EE6506 - Computational Electromagnetics
Fall 2014
Lecture-2
Boundary conditions in electrostatics

Ananth Krishnan, *Electrical Engineering, IIT Madras*

September 19, 2014

Space for handwritten notes ↓

Learning Objectives

1. To study different boundary conditions in electrostatics
2. To be able to ascertain appropriate boundary conditions for a given problem
3. Handling material properties and discontinuities in electrostatics
4. To eliminate usage of For loops and vectorize the codes for electrostatics

Boundary conditions

Dirichlet boundary condition

1. It directly specifies the values of the unknown along the boundary of a simulation domain.
2. In engineering applications, it is also referred to as Fixed Boundary Condition.
3. In the previous lecture,we considered the following problem for understanding numerical solution to Laplace's equation

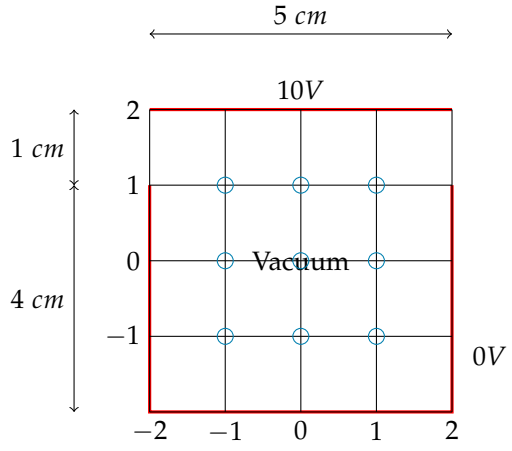

Figure 1: Sample problem from previous lecture

4. The boundaries of this simulation are marked in Red. These boundaries are fixed boundaries where Dirichlet boundary condition is applied. The value of the potential ϕ at $(y = 2)$ is 10 V and ϕ at lower,left and right boundary is fixed at $0V$.

5. The potentials at these points (also referred to as Nodes) on the boundary do not change with number of iterations. They are fixed.

Neumann boundary condition

1. The Neumann boundary condition specifies the value of the derivative of the solution, instead of specifying the solution itself. In this case of electrostatics, while solving Laplace's equation, Neumann's boundary condition may be of the form

$$\frac{\partial \phi}{\partial x} = K_1 \tag{1}$$

or

$$\frac{\partial \phi}{\partial y} = K_2 \tag{2}$$

2. The first equation would mean that the derivative of the potential ϕ along x-direction is constant K_1. If $K_1 = 0$, one can say that the derivative of the potential ϕ along x-direction is zero and that implies that ϕ along x-direction is constant. A slightly more technical term used to describe this, is by saying that the Nodes along the x-direction are Equipotential (meaning that potential is constant).

3. The second equation would mean that the derivative of the potential ϕ along y-direction is constant K_2. If $K_2 = 0$, one can say that the derivative of the potential ϕ along y-direction is zero and that implies that ϕ along y-direction is constant. A slightly more technical term used to describe this, is by saying that the Nodes along the y-direction are Equipotential (meaning that potential is constant).

4. To understand this boundary condition, let us consider the following example of a parallel capacitor. The parallel plate capacitor is made out of a 12 inches diameter Silicon wafer of 650 μm thickness, coated on both sides with aluminum.

5. In order to proceed with the problem, we need to first decide the discretization procedure. More specifically, we need to decide the value of $\Delta x = \Delta y = h$ for this geometry to avoid the denominator in the Laplace's equation.

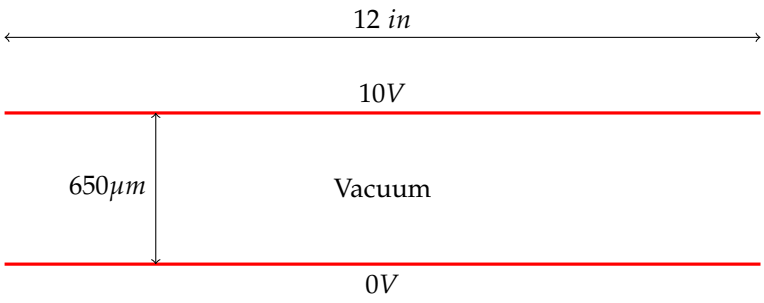

Figure 2: Parallel plate capacitor

6. The structure seems very large along x-axis and very small along y-axis. Let us say that we would like to have 100 nodes along y-axis. This would mean that $\Delta y = \frac{650\mu m}{100} = 6.5\mu m$, which naturally forces $\Delta x = 6.5\mu m = h$.

7. Now, the size of the array needed to calculate would be $N_x = \frac{12in}{6.5\mu m} \approx 46,892$, $N_y = 100$. Now, this is a huge matrix and to calculate the potentials using say 1,000 iterations, it would take a lot of time and consumes a lot of memory to store and manipulate the array.

8. In such cases, certain physical approximations may be made to reduce the size of the problem considerably. While these approximations do not guarantee accurate solutions, the results may be close.

9. In this example, we may choose a small region at the center of the capacitor to perform the simulation. It is alright to say that, in this problem, if one obtains the potential distribution in that small region at the center, the potential distribution in other regions is identical to the calculated distribution at the center.

10. This is shown in the figure below. The dashed box represents the region of simulation.

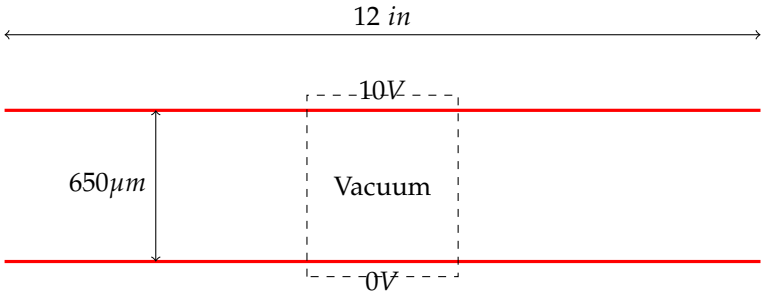

Figure 3: Parallel plate capacitor

11. In the top and bottom of this region, Dirichlet conditions (at top boundary, $10V$, and bottom boundary $0V$) should be applied.

On the left and right side of this dashed box, we need to apply Neumann's boundary condition (i.e. $\frac{\partial \phi}{\partial x} = 0$). This would mean that along the x-direction, the structure is equipotential.

12. The physical meaning of this boundary condition is that we neglect what are known as Fringing Fields. These are the electric field lines that start from the positive plate of the capacitor, travel through outside medium (like air/vacuum) and terminate again on negative plate of capacitor. Details of this may be studied in elementary electrostatics text.

13. The diagramatic representation of the boundary condition implemented in the Laplace's equation solver is shown below.

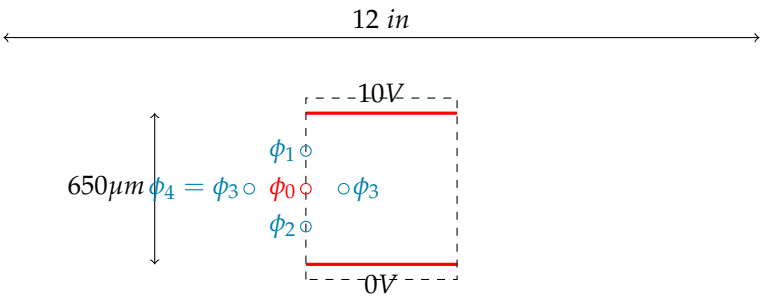

Figure 4: Parallel plate capacitor

14. In the figure, ϕ_4 is outside the simulation domain, but we need it to calculate the value of ϕ_0. Since $\frac{\partial \phi}{\partial x} = 0$, we can take $\phi_4 = \phi_3$ to solve the problem.

15. Note that the Homogeneous Neumann Boundary condition is a dynamic condition unlike Dirichlet condition. In the case of Dirichlet Boundary condition, the values of the potential at the boundary are fixed irrespective of the iteration. However, in the case of Neumann Boundary condition, the potential value at the boundary changes with every iteration.

Handling material inhomogeneities

Simple two layer system

1. In order to model real life systems, one has to carefully consider structures made of different materials.

2. In electrostatics, different materials corresponds to materials having different dielectric constants. How do we calculate the potential and fields in such structures?

3. Consider the following scenario wherein nodes are present at the interface of two materials with dielectric constants ϵ_1 and ϵ_2.

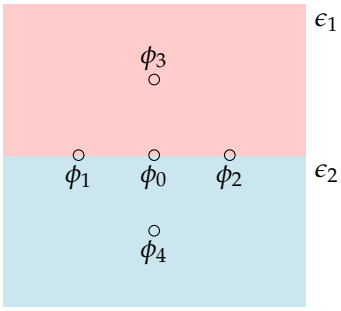

Figure 5: 2 layer material inhomogeneity

4. The points corresponding to ϕ_0, ϕ_1 and ϕ_2 are present at the interface between the two materials and our objective is to find the equation of ϕ_0.

5. Assumption made here are that there are no free charges on the interface between the two materials. Now, Gauss's law can be used to determine ϕ_0.

6. Let us consider a box shaped Gaussian surface around the point ϕ_0.

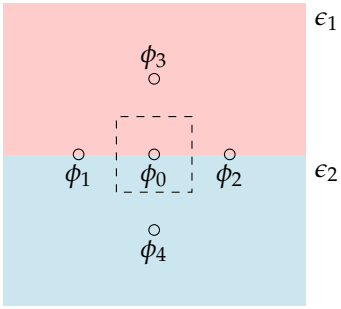

Figure 6: 2 layer material inhomogeneity

7. According to Gauss's law, the net electric flux leaving the surface is equal to the charge enclosed by the surface and that charge in this case is zero.

$$\oiint \epsilon E \cdot ds = q \quad (3)$$

Here q=o, hence,

$$\oiint \epsilon E \cdot ds = 0 \quad (4)$$

Substituting E= $-\nabla\phi$, we get

$$-\oint \epsilon \nabla\phi \cdot dc = 0 \quad (5)$$

where the surface integral has been replaced with closed contour c

The derivative of ϕ can be replaced by $\frac{\partial \phi}{\partial n}$, where n is the outward normal unit vector to contour. We obtain

$$-\oint \epsilon \frac{\partial \phi}{\partial n} dc = 0 \quad (6)$$

To proceed further, we need to treat the 4 sides of the Gaussian surface one at a time and write down the flux crossing the sides. The sum of all these fluxes is zero due to no net charge.

Flux leaving the right side of the contour is

$$\psi_R = \epsilon_1 \frac{\phi_2 - \phi_0}{h} \frac{h}{2} + \epsilon_2 \frac{\phi_2 - \phi_0}{h} \frac{h}{2} \quad (7)$$

Flux leaving the left side of the contour is

$$\psi_L = \epsilon_1 \frac{\phi_1 - \phi_0}{h} \frac{h}{2} + \epsilon_2 \frac{\phi_1 - \phi_0}{h} \frac{h}{2} \quad (8)$$

Flux leaving the top side of the contour is

$$\psi_T = \epsilon_1 \frac{\phi_3 - \phi_0}{h} h \quad (9)$$

Flux leaving the bottom side of the contour is

$$\psi_T = \epsilon_2 \frac{\phi_4 - \phi_0}{h} h \quad (10)$$

Setting the total flux leaving the Gaussian surface to zero, we obtain,

$$\psi_R + \psi_L + \psi_T + \psi_B = 0 \quad (11)$$

Substituting and rearranding the terms, we get,

$$\frac{\epsilon_1 + \epsilon_2}{2} \phi_0 = \frac{1}{4} \left(\frac{\epsilon_1 + \epsilon_2}{2} \phi_1 + \epsilon_1 \phi_2 + \frac{\epsilon_1 + \epsilon_2}{2} \phi_3 + \epsilon_2 \phi_4 \right) \quad (12)$$

8. The equation 12 shows that for nodes 0,1 and 3 with potentials ϕ_0, ϕ_1 and ϕ_3, the average permittivity across the interface has to be considered in the calculation. If the permittivities are identical, then this equation reduces to the Finite difference version of Laplace's equation in a homogeneous medium.

Nodes at a corner

1. Consider the following situation represented in the figure,
2. Using the same procedure described in the previous section, we can form a Gaussian surface, and setting the total flux leaving the Gaussian surface to be zero,

$$\epsilon_1 \frac{\phi_1 - \phi_0}{h} \frac{h}{2} + \epsilon_2 \frac{\phi_1 - \phi_0}{h} \frac{h}{2} + \epsilon_1 \frac{\phi_2 - \phi_0}{h} h + \epsilon_1 \frac{\phi_3 - \phi_0}{h} h + \epsilon_1 \frac{\phi_4 - \phi_0}{h} \frac{h}{2} + \epsilon_2 \frac{\phi_4 - \phi_0}{h} \frac{h}{2} \quad (13)$$

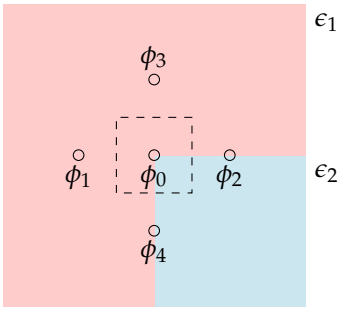

Figure 7: 2 layer material inhomogeneity

Rearranging the terms, we get,

$$\frac{3\epsilon_1+\epsilon_2}{4}\phi_0 = \frac{1}{4}\left(\frac{\epsilon_1+\epsilon_2}{2}\phi_1 + \epsilon_1\phi_2 + \epsilon_1\phi_3 + \frac{\epsilon_1+\epsilon_2}{2}\phi_4\right) \tag{14}$$

Assigning update equations for nodes on any corner or interface

1. Update equations for nodes at any interface (edge or corner) has to be carefully dealt with as a special case. Suppose in a square simulation region, the right side boundary is assigned to be Neumann's boundary and top is also a Neumann's boundary, then the top right corner node, has to be updated accordingly (It should in this case be an average of the potentials at the nodes at the left and below it).
2. Try to use conditional updates for all the special cases using Gauss's law.

Examples of codes with Dirichlet and Neumann's Boundary condition

Example-1

1. The Matlab code for a similar problem is shown below. The problem description can be found in the program comments.
2. The program allows one to view the potential distribution as iterations proceed.
3. At the end, the electric field distribution is calculated as a gradient of the finally calculated potential.

```
%% Computational Electromagnetics Assignment 1
% Written for Course :- Computational Electromagnetics, Fall 2012
%                       Department of Electrical Engineering
%                       Indian Institute of Technology Madras
%                       Chennai - 600036, India
```

```
%
% Authors            :- Nithin Sivadas, Dual Degree. Aerospace Engg.
%
% Instructor :- Ananth Krishnan
%               Assistant Professor
%               Department of Electrical Engineering
%               Indian Institute of Technology Madras

% Assignment 1: "Job of an Integration Engineer: Solving Finite Difference"
% Assignment Date: August 3, 2012

%% Introduction
% *Case 1*
%
% Objective of the program is to solve for the steady state voltage
% distribution in a region 0<x<40, 0<y<40, filled with a dielectric medium
% with one side at a voltage of 10 Volts and the opposite side maintained
% at 0 Volts.
%
% At any iteration, the value of voltage is updated as average of voltages
% of 4 nearest neighbors, until between consecutive iterations, the error
% is less than 0.001 V.
%
% The tolerance in error between iterations is kept at 0.0001 V. This may be
% tweaked to a higher or lower value for lower or higher accuracy
% respectively. Imagesc command by default uses image axis settings, which
% are different from normal plot command and hence x and y axis may look
% flipped. Read Matlab documentation on imagesc for more details.
%
% Program stops when iteration number in plot does not change or can be
% closed anytime by just closing the plot window. On normal completion, the
% program plots the electric field in a quiver plot.


%% Initialising Matrices

%Clearing variables in memory and Matlab command screen
clear all;
clc;

%Dimensions of the simulation grid in x (xdim) and y (ydim) directions
xdim = 40;
ydim = 40;

```

```
% Dimension of the semiconductor
size_sc=40;

%Initializing previous (V_prev) and present (V_now) voltage matrices
V_now=zeros(xdim+1,ydim+1);
V_prev=zeros(xdim+1, ydim+1);

%Initializing permittivity matrix to define permittivity of regions
epsilon=zeros(xdim+1, ydim+1);
epsilon(:,:)=12;
%epsilon((xdim/2)-(size_sc/2):1:(xdim/2)+(size_sc/2),(ydim/2)-(size_sc/2):1:(ydim/2)+(
    size_sc/2))=12;

%Initializing an array defining the length of a metal contact
k =(xdim/2)-(size_sc/2)+1:1:(xdim/2)+(size_sc/2)+1; %Full width contact

%Applying Dirichlet boundary condition for the metal contacts
V_now(k,(ydim/2)+(size_sc/2)+1)=10;
V_now(k,(ydim/2)-(size_sc/2)+1)=0;

%Iteration counter
iter=0;

%Calculation of maximum error between V_now and V_prev at all points
%By setting the applied voltage for only V_now, we have made V_no and
%V_prev different, hence error will be greater than zero and the program
%will enter the while loop following this command.
error=max(max(abs(V_now-V_prev)));

%% Iteration loop
while (error>0.001)

    iter = iter + 1; % Iteration counter increment

    %Applying Neuman boundary conditions on the sides
    V_now(xdim+1,1:1:ydim+1)=V_now(xdim-1,1:1:ydim+1);
    V_now(1,1:1:ydim+1)=V_now(3,1:1:ydim+1);

    % Updating present iteration using 4 point Central diffrence form
    % of Laplace equation obtained using Finite Difference method
    V_now(2:1:xdim,2:1:ydim)=0.5*((epsilon(3:1:xdim+1,3:1:ydim+1)+epsilon(3:1:xdim+1,1:1:
        ydim-1) + epsilon(1:1:xdim-1,3:1:ydim+1) + epsilon (1:1:xdim-1,1:1:ydim-1)).^-1).*(
        V_now(1:1:xdim-1,2:1:ydim).*(epsilon(1:1:xdim-1,3:1:ydim+1)+epsilon(1:1:xdim-1,1:1:
        ydim-1)) + V_now(3:1:xdim+1,2:1:ydim).*(epsilon(3:1:xdim+1,3:1:ydim+1)+epsilon(3:1:
```

```
        xdim+1,1:1:ydim-1)) + V_now(2:1:xdim,1:1:ydim-1).*(epsilon(1:1:xdim-1,1:1:ydim-1)+
        epsilon(3:1:xdim+1,1:1:ydim-1)) + V_now(2:1:xdim,3:1:ydim+1).*(epsilon(1:1:xdim
        -1,3:1:ydim+1)+epsilon(3:1:xdim+1,3:1:ydim+1)));

    error=max(max(abs(V_now-V_prev)));% Estimating the maximum error between V_now and
        V_prev at all points
    V_prev=V_now; % Updating previous iteration matrix to the last iteration performed
    e(iter)=error*100/max(max(abs(V_now))); %Storing the error values for each iteration

    %The following code can be used to display a movie showing the evolution of
    %the potential distribution over each iteration.
    %imagesc(flipud(V_now')); colorbar;
    %title (['Voltage distribution on a ', int2str(xdim),' X ', int2str(ydim), ' grid at
        iteration no ', int2str(iter)], 'Color', 'k');
    %getfigure;
end;

%% Plotting Results
% Color map of the potential distribution in the grid
figure;
imagesc(flipud(V_now')); colorbar;
title (['Voltage distribution on a ', int2str(xdim),' X ', int2str(ydim), ' grid at
    iteration no ', int2str(iter)], 'Color', 'k');
xlabel('X-axis'); ylabel('Y-axis');

% Color map of the material permittivity distribution in the grid
figure;
imagesc(flipud(epsilon'));colorbar;
title (['Material distribution (permittivity)']);
xlabel('X-axis'); ylabel('Y-axis');

% Plot of the Maximum Percentage Error between V_now and V_prev at all
% points w.r.t each iteration
figure;
plot(e);
set(gca,'Xscale','log');
title (['Maximum percentage error between present and previous values of voltage']);
xlabel('Number of iterations'); ylabel('Maximum percentage error in the potential
    distribution (%)');

% Plot the electric field distribution
[ex,ey]=gradient(flipud(V_now(1:1:xdim+1,1:1:ydim+1)'));
figure;
imagesc((ex.^2+ey.^2).^0.5); colorbar; % Ploting color map of electric field intensity
```

```
127 hold on;
128 quiver(-ex,-ey); %Quiver command creates a plot, E=-grad(V), hence the negative sign
129 title (['Electric field distribution on a ',int2str(xdim), ' X ', int2str(ydim), ' grid']);
130 xlabel('X-axis'); ylabel('Y-axis');
```

Example-2

```
1  %% Computational Electromagnetics Assignment 1
2
3  %% Background
4  % Written for Course :- Computational Electromagnetics, Fall 2012
5  %                       Department of Electrical Engineering
6  %                       Indian Institute of Technology Madras
7  %                       Chennai - 600036, India
8  %
9  % Authors             :- Nithin Sivadas, Dual Degree. Aerospace Engg.
10 %
11 % Instructor :- Ananth Krishnan
12 %               Assistant Professor
13 %               Department of Electrical Engineering
14 %               Indian Institute of Technology Madras
15 %
16 % *Assignment 1:*
17 % "Job of an Integration Engineer: Solving Finite Difference"
18 % Assignment Date: August 3, 2012
19
20 %% Introduction
21 % *Case 2*
22 %
23 % Objective of the program is to solve for the steady state voltage
24 % distribution in a region 0<x<100, 0<y<100, filled with air and
25 % a dielectric medium in the region 20<x<60, 20<y<60 with one side
26 % at a voltage of 10 Volts and the opposite side maintained at 0 Volts.
27 %
28 % At any iteration, the value of voltage is updated as average of voltages
29 % of 4 nearest neighbors, until between consecutive iterations, the error
30 % is less than 0.001 V.
31 %
32 % The tolerance in error between iterations is kept at 0.0001 V. This may be
33 % tweaked to a higher or lower value for lower or higher accuracy
34 % respectively. Imagesc command by default uses image axis settings, which
35 % are different from normal plot command and hence x and y axis may look
36 % flipped. Read Matlab documentation on imagesc for more details.
```

```
%
% Program stops when iteration number in plot does not change or can be
% closed anytime by just closing the plot window. On normal completion, the
% program plots the electric field in a quiver plot.

%% Initialising Matrices

%Clearing variables in memory and Matlab command screen
clear all;
clc;

%Dimensions of the simulation grid in x (xdim) and y (ydim) directions
xdim = 100;
ydim = 100;

% Dimension of the semiconductor
size_sc=40;

%Initializing previous (V_prev) and present (V_now) voltage matrices
V_now=zeros(xdim+1,ydim+1);
V_prev=zeros(xdim+1, ydim+1);

%Initializing permittivity matrix to define permittivity of regions
epsilon=zeros(xdim+1, ydim+1);
epsilon(:,:)=1;
epsilon((xdim/2)-(size_sc/2):1:(xdim/2)+(size_sc/2),(ydim/2)-(size_sc/2):1:(ydim/2)+(
    size_sc/2))=12;

%Initializing an array defining the length of a metal contact
k =(ydim/2)-(size_sc/2):1:(ydim/2)+(size_sc/2); %Full width contact

%Applying Dirichlet boundary condition for the metal contacts
V_now(k,(ydim/2)+(size_sc/2))=10;
V_now(k,(ydim/2)-(size_sc/2))=0;

%Iteration counter
iter=0;

%Calculation of maximum error between V_now and V_prev at all points
%By setting the applied voltage for only V_now, we have made V_no and
%V_prev different, hence error will be greater than zero and the program
%will enter the while loop following this command.
error=max(max(abs(V_now-V_prev)));

```

```
%% Iteration loop
while (error>0.0001)

    iter = iter + 1; % Iteration counter increment

    %Applying Dirichlet boundary condition for the metal contacts
    V_now(k,(ydim/2)+(size_sc/2))=10;
    V_now(k,(ydim/2)-(size_sc/2))=0;

    %Applying Neuman boundary conditions on the sides of the grid
    V_now(xdim+1,1:1:ydim+1)=V_now(xdim-1,1:1:ydim+1);
    V_now(1,1:1:ydim+1)=V_now(3,1:1:ydim+1);
    V_now(1:1:xdim+1,1)=V_now(1:1:xdim+1,3);
    V_now(1:1:xdim+1,ydim+1)=V_now(1:1:xdim+1,ydim-1);

    % Updating present iteration using 4 point Central diffrence form
    % of Laplace equation obtained using Finite Difference method
    V_now(2:1:xdim,2:1:ydim)=0.5*((epsilon(3:1:xdim+1,3:1:ydim+1)+epsilon(3:1:xdim+1,1:1:
        ydim-1) + epsilon(1:1:xdim-1,3:1:ydim+1) + epsilon (1:1:xdim-1,1:1:ydim-1)).^-1).*(
        V_now(1:1:xdim-1,2:1:ydim).*(epsilon(1:1:xdim-1,3:1:ydim+1)+epsilon(1:1:xdim-1,1:1:
        ydim-1)) + V_now(3:1:xdim+1,2:1:ydim).*(epsilon(3:1:xdim+1,3:1:ydim+1)+epsilon(3:1:
        xdim+1,1:1:ydim-1)) + V_now(2:1:xdim,1:1:ydim-1).*(epsilon(1:1:xdim-1,1:1:ydim-1)+
        epsilon(3:1:xdim+1,1:1:ydim-1)) + V_now(2:1:xdim,3:1:ydim+1).*(epsilon(1:1:xdim
        -1,3:1:ydim+1)+epsilon(3:1:xdim+1,3:1:ydim+1)));

    error=max(max(abs(V_now-V_prev))); % Estimating the maximum error between V_now and
        V_prev at all points
    V_prev=V_now; % Updating previous iteration matrix to the last iteration performed
    e(iter)=error*100/max(max(abs(V_now))); %Storing the error values for each iteration

    %The following code can be used to display a movie showing the evolution of
    %the potential distribution over each iteration.
    %imagesc(flipud(V_now')); colorbar;
    %title (['Voltage distribution on a ', int2str(xdim),' X ', int2str(ydim), ' grid at
        iteration no ', int2str(iter)], 'Color', 'k');
    %getfigure;
end;
%% Plotting Results
% Color map of the potential distribution in the grid
figure;
imagesc(flipud(V_now')); colorbar;
title (['Voltage distribution on a ', int2str(xdim),' X ', int2str(ydim), ' grid at
    iteration no ', int2str(iter)], 'Color', 'k');
xlabel('X-axis'); ylabel('Y-axis');
```

```
115
116 % Color map of the material permittivity distribution in the grid
117 figure;
118 imagesc(flipud(epsilon'));colorbar;
119 title (['Material distribution (permittivity)']);
120 xlabel('X-axis'); ylabel('Y-axis');
121
122 % Plot of the Maximum Percentage Error between V_now and V_prev at all
123 % points w.r.t each iteration
124 figure;
125 plot(e);
126 set(gca,'Xscale','log');
127 title (['Maximum percentage error between present and previous values of voltage']);
128 xlabel('Number of iterations'); ylabel('Maximum percentage error in the potential
        distribution (%)');
129
130 % Plot the electric field distribution
131 [ex,ey]=gradient(flipud(V_now(3:1:xdim-3,3:1:ydim-3)'));
132 figure;
133 imagesc((ex.^2+ey.^2).^0.5); colorbar; % Ploting color map of electric field intensity
134 hold on;
135 quiver(-ex,-ey); %Quiver command creates a plot, E=-grad(V), hence the negative sign
136 title (['Electric field distribution on a ',int2str(xdim-2), ' X ', int2str(ydim-2), ' grid
        ']);
137 xlabel('X-axis'); ylabel('Y-axis');
```

General Comments on Programming

1. It must be easy to notice that in the above Matlab program, there are NO For loops for calculating the potentials of the unknown nodes.
2. In Matlab (or in Python using NumPy), one can perform such calculations using vectorization. Instead of using loops, one could compute the potential values directly by parsing all the array indices directly as shown in the program above.
3. It will be easy to notice that, this program had more iterations than the program in Lecture-1, however, it ran much faster than that program.
4. There are many ways of achieving faster computing than this. Users of Python have repeatedly shown that using Inline C in Python can decrease compute times significantly. The readers are

advised to go through the keywords, NumPy, Sweave, Inline in Python forums for more information.

5. In the above 2 examples, vectorization has been used in Matlab (equivalent codes in NumPy can be made by minor modifications).

6. In this course, vectorization is mandatory and elimination of as many For Loops as possible for a given problem is mandatory.

7. You may be awarded bonus points for going out of the way to improve speeds of computation. To receive these bonus points, you need to prove that you have made the codes faster by comparing run-times.