ELEC 4700 Assignment 2
Finite Difference Method

Table of Contents

Question 1 1
Question 1a) 1
Question 1b) 5
Question 1 Conclusion 10
Question 2 10
Question 2a) 10
Question 2b) 18
Question 2c) 20
Question 2d) 25

Student: Samuel (Wendi) Zhu

Student Number: 101088968

Date: 2/25/2022

Question 1

In Question 1, the electrostatic potential in a rectangular region is solved using finite difference method and compared with the analytical solution.

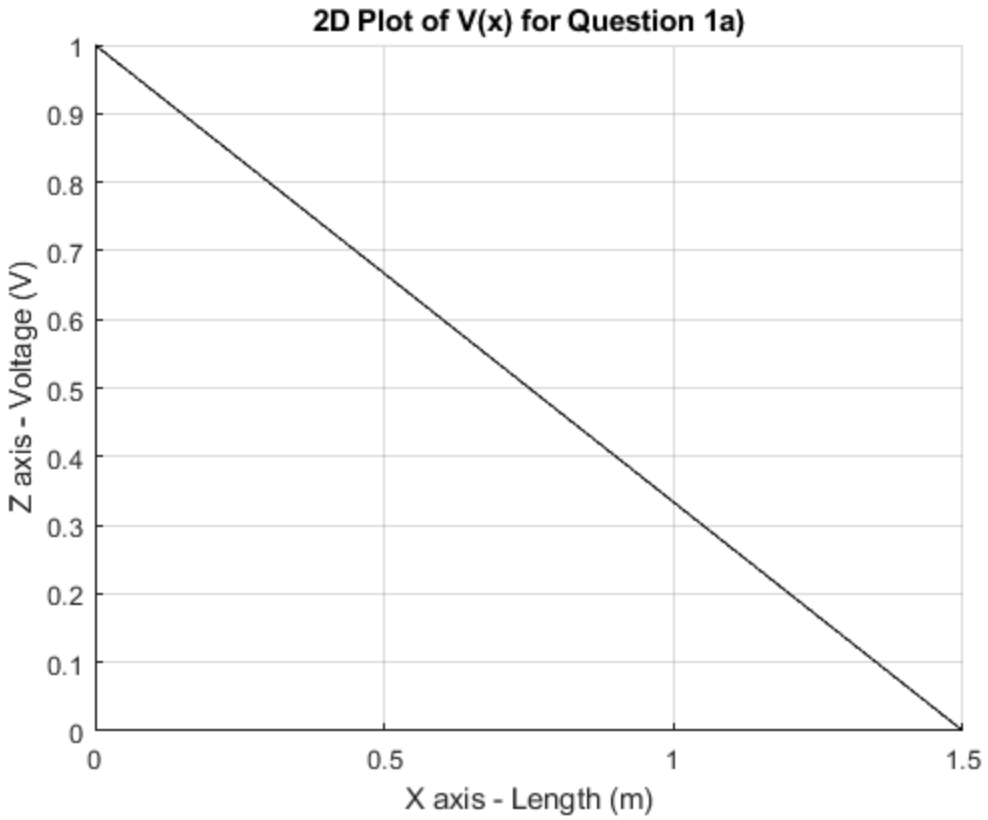
Question 1a)

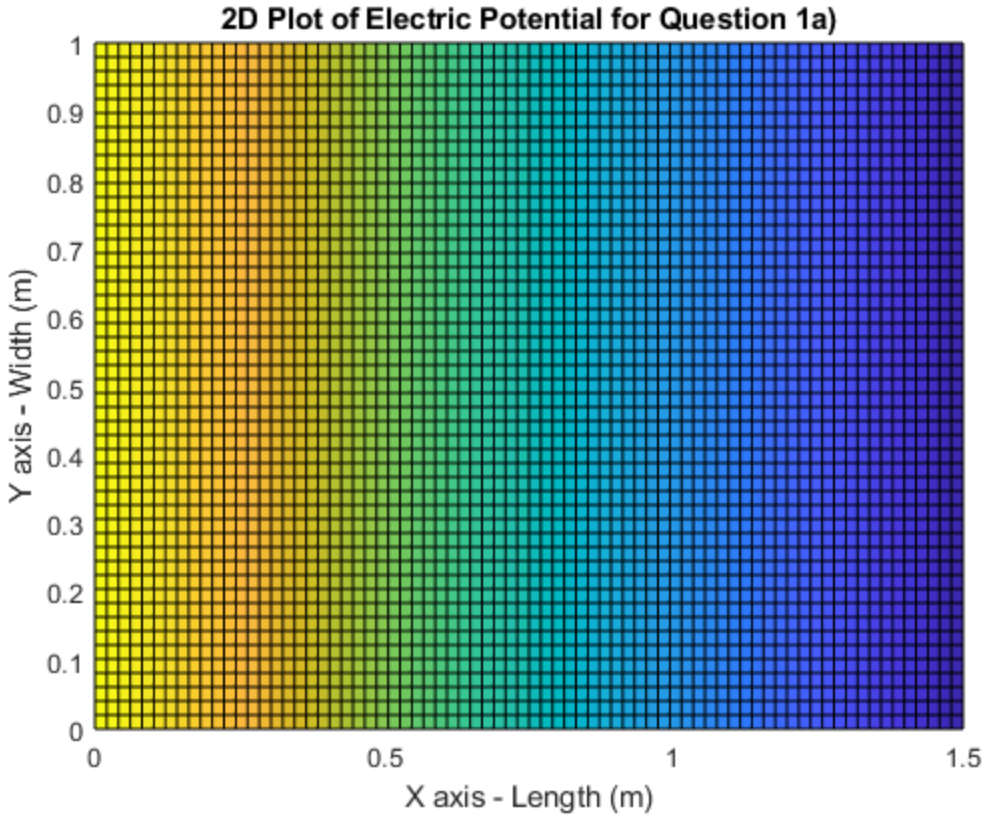
In this part, a simple case where V=V0 at x = 0 and V = 0 at x = L was solved with dV/dy = 0. Assume deltaX = deltaY = 1. The following MATLAB code solves the Question 1a).

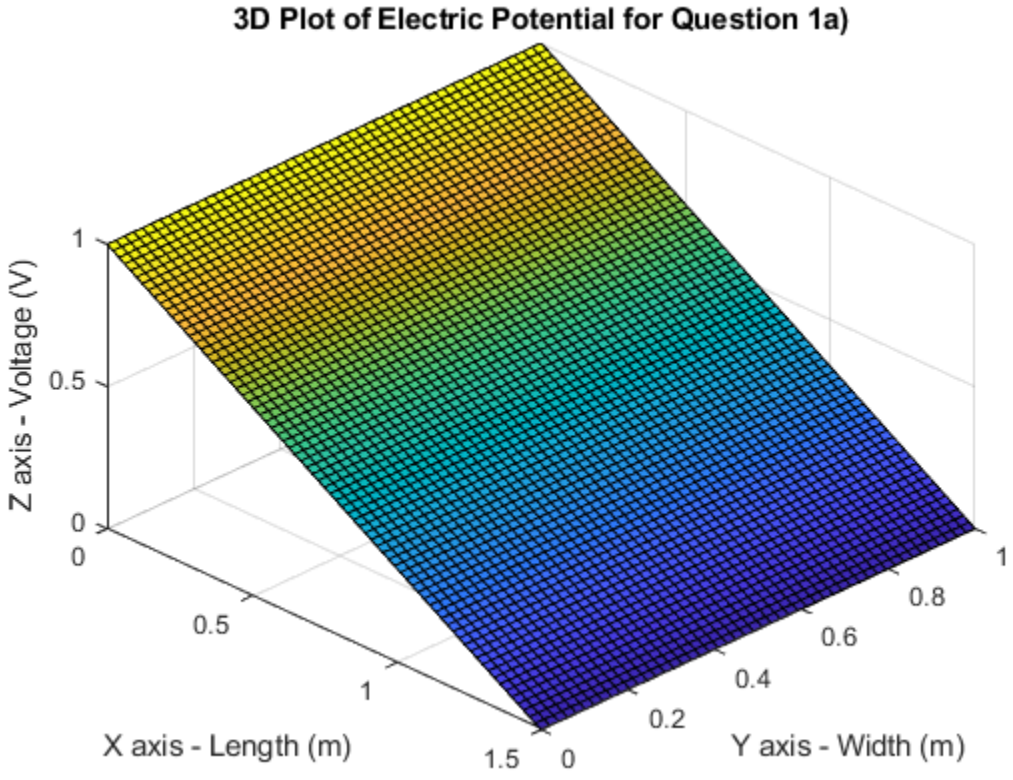
```
% Clear all
clearvars
clearvars -global
close all

% Define the dimension
W = 1;  % Width
L = 3/2 * W;  % Height
deltaXY = 0.02;  % Assume deltaX = deltaY

% Declare the V0
V0 = 1;

% Calculate the dimension of solution matrix
nx = L/deltaXY;
ny = W/deltaXY;
```

```
% GV = F
G = zeros(nx*ny, nx*ny);  % Declare the G matrix
F = zeros(nx*ny, 1);  % Declare the F matrix
matrixSol = zeros(ny, nx);  % Hold the solution matrix

% Constructor the F and G matrix
for ix = 1:nx  % Loop through the width
    for iy = 1:ny  % Loop through the height
        % Calculate the mapping index
        n = mappingEq(ix, iy, ny);
        % Check for boundary condition
        if ix == 1 || ix == nx
            % Setup F matrix
            if ix == 1
                F(n, 1) = V0;  % V = V0 at x = 0
            else
                F(n, 1) = 0;  % V = 0 at x = L
            end
            % Setup G matrix
            G(n, n) = 1;
        else
            % Setup F matrix
            F(n, 1) = 0;
            % Setup G matrix
            % Because dV/dy = 0 => (V(i+1,j) - 2V(i,j) + V(i-1,j))/deltaX^2 =
 0
            G(n, n) = -2/deltaXY^2;
            % Calculate mapping index for V(i+1,j) and V(i-1,j)
            nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
            nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
            G(n, nxp) = 1/deltaXY^2;
            G(n, nxm) = 1/deltaXY^2;
        end
    end
end

% Solve for V from GV = F
V = G\F;

% Map back to the 2D region
for iMap = 1:nx*ny
    % Calculate the index for the 2D region
    ix = ceil(iMap/ny);
    iy = mod(iMap, ny);
    if iy == 0
        iy = ny;
    end
    % Assign the value
    matrixSol(iy, ix) = V(iMap);
end

% Plot the solution
figure(1)  % 2D plot of V(x)
```

```
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
surf(X,Y,matrixSol)
title("2D Plot of V(x) for Question 1a)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
view(0,0)
snapnow
figure(2)  % 2D plot of V(x,y)
surf(X,Y,matrixSol)
title("2D Plot of Electric Potential for Question 1a)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
view(0,90)
snapnow
figure(3)  % 3D plot
surf(X,Y,matrixSol)
title("3D Plot of Electric Potential for Question 1a)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
view(45,45)
snapnow
```




2D Plot of Electric Potential for Question 1a)
Y axis - Width (m)
X axis - Length (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5


3D Plot of Electric Potential for Question 1a)
Z axis - Voltage (V)
X axis - Length (m)
Y axis - Width (m)
0
0.5
1
0
0.5
1
1.5
0
0.2
0.4
0.6
0.8
1

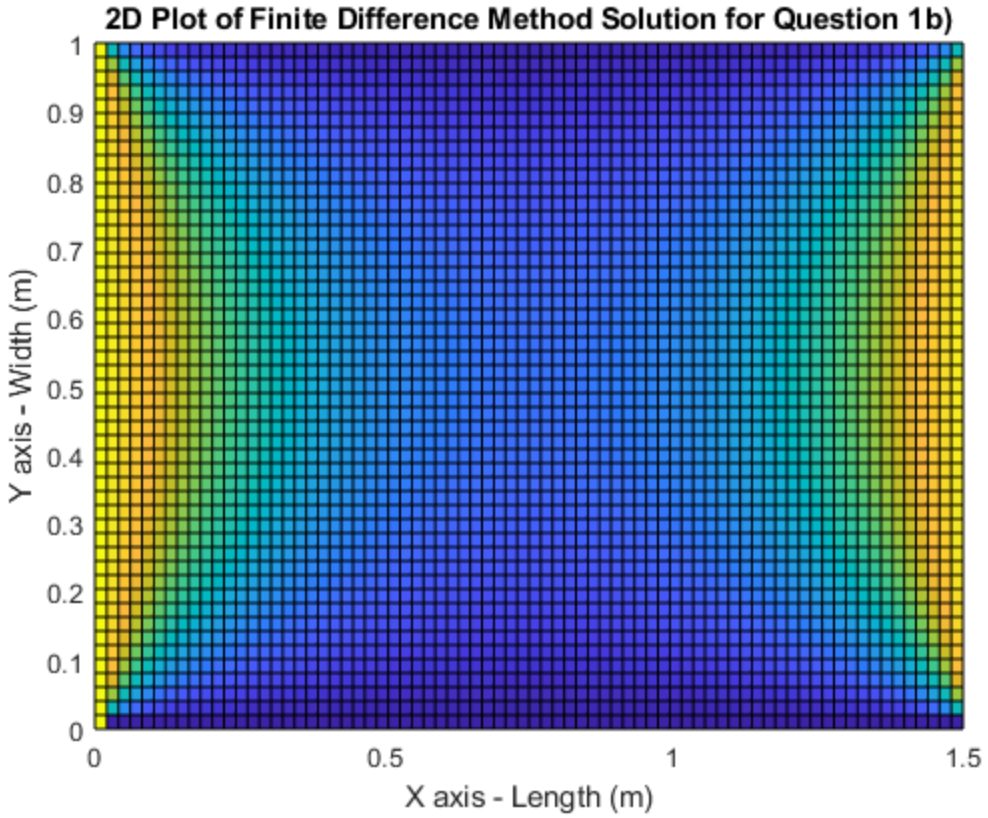
Question 1b)

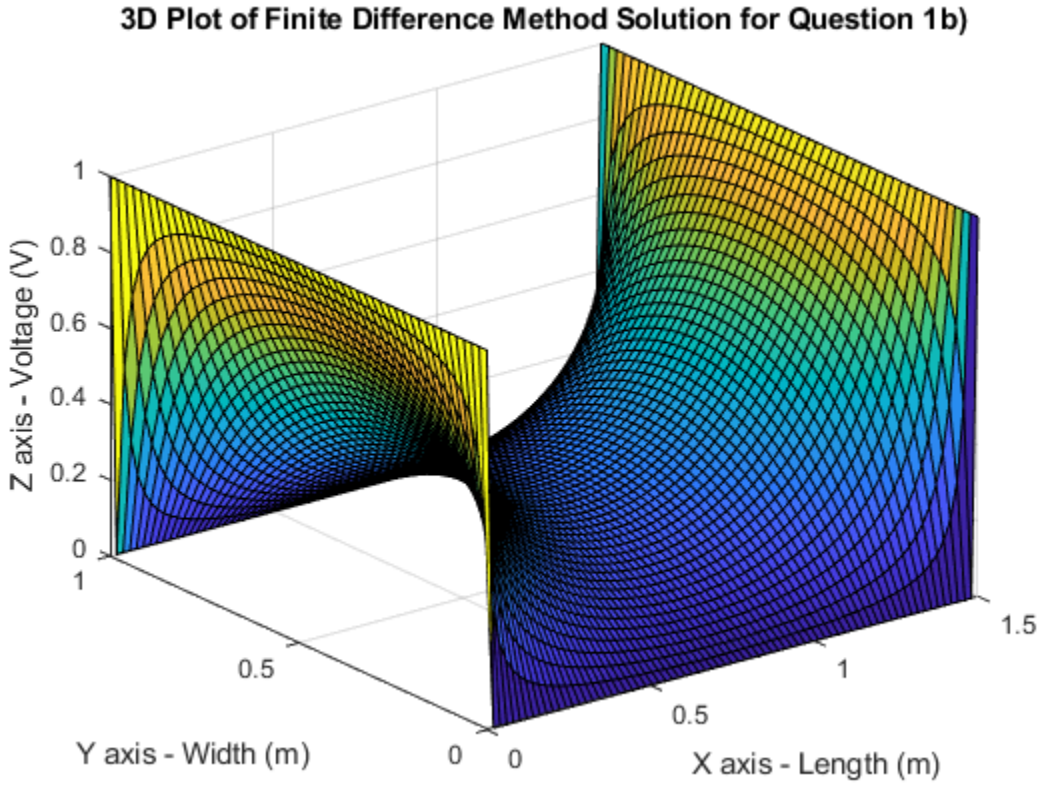
In this part, the case where V=V0 at x=0, x=L and V=0 at y=0, y=W is solved using Finite Difference Method and Analytical series.

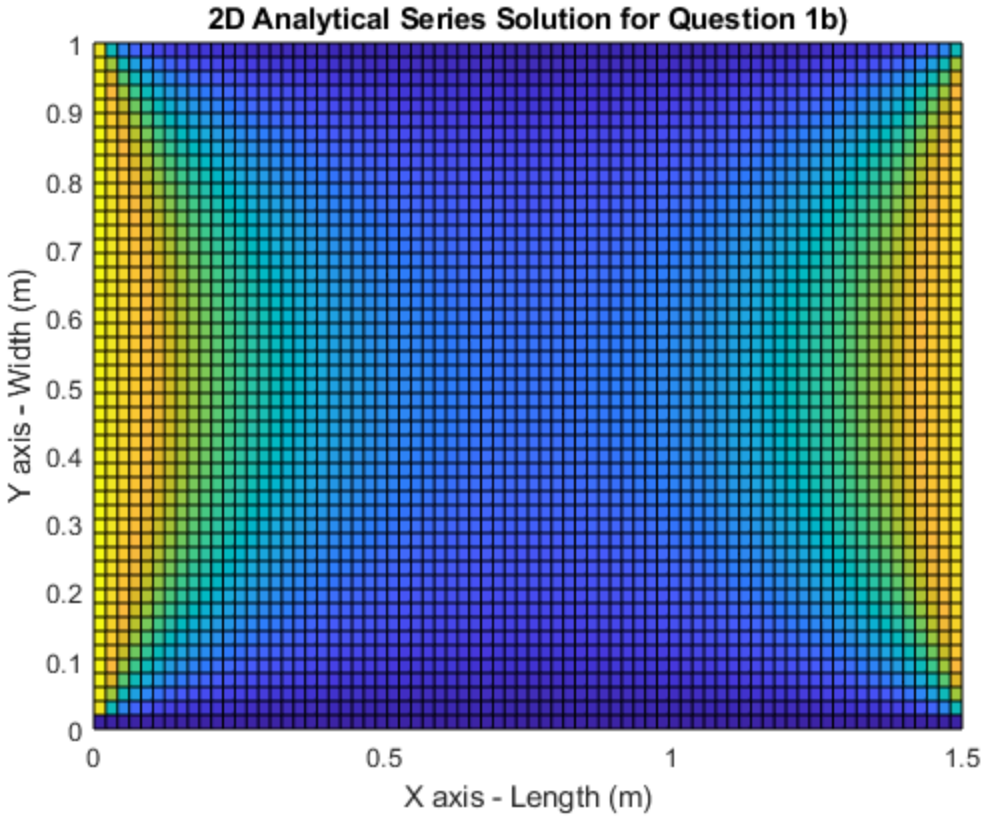
```
% Simulation settings for solving analytical series
simSteps = 100; % Declare the simulation steps
pauseTime = 0.01; % Time paused per simulation step in second

% GV = F
G = zeros(nx*ny, nx*ny);  % Declare the G matrix
F = zeros(nx*ny, 1);  % Declare the F matrix
% Hold the solution matrix for Finite Difference Method (FDM)
matrixSolFDM = zeros(ny, nx);
% Hold the solution matrix for analytical series
matrixSolAnalytic = zeros(ny, nx);

% Constructor the F and G matrix
for ix = 1:nx  % Loop through the width
    for iy = 1:ny  % Loop through the height
        % Calculate the mapping index
        n = mappingEq(ix, iy, ny);
        % Check for boundary condition
        if ix == 1 || ix == nx  % at x = 0 or x = L
            % Setup F matrix
            F(n, 1) = V0;  % V = V0 at x = 0, x = L
            % Setup G matrix
            G(n, n) = 1;
        elseif iy == 1 || iy == ny  % at y = 0 or y = W
            % setup F matrix
            F(n, 1) = 0;  % V = 0 at y = 0, y = W
            % Setup G matrix
            G(n, n) = 1;
        else
            % Setup F matrix
            F(n, 1) = 0;
            % Setup G matrix
            % (V(i+1,j) + V(i, j+1) - 4V(i,j) + V(i-1,j) + V(i, j-1))/delta^2
 = 0
            G(n, n) = -4/deltaXY^2;
            % Calculate mapping index
            nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
            nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
            nyp = mappingEq(ix, iy+1, ny);  % index for V(i, j+1)
            nym = mappingEq(ix, iy-1, ny);  % index for V(i, j-1)
            % Set the G matrix
            G(n, nxp) = 1/deltaXY^2;
            G(n, nxm) = 1/deltaXY^2;
            G(n, nyp) = 1/deltaXY^2;
            G(n, nym) = 1/deltaXY^2;
        end
    end
end
```

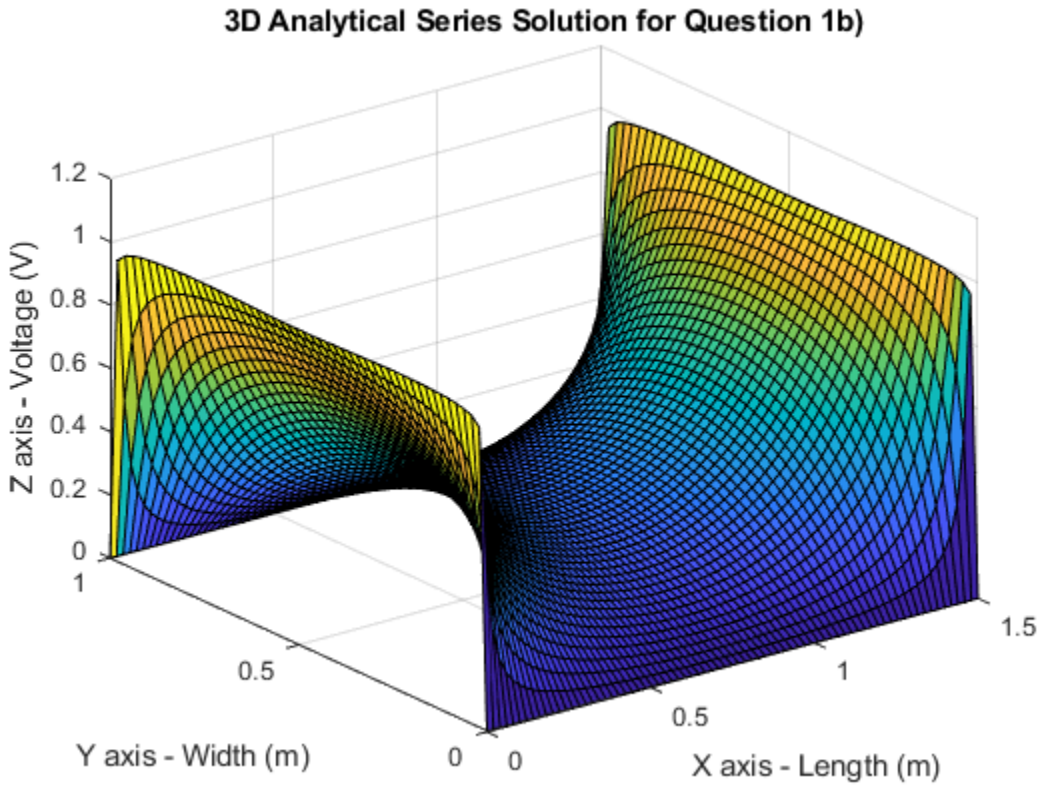
```
% Solve for V from GV = F
V = G\F;

% Map back to the 2D region
for iMap = 1:nx*ny
    % Calculate the index for the 2D region
    ix = ceil(iMap/ny);
    iy = mod(iMap, ny);
    if iy == 0
        iy = ny;
    end
    % Assign the value
    matrixSolFDM(iy, ix) = V(iMap);
end

% Plot the solution from the Finite Difference Method
figure(4)  % 2D
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
surf(X,Y,matrixSolFDM)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("2D Plot of Finite Difference Method Solution for Question 1b)")
view(0, 90)
snapnow
figure(5)  % 3D
surf(X,Y,matrixSolFDM)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("3D Plot of Finite Difference Method Solution for Question 1b)")
snapnow

% Calculate using the analytical series
figure(6)
% Calculate the a and b for the analytical series
a = W;
b = L/2;
Xshifted = X-b;
% Loop for simulation for analytical solution
for iSim = 1:simSteps
    % Calculate the n in the sum: n = 1,3,5,7...
    n = iSim*2-1;
    % Calculate the sum
    matrixSolAnalytic = matrixSolAnalytic + 4*V0/pi* (1/
n)*(cosh(n*pi*Xshifted/a)/cosh(n*pi*b/a)).*sin(n*pi*Y/a);

    % Plot the evolving surface
    figure(6)
    surf(X,Y,matrixSolAnalytic)
    xlabel("X axis - Length (m)")
    ylabel("Y axis - Width (m)")
    zlabel("Z axis - Voltage (V)")
```

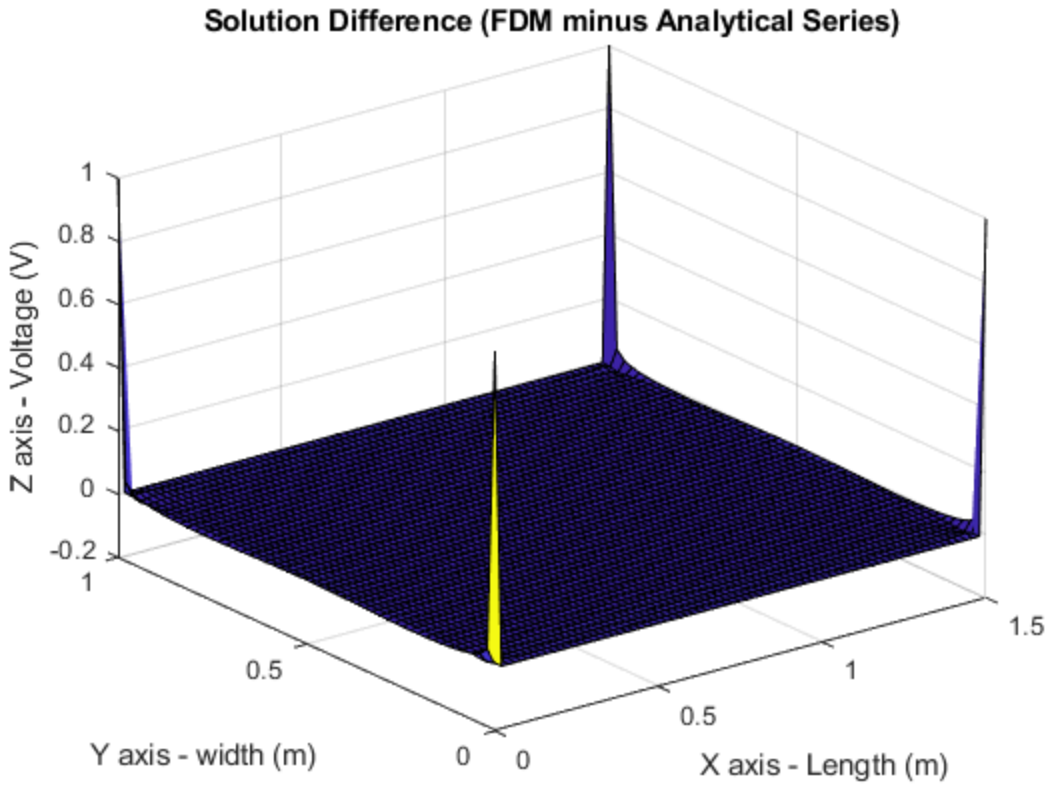
```
    title("Analytical Series Solution")

    % Pause some time
    pause(pauseTime)
end

% Plot the final surface
figure(6)  % 2D
surf(X,Y,matrixSolAnalytic)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("2D Analytical Series Solution for Question 1b)")
view(0, 90)
snapnow
figure(7)  % 2D
surf(X,Y,matrixSolAnalytic)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("3D Analytical Series Solution for Question 1b)")
snapnow

% Calculate the difference between FDM and analytical solution
figure(8)
matrixSolDiff = matrixSolFDM - matrixSolAnalytic;
surf(X,Y, matrixSolDiff)
xlabel("X axis - Length (m)")
ylabel("Y axis - width (m)")
zlabel("Z axis - Voltage (V)")
title("Solution Difference (FDM minus Analytical Series)")
snapnow
```


2D Plot of Finite Difference Method Solution for Question 1b)
Y axis - Width (m)
X axis - Length (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5


3D Plot of Finite Difference Method Solution for Question 1b)
Z axis - Voltage (V)
Y axis - Width (m)
X axis - Length (m)
0
0.2
0.4
0.6
0.8
1
0
0.5
1
0
0.5
1
1.5


2D Analytical Series Solution for Question 1b)
Y axis - Width (m)
X axis - Length (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5


3D Analytical Series Solution for Question 1b)
Z axis - Voltage (V)
Y axis - Width (m)
X axis - Length (m)
0
0.2
0.4
0.6
0.8
1
1.2
0
0.5
1
0
0.5
1
1.5



Question 1 Conclusion

The finite difference method is solved by constructing the G and F matrices. The precision of the solution is depend on the meshing. Finer the meshing, more precise of the solution, but will require more computations and longer time.

The analytical series solution requires an infinite sum. The solution can be more precise by including more terms in the sum. However, more terms in the sum require more computations and longer time. When n is large, the corresponding calculation of cosh() may be too large and may result in computation error (NaN). Therefore, we should stop the analytical series when the solution is converging to a certain level before n is getting too large.

One advantage of the finite difference method versus the analytical solution is that the solutions of the corners are precise because the boundary conditions are used directly. One disadvantage of the finite difference method is when the mesh is big, the MATLAB will generate an error indicating that the maximum storage limit is exceeded. Therefore, the precision of the solution is limited.

Question 2

In Question 2, the Finite Difference Method is used to solve for the current flow in the rectangular region with 2 resistive boxes.

Question 2a)

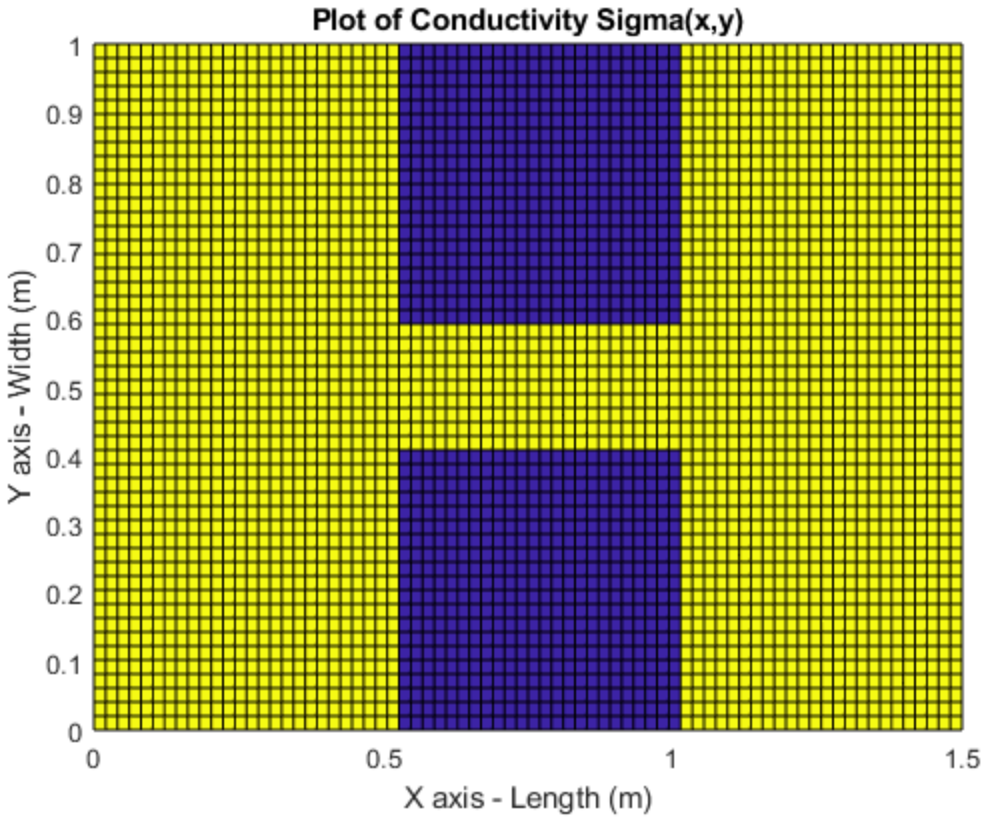
In this part, the current flow at the two contacts is calculated, and the plots of the sigma(x,y), V(x,y), Ex, Ey, and J(x,y) are generated for the rectangular region.

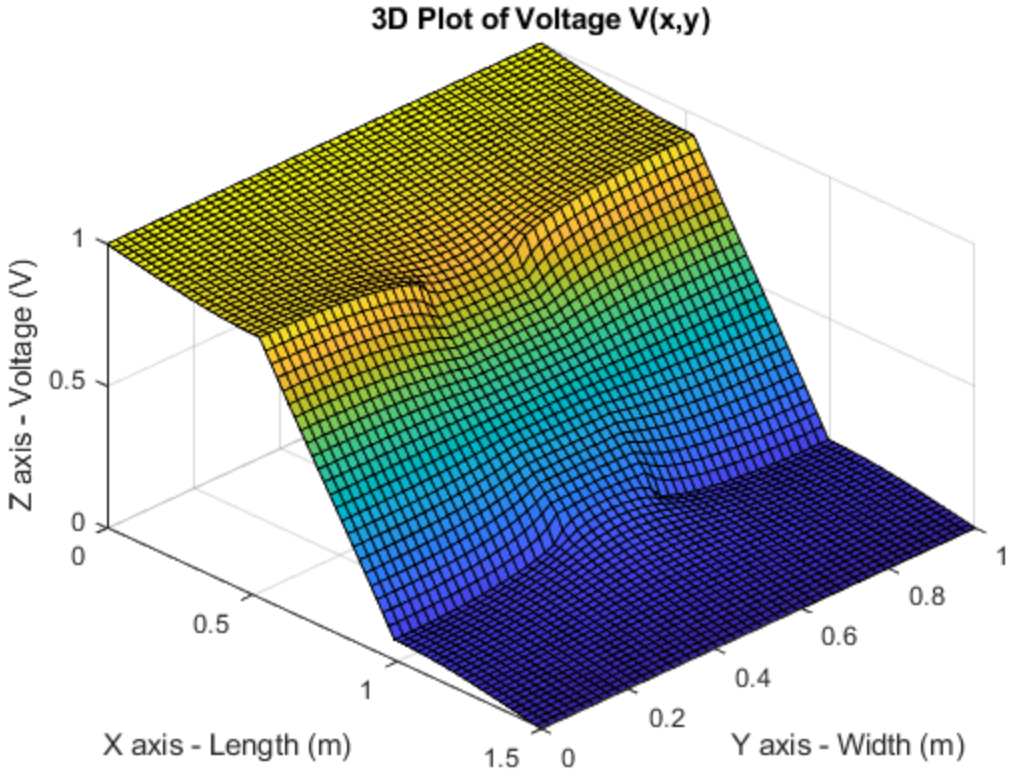
```
% Define the dimension
W = 1;  % Width
L = 1.5;  % Height
Wb = 0.4*W;  % Width of the box
Lb = 0.3*L;  % Length of the box
deltaXY = 0.02;  % Assume deltaX = deltaY

% Declare the V0
V0 = 1;

% Calculate the dimension of solution matrix
nx = L/deltaXY;
ny = W/deltaXY;
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));

% Declare the matrix for conductivity: Sigma(y,x)
matrixSigma = ones(ny, nx);  % Dimension: ny times nx
xIndexBox = ceil((L-Lb)/(2*deltaXY));  % Find the starting x index for the box
LbIndexRange = ceil(Lb/deltaXY);  % Index range for the length of the box
WbIndexRange = ceil(Wb/deltaXY);  % Index range for the width of the box
% Assign the region for the box
matrixSigma(1:WbIndexRange, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;
matrixSigma(ny-WbIndexRange:ny, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;

% Plot the region for conductivity
figure(9)
surf(X,Y, matrixSigma)
title("Plot of Conductivity Sigma(x,y)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Conductivity (S)")
view(0,90)  % View from top
snapnow

% Declare the matrix for voltage V(y,x)
matrixV = zeros(ny, nx);  % Dimension: ny times nx

% Declare the G matrix and F vector: GV = F
G = zeros(nx*ny, nx*ny);
F = zeros(nx*ny, 1);

% Construct the G matrix and F vector
for ix = 1:nx
    for iy = 1:ny
        % Calculate the index
        n = mappingEq(ix, iy, ny);
        % Check for the boundary
        if ix==1 || ix==nx || iy ==1 || iy==ny
            G(n,n) = 1;
            % Boundary condition for x
            if ix == 1
                F(n,1) = V0;  % V = V0 at x = 0
            elseif ix == nx
                F(n,1) = 0;  % and V = 0 at x = L
```

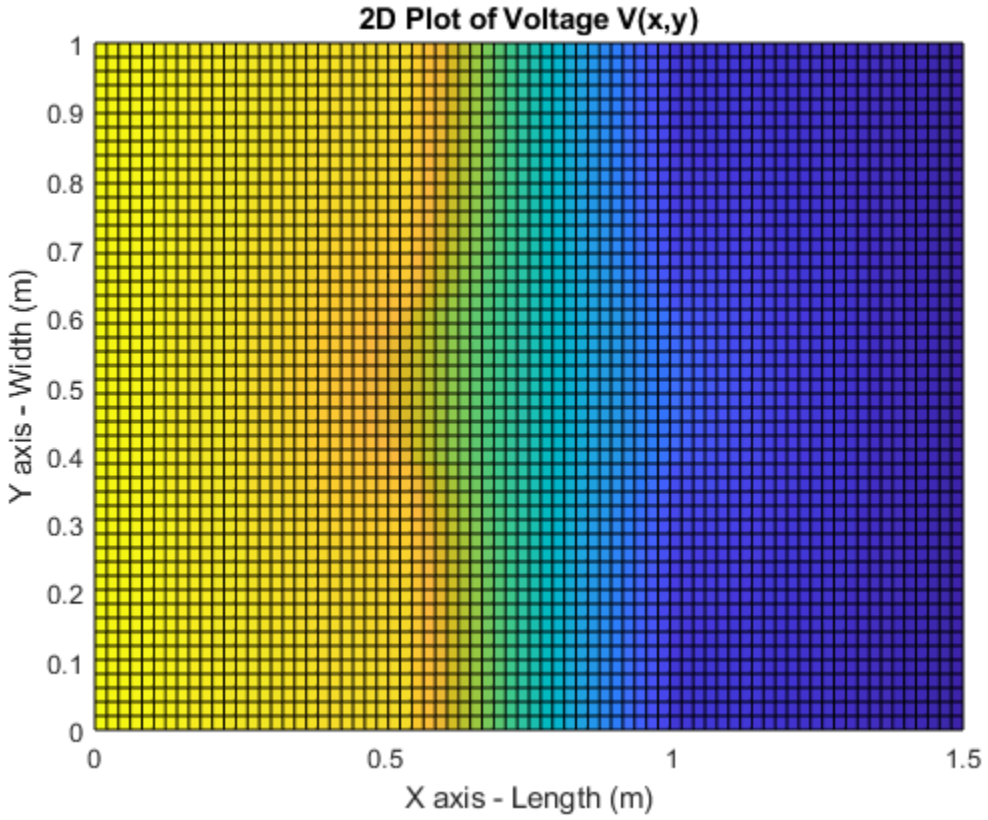
```
            elseif iy == 1
                nyp = mappingEq(ix, iy+1, ny);  % dV/dy=0 at y=0
                G(n,nyp) = -1;
            elseif iy == ny
                nym = mappingEq(ix, iy-1, ny);  % dV/dy=0 at y=W
                G(n, nym) = -1;
            end
        else
            % Calculate the sigma
            sigmaxp = (matrixSigma(iy,ix) + matrixSigma(iy,ix+1))/2;
            sigmaxm = (matrixSigma(iy,ix) + matrixSigma(iy, ix-1))/2;
            sigmayp = (matrixSigma(iy,ix) + matrixSigma(iy+1, ix))/2;
            sigmaym = (matrixSigma(iy,ix) + matrixSigma(iy-1, ix))/2;

            % Calculate mapping index
            nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
            nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
            nyp = mappingEq(ix, iy+1, ny);  % index for V(i,j+1)
            nym = mappingEq(ix, iy-1, ny);  % index for V(i,j-1)

            % Setup the G matrix
            G(n,n) = -(sigmaxp+sigmaxm+sigmayp+sigmaym)/deltaXY^2;
            G(n, nxp) = sigmaxp/deltaXY^2;
            G(n, nxm) = sigmaxm/deltaXY^2;
            G(n, nyp) = sigmayp/deltaXY^2;
            G(n, nym) = sigmaym/deltaXY^2;
        end
    end
end

% Solve for V from GV = F
V = G\F;

% Map back to the 2D region
for iMap = 1:nx*ny
    % Calculate the index for the 2D region
    ix = ceil(iMap/ny);
    iy = mod(iMap, ny);
    if iy == 0
        iy = ny;
    end
    % Assign the value
    matrixV(iy, ix) = V(iMap);
end

% Plot the solution for V from the Finite Difference Method
figure(10)
surf(X,Y,matrixV)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("3D Plot of Voltage V(x,y)")
view(45,45)  % 3-D View
snapnow
```

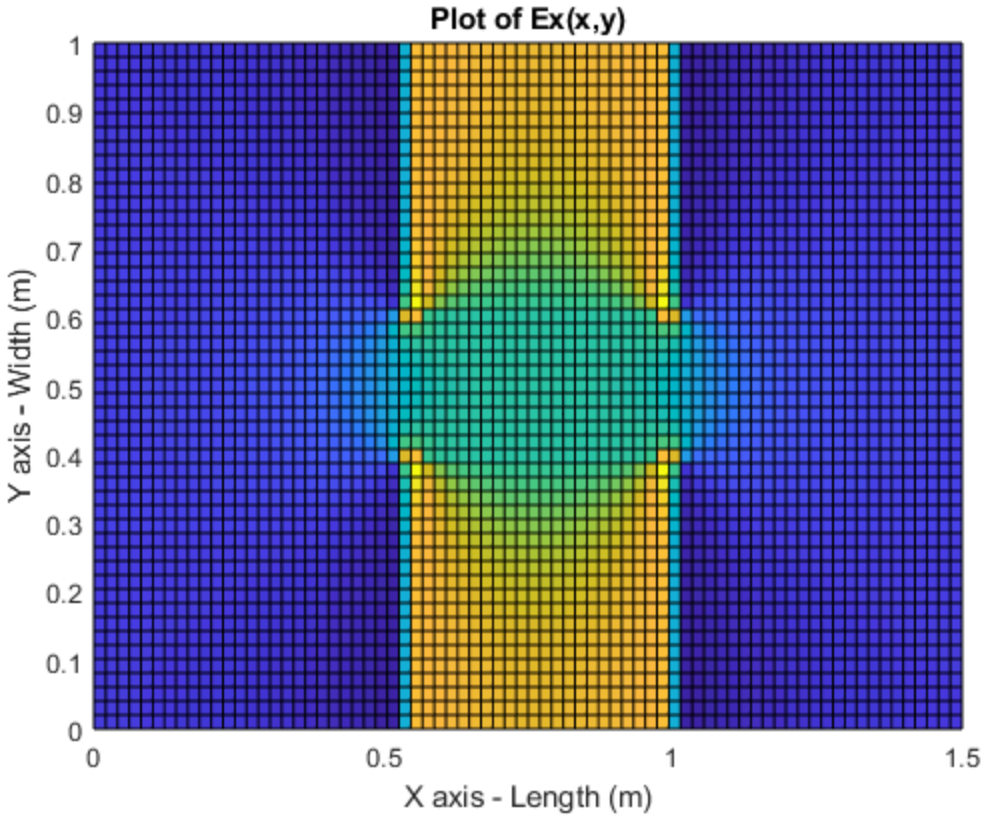
```
figure(11)
surf(X,Y,matrixV)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Voltage (V)")
title("2D Plot of Voltage V(x,y)")
view(0,90)  % View from top
snapnow

% Solve the electric field
[Ex, Ey] = gradient(-matrixV);
Ex = Ex/deltaXY;
Ey = Ey/deltaXY;
% Plot the Ex field
figure(12)
surf(X,Y,Ex)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Ex Field (V/m)")
title("Plot of Ex(x,y)")
view(0,90)  % View from top
snapnow
% Plot the Ey field
figure(13)
surf(X,Y,Ey)
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
zlabel("Z axis - Ey Field (V/m)")
title("Plot of Ey(x,y)")
view(0,90)  % View from top
snapnow

% Plot the electric field
figure(14)
quiver(X, Y, Ex, Ey);
title("Plot of Electric Field E(x,y)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
snapnow

% Solve for current density field J = sigma*E
Jx = matrixSigma .* Ex;
Jy = matrixSigma .* Ey;
% Plot the current density
figure(15)
quiver(X,Y, Jx, Jy)
title("Plot of Current Density J(x,y)")
xlabel("X axis - Length (m)")
ylabel("Y axis - Width (m)")
snapnow

% Calculate the current flow at the two contacts
I_first = sum(Jx(:,1)) * deltaXY;
fprintf("The current flow at the first contact is "+I_first+".\n");
```

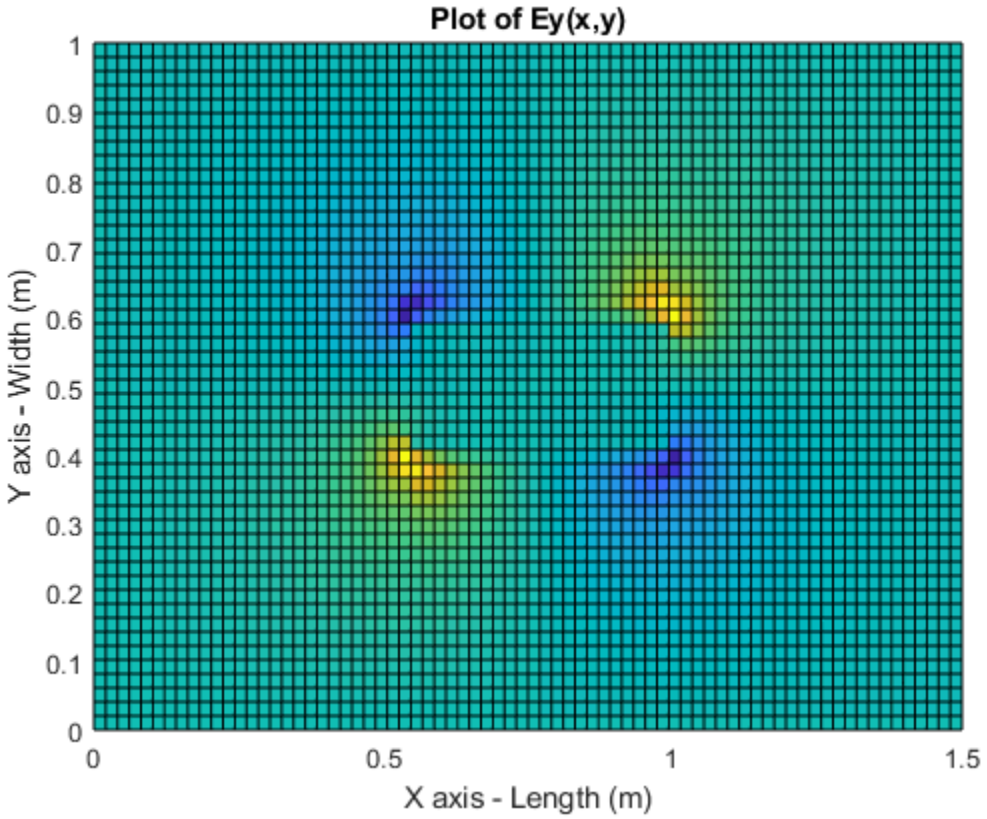
```
I_last = sum(Jx(:, nx)) * deltaXY;
fprintf("The current flow at the last contact is "+I_last+".\n");
```

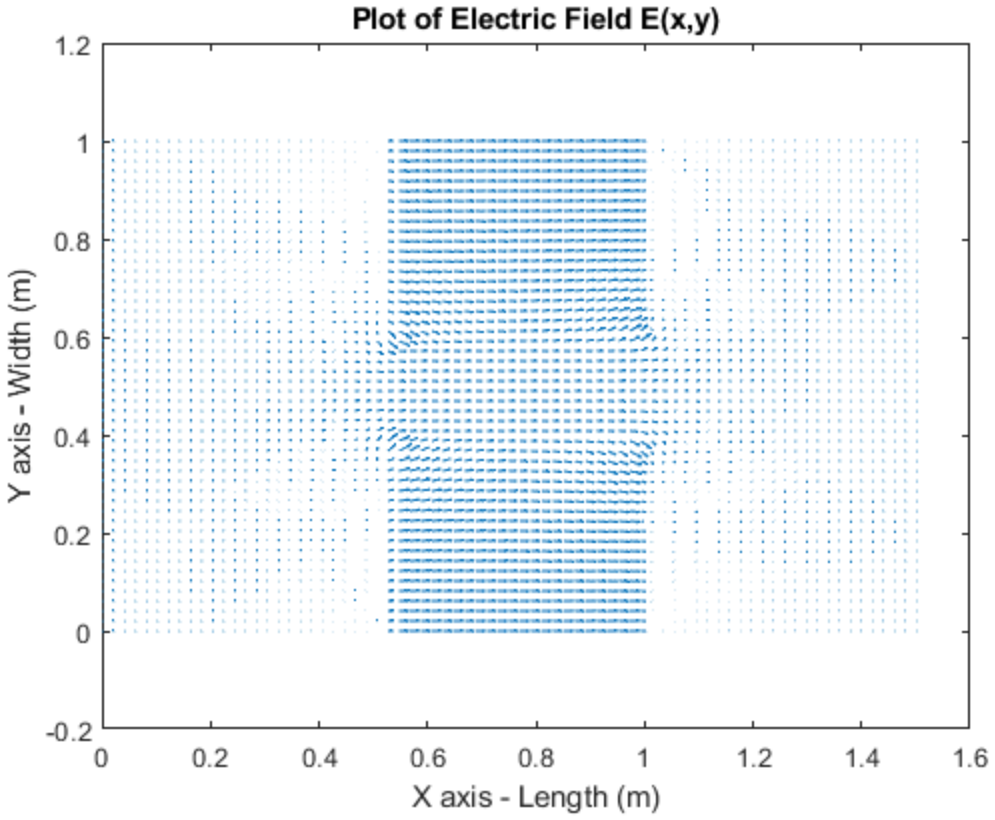


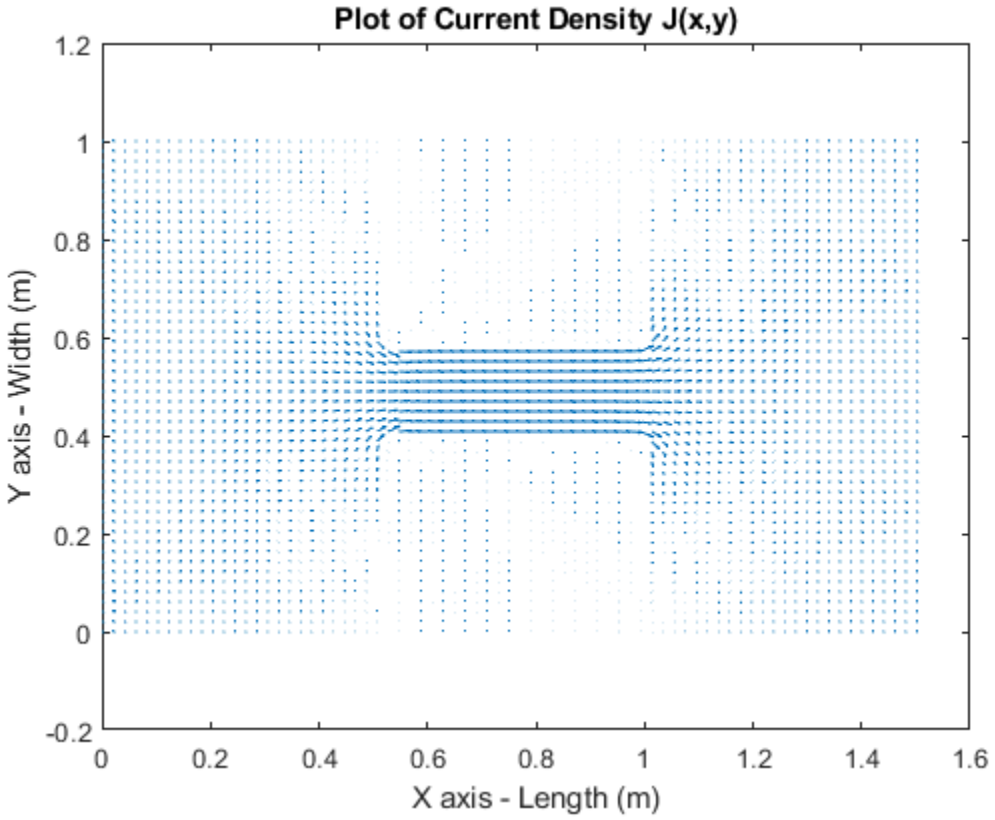

3D Plot of Voltage V(x,y)
Z axis - Voltage (V)
1
0.5
0
0
0.5
1
1.5
X axis - Length (m)
0
0.2
0.4
0.6
0.8
1
Y axis - Width (m)


2D Plot of Voltage V(x,y)
Y axis - Width (m)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
X axis - Length (m)


Plot of Ex(x,y)
Y axis - Width (m)
X axis - Length (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5


Plot of Ey(x,y)
Y axis - Width (m)
X axis - Length (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5


Plot of Electric Field E(x,y)
Y axis - Width (m)
X axis - Length (m)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Plot of Current Density J(x,y)
Y axis - Width (m)
X axis - Length (m)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

```
The current flow at the first contact is 0.24165.
The current flow at the last contact is 0.2415.
```

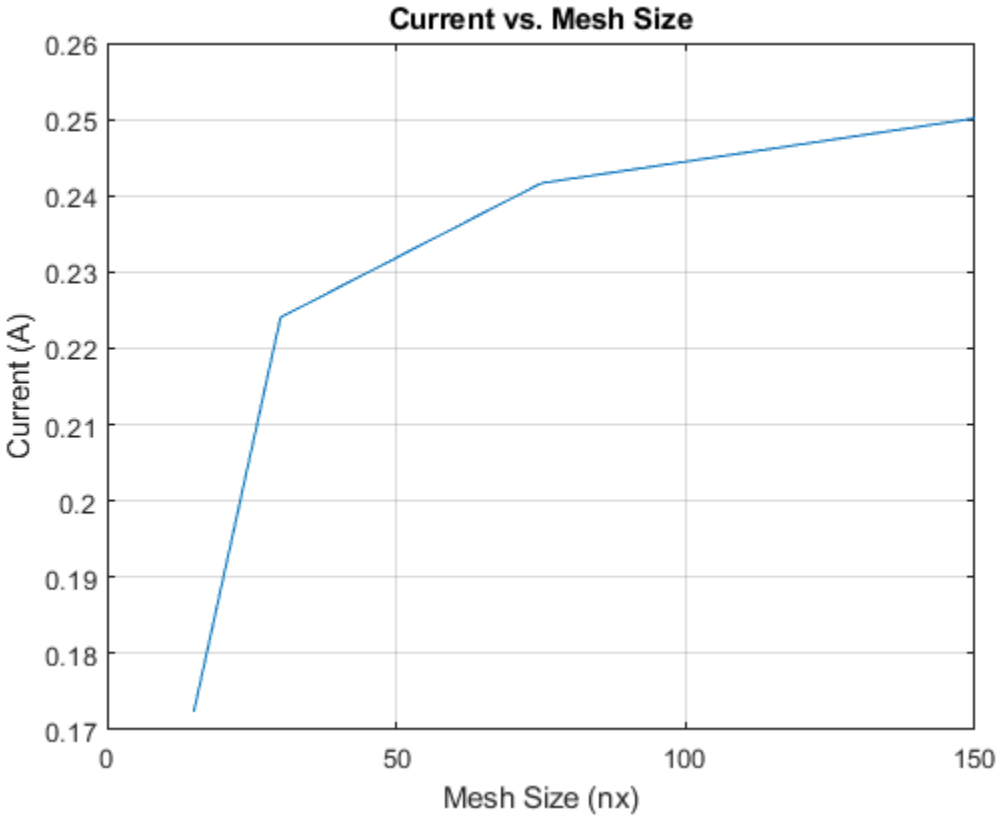
Question 2b)

In this part, the graph of current vs mesh size is generated. As shown the the graph, as the mesh size is increased, the current become more precise and approaches to the "true" value.

```
% Create graph of current vs mesh size
vectorDeltaXY = [0.1, 0.05, 0.02, 0.01];
vectorCurrent = zeros(length(vectorDeltaXY), 1);
vectorMeshSize = zeros(length(vectorDeltaXY), 1);  % Hold the value of nx as
 mesh size
for indexMesh = 1:length(vectorDeltaXY)
    deltaXY = vectorDeltaXY(indexMesh);
    % Calculate the dimension of solution matrix
    nx = ceil(L/deltaXY);
    ny = ceil(W/deltaXY);
    [X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
    % Reconstruct the sigma matrix
    matrixSigma = ones(ny, nx);  % Dimension: ny times nx
    xIndexBox = ceil((L-Lb)/(2*deltaXY));  % Find the starting x index for the
 box
    LbIndexRange = ceil(Lb/deltaXY);  % Index range for the length of the box
    WbIndexRange = ceil(Wb/deltaXY);  % Index range for the width of the box
    % Assign the region for the box
    matrixSigma(1:WbIndexRange, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;
    matrixSigma(ny-WbIndexRange:ny, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;

    % Construct the G matrix and F vector
    G = zeros(nx*ny, nx*ny);
    F = zeros(nx*ny, 1);
    matrixV = zeros(ny, nx);
    for ix = 1:nx
        for iy = 1:ny
            % Calculate the index
            n = mappingEq(ix, iy, ny);
            % Check for the boundary
            if ix==1 || ix==nx || iy ==1 || iy==ny
                G(n,n) = 1;
                % Boundary condition for x
                if ix == 1
                    F(n,1) = V0;  % V = V0 at x = 0
                elseif ix == nx
                    F(n,1) = 0;  % and V = 0 at x = L
                elseif iy == 1
                    nyp = mappingEq(ix, iy+1, ny);  % dV/dy=0 at y=0
                    G(n,nyp) = -1;
                elseif iy == ny
                    nym = mappingEq(ix, iy-1, ny);  % dV/dy=0 at y=W
                    G(n, nym) = -1;
                end
            else
```

```
                % Calculate the sigma
                sigmaxp = (matrixSigma(iy,ix) + matrixSigma(iy,ix+1))/2;
                sigmaxm = (matrixSigma(iy,ix) + matrixSigma(iy, ix-1))/2;
                sigmayp = (matrixSigma(iy,ix) + matrixSigma(iy+1, ix))/2;
                sigmaym = (matrixSigma(iy,ix) + matrixSigma(iy-1, ix))/2;

                % Calculate mapping index
                nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
                nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
                nyp = mappingEq(ix, iy+1, ny);  % index for V(i,j+1)
                nym = mappingEq(ix, iy-1, ny);  % index for V(i,j-1)

                % Setup the G matrix
                G(n,n) = -(sigmaxp+sigmaxm+sigmayp+sigmaym)/deltaXY^2;
                G(n, nxp) = sigmaxp/deltaXY^2;
                G(n, nxm) = sigmaxm/deltaXY^2;
                G(n, nyp) = sigmayp/deltaXY^2;
                G(n, nym) = sigmaym/deltaXY^2;
            end
        end
    end

    % Solve for V from GV = F
    V = G\F;

    % Map back to the 2D region
    for iMap = 1:nx*ny
        % Calculate the index for the 2D region
        ix = ceil(iMap/ny);
        iy = mod(iMap, ny);
        if iy == 0
            iy = ny;
        end
        % Assign the value
        matrixV(iy, ix) = V(iMap);
    end

    % Solve the electric field
    [Ex, Ey] = gradient(-matrixV);
    Ex = Ex/deltaXY;
    Ey = Ey/deltaXY;
    % Solve for current density
    Jx = matrixSigma .* Ex;
    % Solve for current
    vectorCurrent(indexMesh) = sum(Jx(:,2)) * deltaXY;
    vectorMeshSize(indexMesh) = nx;  % Save the mesh size
end
% Plot for current vs. mesh size (nx)
figure(16)
plot(vectorMeshSize, vectorCurrent)
title("Current vs. Mesh Size")
xlabel("Mesh Size (nx)")
ylabel("Current (A)")
grid on
```

```
snapnow
```



Question 2c)

In this part, the graphs for current versus various bottle-necks are generated. The first graph is the current versus the different Wb (Width of the box) while keeping Lb (Length of the box) constant. The second graph is the current versus different Lb while keeping Wb constant. As shown in the graphs, the current dropped while the "bottle-neck" is narrowed with the box size is increased.

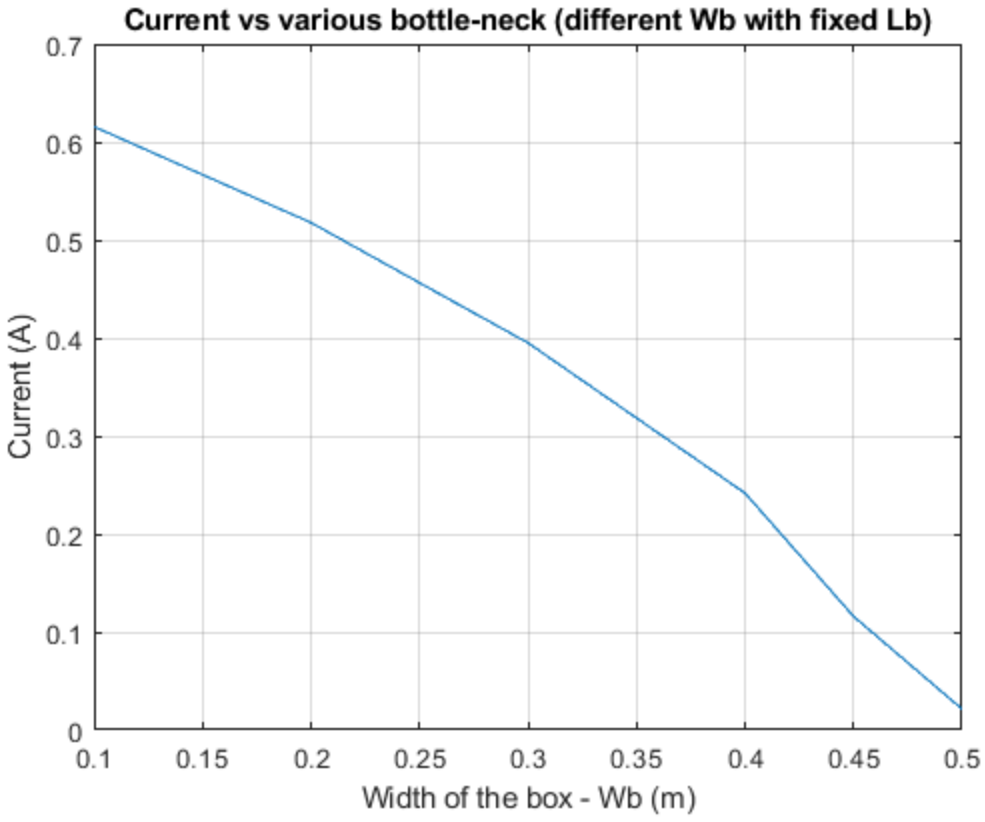
```
% Create graph of current vs. various bottle necks (different Wb with fixed
 Lb)
vectorWb = [0.1, 0.2, 0.3, 0.4, 0.45, 0.5]*W;  % Width of the box
Lb = 0.3*L;  % Length of the box
vectorCurrent = zeros(length(vectorWb), 1);
deltaXY = 0.02;
% Calculate the dimension of solution matrix
nx = L/deltaXY;
ny = W/deltaXY;
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
for indexWb = 1:length(vectorWb)
    Wb = vectorWb(indexWb);
    % Reconstruct the sigma matrix
    matrixSigma = ones(ny, nx);  % Dimension: ny times nx
    xIndexBox = ceil((L-Lb)/(2*deltaXY));  % Find the starting x index for the
 box
    LbIndexRange = ceil(Lb/deltaXY);  % Index range for the length of the box
```

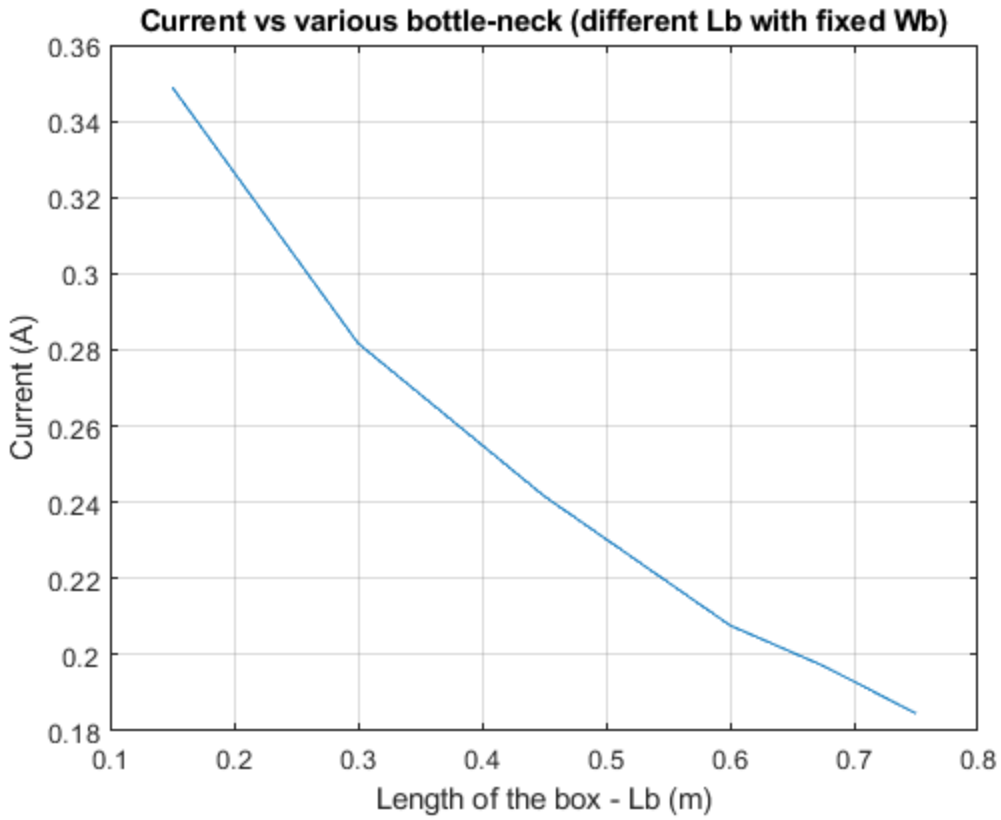
```
WbIndexRange = ceil(Wb/deltaXY);  % Index range for the width of the box
% Assign the region for the box
matrixSigma(1:WbIndexRange, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;
matrixSigma(ny-WbIndexRange:ny, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;

% Construct the G matrix and F vector
G = zeros(nx*ny, nx*ny);
F = zeros(nx*ny, 1);
matrixV = zeros(ny, nx);
for ix = 1:nx
    for iy = 1:ny
        % Calculate the index
        n = mappingEq(ix, iy, ny);
        % Check for the boundary
        if ix==1 || ix==nx || iy ==1 || iy==ny
            G(n,n) = 1;
            % Boundary condition for x
            if ix == 1
                F(n,1) = V0;  % V = V0 at x = 0
            elseif ix == nx
                F(n,1) = 0;  % and V = 0 at x = L
            elseif iy == 1
                nyp = mappingEq(ix, iy+1, ny);  % dV/dy=0 at y=0
                G(n,nyp) = -1;
            elseif iy == ny
                nym = mappingEq(ix, iy-1, ny);  % dV/dy=0 at y=W
                G(n, nym) = -1;
            end
        else
            % Calculate the sigma
            sigmaxp = (matrixSigma(iy,ix) + matrixSigma(iy,ix+1))/2;
            sigmaxm = (matrixSigma(iy,ix) + matrixSigma(iy, ix-1))/2;
            sigmayp = (matrixSigma(iy,ix) + matrixSigma(iy+1, ix))/2;
            sigmaym = (matrixSigma(iy,ix) + matrixSigma(iy-1, ix))/2;

            % Calculate mapping index
            nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
            nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
            nyp = mappingEq(ix, iy+1, ny);  % index for V(i,j+1)
            nym = mappingEq(ix, iy-1, ny);  % index for V(i,j-1)

            % Setup the G matrix
            G(n,n) = -(sigmaxp+sigmaxm+sigmayp+sigmaym)/deltaXY^2;
            G(n, nxp) = sigmaxp/deltaXY^2;
            G(n, nxm) = sigmaxm/deltaXY^2;
            G(n, nyp) = sigmayp/deltaXY^2;
            G(n, nym) = sigmaym/deltaXY^2;
        end
    end
end

% Solve for V from GV = F
V = G\F;
```

```
    % Map back to the 2D region
    for iMap = 1:nx*ny
        % Calculate the index for the 2D region
        ix = ceil(iMap/ny);
        iy = mod(iMap, ny);
        if iy == 0
            iy = ny;
        end
        % Assign the value
        matrixV(iy, ix) = V(iMap);
    end

    % Solve the electric field
    [Ex, Ey] = gradient(-matrixV);
    Ex = Ex/deltaXY;
    Ey = Ey/deltaXY;
    % Solve for current density
    Jx = matrixSigma .* Ex;
    % Solve for current
    vectorCurrent(indexWb) = sum(Jx(:,2)) * deltaXY;
end
% Plot the current vs various bottle-neck (different Wb with fixed Lb)
figure(17)
plot(vectorWb, vectorCurrent)
title("Current vs various bottle-neck (different Wb with fixed Lb)")
xlabel("Width of the box - Wb (m)")
ylabel("Current (A)")
grid on
snapnow


% Create graph of current vs. various bottle necks (different Lb with fixed
 Wb)
vectorLb = [0.1, 0.2, 0.3, 0.4, 0.45, 0.5]*L;  % Length of the box
Wb = 0.4*W;  % Width of the box
vectorCurrent = zeros(length(vectorLb), 1);
deltaXY = 0.02;
% Calculate the dimension of solution matrix
nx = L/deltaXY;
ny = W/deltaXY;
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
for indexLb = 1:length(vectorLb)
    Lb = vectorLb(indexLb);
    % Reconstruct the sigma matrix
    matrixSigma = ones(ny, nx);  % Dimension: ny times nx
    xIndexBox = ceil((L-Lb)/(2*deltaXY));  % Find the starting x index for the
 box
    LbIndexRange = ceil(Lb/deltaXY);  % Index range for the length of the box
    WbIndexRange = ceil(Wb/deltaXY);  % Index range for the width of the box
    % Assign the region for the box
    matrixSigma(1:WbIndexRange, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;
    matrixSigma(ny-WbIndexRange:ny, xIndexBox:xIndexBox+LbIndexRange) = 10^-2;

    % Construct the G matrix and F vector
```

```
G = zeros(nx*ny, nx*ny);
F = zeros(nx*ny, 1);
matrixV = zeros(ny, nx);
for ix = 1:nx
    for iy = 1:ny
        % Calculate the index
        n = mappingEq(ix, iy, ny);
        % Check for the boundary
        if ix==1 || ix==nx || iy ==1 || iy==ny
            G(n,n) = 1;
            % Boundary condition for x
            if ix == 1
                F(n,1) = V0;  % V = V0 at x = 0
            elseif ix == nx
                F(n,1) = 0;  % and V = 0 at x = L
            elseif iy == 1
                nyp = mappingEq(ix, iy+1, ny);  % dV/dy=0 at y=0
                G(n,nyp) = -1;
            elseif iy == ny
                nym = mappingEq(ix, iy-1, ny);  % dV/dy=0 at y=W
                G(n, nym) = -1;
            end
        else
            % Calculate the sigma
            sigmaxp = (matrixSigma(iy,ix) + matrixSigma(iy,ix+1))/2;
            sigmaxm = (matrixSigma(iy,ix) + matrixSigma(iy, ix-1))/2;
            sigmayp = (matrixSigma(iy,ix) + matrixSigma(iy+1, ix))/2;
            sigmaym = (matrixSigma(iy,ix) + matrixSigma(iy-1, ix))/2;

            % Calculate mapping index
            nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
            nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
            nyp = mappingEq(ix, iy+1, ny);  % index for V(i,j+1)
            nym = mappingEq(ix, iy-1, ny);  % index for V(i,j-1)

            % Setup the G matrix
            G(n,n) = -(sigmaxp+sigmaxm+sigmayp+sigmaym)/deltaXY^2;
            G(n, nxp) = sigmaxp/deltaXY^2;
            G(n, nxm) = sigmaxm/deltaXY^2;
            G(n, nyp) = sigmayp/deltaXY^2;
            G(n, nym) = sigmaym/deltaXY^2;
        end
    end
end

% Solve for V from GV = F
V = G\F;

% Map back to the 2D region
for iMap = 1:nx*ny
    % Calculate the index for the 2D region
    ix = ceil(iMap/ny);
    iy = mod(iMap, ny);
    if iy == 0
```

```
            iy = ny;
        end
        % Assign the value
        matrixV(iy, ix) = V(iMap);
    end

    % Solve the electric field
    [Ex, Ey] = gradient(-matrixV);
    Ex = Ex/deltaXY;
    Ey = Ey/deltaXY;
    % Solve for current density
    Jx = matrixSigma .* Ex;
    % Solve for current
    vectorCurrent(indexLb) = sum(Jx(:,2)) * deltaXY;
end
% Plot the current vs various bottle-neck (different Lb with fixed Wb)
figure(18)
plot(vectorLb, vectorCurrent)
title("Current vs various bottle-neck (different Lb with fixed Wb)")
xlabel("Length of the box - Lb (m)")
ylabel("Current (A)")
grid on
snapnow
```





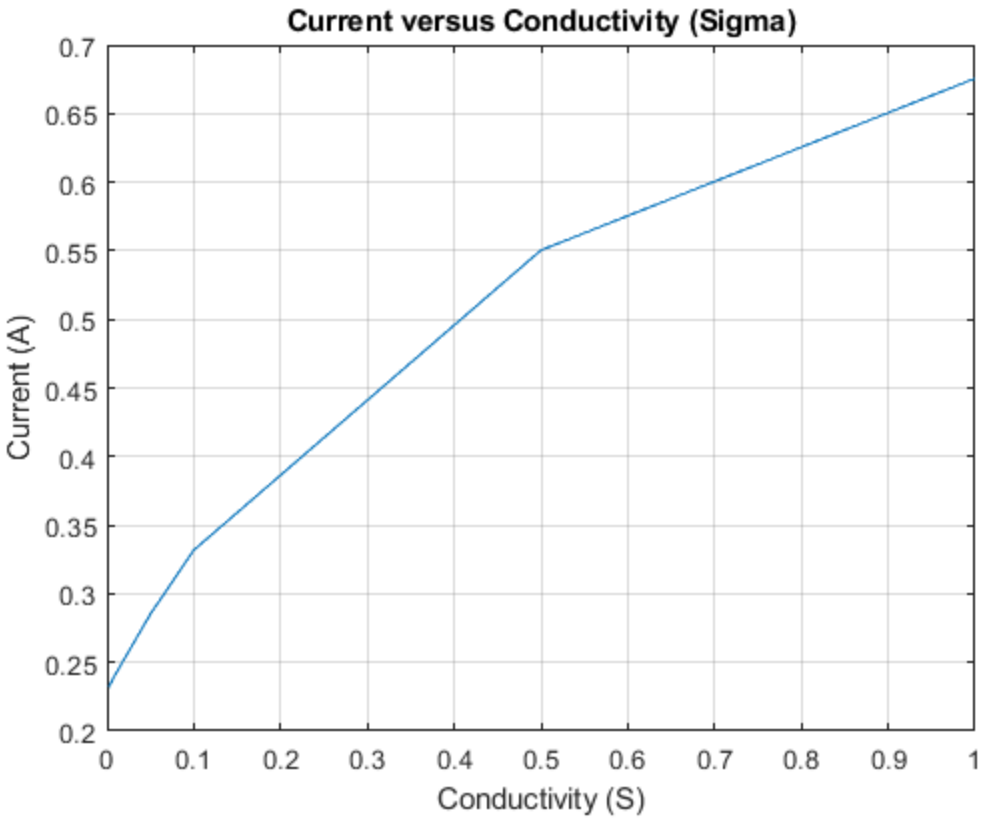
Question 2d)

In this part, the graph of current vs conductivity (sigma) of the boxes is generated. As shown in the graph, the current increased as the conductivity of the boxes is increased.

```
% Graph of current vs sigma inside the box
vectorSigma = [1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001];  % Length of the box
Wb = 0.4*W;  % Width of the box
Lb = 0.3*L;  % Length of the box
vectorCurrent = zeros(length(vectorSigma), 1);
deltaXY = 0.02;
% Calculate the dimension of solution matrix
nx = L/deltaXY;
ny = W/deltaXY;
[X,Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
for indexSigma = 1:length(vectorSigma)
    sigmaInBox = vectorSigma(indexSigma);
    % Reconstruct the sigma matrix
    matrixSigma = ones(ny, nx);  % Dimension: ny times nx
    xIndexBox = ceil((L-Lb)/(2*deltaXY));  % Find the starting x index for the
 box
    LbIndexRange = ceil(Lb/deltaXY);  % Index range for the length of the box
    WbIndexRange = ceil(Wb/deltaXY);  % Index range for the width of the box
    % Assign the region for the box
    matrixSigma(1:WbIndexRange, xIndexBox:xIndexBox+LbIndexRange) =
 sigmaInBox;
```

```
    matrixSigma(ny-WbIndexRange:ny, xIndexBox:xIndexBox+LbIndexRange) =
sigmaInBox;

    % Construct the G matrix and F vector
    G = zeros(nx*ny, nx*ny);
    F = zeros(nx*ny, 1);
    matrixV = zeros(ny, nx);
    for ix = 1:nx
        for iy = 1:ny
            % Calculate the index
            n = mappingEq(ix, iy, ny);
            % Check for the boundary
            if ix==1 || ix==nx || iy ==1 || iy==ny
                G(n,n) = 1;
                % Boundary condition for x
                if ix == 1
                    F(n,1) = V0;  % V = V0 at x = 0
                elseif ix == nx
                    F(n,1) = 0;  % and V = 0 at x = L
                elseif iy == 1
                    nyp = mappingEq(ix, iy+1, ny);  % dV/dy=0 at y=0
                    G(n,nyp) = -1;
                elseif iy == ny
                    nym = mappingEq(ix, iy-1, ny);  % dV/dy=0 at y=W
                    G(n, nym) = -1;
                end
            else
                % Calculate the sigma
                sigmaxp = (matrixSigma(iy,ix) + matrixSigma(iy,ix+1))/2;
                sigmaxm = (matrixSigma(iy,ix) + matrixSigma(iy, ix-1))/2;
                sigmayp = (matrixSigma(iy,ix) + matrixSigma(iy+1, ix))/2;
                sigmaym = (matrixSigma(iy,ix) + matrixSigma(iy-1, ix))/2;

                % Calculate mapping index
                nxp = mappingEq(ix+1, iy, ny);  % index for V(i+1,j)
                nxm = mappingEq(ix-1, iy, ny);  % index for V(i-1,j)
                nyp = mappingEq(ix, iy+1, ny);  % index for V(i,j+1)
                nym = mappingEq(ix, iy-1, ny);  % index for V(i,j-1)

                % Setup the G matrix
                G(n,n) = -(sigmaxp+sigmaxm+sigmayp+sigmaym)/deltaXY^2;
                G(n, nxp) = sigmaxp/deltaXY^2;
                G(n, nxm) = sigmaxm/deltaXY^2;
                G(n, nyp) = sigmayp/deltaXY^2;
                G(n, nym) = sigmaym/deltaXY^2;
            end
        end
    end

    % Solve for V from GV = F
    V = G\F;

    % Map back to the 2D region
    for iMap = 1:nx*ny
```

```
        % Calculate the index for the 2D region
        ix = ceil(iMap/ny);
        iy = mod(iMap, ny);
        if iy == 0
            iy = ny;
        end
        % Assign the value
        matrixV(iy, ix) = V(iMap);
    end

    % Solve the electric field
    [Ex, Ey] = gradient(-matrixV);
    Ex = Ex/deltaXY;
    Ey = Ey/deltaXY;
    % Solve for current density
    Jx = matrixSigma .* Ex;
    % Solve for current
    vectorCurrent(indexSigma) = sum(Jx(:,2)) * deltaXY;
end
% Plot of current vs sigma
figure(19)
plot(vectorSigma, vectorCurrent)
title("Current versus Conductivity (Sigma)")
xlabel("Conductivity (S)")
ylabel("Current (A)")
grid on
snapnow

% Helper function for mapping index
% @param iRow = i index for the row
%        jRow = j index for the column
%        ny   = size of the y
function [n] = mappingEq(iRow, jCol, ny)
    n = jCol + (iRow - 1) * ny;
end
```



Published with MATLAB® R2021b