Suman Patra, Assistant Professor, Department of Physics,

Netaji Nagar Day College

Topic for

Semester – 4, Paper – PHSA CC8

SOLVING LAPLACE EQUATION NUMERICALLY

We will now discuss the numerical solution of Laplace equation for electrical potentials in a certain region of space, knowing its behaviour or value at the border of said region.

We already know that Laplace equation is used to represent various physical problems dealing with the potential of an unknown variable.

Generalized form of Laplace equation can be written as -

$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$, where u is the potential of an unknown variable.

So, Laplace equation for electrical potential can be written as –

$\boldsymbol{\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0}$, where **V** is the electrical potential.

Before discussing the solution let us first specify the boundary conditions.

Our region of interest is bounded by $0 < x, y < 1$

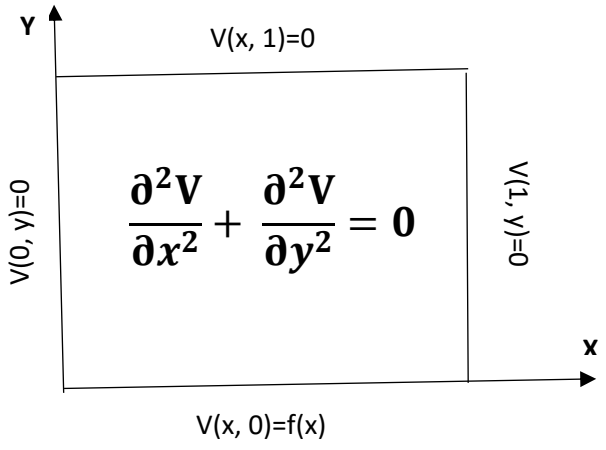
Let us specify the values of electricals potential at the boundary.

At x=0, V = 0 (for any value of y)

At x=1, V = 0 (for any value of y)

At y=0, V = f(x) (for any value of x, f(x) is some function of x)

At y=0, V = 0 (for any value of x)



Let us now proceed with numerical solution that is based on finite difference method.

Again, let us start by writing the equation $\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$

We can write, $\frac{\partial V}{\partial x} = \frac{V(x+\Delta x)-V(x)}{\Delta x}$

Therefore, $\frac{\partial^2 V}{\partial x^2} = \frac{\frac{\partial V}{\partial x}(at\ x) - \frac{\partial V}{\partial x}(at\ x-\Delta x)}{\Delta x}$

$$= \frac{\frac{V(x+\Delta x)-V(x)}{\Delta x} - \frac{V(x)-V(x-\Delta x)}{\Delta x}}{\Delta x} = \frac{V(x+\Delta x) - 2V(x) + V(x-\Delta x)}{(\Delta x)^2}$$

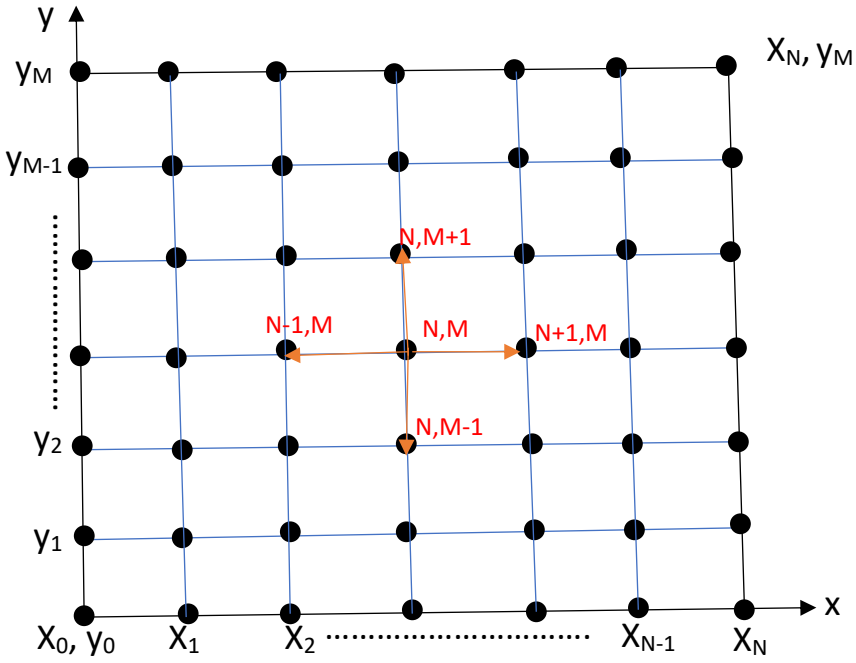
Similarly, $\frac{\partial^2 V}{\partial y^2} = \frac{\frac{\partial V}{\partial y}(at\ y) - \frac{\partial V}{\partial y}(at\ y-\Delta y)}{\Delta y}$

$$= \frac{\frac{V(y+\Delta y)-V(y)}{\Delta y} - \frac{V(y)-V(y-\Delta y)}{\Delta y}}{\Delta y} = \frac{V(y+\Delta y) - 2V(y) + V(y-\Delta y)}{(\Delta y)^2}$$

$$\therefore \frac{V(x+\Delta x) - 2V(x) + V(x-\Delta x)}{(\Delta x)^2} + \frac{V(y+\Delta y) - 2V(y) + V(y-\Delta y)}{(\Delta y)^2} = 0 \qquad (1)$$

We have discretized the equation.

This equation is two dimensional. So, we will treat the region of interest as a mesh of discrete points.



Let us divide the interval 0≤x≤1 into (N+1) equally spaced points.

$\therefore \mathrm{dx} = \frac{1}{N} = \Delta x$

Now $x=0$ is denoted by x_0, $x=0+\Delta x$ is denoted by x_1, ... $x=1$ is denoted by x_{N+1}

Let us also divide the interval 0≤y≤1 into (M+1) equally spaced points.

$\therefore \mathrm{dy} = \frac{1}{M} = \Delta y$

Now $y=0$ is denoted by y_0, $y=0+\Delta y$ is denoted by x_1, ... $y=1$ is denoted by y_{M+1}

Now consider a point (x,y) in the region, which is represented by the discrete point (x_N,y_M) or (N,M) in the mesh.

So, the points (x+Δx,y) and (x−Δx,y) are represented by the discrete points (N+1, M) and (N-1, M) respectively in the mesh.

Now it is easy to understand that the points (x,y+Δy) and (x,y−Δy) are represented by the discrete points (N, M+1) and (N, M-1) respectively in the mesh.

Therefore, the equation (1) can be rewritten as –

$$\frac{\boldsymbol{V(N+1,M) - 2V(N,M) + V(N-1,M)}}{(\Delta \mathbf{x})^2} + \frac{\boldsymbol{V(N,M+1) - 2V(N,M) + V(N,M-1)}}{(\Delta \mathbf{y})^2} = \mathbf{0} \qquad (2)$$

Let's choose, $\Delta x = \Delta y$

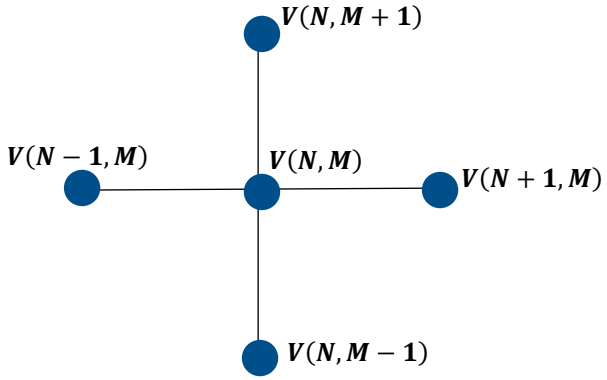
Now we can write,

$$\boldsymbol{V(N+1,M) - 2V(N,M) + V(N-1,M) + V(N,M+1) - 2V(N,M) + V(N,M-1) = 0}$$

$$\boldsymbol{\therefore V(N+1,M) + V(N-1,M) + V(N,M+1) + V(N,M-1) - 4V(N,M) = 0}$$

$$\boldsymbol{\therefore 4V(N,M) = V(N+1,M) + V(N-1,M) + V(N,M+1) + V(N,M-1)}$$

$$\boldsymbol{\therefore V(N,M) = \frac{V(N+1,M) + V(N-1,M) + V(N,M+1) + V(N,M-1)}{4}} \qquad (3)$$



This is known as finite difference stencil that relates V(N,M) to its 4 nearest neighbours.

So, the procedure should be as follows ---

We have to determine the electrical potential V(N,M) at any point (N,M) using the electrical potential of its 4 nearest neighbours.

Using this method, we have to determine electrical potential V at all points in the mesh. We should continue doing this until the electrical potential V at all points attains a steady value.

Let us discuss the procedure again step by step.
Firstly, we have to assign a guess value V_{guess} to electrical potential at all the points in the mesh.
Then using this guess value, we have to determine electrical potential V at all points in the mesh.
In the next iteration, we have to determine electrical potential V at all points in the mesh using the values at previous stage.
We have to do this again and again, until the electrical potential V at all points attains a steady value.
Now the question remains - how do we know that V has got that steady value?
Its super easy. We have to set a desired accuracy ΔV.
For a particular mesh point (Let's say N,M) when the difference between value of V at a stage and value of V at previous stage will become less than ΔV [$V_{k+1} \sim V_k \leq \Delta V, k \ denotes \ no \ of \ iteration$], then we can say that V at that mesh point has achieved the steady value.
When this will be true for all the mesh points then we can say that V has got the steady value at all the points.
Once V at all points reaches the steady value then we can stop our calculation.

Let us denote the value of electrical potential V at the mesh point (N,M) during K-th iteration by $V_{N,M}^{K}$.
Value of electrical potential V at the mesh point (N,M) during (K+1)th iteration can be calculated using equation (3).

$$V_{N,M}^{K+1} = \frac{V_{N+1,M}^{K} + V_{N-1,M}^{K} + V_{N,M+1}^{K} + V_{N,M-1}^{K}}{4}$$

Let us start writing our program.
Let us divide both the x-range 0 < *x*<1 and y-range 0 <*y* <1 into 14 equally spaced points.
So, N=13=M (as N+1=14)

$$\therefore \Delta x = \Delta y = \frac{1}{13}$$

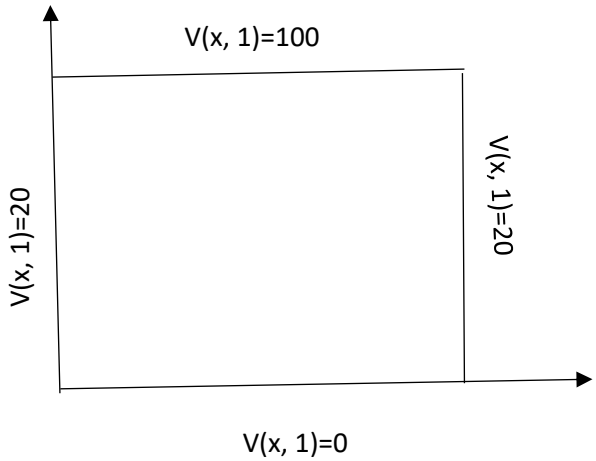
So, we have converted our region into a mesh grid of 14 X 14 points.
Suppose the boundary condition is:

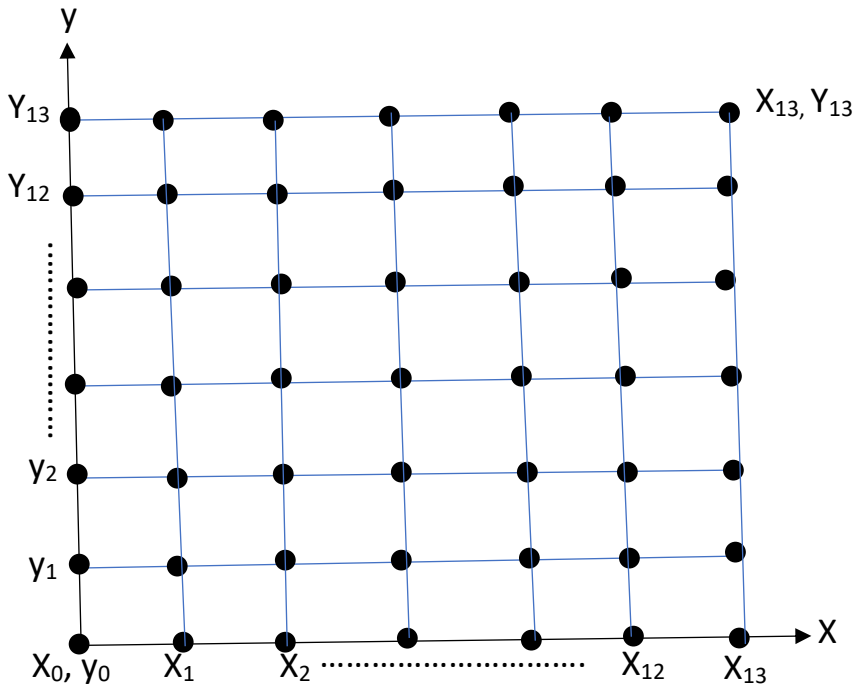
At x=0, V = 20 (for any value of y)

At x=1, V = 20 (for any value of y)

At y=0, V = 0 (for any value of x)

At y=1, V = 100 (for any value of x)





Using the boundary condition we can write,
$V_{0,0}$, $V_{0,1}$, $V_{0,2}$, $V_{0,3}$, ……., $V_{0,12}$, $V_{0,13}$ = 20

$V_{13,0}$, $V_{13,1}$, $V_{13,2}$, $V_{13,3}$, ……., $V_{13,12}$, $V_{13,13}$ = 20

$V_{0,0}$, $V_{1,0}$, $V_{2,0}$, $V_{3,0}$, ……., $V_{12,0}$, $V_{13,0}$ = 0

$V_{0,13}$, $V_{1,13}$, $V_{2,13}$, $V_{3,13}$, ……., $V_{13,13}$, $V_{13,13}$ = 100

*We can assign either 20 or 100 to $V_{13,13}$

Now we will assign a guess value V_{guess} to all other mesh points.

Let's say V_{guess} = 50

Alternatively, we can also assign the guess value to all the mesh points first and then impose the boundary conditions to the border points.

Now we will set V_{steady} =0. Our aim is to update the value of V_{steady}.

Initially V_{steady} is zero.

Hence $\{V_{steady} \sim V_{guess}\} >> \Delta V$

We will now update $V_{i,j}$ at each mesh point using equation (3) and these values will be assigned to V_{steady}.

Now if we see $\{V_{steady} \sim V_{guess}\} \leq \Delta V$ then we will exit the loop; else we will set $V_{guess} = V_{steady}$ and go to next iteration.

Now let's choose our favourite language and develop the source code.

Python Code

```
# Simple Numerical Laplace Equation Solution using Finite Difference Method
import numpy as np
import matplotlib.pyplot as plt

# Set Dimension and delta
lenX = lenY = 14
delta = 1

# Boundary condition
Vtop = 100
Vbottom = 0
Vleft = 20
Vright = 20

# Initial guess of interior grid
Vguess = 50

# Set colour interpolation and colour map
colorinterpolation = 20
colourMap = plt.cm.coolwarm

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))

# Set array size and set the interior value with Tguess
V = np.empty((lenX, lenY))
V1 = np.empty((lenX, lenY))
V.fill(Vguess)

# Set Boundary condition
V[(lenY-1):, :] = Vtop
V[:1, :] = Vbottom
V[:, (lenX-1):] = Vright
V[:, :1] = Vleft


#Set Tsteady as zero to find out the convergence
Vsteady=0

#set initial no of iteration as zero
n=0

while True:
        acc=abs(Vsteady-Vguess)
        if(acc<=0.000001):
                break
        Vguess=Vsteady
        n=n+1
        for i in range(1, lenX-1, delta):
                for j in range(1, lenY-1, delta):
```

```
                    V[i, j] = 0.25 * (V[i+1][j] + V[i-1][j] + V[i][j+1] + V[i][j-1])
                    Vsteady=V[i,j]

print('Potential Mesh')
for i in range(0,lenX):
   V1[i,:]=V[(lenY-1)-i,:]

np.savetxt('suman-laplace.txt',V1,fmt='%.2f')
print(np.around(V1, decimals=0))

print('Total no of Iteration : ',n)

# Configure the contour
plt.title("Contour of Potential")
plt.contourf(X, Y, V, colorinterpolation, cmap=colourMap)

# Set Colorbar
plt.colorbar()
plt.show()
```

Output

Potential Mesh
[[20. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 20.]
 [20. 59. 74. 81. 85. 86. 87. 87. 86. 85. 81. 74. 59. 20.]
 [20. 43. 57. 66. 71. 73. 75. 75. 73. 71. 66. 57. 43. 20.]
 [20. 35. 46. 54. 59. 62. 64. 64. 62. 59. 54. 46. 35. 20.]
 [20. 30. 39. 45. 50. 53. 54. 54. 53. 50. 45. 39. 30. 20.]
 [20. 27. 33. 38. 42. 44. 45. 45. 44. 42. 38. 33. 27. 20.]
 [20. 25. 29. 33. 35. 37. 38. 38. 37. 35. 33. 29. 25. 20.]
 [20. 23. 26. 28. 30. 31. 32. 32. 31. 30. 28. 26. 23. 20.]
 [20. 21. 23. 24. 25. 26. 26. 26. 26. 25. 24. 23. 21. 20.]
 [20. 20. 20. 20. 20. 21. 21. 21. 21. 20. 20. 20. 20. 20.]
 [20. 18. 17. 16. 16. 16. 16. 16. 16. 16. 16. 17. 18. 20.]
 [20. 15. 13. 11. 11. 11. 10. 10. 11. 11. 11. 13. 15. 20.]
 [20. 11.   7.   6.   6.   5.   5.   5.   5.   6.   6.   7. 11. 20.]
 [20.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0. 20.]]
Total no of Iteration : 182