ME 614 Spring 2017-Homework 3

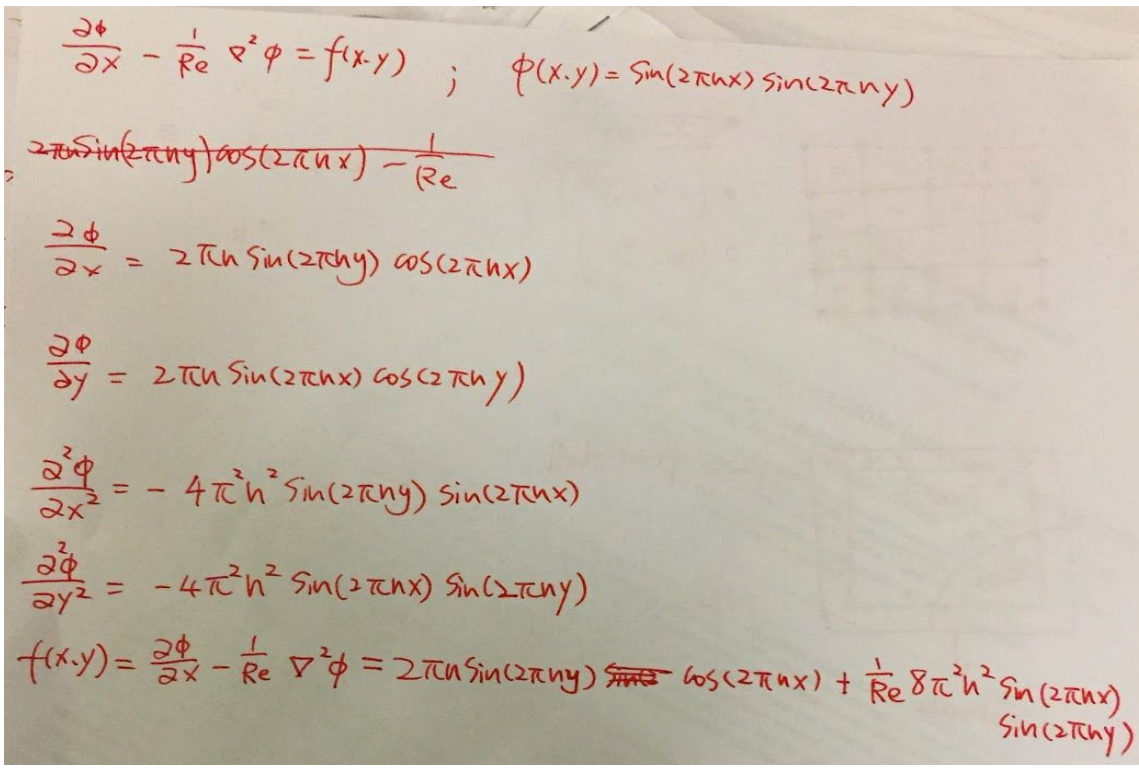
Poisson Solver & Taylor Green Vortex

Zitao He

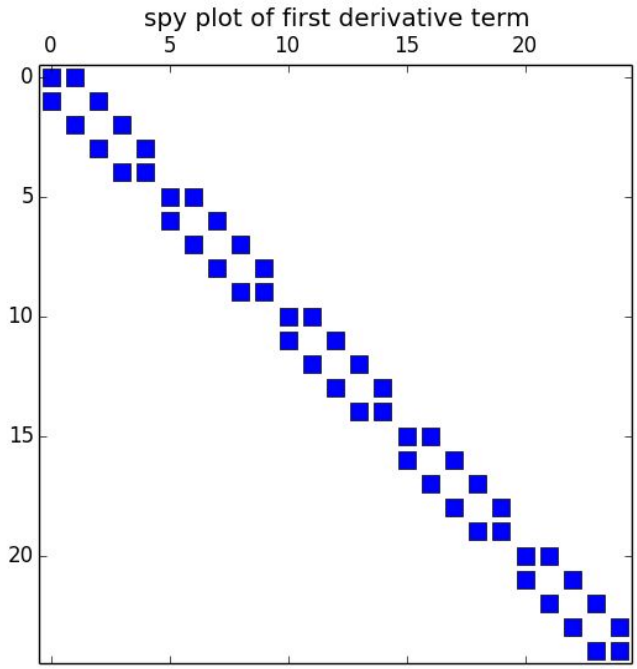
4/14/2017

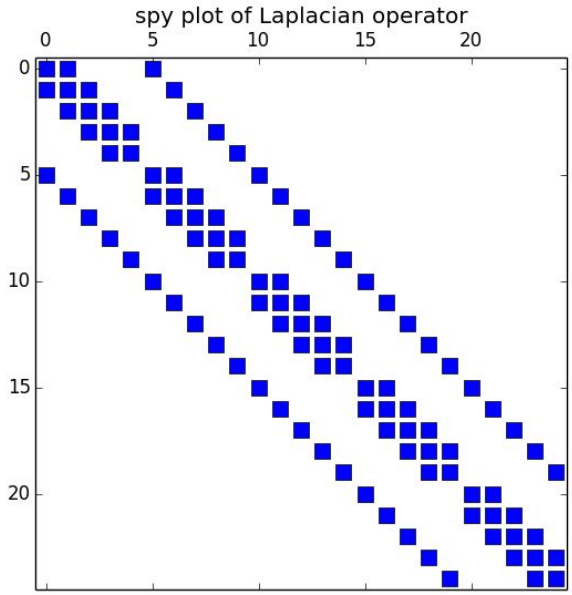
Problem 1

(a) In this part, we are asked to find the expression of f(x,y). The expression was listed below:

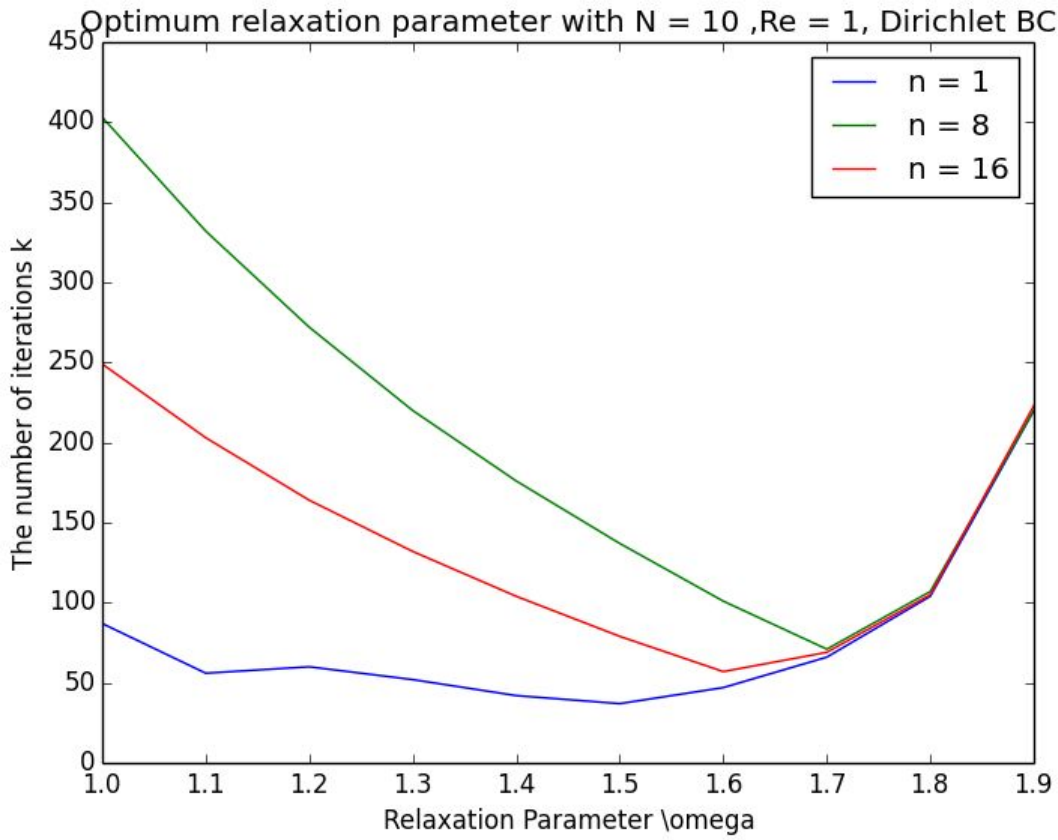


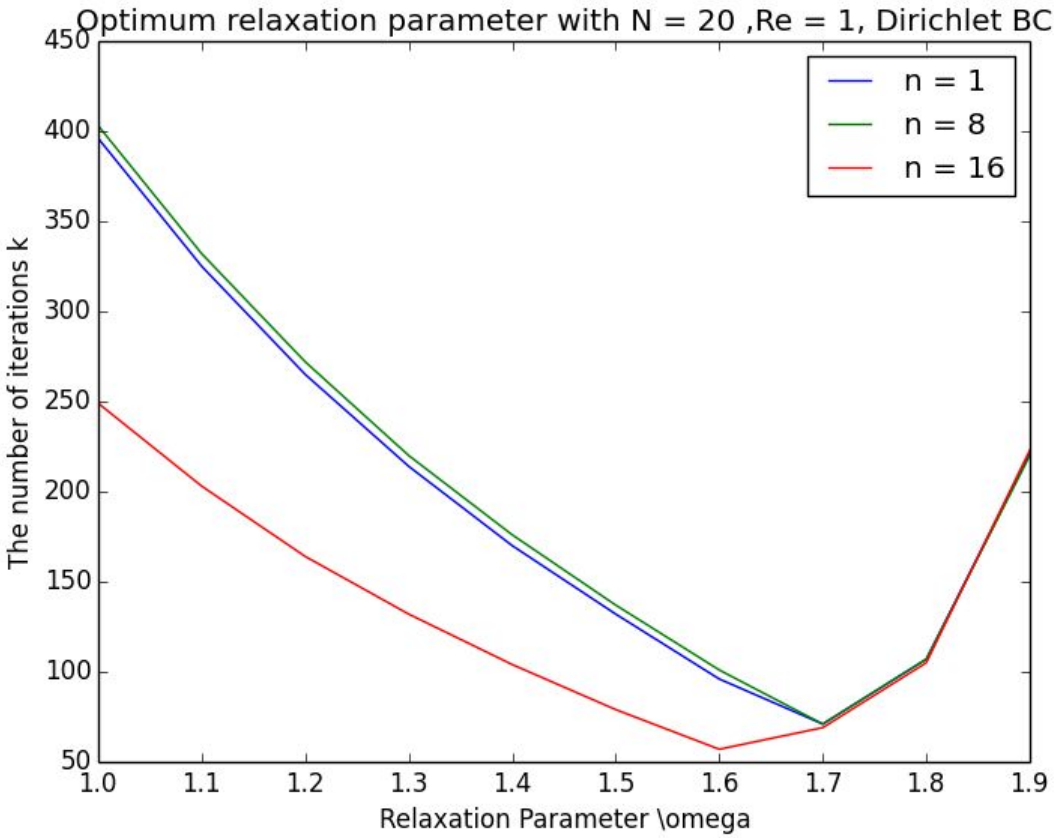
The discrete Laplacian operator for both homogeneous Neumann and homogeneous Dirichlet boundary conditions for a generic non-uniform grid have been prepared in code. The spy plots of operators are listed below:

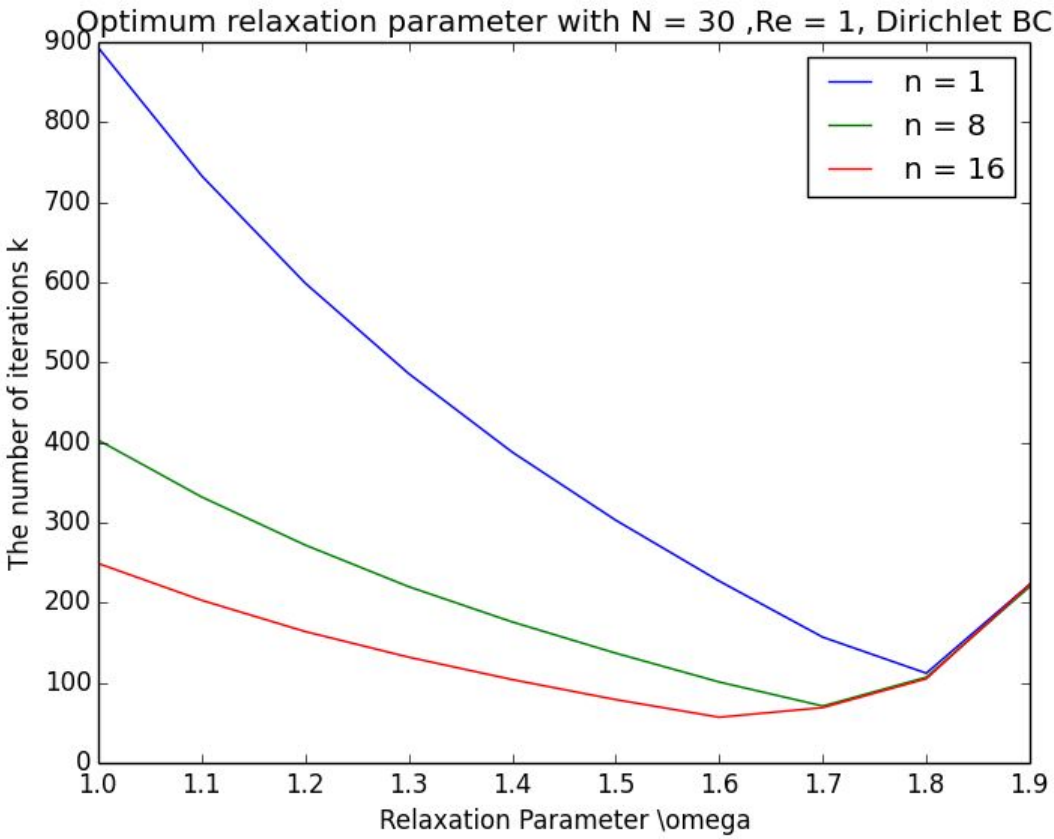




(b) To find the optimum relaxation parameter, I plotted the number of iterations versus the relaxation parameter with different values of wavenumber n and different grid sizes. The results are shown below:

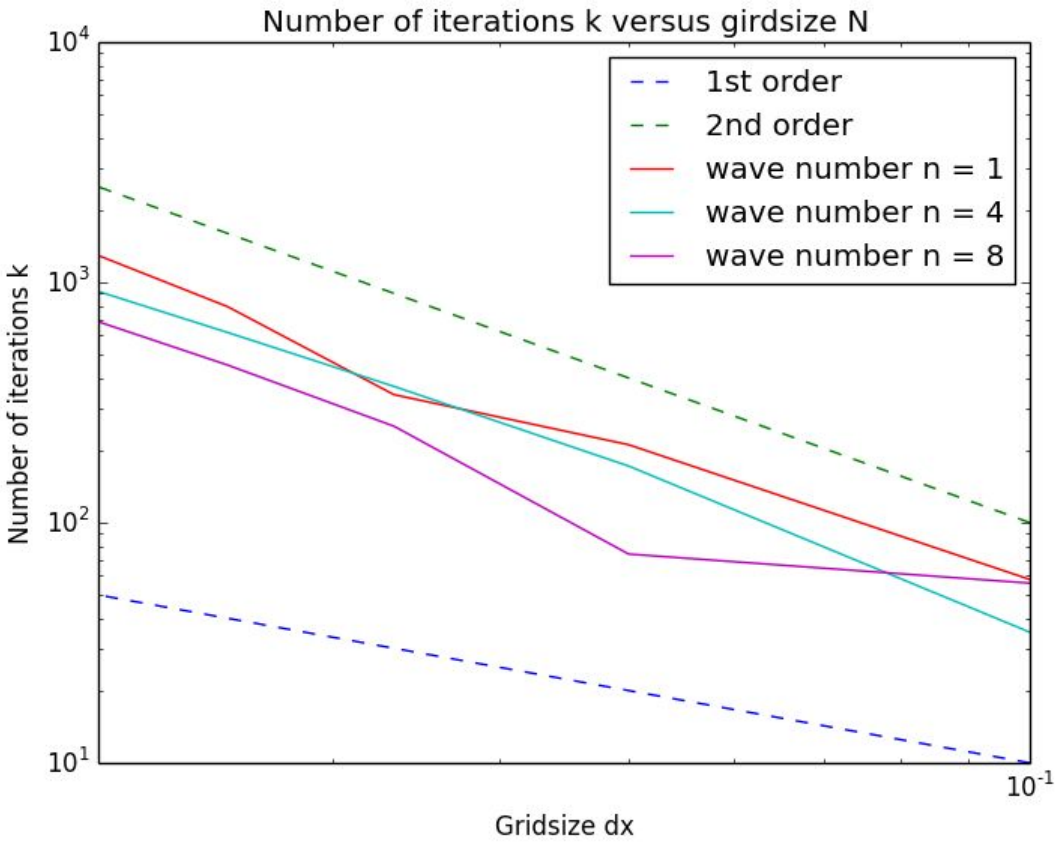



Optimum relaxation parameter with N = 20 ,Re = 1, Dirichlet BC
The number of iterations k
Relaxation Parameter \omega
n = 1
n = 8
n = 16
450
400
350
300
250
200
150
100
50
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9


Optimum relaxation parameter with N = 30 ,Re = 1, Dirichlet BC
The number of iterations k
Relaxation Parameter \omega
n = 1
n = 8
n = 16
900
800
700
600
500
400
300
200
100
0
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9

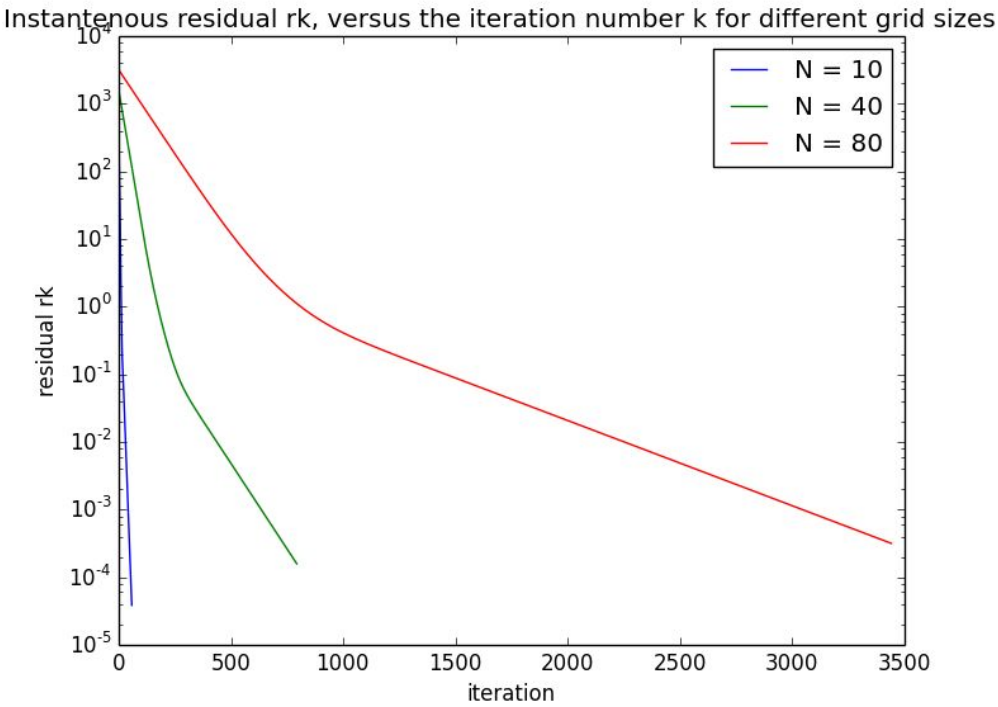
As we can see, at each grid size, as the wave number increased, the optimum relaxation parameter will increase as well. If we compare the the relaxation parameter with different grid size under the same wavenumber, it will show that decreasing grid size (L/N where L is the length of computation domain and N is the number of nodes we took into consideration) will lead to bigger optimum relaxation parameter. I adopted varying number of nodes of [10, 20, 30] which might make it not obvious for optimum relaxation parameter to change. However, for wave number n = 1, it is clear that the optimum relaxation parameter has changed which reveals that the properties of low wave number will be more sensitive for changes of grid size.

(c) To see the effects of wave number and grid size on the total number of iterations to reach the same convergence condition. I plotted the following figure:

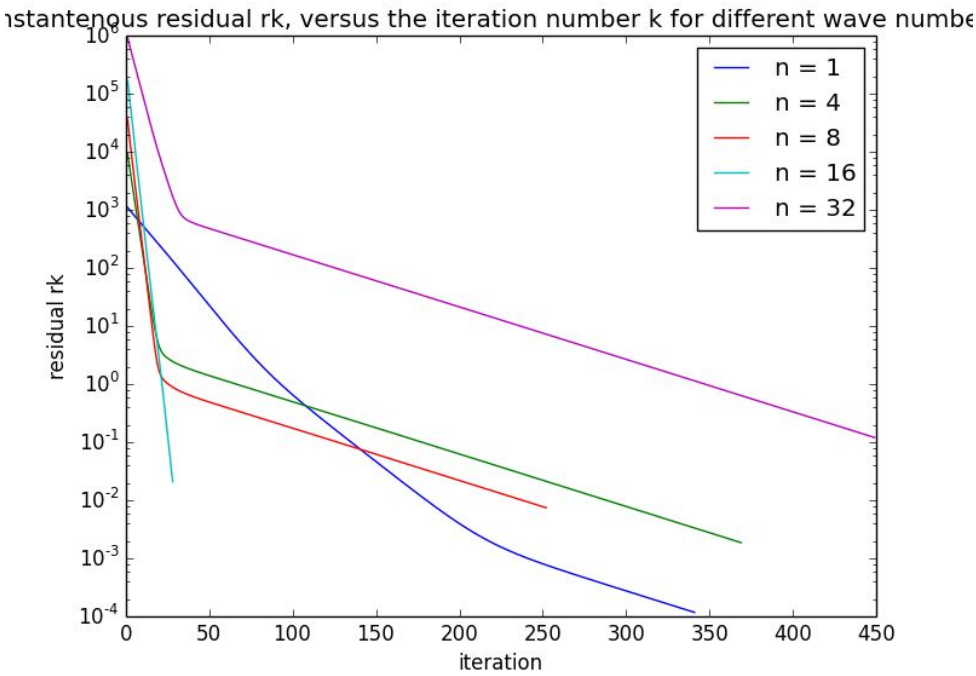


From the plot, generally, increasing wave number will lead to faster convergence(the number of iterations will be smaller). With the grid getting finer, the number of iterations getting larger. It has to be mentioned that the value on the left bottom corner(10^1) belongs to y-axis. The order of the plots seems to follow the second order reference line.

(d) The instantaneous residual rk versus the iteration number for different grid sizes N is shown below. In this plot, we can see finer grid convergences with smaller number of total iterations(the end of the line shows the end of the iteration process). Also, coarse grid convergences very quickly which was indicated by the very precipitous line of N =10 case.

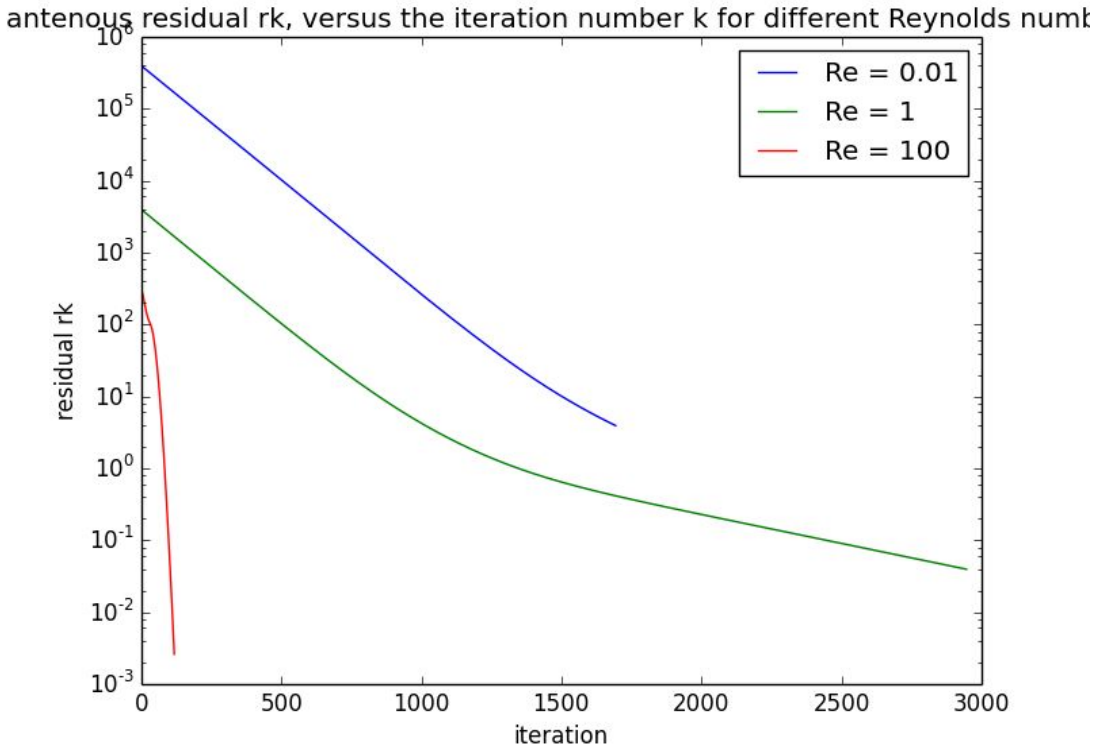


The instantaneous residual rk versus the iteration number for different wave number n is shown below. The properties of convergence are not that obvious. The cases of n = 4, 8, 32 look very similar but n=32 has larger residual and the value of rk is very dependent on the equation itself. The case of n = 16 convergences quickly than other cases and the case of n = 1 is between n = 4, 8, 32 and n =16.



The instantaneous residual rk versus the iteration number for different wave number n is shown below. In this part, we are supposed to plot Reynolds number Re from 0.01 to 10^8, however, I cannot get the result from with Re = 10000 to larger Re. This is because very high Re will make the LHS operator to be very ill-conditioned to solve the equations. The matrix is no longer a diagonal-dominant matrix which make it hard to solve. Other schemes might be helpful to get the result with very high Reynolds number but that's not what we should concern about.

From this plot we can see that the line of Re = 100 go down very sharply which means it convergences very quickly, and Re = 0.01 convergence relatively slowly compared to Re = 1. This might be because very small Re and very high Re make the one of the two operators on the left hand side of the equation to be small enough(either the Laplacian term or first derivative term) which leads to complexity reduction of left-hand-side matrix, hence make it convergences faster.

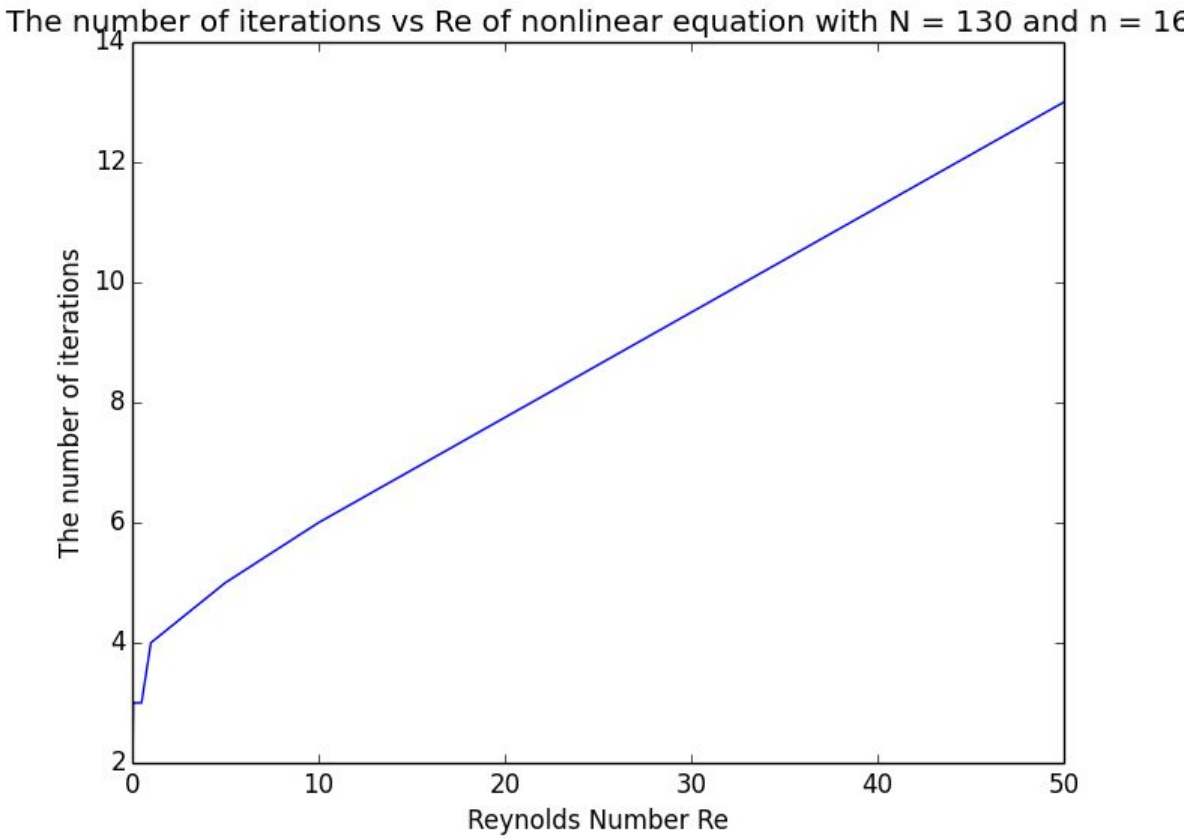


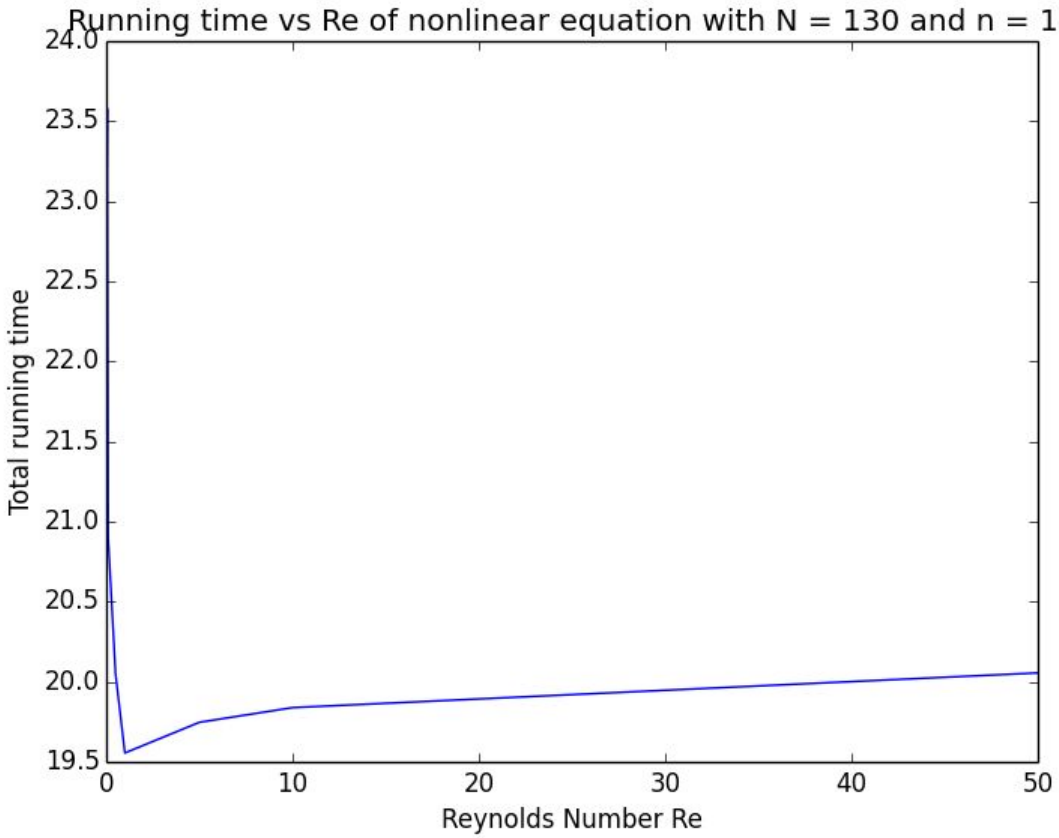
(e) For nonlinear equation, I left the Laplace operator on the left-hand-side and solved the phi. The result looks similar with linear equation as in sample code(without pressure mask). I plotted the number of iterations to converge with different Reynolds number, and also the elapsed time for different value of Re. The results are shown below:

As yo ucan see, with increasing value of Re, the number of iterations will increase accordingly. It looks like that it follows linearly except for very small value of Re = 0.01. For

running time plot, There is a value of which the running time is smallest. With Re increases, the total running time will increase slightly.

We are supossed to solve linear equation by using sub-iterations method. However, because of the rank deficiency issue, the problem is insolvable. Because of that I cannot compare the performance between first solving method and sub-iteration method in terms of the number of iterations to converge and running time.





(f) Applying LU pre-factorization of the Laplace operator through ***scipy.sparse.linalg.splu*** will make the solver much faster than calling ***spsolve*** function in Python. The reason is that it will take LU pre-factorization one time to get the final solution for all of the iteration step, but it will take ***spsolve*** to operation in each iteration which makes it very time-consuming.

Problem2.

In this part I applied staggered grid for P, u and v to solve Naiver-Stokes equation through projection method as explained in class. The operator of convective term and viscous term have been developed. Poisson equation for pressure was solved and Runge-Kutta method was applied for prediction step.

Details for Runge-Kutta method are following:

Runge-Kutta 1:

This method is also called explicit euler method for approximating the solution of the initial value problem $\boldsymbol{y'(x) = f(x,y)}$; $\boldsymbol{y(x_0) = y_0}$ which evaluates the integrand.
For step *i+1*,

$$y_{i+1} = y_i + h\, f(x_i, y_i),$$

Runge-Kutta 2:

This method is also called Heun euler method or midpoint method(there are many other RK methods such as modified euler method and Ralston method).
For step *i+1*,

$$y_{i+1} = y_i + 1/2 \, (k_1 + k_2 \,),$$

$$k_1 = h \, f(x_i, y_i),$$

$$k_2 = h \, f(x_i + h, y_i + k_1 \,),$$

Runge-Kutta 3:

This method is a third order Runge-Kutta. Procedures are:
For step *i+1*,

$$y_{i+1} = y_i + 1/6 \, (k_1 + 4 \, k_2 + k_3 \,),$$

where

$$k_1 = h \, f(x_i, y_i),$$

$$k_2 = h \, f(x_i + h / 2, y_i + k_1 / 2 \,),$$

$$k_3 = h \, f(x_i + h, y_i - k_1 + 2 \, k_2 \,),$$

and $x_i = x_0 + i \, h$.

Runge-Kutta 4:

The most widely known member of the Runge–Kutta family is generally referred to as "RK4", "classical Runge–Kutta method" or simply as "the Runge–Kutta method".
For step *i+1*,

$$y_{i+1} = y_i + 1/6 \, (k_1 + 2 \, k_2 + 2 \, k_3 + k_4),$$

where

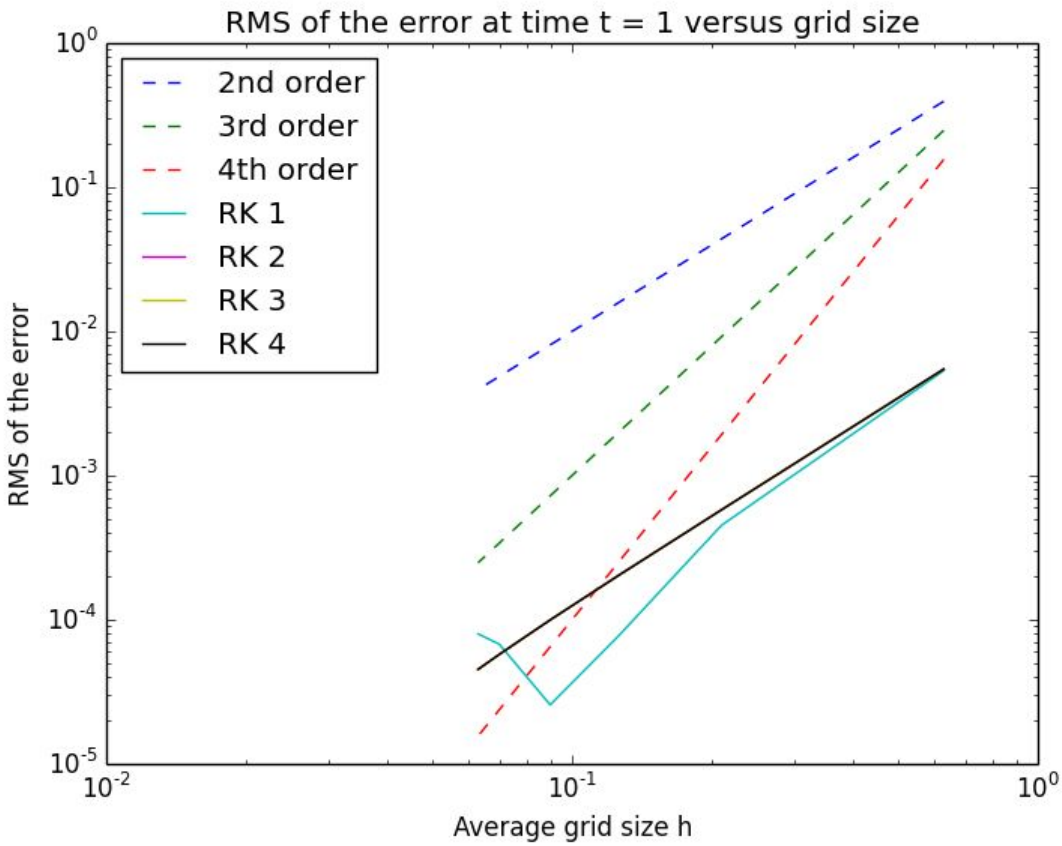
$$k_1 = h \, f(x_i, y_i),$$

$$k_2 = h \, f(x_i + h / 2, y_i + k_1 / 2 \,),$$

$$k_3 = h \, f(x_i + h/2, y_i + k_2 / 2),$$

$$k_4 = h\, f(x_i + h,\ y_i + k_3)$$

The numerical solution for P, u and v match analytical solution very well. The RMS error of u and v are plotted here. For the square domain we computed, u and v are symmetric, so the value of u is exactly the same as v. So the following plot can be represented for both u and v.
The time step is 0.001.



As we can see, the line of RK1, RK2, RK3 are almost the coincide exactly and RK has a turning point which means RK 2, 3, 4 have better numerical performance than RK1. All of these RK method have the same order of convergence (2nd order).