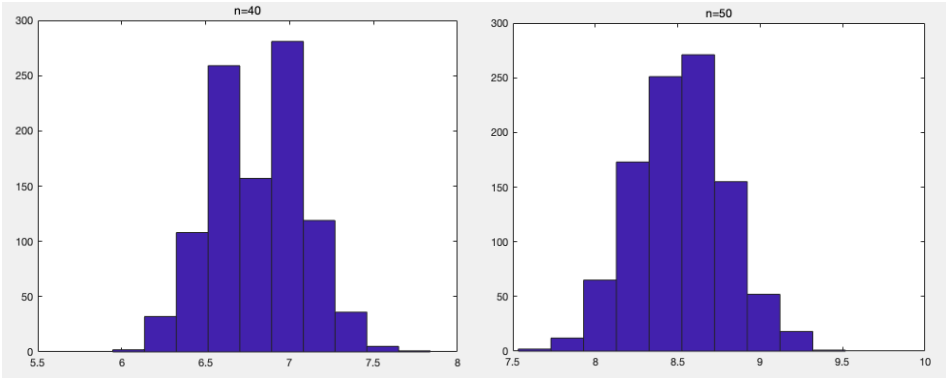
- For each n, display a histogram of the orders of magnitude of the growth factors. You can use `round(log10(growthRate),1)`.

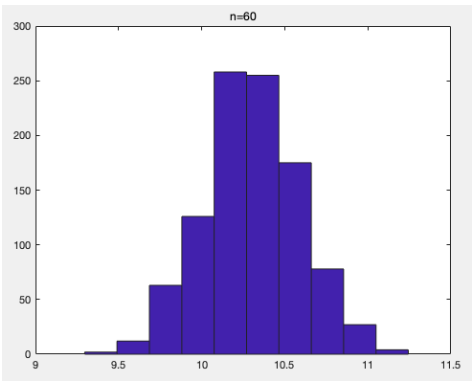
Answer:

The result pictures of histogram of orders magnitude of growth factor for n=5, 10, 20, 30, 40, 50, 60 are shown in there:





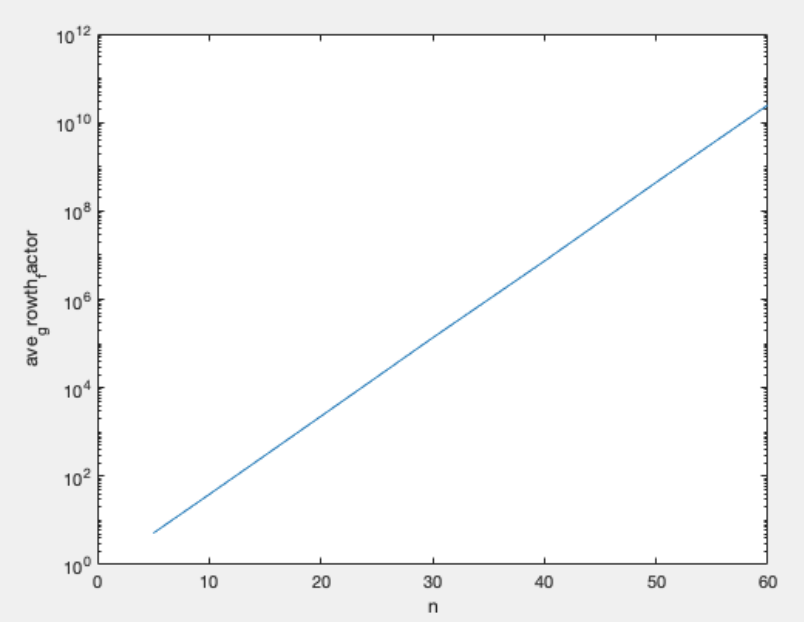




- For each n, compute the average growth rate from your experiments and plot them against n using `semilogy()`. Interpret what you see in this figure.

Answer:

The result pictures are shown in there:



2. Use the included file to generate the finite difference matrix approximation of the three-dimensional Laplacian operator, `L=hw4_3d2ndDeriv(m)`. Using the other attached file, extract the tridiagonal of this matrix, `M=tridiag(L)`. Consider the fixed-point iteration induced by the splitting $L = M - N$.

   - Implement this fixed-point iteration but without computing N. *Hint*: consider the Richardson iteration version of the FPI. In your implementation, you should precompute the stable partially-pivoted LU factorization of M and use it to apply M^{-1} at each iteration.
   - What is the upper bound on the convergence rate? *Note*: use `eig(full(B))` to get the eigenvalues of a sparse matrix B.
   - For $\vec{x}_0 = \vec{0}$ and $\vec{b} = \vec{1}$, solve $L\vec{x} = \vec{b}$ using this FPI, to a relative residual tolerance of 10^{-8}. Using `semilogy()`, plot the relative residual at each iteration. How does this rate compare with the convergence rate bound?

Answer:

The approximately solution of this question is:

x =

-0.077427455344716
-0.092354910701879
-0.077427455351435
-0.092354910702752
-0.111328124989157
-0.092354910710212
-0.077427455352542
-0.092354910710868
-0.077427455357263
-0.092354910703981
-0.111328124990602
-0.092354910711170
-0.111328124991630
-0.135602678566023
-0.111328125001715
-0.092354910712843
-0.111328125003068
-0.092354910722150
-0.077427455354216
-0.092354910713278
-0.077427455359346
-0.092354910714024
-0.111328125005213
-0.092354910724445
-0.077427455361090

-0.092354910726246
-0.077427455373824
-0.092354910705916
-0.111328124993060
-0.092354910712976
-0.111328124994086
-0.135602678569706
-0.111328125005057
-0.092354910714826
-0.111328125006681
-0.092354910725974
-0.111328124995764
-0.135602678572340
-0.111328125007563
-0.135602678574289
-0.166852678588858
-0.135602678600268
-0.111328125012566
-0.135602678605592
-0.111328125046041
-0.092354910718886
-0.111328125014442
-0.092354910734321
-0.111328125017027
-0.135602678615015
-0.111328125056901
-0.092354910742010
-0.111328125065431
-0.092354910795712
-0.077427455358138
-0.092354910719940
-0.077427455365779
-0.092354910721090
-0.111328125018681
-0.092354910738700
-0.077427455368803
-0.092354910741907
-0.077427455390986
-0.092354910723294
-0.111328125023140
-0.092354910743662
-0.111328125026597
-0.135602678634915
-0.111328125079254

-0.092354910753959
-0.111328125090516
-0.092354910823863
-0.077427455376593
-0.092354910757937
-0.077427455408589
-0.092354910763375
-0.111328125110423
-0.092354910846354
-0.077427455424786
-0.092354910863838
-0.077427455533041

The upper Bound Error Convergence and Error shows as below: