Project 1

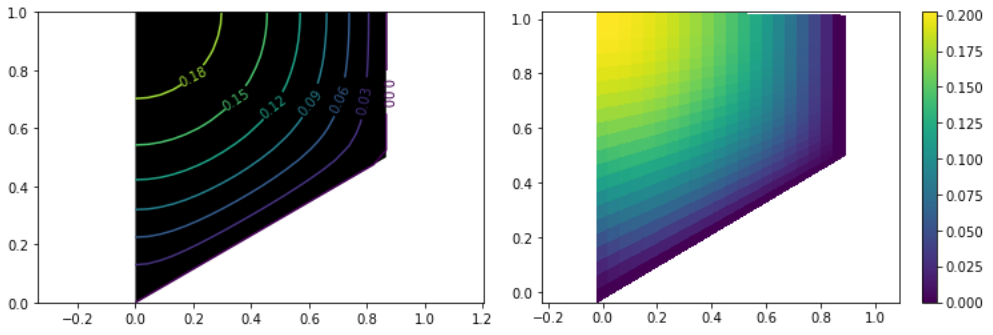
Wenhao He

Problem 1:

1-3. see atachment derivation1-3.pdf

4. The result is shown below, and the corresponding code is attached as 1-4.ipynb.

Q = 0.129



We can see that our solution agrees well with the reference one.

5. the overall strategy is to transform the domain into a unit square, solve equation in the unit square and transform back

- Find out a transformation $T: (x, y) \to (\xi, \eta)$ such that in (ξ, η) space, we have a unit square domain
- According to the transformation, determin the corresponding function in the new domain, i.e. determine coefecients in this equation:

$$\boxed{\frac{-1}{J^2}\left(a\, u_{\xi\xi} - 2b\, u_{\xi\eta} + c\, u_{\eta\eta} + d\, u_{\xi} + e\, u_{\eta}\right) = f} \quad ,$$

$$a = x_\eta^2 + y_\eta^2 \qquad d = \frac{x_\eta \beta - y_\eta \alpha}{J} \qquad \alpha = a x_{\xi\xi} - 2b x_{\xi\eta} + c x_{\eta\eta}$$

$$b = x_\xi x_\eta + y_\xi y_\eta \qquad e = \frac{y_\xi \alpha - x_\xi \beta}{J} \qquad \beta = a y_{\xi\xi} - 2b y_{\xi\eta} + c y_{\eta\eta}$$

$$c = x_\xi^2 + y_\xi^2$$

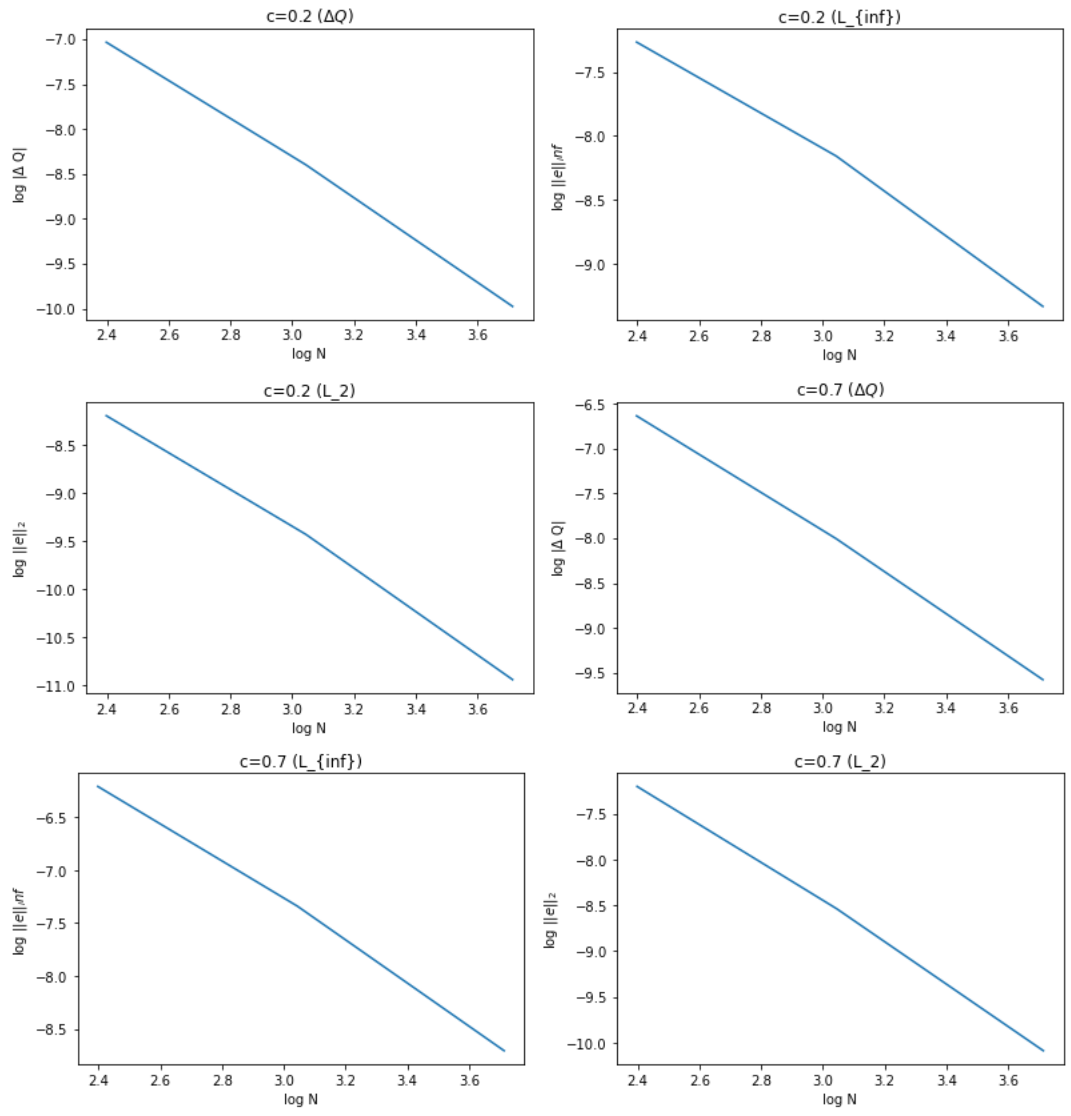
- Determine boundary condition in the new domain, by

$$\frac{\partial u}{\partial n} = u_x n^x + u_y n^y = \frac{1}{J}\left[\left(y_\eta n^x - x_\eta n^y\right) u_\xi + \left(-y_\xi n^x + x_\xi n^y\right) u_\eta\right]$$

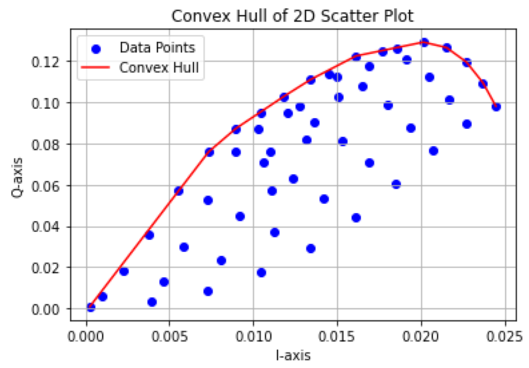
- Solve the equaiton in the new domain.

- Transform back to original domain $T^{-1}: (\xi, \eta) \rightarrow (x, y)$

6. The result is shown below, and the corresponding code is attached as 1-6.ipynb. We can see that the lines are straght, and slope are all approximately -2, so second order convergence rate is verified.



7. The result is shown below, and the corresponding code is attached as 1-7.ipynb.



About the Pareto front, we can see that when I increases, Q increases first and decreases then, the maximum Q corresponds to I = 0.2, and corresponding h and c is 1 and 0.5 respectively.

Problem 2

1.

Jacobi iteration scheme:

$$\phi_{i,j}^{t+1} = \frac{1}{4}\left(\phi_{i,j-1}^{t} + \phi_{i,j+1}^{t} + \phi_{i-1,j}^{t} + \phi_{i+1,j}^{t}\right) + \frac{1}{4} f_{ij}\Delta x^2$$

for internal points, and

$$\phi_{0,j}^{t} = \phi_{N-1,j}^{t} = \phi_{i,0}^{t} = \phi_{i,N-1}^{t} = 0$$

For boundary points

G-Siteration scheme:

$$\phi_{i,j}^{t+1} = \frac{1}{4}\left(\phi_{i,j-1}^{t+1} + \phi_{i,j+1}^{t} + \phi_{i-1,j}^{t+1} + \phi_{i+1,j}^{t}\right) + \frac{1}{4} f_{ij}\Delta x\Delta y$$

for internal points, and

$$\phi_{0,j}^{t} = \phi_{N-1,j}^{t} = \phi_{i,0}^{t} = \phi_{i,N-1}^{t} = 0$$

For boundary points.

In this scheme, we must iterate i and j in ascend order.

2.a

for relaxation scheme, we have

Jacobi iteration scheme:

$$\phi_{i,j}^{t+1} = \omega\left(\frac{1}{4}\left(\phi_{i,j-1}^{t} + \phi_{i,j+1}^{t} + \phi_{i-1,j}^{t} + \phi_{i+1,j}^{t}\right) + \frac{1}{4} f_{ij}\Delta x^2\right) + (1-\omega)\phi_{i,j}^{t}$$

for internal points, and

$$\phi_{0,j}^{t} = \phi_{N-1,j}^{t} = \phi_{i,0}^{t} = \phi_{i,N-1}^{t} = 0$$

For boundary points

G-Siteration scheme:

$$\phi_{i,j}^{t+1} = \omega(\frac{1}{4}\left(\phi_{i,j-1}^{t+1} + \phi_{i,j+1}^{t} + \phi_{i-1,j}^{t+1} + \phi_{i+1,j}^{t}\right) + \frac{1}{4} f_{ij}\Delta x \Delta y) + (1-\omega)\phi_{i,j}^{t}$$

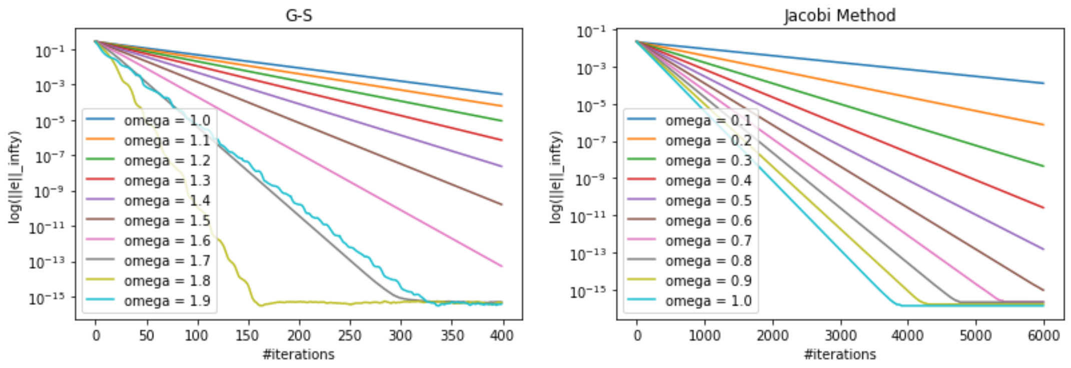
for internal points, and

$$\phi_{0,j}^{t} = \phi_{N-1,j}^{t} = \phi_{i,0}^{t} = \phi_{i,N-1}^{t} = 0$$

For boundary points.

In this scheme, we must iterate i and j in ascend order.

2.b. The result is shown below, and the corresponding code is attached as 2-2b.ipynb. We calculate the error by comparing our solution with the exact one



We cans see that for G-S method, the best omega is 1.8$\pm$0.1

2.c The result is shown below, and the corresponding code is attached as 2-2c.ipynb. We can see that our result fits very well with the reference one.

```
Gradient on the left boundary (j = 0):
[-0.         0.01218417  0.02426034  0.03587945  0.04632985  0.05488432
  0.06083544  0.06393579  0.0644587   0.06312989  0.0606809   0.05780875
  0.05482042  0.05172878  0.04842216  0.04478362  0.04074973  0.03632396
  0.03156206  0.02654637  0.02136069  0.01607405  0.01073474  0.00537173
 -0.        ]

Gradient on the right boundary (j = L):
[-0.         0.00710492  0.01438734  0.02201717  0.03013939  0.03882822
  0.04799486  0.05724959  0.06583454  0.07299798  0.07804144  0.08076095
  0.08149332  0.08100438  0.07994885  0.07868347  0.07682427  0.07368601
 0.06846581  0.06077975  0.05077159  0.03906413  0.02631866  0.01319513
-0.        ]


Gradient on the bottom boundary (i = 0):
[-0.          0.01217454  0.02422181  0.03577382  0.04610126  0.05446274
 0.06014409  0.06290426  0.06304296  0.06133815  0.0586012   0.05562897
 0.05282854  0.05028221  0.047885    0.04544258  0.0427328   0.0395492
 0.03573479  0.03120595  0.02596157  0.02007561  0.0136772   0.00692739
-0.        ]
Gradient on the top boundary (i = L):
[0.         0.00536753 0.0107528  0.01616545 0.0215996  0.02702408
 0.03237269 0.03753984 0.0423922  0.04680758 0.0507357  0.05424237
 0.05752455 0.06088888 0.06466242 0.0689461  0.07299743 0.07416093
 0.07150742 0.06483316 0.05427728 0.04071888 0.02679009 0.01324138
 0.        ]
```

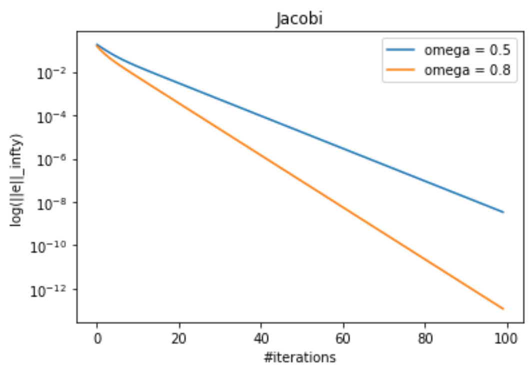
3a. the code is attached in 2-3.ipynb, where
the restriction I use:

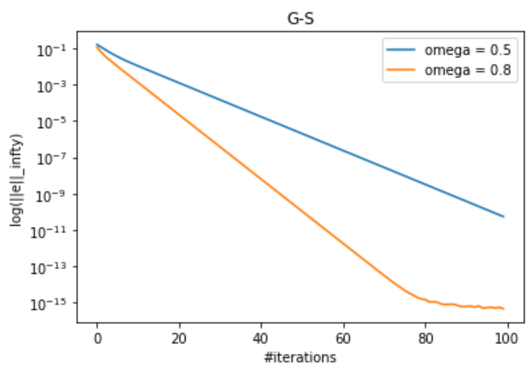
$$r_{i,j}^{coarse} = r_{2i,2j}^{fine}$$

The prolongation I use:

$$e_{i,j}^{\text{fine}} = e_{i//2,j//2}^{\text{coarse}}$$

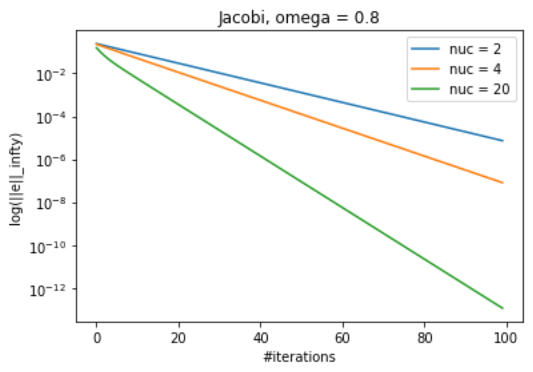
Where the //2 means to divide 2 and round down

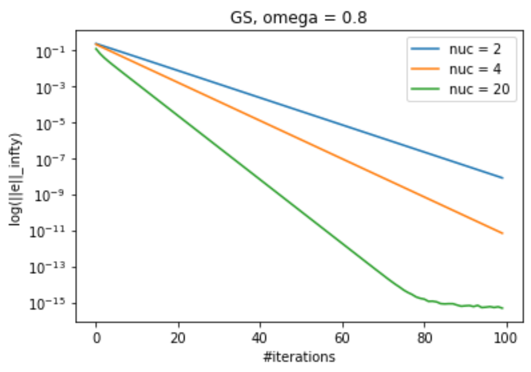




Interms of iteration number, we can see that multigrid routine is better than Jacobi or G-S. Relaxation factor 0.8 is better.

3.b





We can see that interms of iteration number, the larger nuc, the better performance of convergence. However, when using larger nuc, in each iteration we need more steps, so the running time is not necessasarily shorter.

4. in 2-4.ipynb attachment we implement multi grid method. we show that when the activated blockes are (2,7,10,11), the gradient is almost the same as the data, so we can conclude that this configuration is true:

```
Gradient on the left boundary (j = 0):
[-0.         0.00836951  0.0169094   0.02577795  0.03509749  0.04490336
  0.05504633  0.06504974  0.07403776  0.08110659  0.08537286  0.08642597
  0.08440623  0.07996028  0.07381995  0.06677717  0.05932249  0.05170792
  0.04406505  0.03647155  0.02897387  0.021592    0.01432145  0.00713728
 -0.        ]

Gradient on the right boundary (j = L):
[-0.         0.00620416  0.01240569  0.01859024  0.02472025  0.03072447
  0.03649082  0.04186592  0.04666461  0.05068915  0.05375301  0.05570379
  0.0564398   0.05591947  0.05416506  0.05126162  0.04734968  0.04260966
  0.03723878  0.03142627  0.02533267  0.01907878  0.01274502  0.00637785
 -0.        ]

Gradient on the bottom boundary (i = 0):
[-0.         0.00819538  0.01621291  0.02387195  0.03099335  0.03741286
  0.04299833  0.0476635   0.05136969  0.05411265  0.05590009  0.05673141
  0.05659017  0.05545471  0.05332061  0.05022422  0.04625467  0.04154868
  0.03627077  0.03058766  0.02464529  0.01855543  0.01239297  0.00620098
 -0.        ]
Gradient on the top boundary (i = L):
[0.         0.00718794 0.01433649 0.02139899 0.02831341 0.0349928
 0.0413146  0.04711249 0.0521797  0.05629462 0.05926437 0.06095185
 0.06128255 0.06024305 0.0578806  0.05430628 0.04969715 0.0442865
 0.03833109 0.03206069 0.02564711 0.01919859 0.01276916 0.0063732
 0.        ]
```