EE2703 : Applied Programming Lab
Assignment 5
The Resistor Problem

T.M.V.S GANESH
EE19B124

April 8, 2021

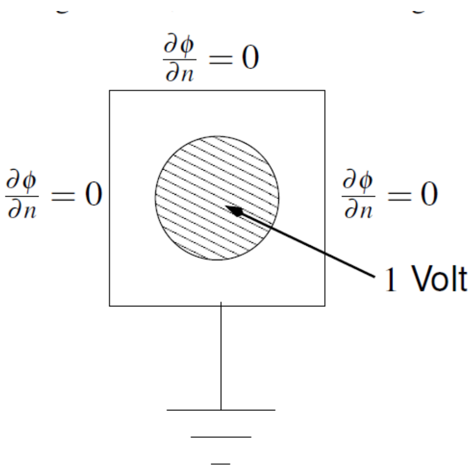
Introduction

The assignment is about

- Voltage and Current profiles in a square Resistor with a circular wire passing through it.
- Temperature Contour in the resistor

The Resistor Problem

A wire is soldered to the middle of a copper plate and its voltage is held at 1 Volt. One side of the plate is grounded, while the remaining are floating. The plate is 1 cm by 1 cm in size.



$\frac{\partial\phi}{\partial n} = 0$
$\frac{\partial\phi}{\partial n} = 0$
$\frac{\partial\phi}{\partial n} = 0$
1 Volt

Figure 1: Resistor with Soldered wire

Equations:

$$\vec{j} = \sigma.\vec{E} \tag{1}$$

$$\vec{E} = -\nabla.\phi \tag{2}$$

$$\nabla.\vec{J} = -\frac{\partial\rho}{\partial t} \tag{3}$$

$$\nabla^2.\phi = \frac{1}{\sigma}.\frac{\partial\rho}{\partial t} \tag{4}$$

For DC currents the RHS in 4th equation is Zero

$$\nabla^2.\phi = 0 \tag{5}$$

Numerical Solution in two dimension

Laplace's equation is easily transformed into a difference equation.The equation can be written out in Cartesian coordinates

$$\frac{\partial^2 \phi}{\partial^2 x} + \frac{\partial^2 \phi}{\partial^2 y} = 0 \tag{6}$$

Solving the above equation at (i, j) gives

$$\phi_{i,j} = \frac{\phi_{i,j+1} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i-1,j}}{4} \tag{7}$$

The potential at certain (i,j) is the average of potentials of neighbours.This values are iterated till the solution converges

Code

Libaries:

```
 #Importing the Required Libraries
import pylab
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
import numpy as np
import scipy
import scipy.special as sp

from sys import argv, exit

```

Defining the Parameters:

The parameters are either taken as from command line or the default values in the document will be used:

```
  #Initialising the Parameters
Nx = 25;  #size along x
Ny = 25;  #size along y
radius = 8;#radius of central lead
Niter = 1500; # number of iterations to perform
temp = 1
if(len(sys.argv)>1):
    if len(argv)!=5:
        print('Enter the correct number of values')
        print("The default values are being used")
    if len(argv)==5:
        if argv[1]!=argv[2]:
            print("Enter equal values of Nx,Ny as we are using a square plate")
            print("The default values are being used")
            temp = 0
        if int(argv[3])>(int(argv[2])//2):
            print("Enter a smaller value of radius")
            print("The default values are being used")
            temp = -1
        if temp == 1:
            Nx=int(argv[1]); #size along x
            Ny=int(argv[2]); # size along y
            radius=int(argv[3]);#radius of central lead
            Niter=int(argv[4]); # number of iterations to perform
else:
    print("The default values are being used")
```

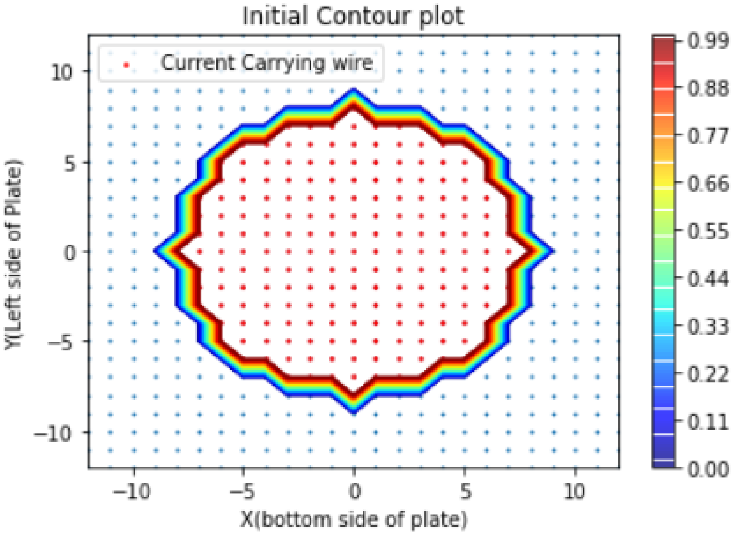
Phi Array

```
#Allocating and Initialising the potential array
phi = np.zeros((Nx,Ny))

x = arange(0,Nx,1)
x = x - (Nx//2)#Coordinates with (0,0) as midpoint
y = arange(0,Ny,1)
y = y - (Ny//2)
X,Y = meshgrid(x,y)

ii = where(X**2+Y**2 <= radius**2)#Indices of the 1V potential in phi array
phi[ii] = 1.0
#Graph:
figure(1)
contour(X,Y,phi,100,cmap=cm.jet)
colorbar()
scatter(X,Y, s = 1)
scatter(ii[0]-Ny//2,ii[1]-Nx//2,s = 2, color = ’r’,label =’Current Carrying wire’
    )#adjusting the indices ii to coordinates
xlabel(’X(bottom side of plate)’)
ylabel(’Y(Left side of Plate)’)
title(’Initial Contour plot’)
legend(loc = ’upper left’,fontsize = 10)
```

The x and y coordinates are adjusted such that the (0,0) is located in middle of the plate and the indices of phi array are matched with the shape of the plate i.e the value of phi[0,0] is the potential at the top left corner of plate



The red dots represent the wire region.

Performing the Iterations:

The array oldphi stores a copy of the phi array as a reference in calculating the error obtained after each iteration
Python code:

```
#Iterations:

def update_phi(phi): #function to update value of phi with average of neighbouring values
    phi[1:-1,1:-1] = 0.25*(phi[1:-1,0:-2]+phi[1:-1,2:]+phi[0:-2,1:-1]+phi[2:,1:-1])
def boundaries_phi(phi): #Asserting the Boundary conditions
    phi[1:-1,[0,-1]] = phi[1:-1,[1,-2]]
    phi[0] = phi[1]
    phi[ii] = 1
oldphi = phi.copy() #Initialisng the copy of phi
errors = np.zeros(Niter)#array of Errors
for i in range(Niter):#Iterating Niter no of times
    oldphi = phi.copy()
    update_phi(phi)
    boundaries_phi(phi)
    errors[i]=(abs(phi-oldphi)).max()
```

Updating the potential

The function "*update_phi*" replaces the current value of potential at a point with that of the average of the potentials of neighbours except for the edges of the plate.
We do this using the python subarrays instead of using a nested for loop

Boundary Conditions

The function "*boundaries_phi*" takes care of the potential at the edges of the plate as there cannot be a normal gradient of ϕ at the edges except the grounded one as the charge cannot leap out into air making the current tangential
As a result the potential at the edge is same as that of the values at the points just adjacent to the edge.
This also maintains the voltage of wire to 1V at the points where the wire is present.

The above process is iterated Niter number of times and the error in each iteration is stored in the array "*errors*"

Errors

Python Code:

```
 n_iter = arange(1,Niter+1,1)#array of no of iterations
plot(n_iter[::50],errors[::50],'o',label='Error')#Every 50th Value of Error
xlabel('iteations')
ylabel('Error')
title('Error in every iteration with a step of 50')
legend(loc = 'upper right',fontsize = 10)
figure(3)
loglog(n_iter,errors,label = 'Error')
loglog(n_iter[::50],errors[::50],'o',markersize =5)
xlabel('iteations(log)')
ylabel('Error(log)')
title('Error in every iteration with a step of 50 in log scale')
legend(loc = 'upper right',fontsize = 10)
show()
figure(4)
semilogy(n_iter,errors,label ="Error")
semilogy(n_iter[::50],errors[::50],'o',markersize =5)
xlabel('iteations')
ylabel('Error(log)')
title('Error in every iteration with a step of 50 in semilog axis')
legend(loc = 'upper right',fontsize = 10)
```

The errors in each iteration are plotted in log-log and semilog axis from semilog plot of the errors we can observe that the log(error) varies linearly with iterations So,we try to fit a exponential to the error curve using all the values of error and only considering the errors after 500 iterations resulting in two exponentials

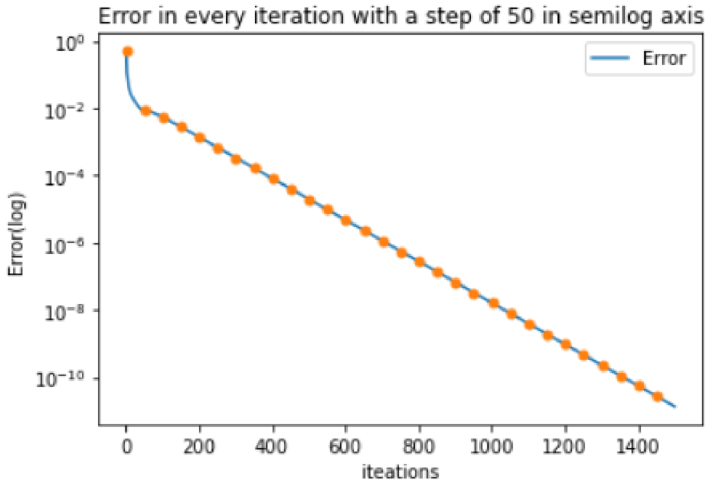


Figure 2: Semilogy plot of error vs iterations

Fit is of the form :

$$y = A.e^{B.x} \tag{8}$$

Applying log on both sides : log(y) = log(A) + B.x

```
#Finding the best fit to errors using least squares method
log_errors = np.log(errors)
M_1 = c_[ones(1500),n_iter]
X_1 = c_[log_errors]
p_1=lstsq(M_1,X_1,rcond=None)
fit1 = (exp(p_1[0][0]))*(exp(p_1[0][1]*n_iter))
#Finding the best fit to errors beyond 500 iterations using least squares method
M_2 = c_[ones(1500),n_iter]
X_2 = c_[log_errors]
p_2=lstsq(M_2,X_2,rcond=None)
fit2 = (exp(p_2[0][0]))*(exp(p_2[0][1]*n_iter))
#Graphs:
figure(5)
semilogy(n_iter,fit1,label = ’fit_1’,linewidth = 3,color = ’b’)
semilogy(n_iter,errors,label = ’original errors’,linewidth = 2)
semilogy(n_iter,fit2,label = ’fit_2’,linewidth = 1)
xlabel(’iteations’)
ylabel(’Error(log)’)
title(’Error and fitted values in semilog plot’)
```

```
legend(loc = 'upper right',fontsize = 10)
show()
```

The values of A and B are found using the least squares approach and are plotted

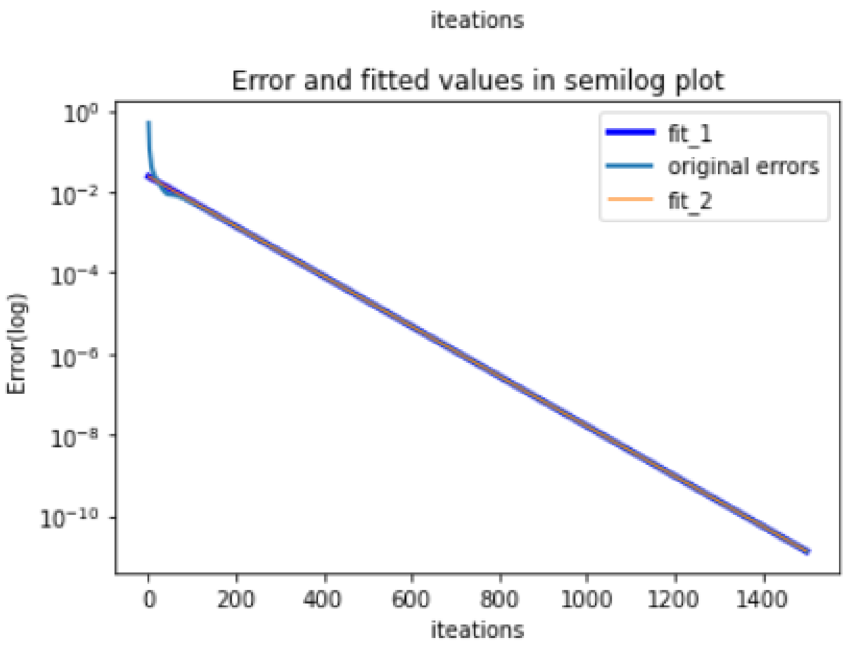


Figure 3: Semilogy plot of error vs iterations

Stopping Condition

We have iterated only Niter number of times and the error seems to decrease exponentially with iterations after certain iterations.We now compute the maximum possible error after Niter iterations if the process continues to infinity

```
#Stopping Condition
error_beyond_N= exp(p_1[0][0]+(p_1[0][1]*Niter))/(-p_1[0][1])#Maximum Error between the function from Niter to that of with infinite value
```

$$Error = -\frac{A}{B}.\exp B(N+0.5) \tag{9}$$

Graphs

Surfacce plot of the potential:

Python Code:

```
#Surface plot of the Potential
fig1=figure(6) # open a new figure
ax=p3.Axes3D(fig1) # Axes3D is the means to do a surface plot
title('The 3-D surface plot of the potential')
surf = ax.plot_surface(Y,-X, phi, rstride=1, cstride=1, cmap=cm.jet)
```

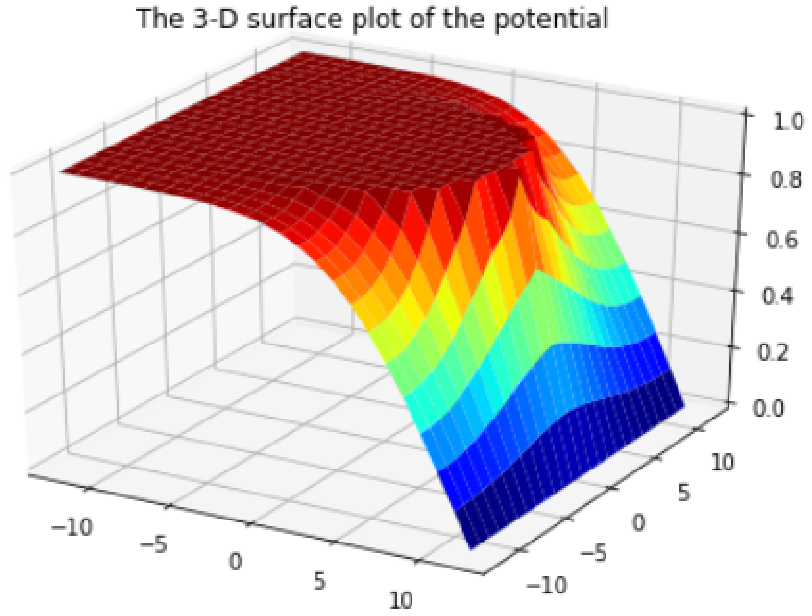


Figure 4: Surface Plot of potential

Contour Plot of the Potential

code:

```
#Contour Plot of Potential
figure(7)
contour(X,-Y,phi,100,cmap=cm.jet)
colorbar()
scatter(ii[0]-Ny//2,ii[1]-Nx//2,s = 2, color = 'r')
xlabel('X(bottom side of Plate)')
ylabel('Y(Left side of Plate)')
title(' Contour plot of Potential after N iterations')
```

```
#legend()
grid()
```

The indices are adjusted so that the graph aligns with the true form of potential in the plate.

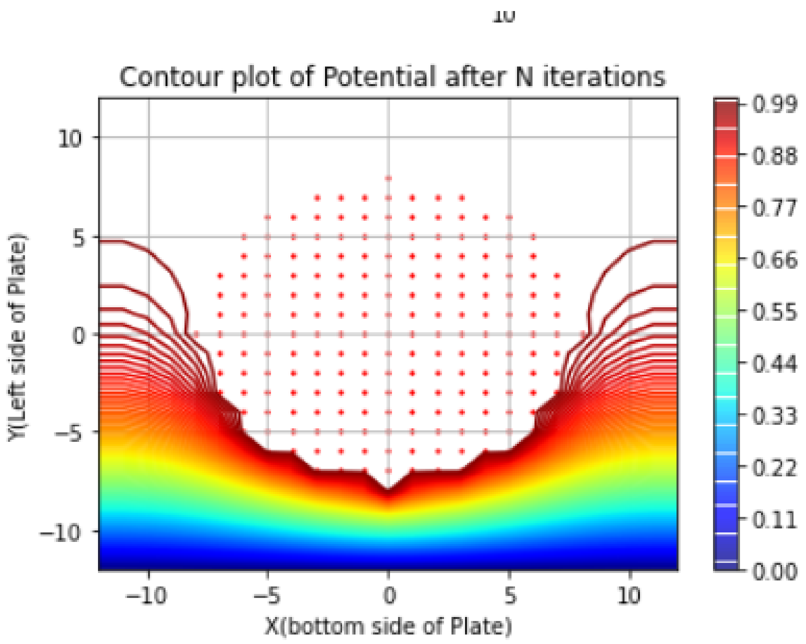


Figure 5: Contour Plot of potential

Vector Plot of Currents:

Taking the value of σ as 1.The equation of currents are :

$$j_x = -\frac{\partial \phi}{\partial x} \tag{10}$$

$$j_y = -\frac{\partial \phi}{\partial y} \tag{11}$$

This Numerically transfers to :

$$j_{x,ij} = 05 * (\phi_{i,j-1} - \phi_{i,j+1}) \tag{12}$$

$$j_{y,ij} = 05 * (\phi_{i-1,j} - \phi_{i+1,j}) \tag{13}$$

Code:

```
#Currents:

J_x = np.zeros((Nx,Ny))#initialising the Values of Currents
J_y = np.zeros((Nx,Ny))
```

```

J_x[1:-1,1:-1] = 0.5*(phi[2:,1:-1] - phi[0:-2,1:-1])#Partial derivative of
    potential with sigma = 1
J_y[1:-1,1:-1] = 0.5*(phi[1:-1,2:] - phi[1:-1,0:-2])
#Vector Plolt of Currents
figure(8)
quiver(y,x,J_y[::-1,:],J_x[::-1,:],label = 'Current')
scatter(ii[0]-Ny//2,ii[1]-Nx//2,s = 2, color = 'r')
xlabel('X(bottom side of Plate)')
ylabel('Y(Left side of Plate)')
title('Current Flow inside the Plate')
legend(loc = 'upper right',fontsize = 10)
```

Graph:

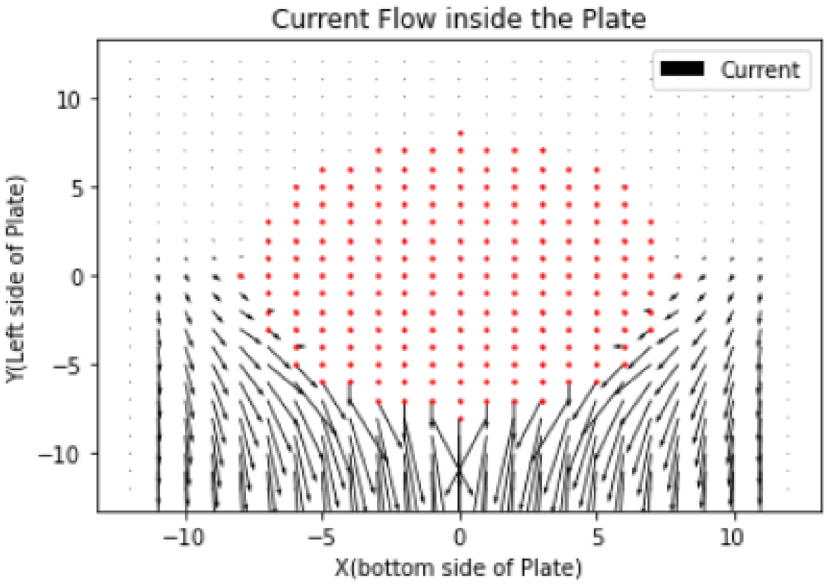


Figure 6: Vector Plot of Currents

From Figure we can see that the normal component of current exists only in the bottom side and is tangential at the remaining edges and most of current flow is located at the bottom of the plate as the value of ϕ saturates to value 1 because of the wire and boundary conditions resulting in a very low gradient.

Temperature

Code:

```
#Temperature:

```

```
Temperature = np.zeros((Nx,Ny))#initialising the Temperature array
ohmic = np.zeros((Nx,Ny))##initialising the ohmic loss array
Temperature = Temperature + 300 #initial Temperature
k = 1 #thermal Conductivity
sigma = 1
delta = 1
const  = (delta**2/(4*sigma*k))
def update_temp(Temperature):
    Temperature[1:-1,1:-1] = 0.25*(Temperature[1:-1,0:-2]+Temperature[1:-1,2:]+
   Temperature[0:-2,1:-1]+Temperature[2:,1:-1])
    ohmic = const*(J_x**2 + J_y**2)#Due to ohmic losses
    Temperature[1:-1,1:-1]  = Temperature[1:-1,1:-1] + ohmic[1:-1,1:-1]
def boundaries_temp(Temperature):
    Temperature[1:-1,[0,-1]] = Temperature[1:-1,[1,-2]]
    Temperature[0] = Temperature[1]
    Temperature[ii] = 300

for i in range(Niter):
    update_temp(Temperature)
    boundaries_temp(Temperature)
figure(9)
contourf(X,-Y,Temperature,cmap=cm.jet)
colorbar()
xlabel('X(bottom side of Plate)')
ylabel('Y(Left side of Plate)')
title('Temperature Contour')
#legend()
```

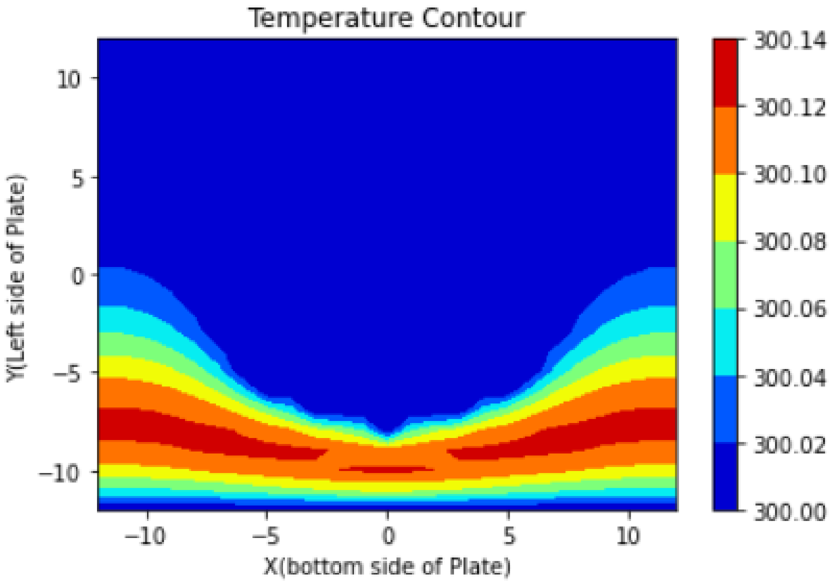


Figure 7: Contour plot of the Temperature

We can observe that the bottom part of the plate is hottest as the currents are mostly concentrated in the bottom region which results in ohmic loss to be high along with the temperature

of boundaries.
The scenario changes if the temperature of wire is higher than that of the edge temperature since the current is nearly zero in the top region the temperature saturates to the temperature of wire
For example if temp of wire is 300.8 instead of 300K

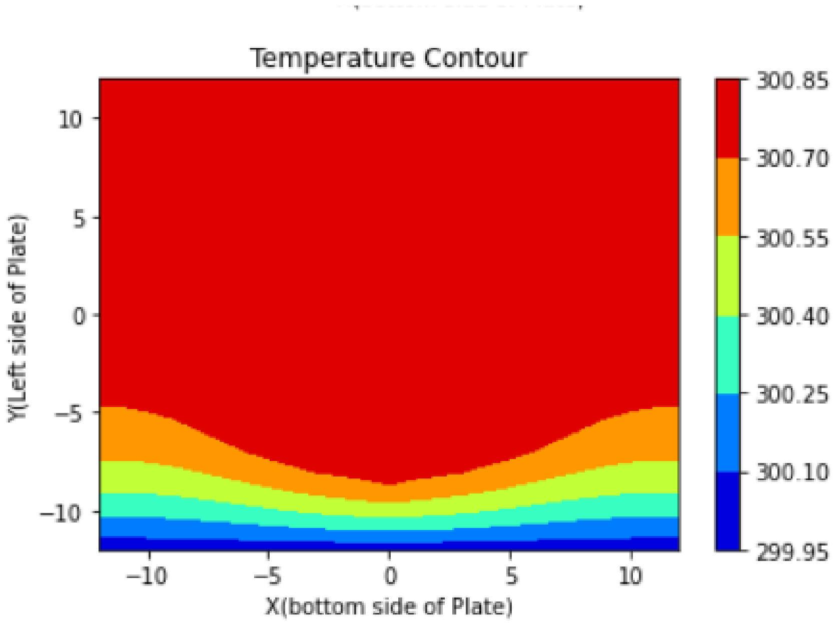


Figure 8: Contour plot of the Temperature with T(wire) = 300.8

Here the boundary condition dominates over ohmic losses.