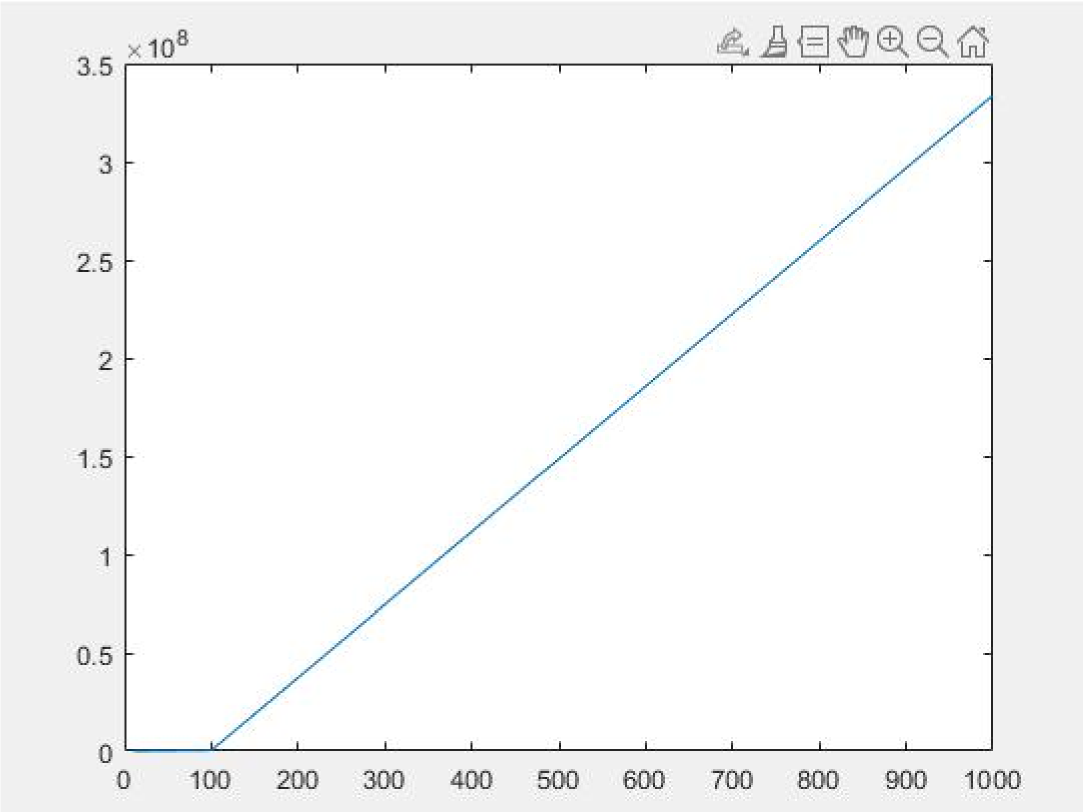
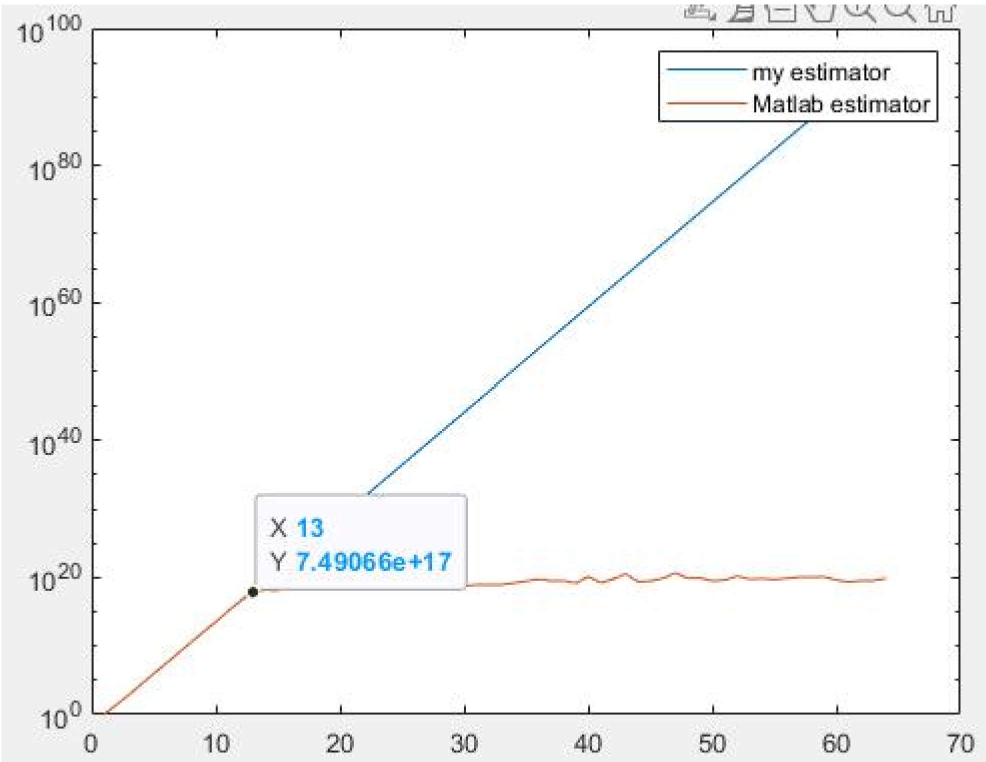
Assignment 3
CS 3200
Robert Li
u1212360

Q1



By calculate the flops in 10 , 100, 1000, I find out after 100 flops to 1000 flops, the time complexity become stable for O(n^2) after we elimination, I will say the it performs the backsolves which we have saw in the lecture slide. I think before 100 flops will occur some errors since it look like not performing O(n^2) , I will assume in some some complex matrix, more flops will eventually lead us to the right result, less flops sometime will not enough for calculations, more times of flops will make the result more precise. I have translate the pseudo-code in to a gauss elimination function to calculate the flops, I replace all the computation with addition since the time complexity between them are the same.

Q2



Compare the 2 estimator, the Matlab's one is good until x = 13, after this point, it stop increasing, and stop the value around 10^20 in my graph.

I have used `max(sum(abs(matrix)));` for both hilb(n) and its inverse, times them up to create my own estimator.

Q3

cnum

617.747646572987	2276.52488810499	8706.13980201778	34010.830942584
4083703.98034936	16221090.4706399	64786994.6114976	265847188.315949
40835812115.6787	162210413191.071	647856686880.122	2658418738016.45
401937015343322	1.53042740576085e+15	6.47856685060422e+15	2.65841873270276e+16
4.49515485037293e+16	1.41785994900117e+17	6.47529757904441e+17	2.65841873270184e+18

const

0.0161878398978546	0.00439266008127156	0.00114861468198367	0.000294023983621032
0.0244875731642638	0.00616481365300315	0.00154351966161822	0.000376155943696323
0.0244883093586391	0.00616483233306409	0.00154355125176787	0.000376163463528
0.0248795199702081	0.00653412240421067	0.001543551256103 4	0.00037616346427988
0.022246174676653	0.00705288276676737	0.00154433056982004	0.00037616346428001

The nor stands for original list and norr is the new list.

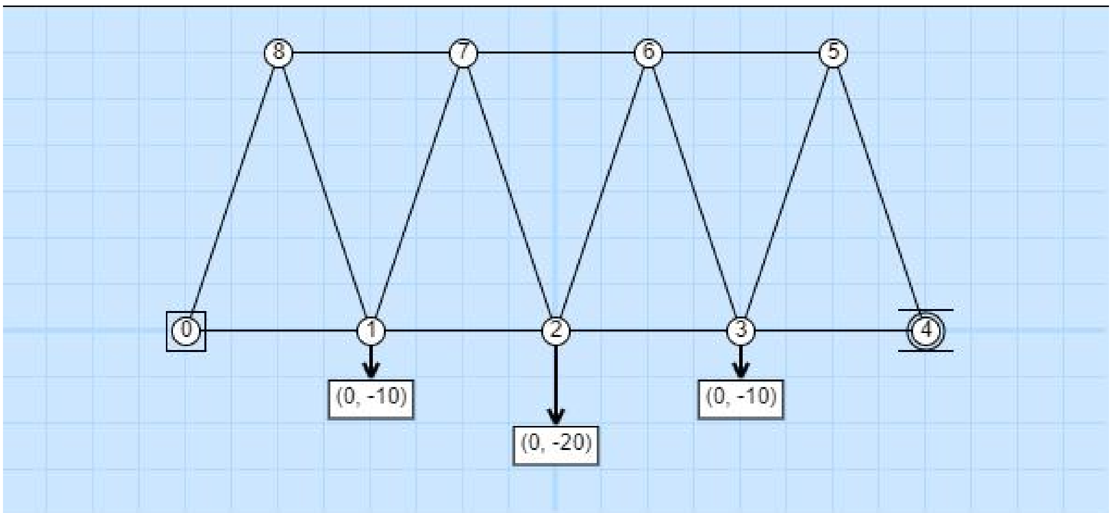
nor(:,:,1)		nor(:,:,1)
1.0e-13 *		
0.079936057773011	0.159872115546023	[7.3468396926392969248046033576 39e-39, 1.4693679385278593849609206715278e-38]
0.044408920985006	0.097699626167014	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.035527136788005	0.071054273576010	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.035527136788005	0.071054273576010	[8.8162076311671563097655240291668e-39, 1.7632415262334312619531048058334e-38]
0.026645352591004	0.062172489379009	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
nor(:,:,2)		norr(:,:,2)
1.0e-13 *		
0.062172489379009	0.124344978758018	[7.3468396926392969248046033576 39e-39, 1.4693679385278593849609206715278e-38]
0.044408920985006	0.097699626167014	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.035527136788005	0.071054273576010	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.035527136788005	0.071054273576010	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.044408920985006	0.097699626167014	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
nor(:,:,3)		norr(:,:,3)
1.0e-13 *		
0.071054273576010	0.150990331349021	[7.3468396926392969248046033576 39e-39, 1.4693679385278593849609206715278e-38]
0.044408920985006	0.097699626167014	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.044408920985006	0.097699626167014	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.035527136788005	0.071054273576010	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.044408920985006	0.097699626167014	[8.8162076311671563097655240291668e-39, 2.0571151139390031389452889401389e-38]
nor(:,:,4)		norr(:,:,4)
1.0e-13 *		
0.097699626167014	0.186517468137026	[7.3468396926392969248046033576 39e-39, 1.4693679385278593849609206715278e-38]
0.079936057773011	0.168753899743024	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.071054273576010	0.142108547152020	[5.8774717541114375398436826861112e-39, 1.1754943508222875079687365372222e-38]
0.071054273576010	0.142108547152020	[8.8162076311671563097655240291668e-39, 1.7632415262334312619531048058334e-38]
0.071054273576010	0.142108547152020	[8.8162076311671563097655240291668e-39, 2.0571151139390031389452889401389e-38]

I performed the table as matrix form to compare them up, with the algorithm from class, I got the condition number and constants with vpa form, and reduced the norm of residual when k<=2 to compare the first iterative refinement and other one with vpa.

Q4

```
x = 18×1
    6.6596
   16.6491
   16.6491
    6.6596
  -21.0748
  -13.3193
  -19.9789
  -13.3193
  -21.0748
   21.0748
```

The result has pretty high rate of precision, however, still contain some errors compare to solution.I think it because the potential growth and the same value in A and B will need larger digit for more precise. The error will get larger after more iterations, as well as the condition number, therefore, the accuracy will be low with small error in larger iteration computing.



The challenge in the real- world will become complex, the first of all is the bridge is 3D in the real world, we need different angle of view the building the bridge. Second, we need to consider more about the force acting on the “nodes” and “member”, which include the physics knowledge. And do not even mention the affect of materials for the bridge. The equation will be at least be double or more, since 1 more dimension in real-world.