Table of Contents

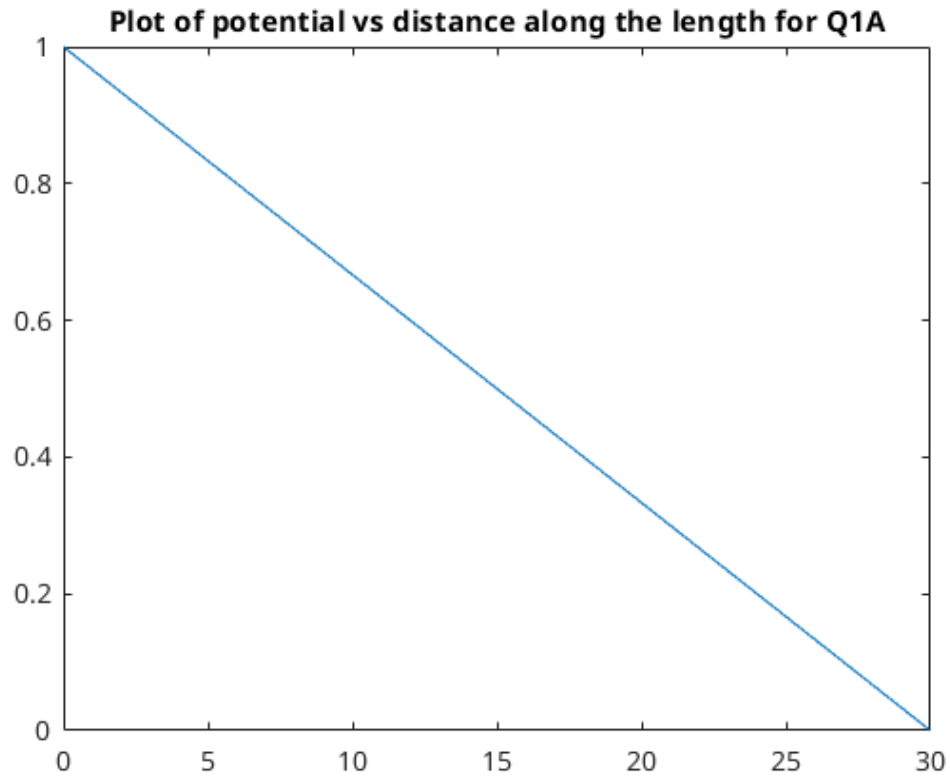

```
% Reyad ElMahdy
% 101064879
```

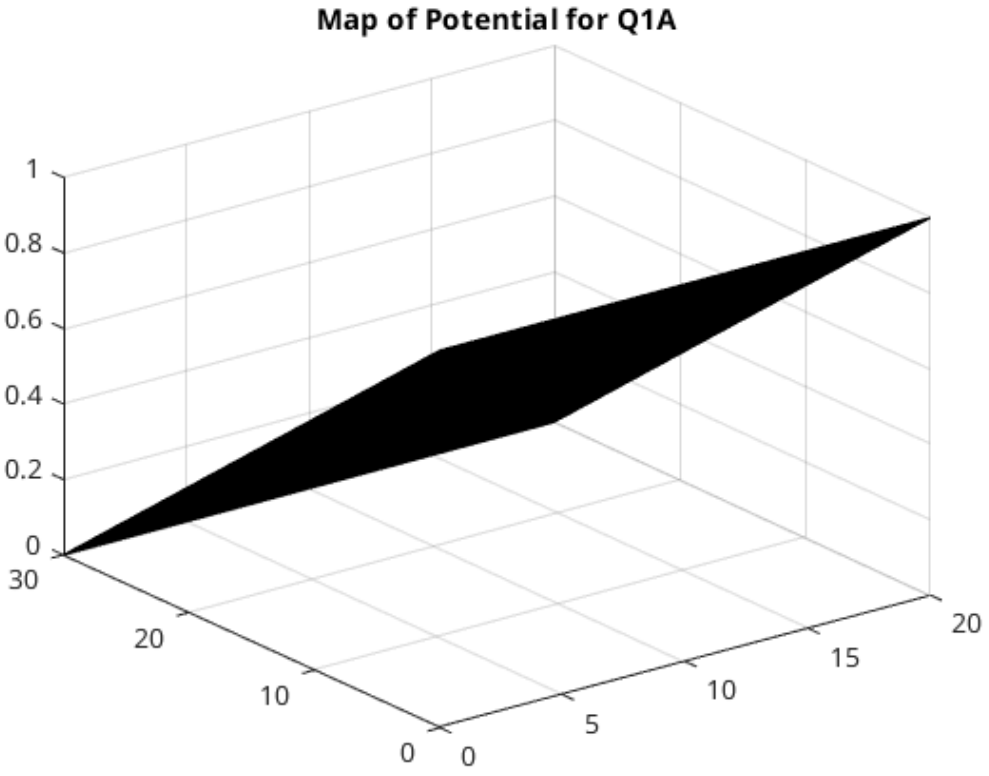
Question 1 a)

The code for this section uses the FDM (Finite Difference Method) to solve Laplace's equation for a 2D rectangle of dimentions L and W. The boundary conditions of the region are V=1 at x=0 and V=0 at x=L, The top and bottom boundaries have dV/dx=0. The code discretizes the region and the laplacian and turns them into matrices so that the potential can be displayed in the form of a vector which can easily be solved by matlab.

```
W = 20;
L = 1.5*W;

% Getting matrix dimensions
s = 0.5; % Space between each point in the meshgrid
nx = floor(L/s + 1);
ny = floor(W/s + 1);

%Creating matricies and mapping the 2D geometry
G = sparse(nx*ny);
Bc = zeros(1,nx*ny);

for i = 1:nx
    for j = 1:ny
        n = j+(i-1)*ny; % Mapping Equation

        % Setting boundary conditions
        if i == 1
            G(n,n) = 1;
            Bc(n) = 1;
        elseif i == nx
            G(n,n) = 1;
        elseif j == 1 || j == ny
            nxm = j+(i-2)*ny;
            nxp = j+i*ny;
            if j == 1
                nyp = j+1+(i-1)*ny;
                G(n,nyp) = 1;
            else
                nym = j-1+(i-1)*ny;
                G(n,nym) = 1;
            end
```

```
            G(n,n) = -3;
            G(n,nxm) = 1;
            G(n,nxp) = 1;
        else
            % Rest of the nodes
            nxm = j+(i-2)*ny;
            nxp = j+i*ny;
            nym = j-1+(i-1)*ny;
            nyp = j+1+(i-1)*ny;
            G(n,n) = -4;
            G(n,nxm) = 1;
            G(n,nxp) = 1;
            G(n,nym) = 1;
            G(n,nyp) = 1;
        end
    end
end

% Find V and map it to the space
V = G\Bc';
mappedV = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j+(i-1)*ny;
        mappedV(i,j) = V(n);
    end
end

% Plot
xaxis = 0:s:L;
figure(1)
plot(xaxis,mappedV(:,floor(ny/2)))
title('Plot of potential vs distance along the length for Q1A')
figure(2)
[xaxis, yaxis] = meshgrid(0:s:W, 0:s:L);
surf(xaxis,yaxis,mappedV)
title('Map of Potential for Q1A')
```


Plot of potential vs distance along the length for Q1A
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30

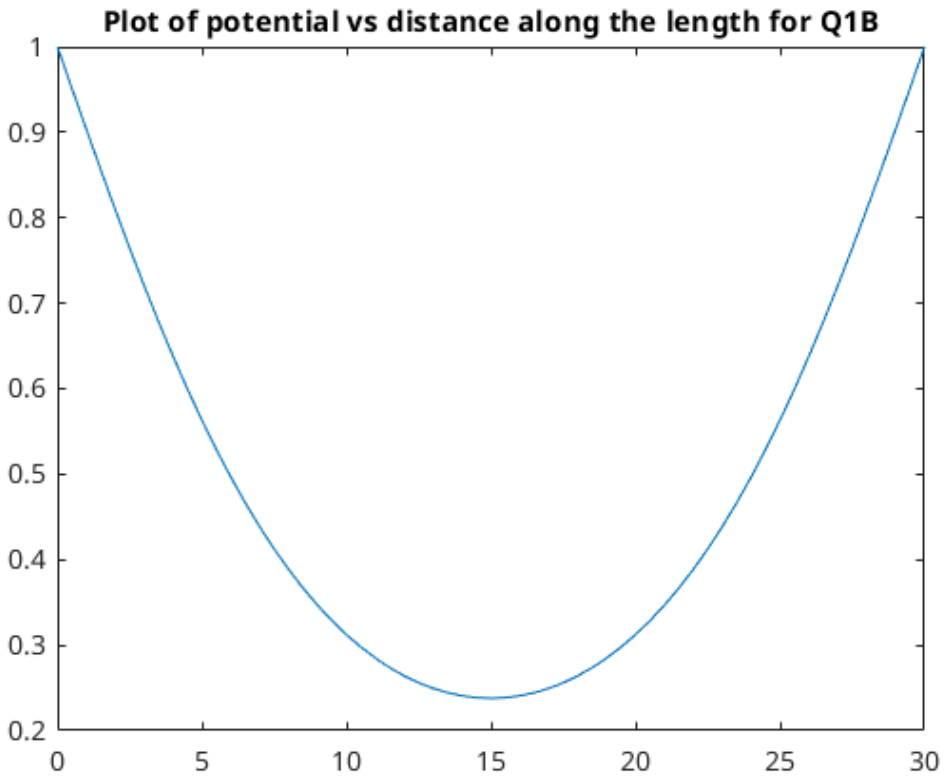

Map of Potential for Q1A
1
0.8
0.6
0.4
0.2
0
30
20
10
0
0
5
10
15
20

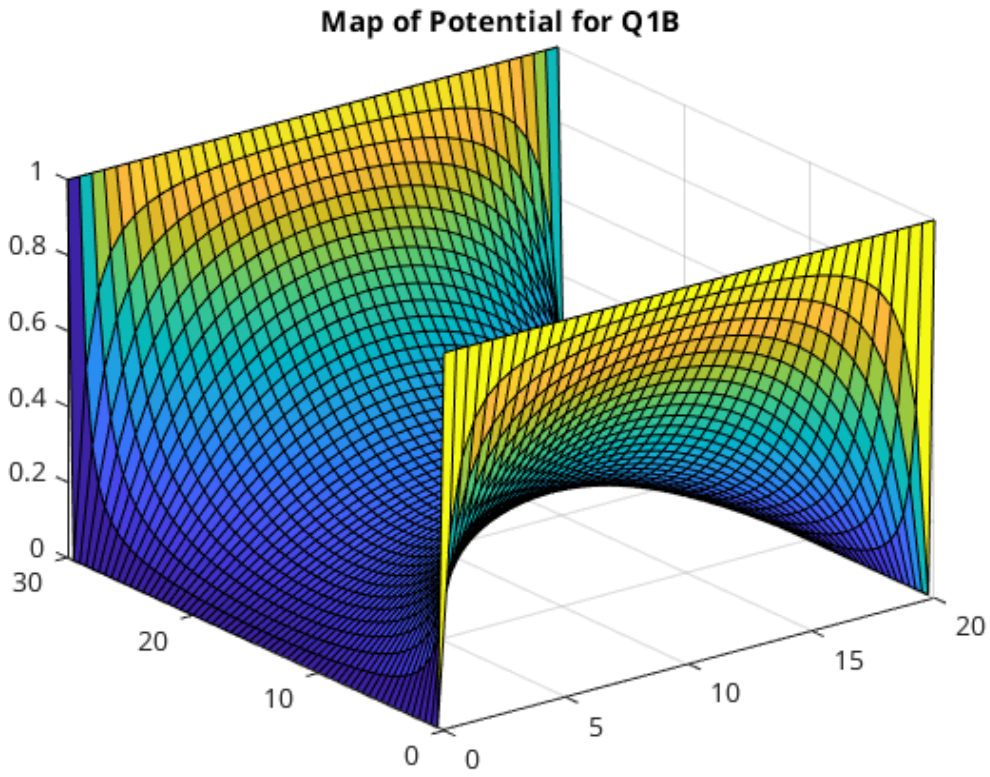
Question 1 b)

In this part the code is modified so that the boundary conditions are V=0 for the top and bottom boundaries and V=1 for the right and left sides, this is then compared with an analytical solution of the problem to see if they produce the same results and to find out the pros and cons of using each method

```
W = 20;
L = 1.5*W;

% Getting matrix dimensions
s = 0.5; % Space between each point in the meshgrid
nx = floor(L/s + 1);
ny = floor(W/s + 1);

% Creating matricies and mapping the 2D geometry
G = sparse(nx*ny);
Bc = zeros(1,nx*ny);

for i = 1:nx
    for j = 1:ny
        n = j+(i-1)*ny; % Mapping Equation

        % Setting boundary conditions
        if i == 1 || i == nx
            G(n,n) = 1;
            Bc(n) = 1;
        elseif j == 1 || j == ny
            G(n,n) = 1;
        else
            % Rest of the nodes
            nxm = j+(i-2)*ny;
            nxp = j+i*ny;
            nym = j-1+(i-1)*ny;
            nyp = j+1+(i-1)*ny;
            G(n,n) = -4;
            G(n,nxm) = 1;
            G(n,nxp) = 1;
            G(n,nym) = 1;
            G(n,nyp) = 1;
        end
    end
end

% Find V and map it to the space
V = G\Bc';
mappedV = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j+(i-1)*ny;
        mappedV(i,j) = V(n);
    end
end
```

```
% Plot
xaxis = 0:s:L;
figure(3)
plot(xaxis,mappedV(:,floor(ny/2)))
title('Plot of potential vs distance along the length for Q1B')

figure(4)
[xaxis, yaxis] = meshgrid(0:s:W, 0:s:L);
surf(xaxis,yaxis,mappedV)
title('Map of Potential for Q1B')
```

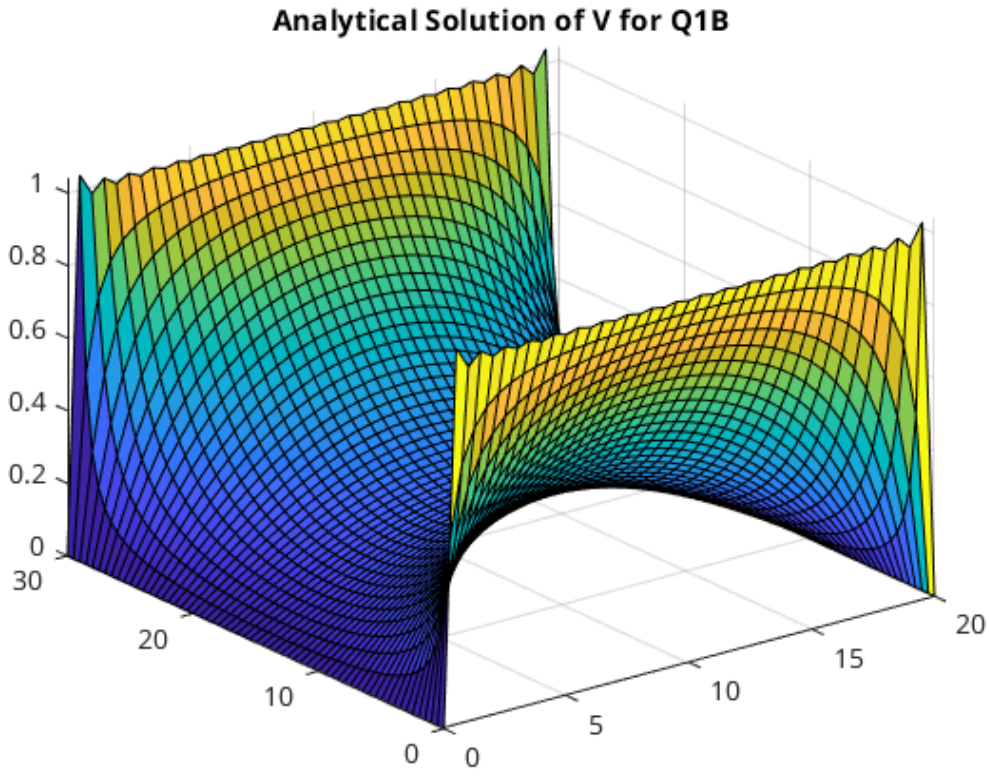




Analytic Solution

```
sol = zeros(nx,ny);
iMAX = 100;

[x,y] = meshgrid(linspace(-L/2,L/2,nx), linspace(0,W,ny));
a = W;
b = L/2;

for i = 1:iMAX
     n = 2*i-1;
     sol = sol+(4/pi).*(1/n).*(cosh((n*pi).*x'./a)./cosh((n*pi).*b./
a)).*sin((n*pi).*y'./a);
     figure(5)
     surf(xaxis,yaxis,sol)
     title('Analytical Solution of V for Q1B')
     pause(0.001)
end

% The analytical solution and the numerical solution both produce very
% similar results. The numerical solution is much faster, but it
 requires
% large matrix operations that make the iterations and mapping complex
 and harder to
% follow.
```

```
% The numerical solution would probably consume a lot of memory due
% to the large amount of data points (This is somwewhat mitigated when
% using sparse matrices but the effect still exists).
% The analytical solution is simpler to follow and could be more
 accurate,
% but it is much slower than the numerical solution due to the time
 and
% number of iterations it takes to converge. The movie helped to watch
 the
% solution to see how fast it converges.
```



Question 2

In this part, the code was modified so that there are two resistive boxes inside the region forming a 'bottle's neck' for the current to pass through. The code uses the same boundary confitions as the previous part. After remapping the G matrix and generating the V plot, the current is then examined with varying box with, conductivity, and mesh spacing to see how it changes.

```
W = 20;
L = 1.5*W;

Lb = 9;
Wb = 6;
s = 0.5;
nx = floor(L/s + 1);
ny = floor(W/s + 1);
```

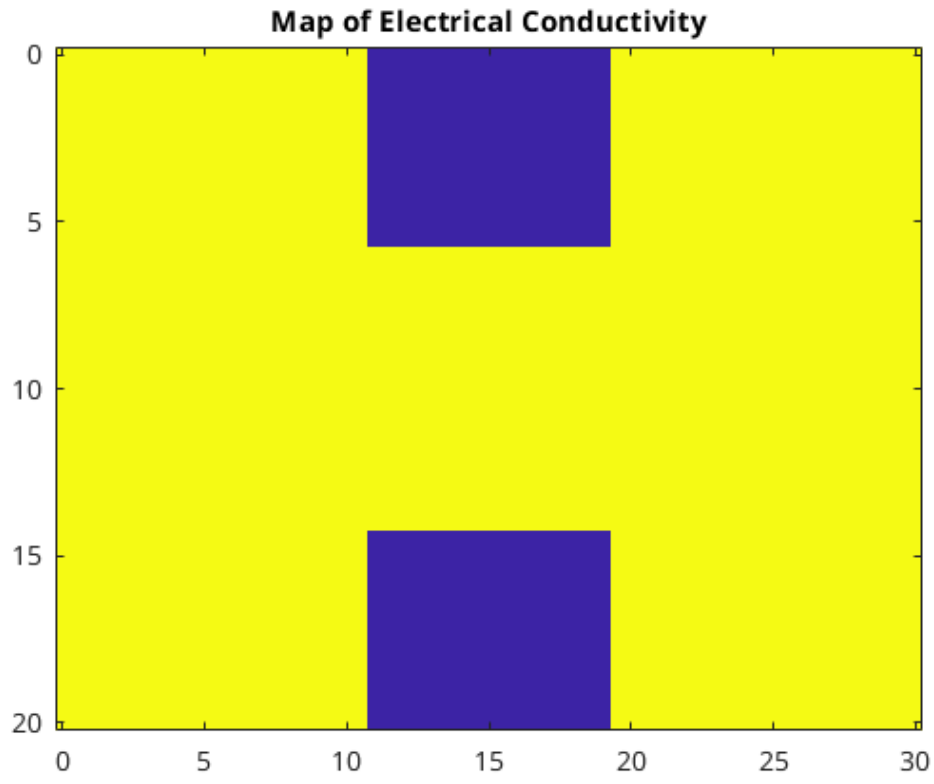
```
cond = 1; % Conductive
res = 1e-2; % Resistive

% Generating the conductivity map
condMap = zeros(nx,ny);
for i = 1:nx
   for j = 1:ny
       if (i-1>0.5*(L-Lb)/s) && ((i-1)<0.5*(L+Lb)/s) && (((j-1)<Wb/s||
(j-1)>(W-Wb)/s))
           condMap(i,j) = res;
       else
           condMap(i,j) = cond;
       end
   end
end

% Adding the map to the plot
figure(6)
imagesc([0 L],[0 W],condMap');
title('Map of Electrical Conductivity')
% Yellow region is conductive, blue is resistive

G = sparse(nx*ny);
Bc = zeros(1,nx*ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny; %Mapping Eq
        %Setting Boundary conditions
        if i == 1 || i == nx
            G(n,n) = 1;
            if i == 1
                Bc(n) = 1;
            end
        elseif j == 1 || j == ny
            nxm = j+(i-2)*ny;
            nxp = j+i*ny;
            nyp = j+1+(i-1)*ny;
            nym = j-1+(i-1)*ny;

            %Resistances from the conductivity map
            rxm = (condMap(i,j)+condMap(i-1,j))/2;
            rxp = (condMap(i,j)+condMap(i+1,j))/2;

            if j == 1
                ryp = (condMap(i,j)+condMap(i,j+1))/2;
                G(n,n) = -(rxm+rxp+ryp);
                G(n,nyp) = ryp;
            else
                rym = (condMap(i,j)+condMap(i,j-1))/2;
                G(n,n) = -(rxm+rxp+rym);
                G(n,nym) = rym;
            end
            G(n,nxm) = rxm;
```

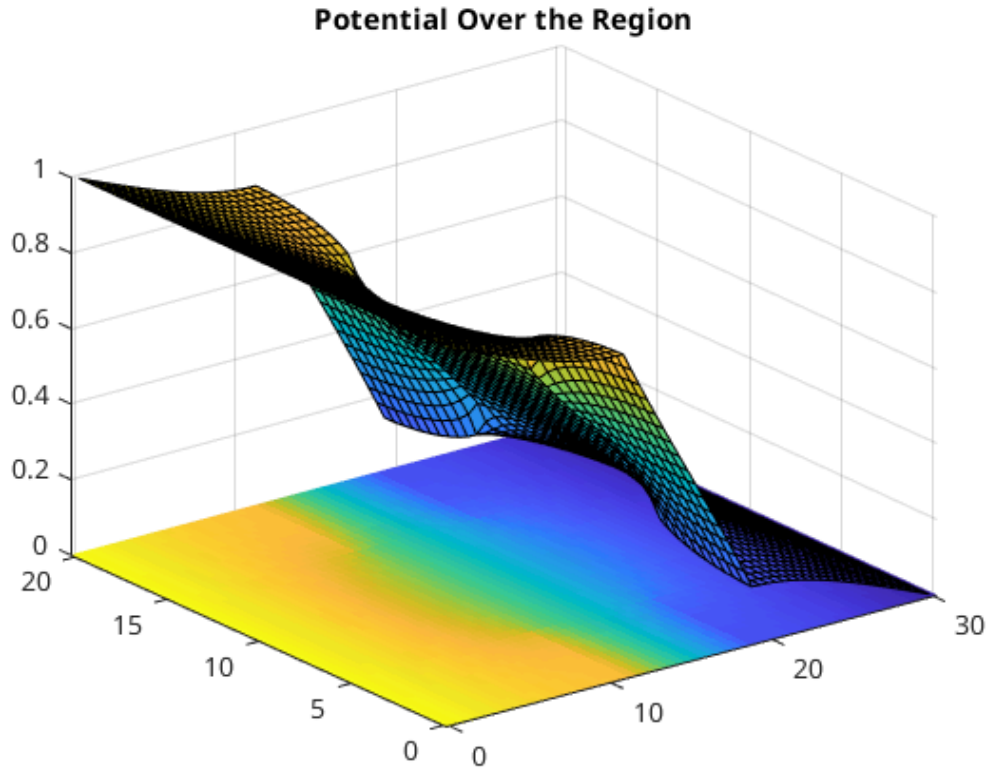
```
            G(n,nxp) = rxp;

        % Rest of the nodes
        else
            nxm = j+(i-2)*ny;
            nxp = j+i*ny;
            nym = j-1+(i-1)*ny;
            nyp = j+1+(i-1)*ny;

            % Getting the resistances from the conduction map
            rxm = (condMap(i,j)+condMap(i-1,j))/2;
            rxp = (condMap(i,j)+condMap(i+1,j))/2;
            ryp = (condMap(i,j)+condMap(i,j+1))/2;
            rym = (condMap(i,j)+condMap(i,j-1))/2;

            % Assigning the node equations
            G(n,n) = -(rxm+rxp+rym+ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;
        end
    end
end

%solving and plotting
V = G\Bc';
mappedV = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j+(i-1)*ny;
        mappedV(i,j) = V(n);
    end
end

[xaxis, yaxis] = meshgrid(0:s:L,0:s:W);
figure(7)
surf(xaxis',yaxis',mappedV)
hold on
imagesc([0 L],[0 W],mappedV')
title('Potential Over the Region')
hold off

[Ey, Ex] = gradient(-mappedV);

figure(8)
quiver(xaxis',yaxis',Ex,Ey)
xlim([0 L])
ylim([0 W])
title('E Field Quiver')
Jx = condMap.*Ex;
Jy = condMap.*Ey;

figure(9)
```

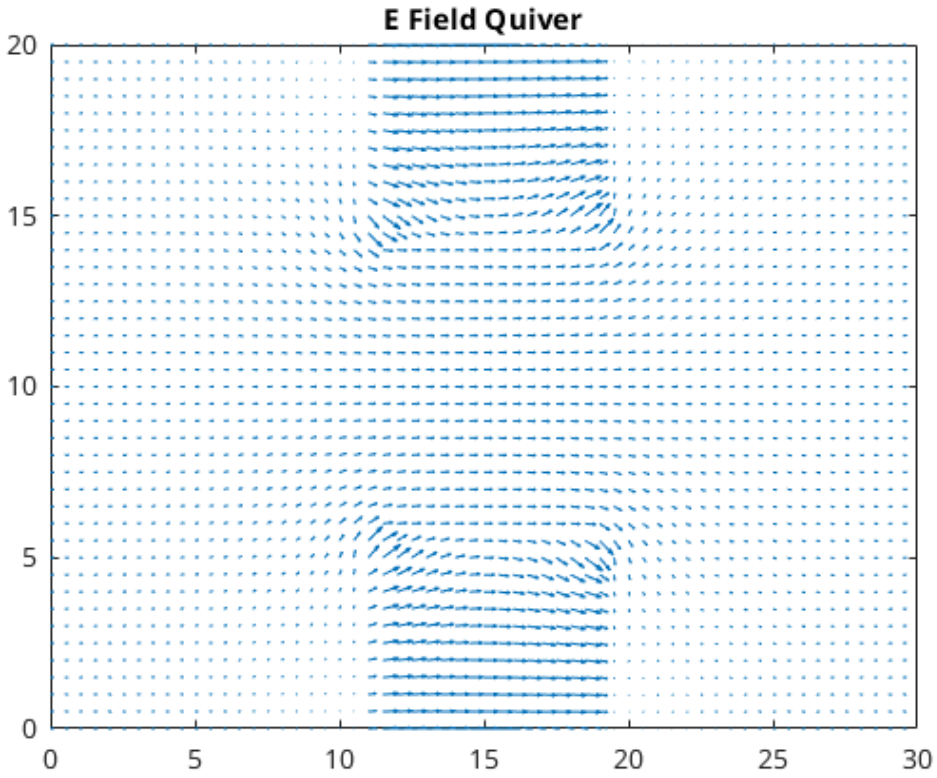
```
quiver(xaxis',yaxis',Jx,Jy)
xlim([0 L])
ylim([0 W])
title('Current Density Quiver')

% Currents at the two contacts
I1 = sum(Jx(1,:))
I2 = sum(Jx(nx,:))


% Generating the other plots
s = linspace(0.5, 10, 20);
for i = 1:20
    I(i) = Curr(s(i), res, Wb);
end
figure(10)
plot(s,I)
title('Plot of current vs mesh density')
% The current generally decreases with a higher mesh density, but
% specifying a lower mesh density leads to innaccurate results as seen
 in
% the figure

cond = linspace(0, 1, 20);
for i = 1:20
    I(i) = Curr(0.5, cond(i), Wb);
end
figure(11)
plot(cond,I)
title('Plot of current vs conductivty of the boxes')
% As expected the current increases when the box is more conductive,
 after
% the conductivity reaches 1, the box has the same conductivity as the
 rest
% of the region, so the current would eventually stop changing.

Wb = linspace(0,12,20);
for i = 1:20
    I(i) = Curr(0.5, 0.01, Wb(i));
end
figure(12)
plot(Wb,I)
title('Plot of current vs box width')
% The wider the box the lower the current that passes, when the width
% reaches 10, the current is constant because the resistive boxes
 would
% have the same width as the region.


% Creating a reuseable function that outputs the current
% This will be used for the remaining plots

function [I] = Curr(s, cond, Wb)
W = 20;
```

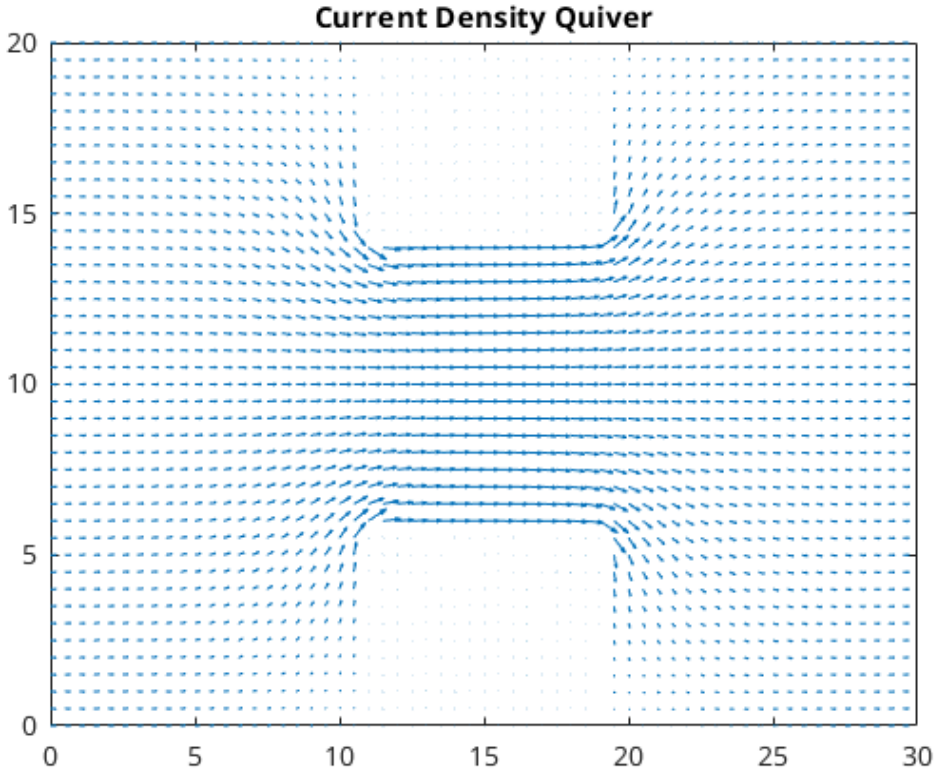
```
L = 1.5*W;
Lb = 9;
nx = floor(L/s+1);
ny = floor(W/s+1);

    condMap = zeros(nx,ny);
    for i = 1:nx
        for j = 1:ny
            if (i-1>0.5*(L-Lb)/s) && ((i-1)<0.5*(L+Lb)/s) &&
 (((j-1)<Wb/s||(j-1)>(W-Wb)/s))
                condMap(i,j) = cond;
            else
                condMap(i,j) = 1;
            end
        end
    end

    G = sparse(nx*ny);
    Bc = zeros(1,nx*ny);

    for i = 1:nx
        for j = 1:ny
            n = j +(i-1)*ny; %Mapping Eq
            %Setting Boundary conditions
            if i == 1 || i == nx
                G(n,n) = 1;
                if i == 1
                    Bc(n) = 1;
                end
            elseif j == 1 || j == ny
                nxm = j+(i-2)*ny;
                nxp = j+i*ny;
                nyp = j+1+(i-1)*ny;
                nym = j-1+(i-1)*ny;

                % Resistances from the conductivity map
                rxm = (condMap(i,j)+condMap(i-1,j))/2;
                rxp = (condMap(i,j)+condMap(i+1,j))/2;

                if j == 1
                    ryp = (condMap(i,j)+condMap(i,j+1))/2;
                    G(n,n) = -(rxm+rxp+ryp);
                    G(n,nyp) = ryp;
                else
                    rym = (condMap(i,j)+condMap(i,j-1))/2;
                    G(n,n) = -(rxm+rxp+rym);
                    G(n,nym) = rym;
                end

            G(n,nxm) = rxm;
            G(n,nxp) = rxp;

            % Rest of the nodes
            else
```

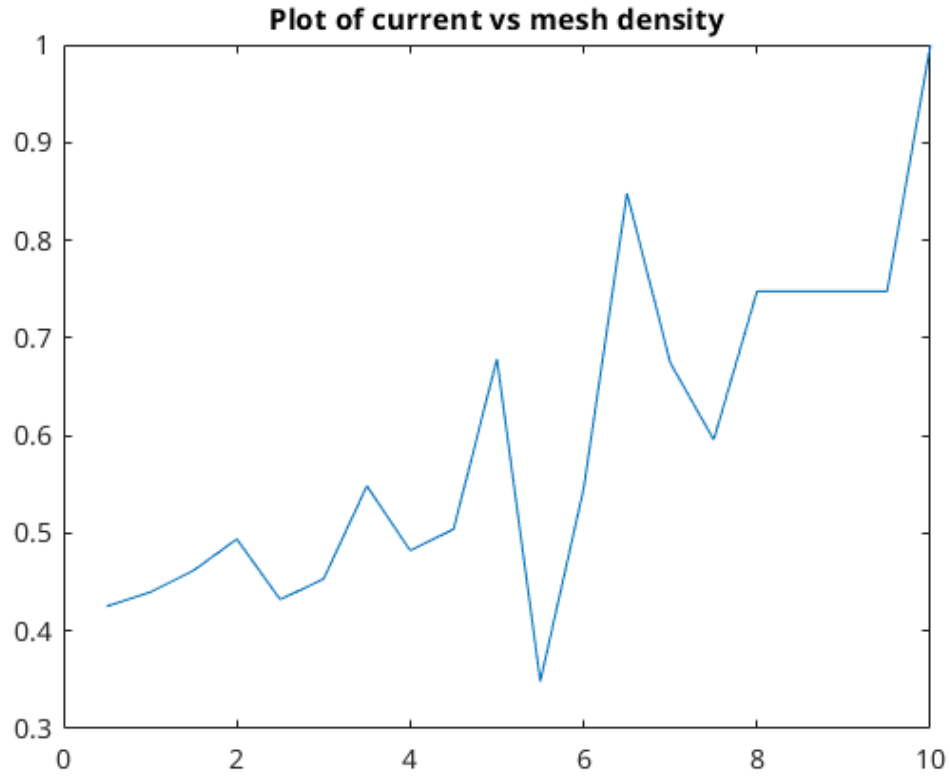
```
                nxm = j+(i-2)*ny;
                nxp = j+i*ny;
                nym = j-1+(i-1)*ny;
                nyp = j+1+(i-1)*ny;

                % Getting the resistances from the conduction map
                rxm = (condMap(i,j)+condMap(i-1,j))/2;
                rxp = (condMap(i,j)+condMap(i+1,j))/2;
                ryp = (condMap(i,j)+condMap(i,j+1))/2;
                rym = (condMap(i,j)+condMap(i,j-1))/2;

                % Assigning the node equations
                G(n,n) = -(rxm+rxp+rym+ryp);
                G(n,nxm) = rxm;
                G(n,nxp) = rxp;
                G(n,nym) = rym;
                G(n,nyp) = ryp;
            end
        end
    end

    V = G\Bc';
    mappedV = zeros(nx,ny);
    for i = 1:nx
        for j = 1:ny
            n = j+(i-1)*ny;
            mappedV(i,j) = V(n);
        end
    end

    [Ey, Ex] = gradient(-mappedV);
    Jx = condMap.*Ex;
    Jy = condMap.*Ey;

    I = sum(Jx(1,:));
end


I1 =

    0.4253


I2 =

    0.4253

Warning: Matrix is singular to working precision.
```

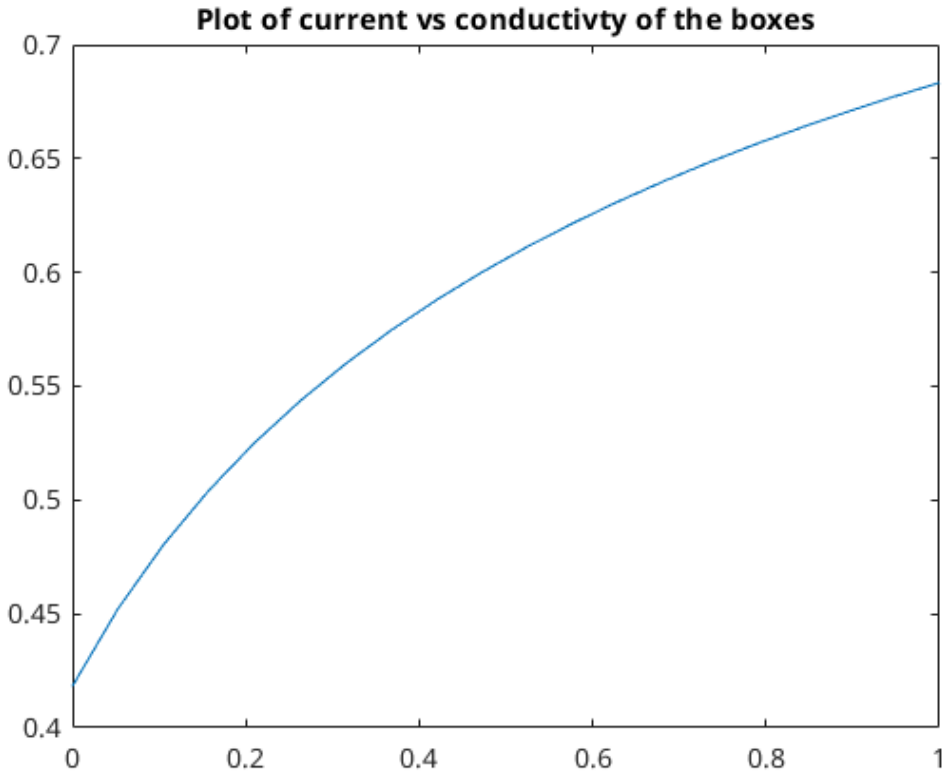

Map of Electrical Conductivity
0
5
10
15
20
0
5
10
15
20
25
30

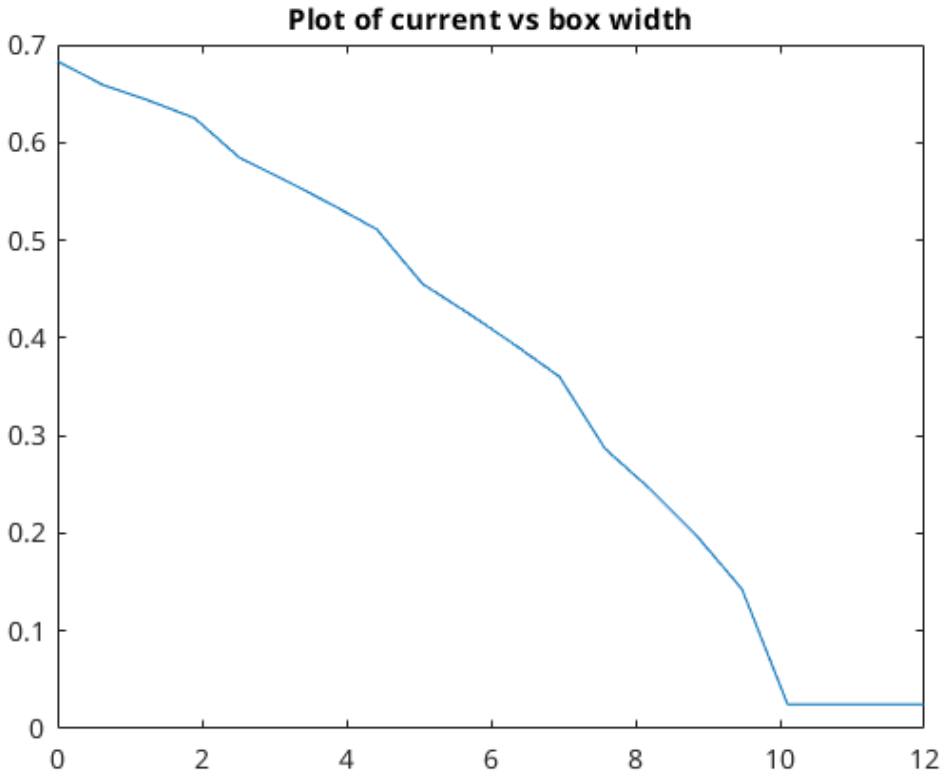

Potential Over the Region
1
0.8
0.6
0.4
0.2
0
20
15
10
5
0
0
10
20
30


E Field Quiver
20
15
10
5
0
0
5
10
15
20
25
30


Current Density Quiver
20
15
10
5
0
0
5
10
15
20
25
30


Plot of current vs mesh density
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10


Plot of current vs conductivty of the boxes
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0
0.2
0.4
0.6
0.8
1



Published with MATLAB® R2020b