Assignment 5

Lakshya J ee19b035

March 24th, 2021

Introduction

In this assignment we are asked to analyze how potential varies across a plat which is kept at a constant voltage of 1 volt within a given radius. I start by importing all the required libraries and writing else if blocks for all the possible inputs the user may enter. I make sure N_x and N_y are the same in all cases and a warning is seen when the radius is bigger then the specified axes. Here the *radius* accepted as input is with respect to the N_x and N_y units which is later scaled accordingly for a 1cm by 1cm plate. The bottom of the plate is assumed to be grounded.
I create a meshgrid and create the vector ii which holds all the points where a potential of 1 V is to be maintained.

```
x = np.linspace(-floor(Nx/2), floor(Nx/2), Nx)
y = np.linspace(-floor(Ny/2), floor(Ny/2), Ny)
Y,X = np.meshgrid(y,x)
ii =np.where(Y**2+X**2<(radius**2))
phi[ii]=1
ii=ii-floor(Nx/2)
```

1 3D Potential Plot

I begin by showing the initial 3D plot of how the potential ϕ varies on the plane. Here is the result obtained for the default parameters defined by

```
Nx=31
Ny=31
radius=8
Niter=1500
```

Here is the code utilised to achieve the same:

```
f1 = plt.figure(1)
ax = p3.Axes3D(f1)
surf = ax.plot_surface(Y, X, phi.T, rstride=1, cstride=1, cmap=cm.
    jet)
plt.show()
```

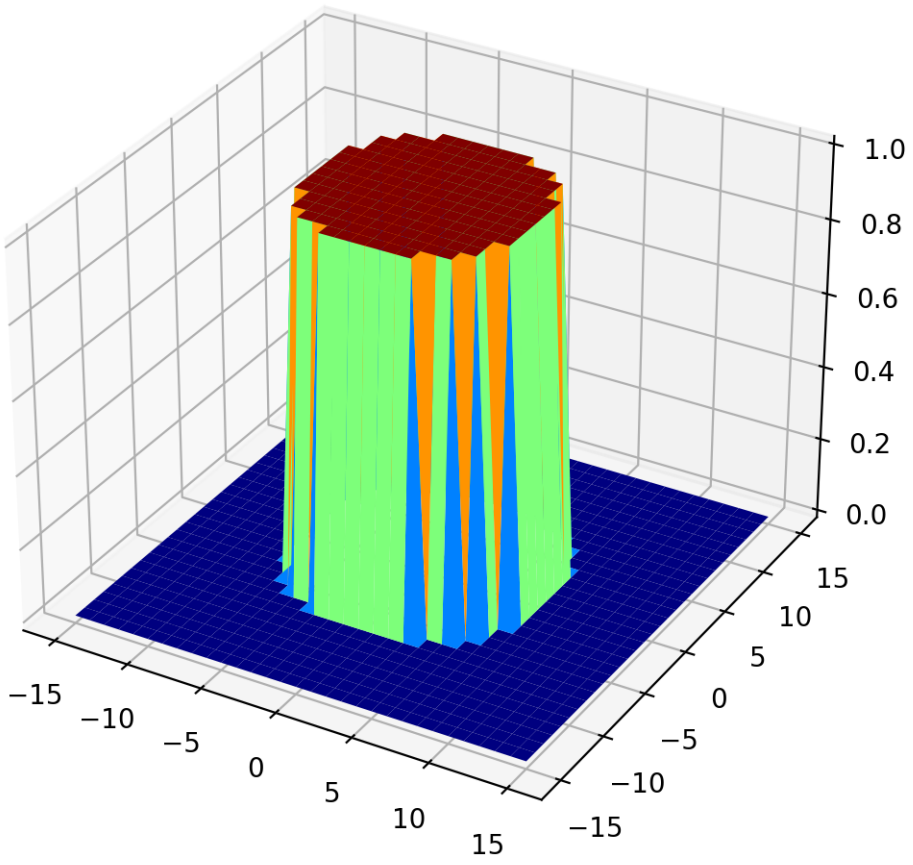

Figure 1: 3D Surface Plot of ϕ

2 Finding Potential

Now i begin the loop for *Niter* to update the potential distribution on the plate. I make another copy of *phi* called *phinew* to store the updated values of phi as sir had asked us to do. Here are the lines I use:

```
phinew[1:-1,1:-1] = 0.25*(phi[1:-1,0:-2]+phi[1:-1,2:]+phi
    [0:-2,1:-1]+phi[2:,1:-1])
# Finding the error
errors[i] = np.amax(abs(phinew-phi))
phi = phinew.copy()
```

As you can see I simultaneously find the error obtained while the value of phi changes.
Now I update the boundary conditions. In all these cases I make use of Python's vectorized code.

```
# Left boundary
phinew[1:-1,0] = phinew[1:-1,1]
# Right boundary
phinew[1:-1,-1] = phinew[1:-1,-2]
# Top boundary
phinew[0,1:-1] = phinew[1,1:-1]
# Bottom Boundary
phinew[-1,1:-1] = phinew[-2,1:-1]
```

3 Finding Current

I find the current in the plate for each iteration by running through the loop for 50 times when I can be sure the error will be low. Here I don't carry out any error analysis. I update the boundary conditions for the currents by making them equal to the values which are present in the first and last but one row and columns.
Here is the code for the same.

```
Jx[:,1:-1] = 0.5*(phi[:, 2:]-phi[:, :-2])
# Left boundary and right boundary
Jx[:,0] = Jx[:,1]
Jx[:,-1] = Jx[:,-2]
Jy[1:-1,:] = 0.5*(phi[2:, :]-phi[:-2, :])
# Top boundary and Bottom Boundary
Jy[0,:] = Jy[1,:]
Jy[-1,:] = Jy[-2,:]
```

I plot every 100th plot obtained after the change in ϕ. Here are few of the results I have obtained.

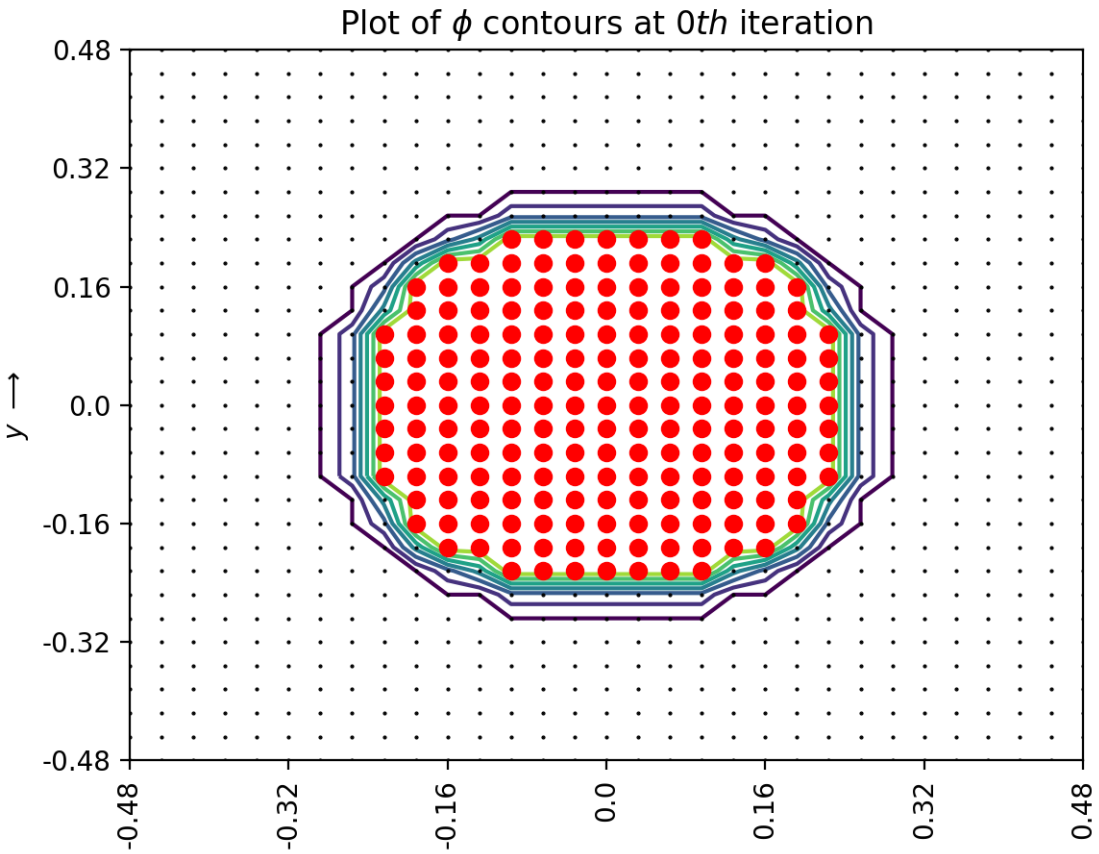

Figure 2: Initial ϕ contour plot for default values

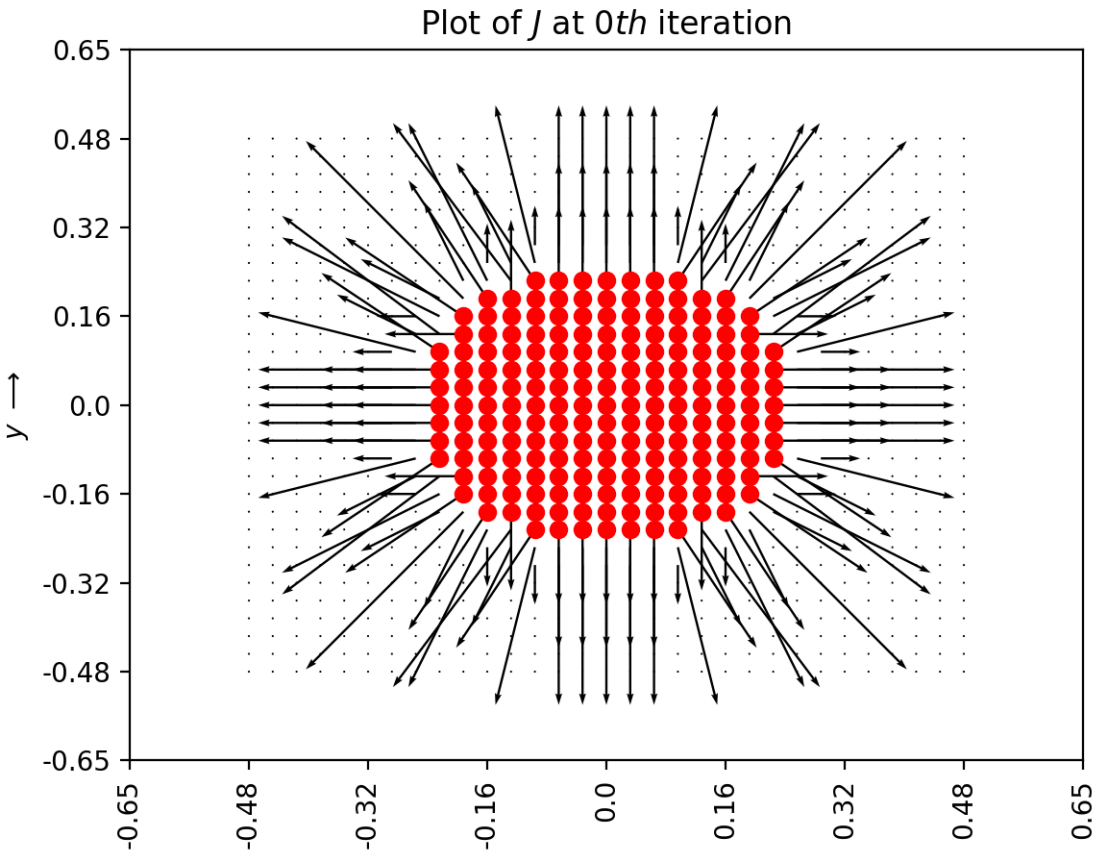

Figure 3: Initial J plot for default values

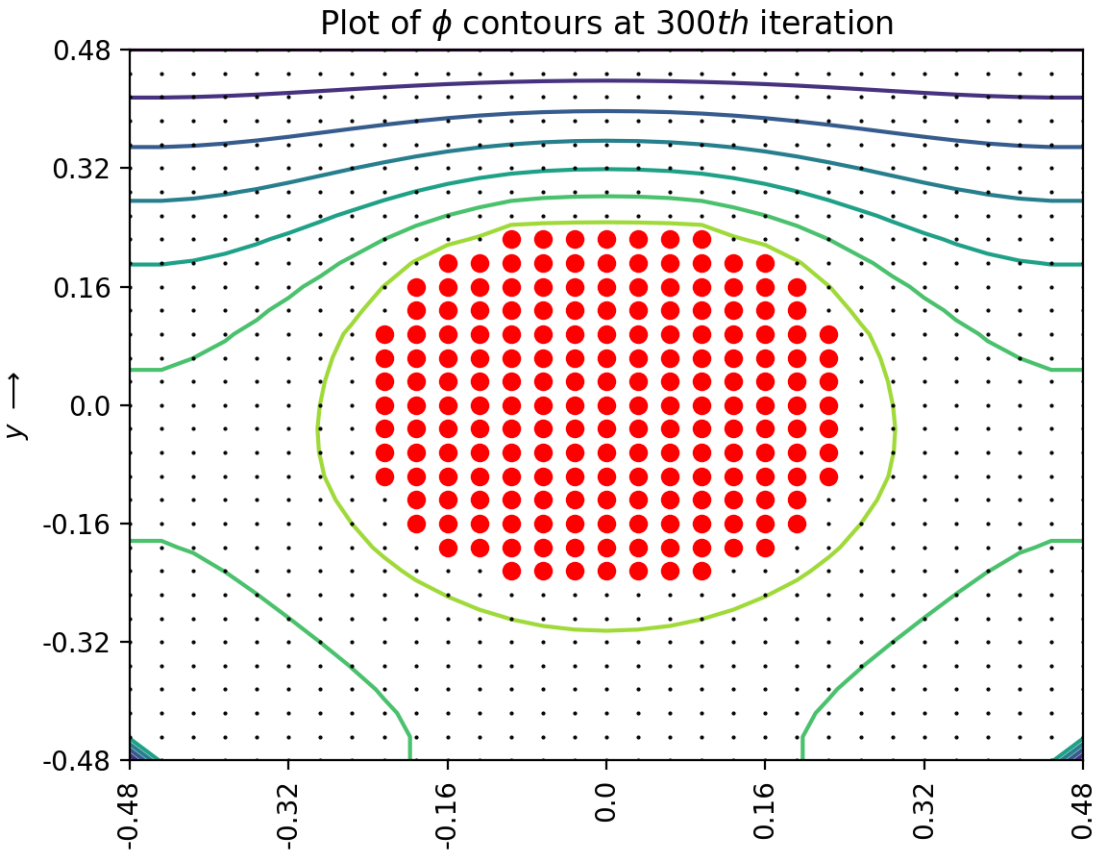

Figure 4: ϕ contour plot at 300th iteration for default values

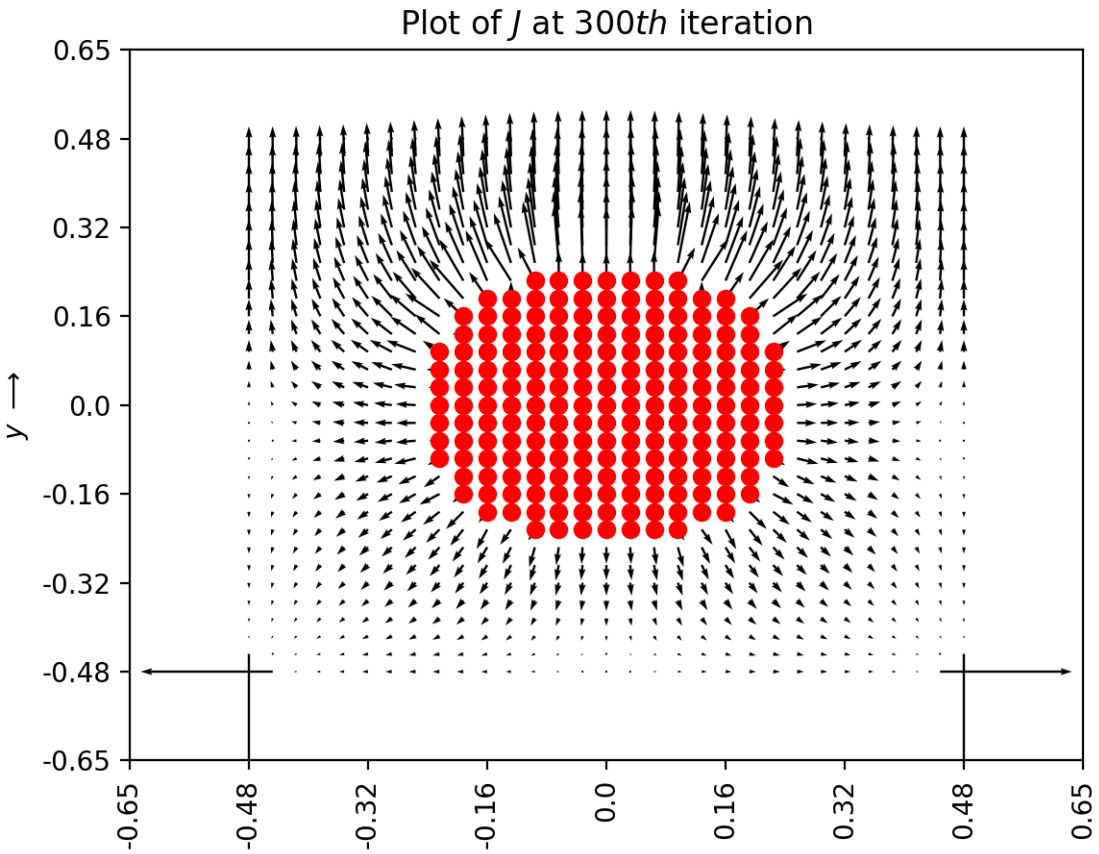

Figure 5: J plot at 300th iteration for default values

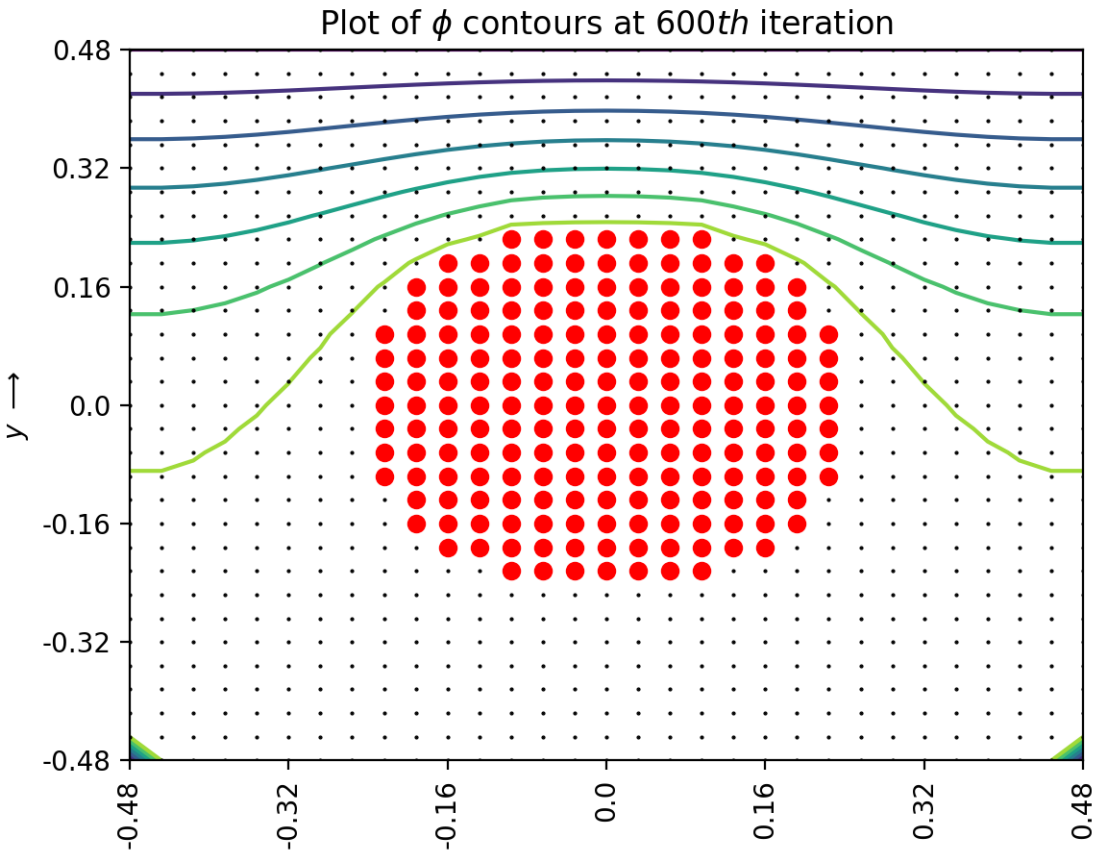

Figure 6: ϕ contour plot at 600th iteration for default values

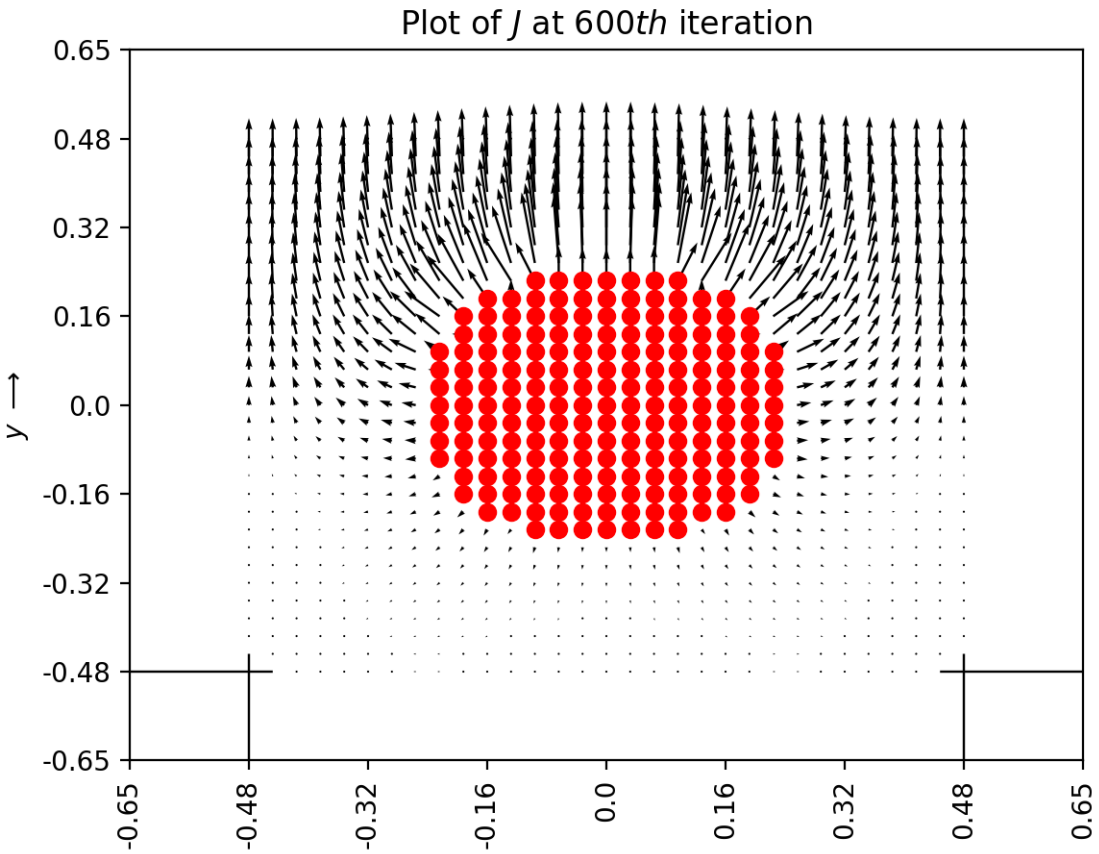

Figure 7: J plot at 600th iteration for default values

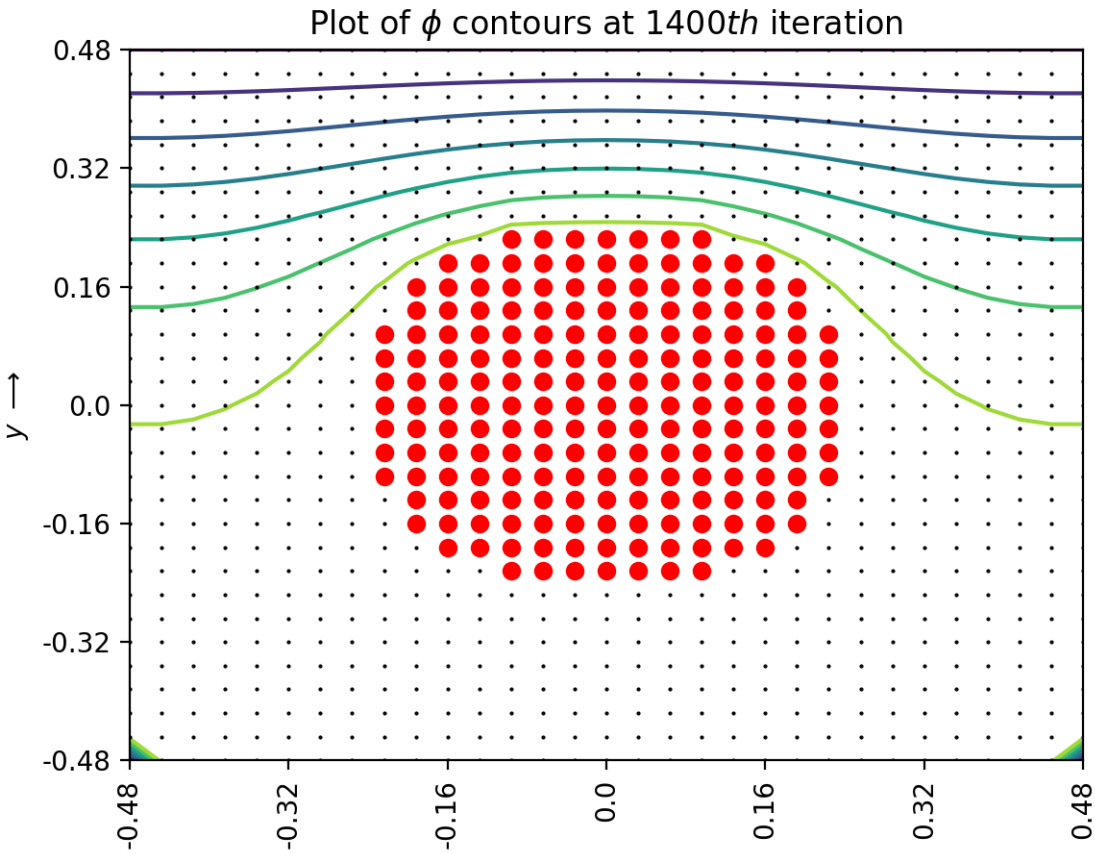

Figure 8: ϕ contour plot at 1400th iteration for default values

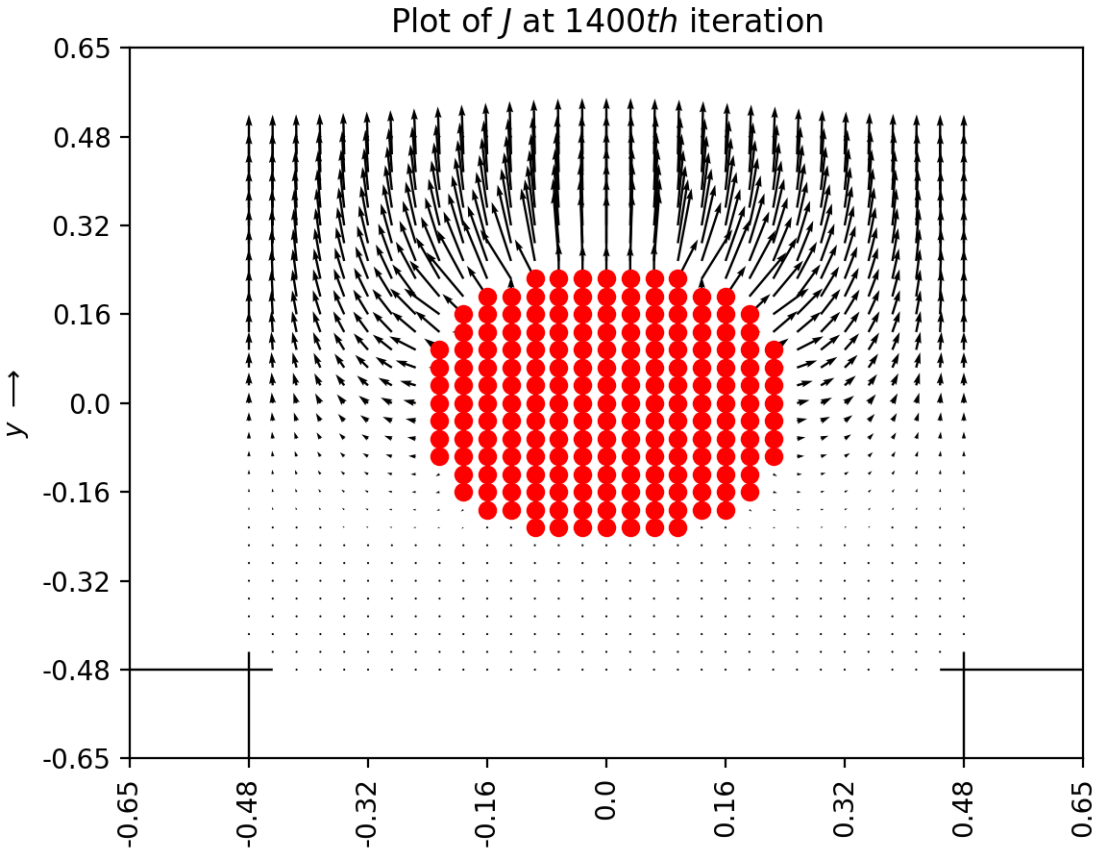

Figure 9: J plot at 1400th iteration for default values

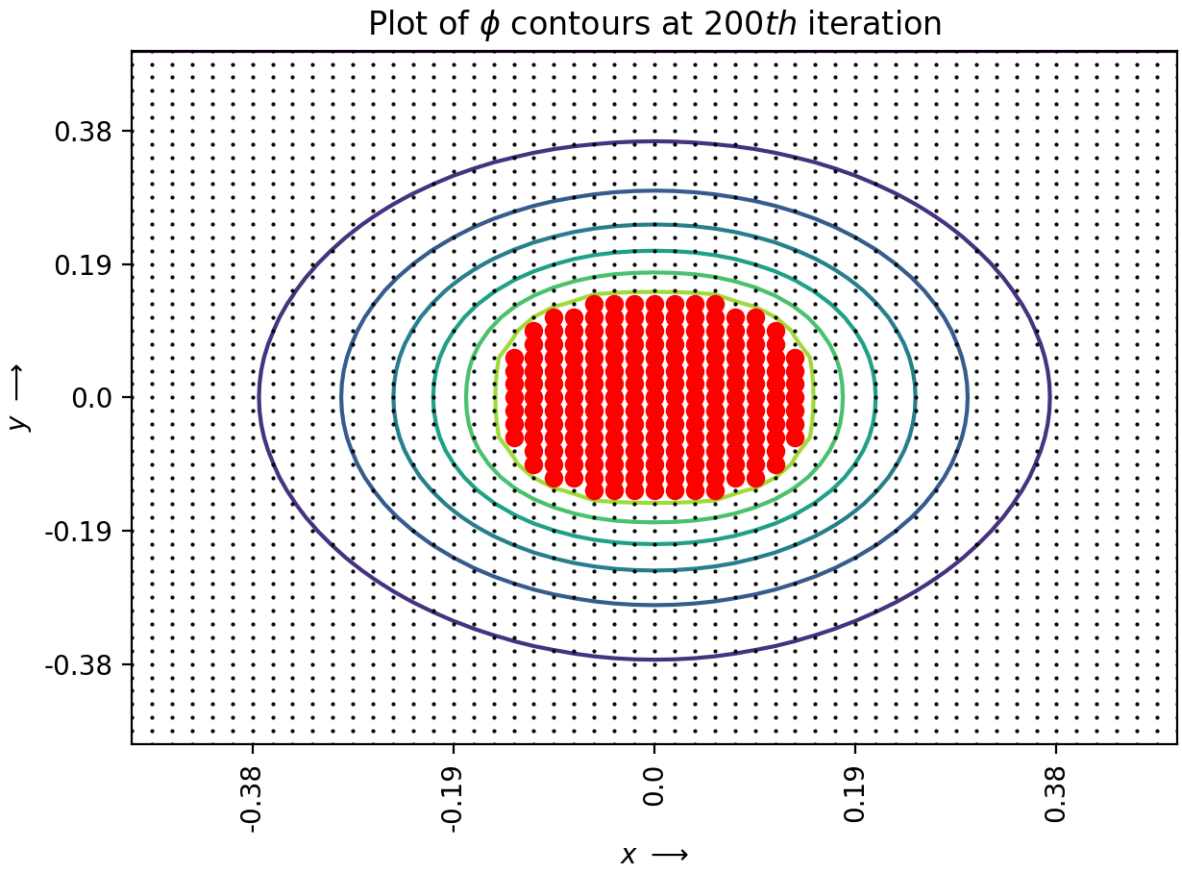

Figure 10: ϕ contour plot at 200th iteration for $Niter$=1000, N_x=53, $radius$=8

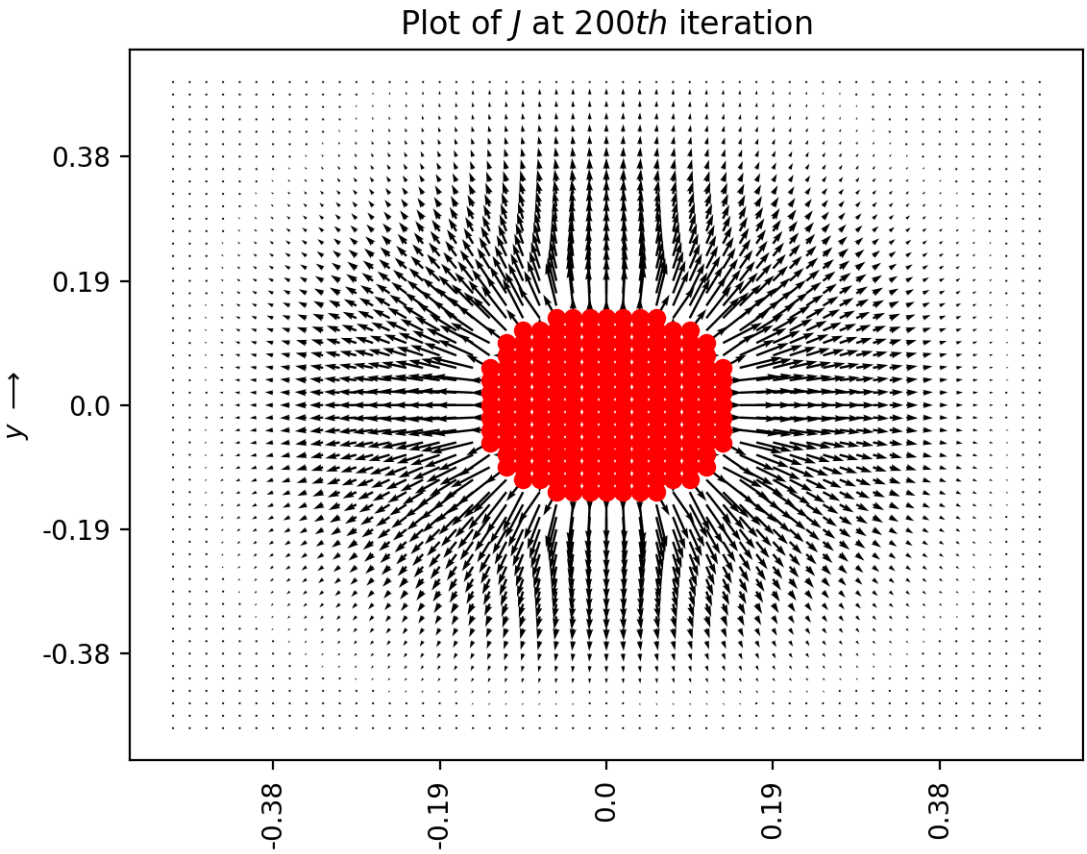

Figure 11: J plot at 1400th iteration for $Niter$=1000, N_x=53, $radius$=8

4 Finding Errors

Here I try and find an expression to fit the errors in finding ϕ in two cases. In the first case I consider all the $Niter$ points and in the second case I consider points after the 500th iteration as the error graph can be considered to settle after this point.

```
# Finding the fits using least squares approximation
ln_err = np.log(errors)
col = np.ones(Niter)
# Stacking the columns for approximating values through linear
    regression
A = np.column_stack((col, iter_array))
ls_co = np.linalg.lstsq(A, ln_err, rcond=None)[0]
A = np.exp(ls_co[0])
B = ls_co[1]
error_est = A*np.exp(B*iter_array)
```

This is the general algorithm I use in both cases for the A and B values approximation. I find the log of the actual errors obtained because I want to try and fit them in a linear plot. Using the least squares method, I find good enough approximations.
Here is the plot obtained on comparing the errors. I plot the errors for 1500 iterations and 1000 iterations with $N_x = 53$ and $radius$=8.
The settling time is longer for the second plot because there are a lesser number of iterations and the granularity for the grid has increased as well.

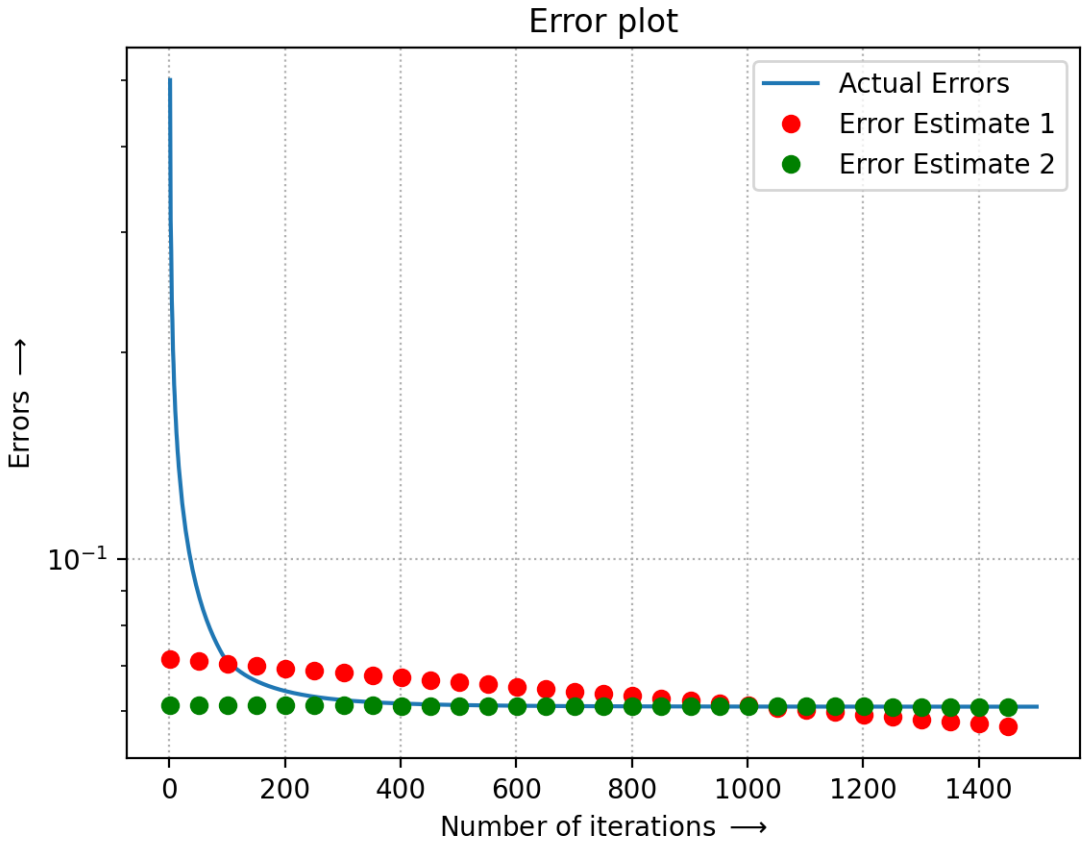

Figure 12: Error Estimates for 1500 iterations

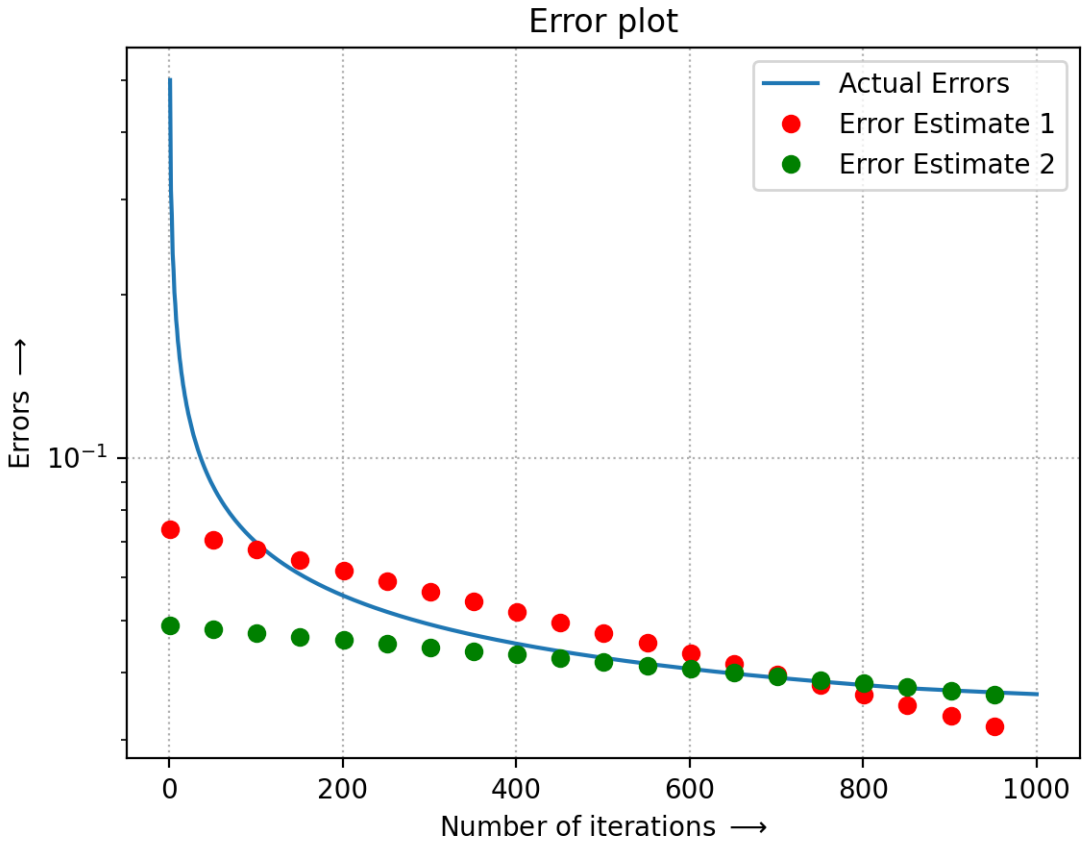

Figure 13: Error Estimates for 500 iterations

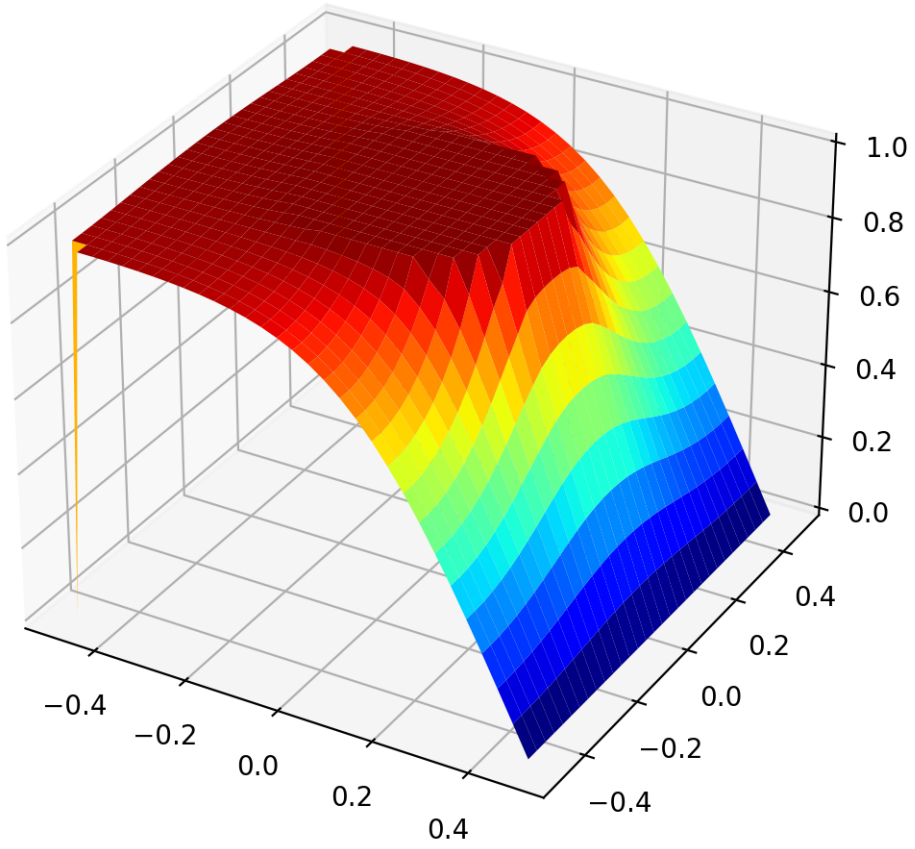

Figure 14: Final 3D Surface plot for the default values

5 Surface Plot

After *Niter* iterations here is the surface plot I have obtained for the default values above. Below is the plot I obtained for 1000 iterations.

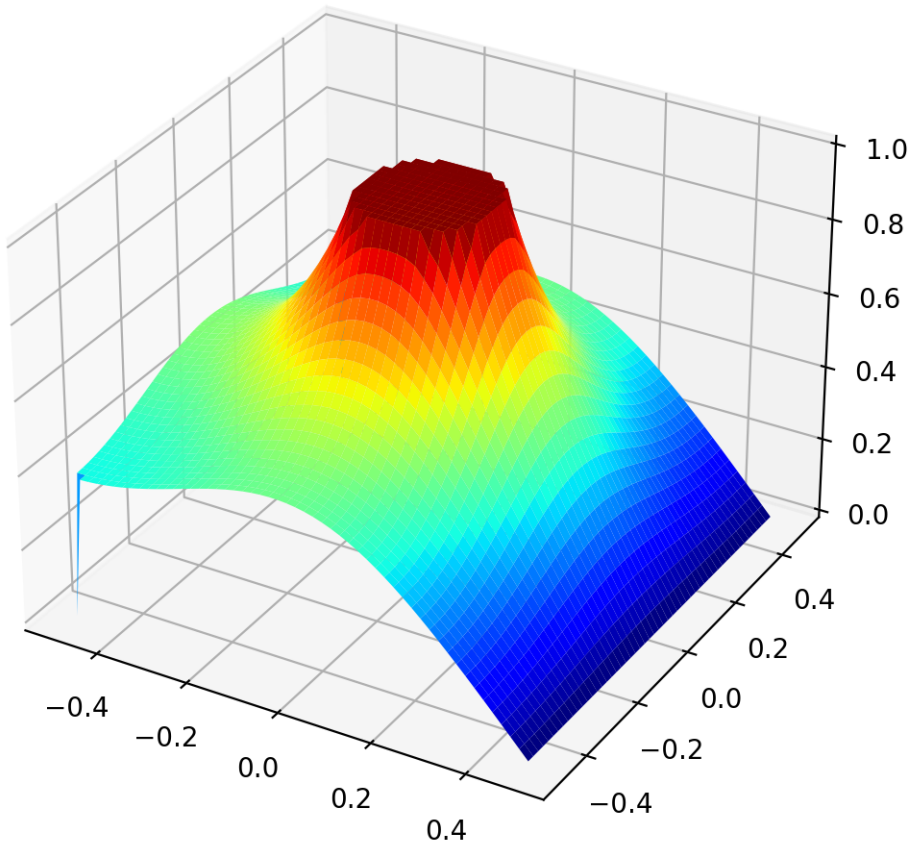

Figure 15: Final 3D Surface plot for $Niter$=1000, N_x=53, $radius$=8