ELEC 4700 Assignment 2

Table of Contents

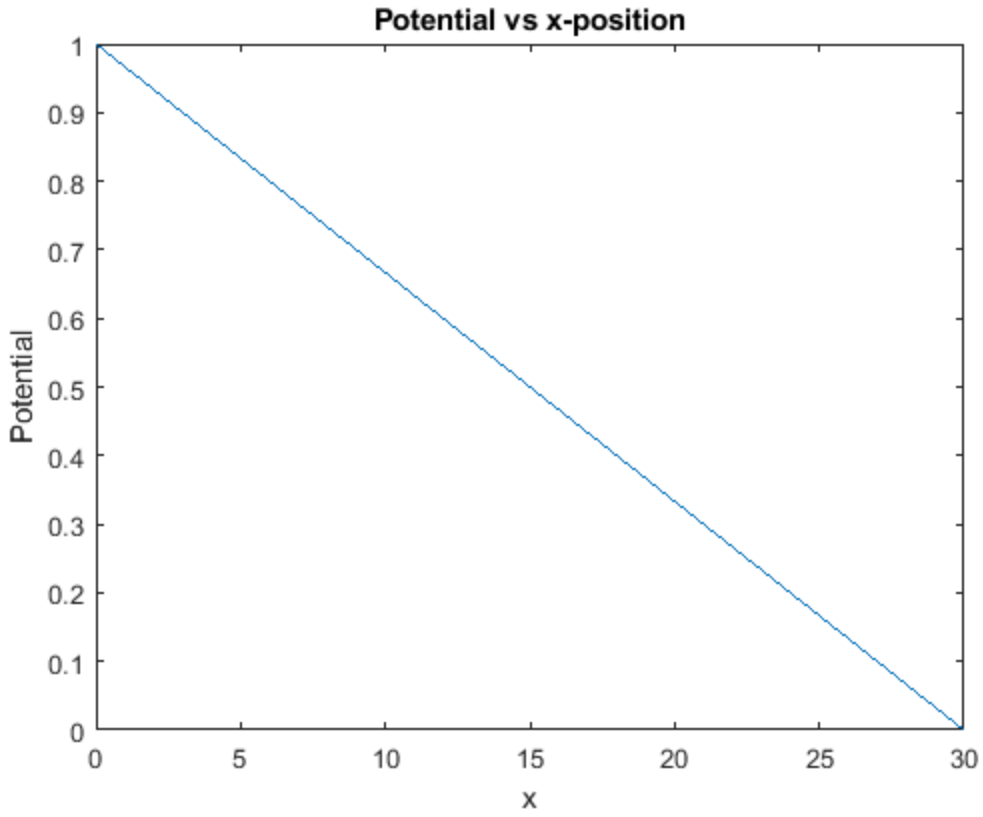
Question 1a)

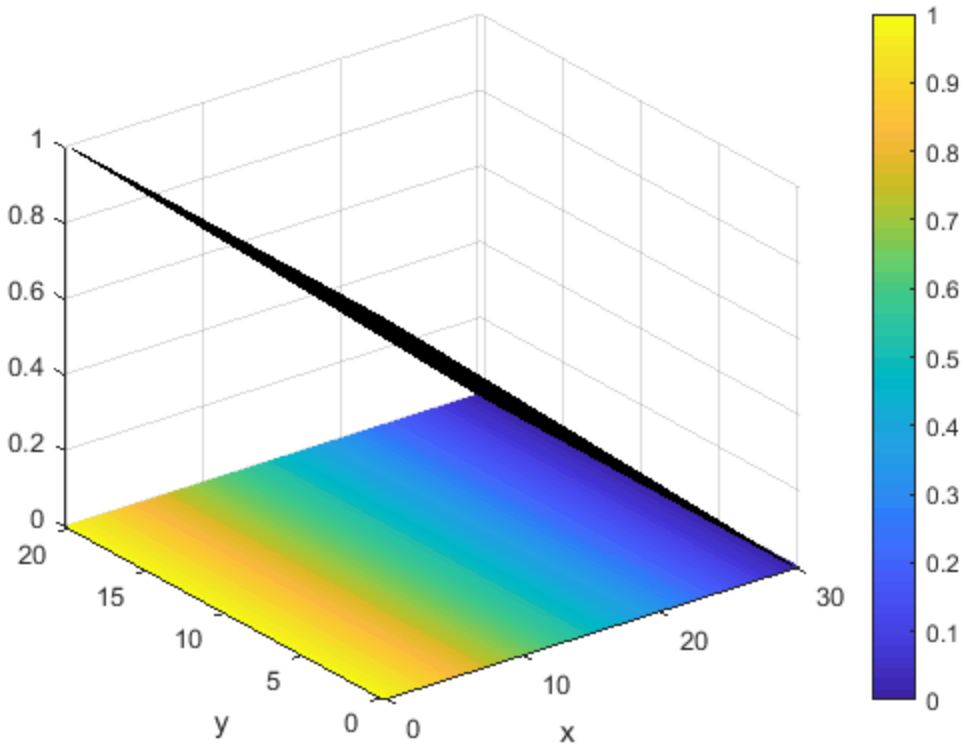
In this question I developed code that uses the finite difference method to solve the Laplace equation in a rectagular region with voltage fixed at V=1 at x=0 and V=0 at x=L. The other 2 boundaries (y=0 and y=W) are absorbing boundary conditions (dV/dx=0). In order to this the region must be discretized (meshed) and the laplacian operator must be discretized and converted into a Matrix, and the potential function will be discretized and turned into a vector. The problem can now be formulated as GV = B. Where G is the discrete laplacian operator, V is the potential and B is a vector of mostly 0s and a few 1s due to boundary conditions. This type of problem can be easily solved in MATLAB to determine the potential in the rectangular region.

```
L = 30; %set the geometry of the region
W = 20;

%set the distance between meshpoints and determine the number points
 used
meshspace = 0.5;
nx = floor(L/meshspace + 1);
ny = floor(W/meshspace + 1);

G = sparse(nx*ny); %% Discretized Laplacian Operator
B = zeros(1,nx*ny); %% Mostly zeros but a few ones for BCs
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny; %Map 2d Geometry to 1D Vector

        %Left side V=1 BC
        if i == 1
            G(n,n) = 1;
            B(n) = 1;

        %Right side V=0 BC
        elseif i == nx
            G(n,n) = 1;

        %Bottom Side Absorbing boundary condition
        elseif j == 1
            nxm = j + (i-2)*ny;
            nxp = j + i*ny;
            nyp = j+1 + (i-1)*ny;
```

```
            G(n,n) = -3;
            G(n,nxm) = 1;
            G(n,nxp) = 1;
            G(n,nyp) = 1;


        %Top side absorbing boundary condition
        elseif j == ny
            nxm = j + (i-2)*ny;
            nxp = j + i*ny;
            nym = j-1 + (i-1)*ny;

            G(n,n) = -3;
            G(n,nxm) = 1;
            G(n,nxp) = 1;
            G(n,nym) = 1;


        %Inner Nodes
        else
            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

%Solve Matrix equation to find V
V = G\B';

%Map solution back to 2D space
Vmap = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny;
        Vmap(i,j) = V(n);
    end
end

%Plot potential
figure(1)
x = 0:meshspace:L;
plot(x,Vmap(:,floor(ny/2)))
title('Potential vs x-position')
xlabel('x')
```

```
ylabel('Potential')
[X, Y] = meshgrid(0:meshspace:L,0:meshspace:W);
figure(2)
surf(X',Y',Vmap)
colorbar
hold on
imagesc([0 L],[0 W],Vmap')
xlabel('x')
ylabel('y')
hold off
```





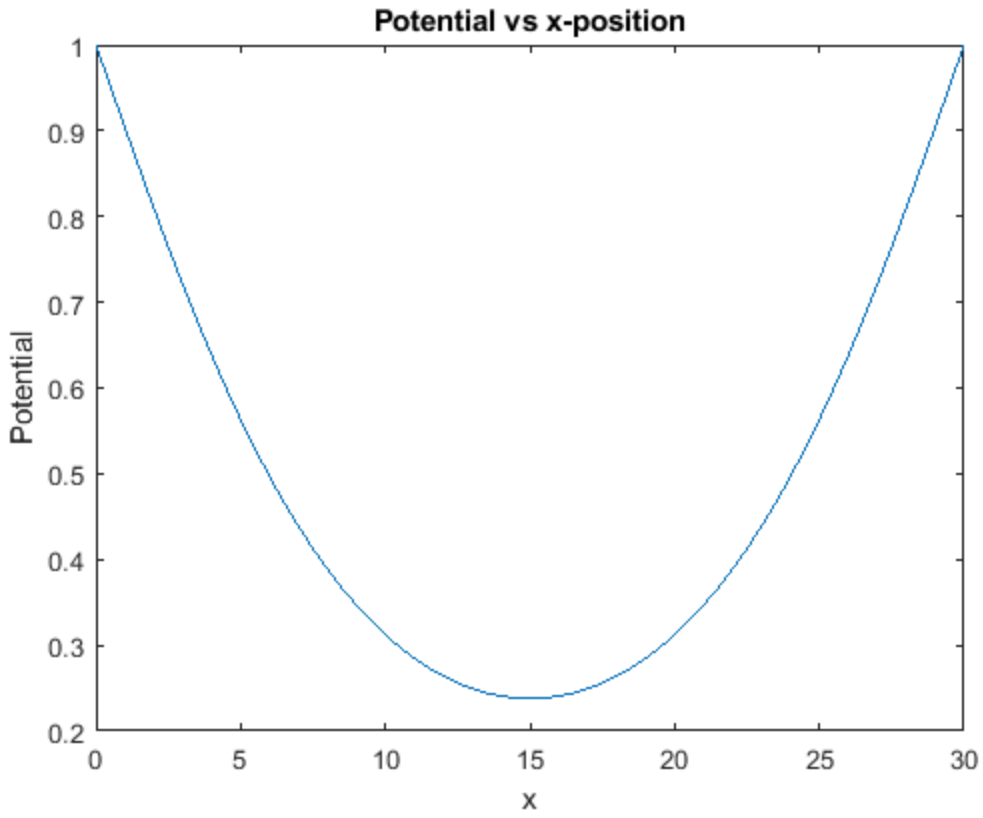
Question 1b)

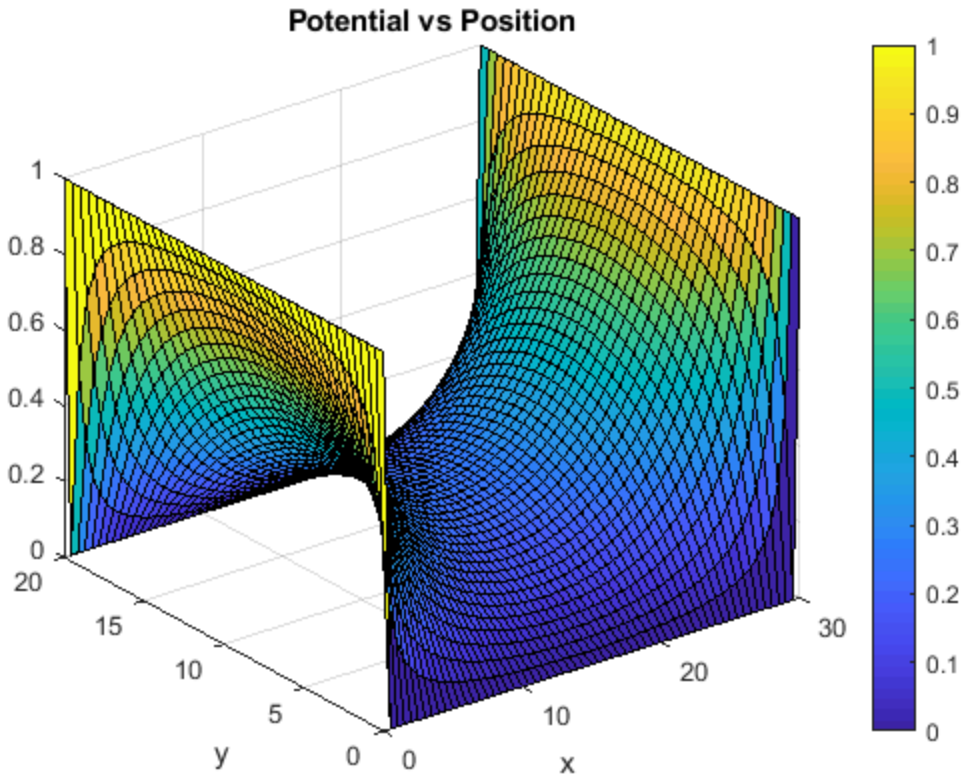
In this question the above code is modified so that the absorbing boundary conditions are now instead fixed at V=0 and the other boundary conditions are modied so that V=1 at x=0 and x=L. Then this code will be ran at several different mesh densities and compared to the analytical solution. The code to solve this problem is shown below.

```
%set the geometry of the region
L = 30;
W = 20;

%set the distance between meshpoints and determine the number points
 used
meshspace = 0.5;
nx = floor(L/meshspace + 1);
ny = floor(W/meshspace + 1);

G = sparse(nx*ny); %% Discretized Laplacian Operator
B = zeros(1,nx*ny); %% Mostly zeros but a few ones for BCs

for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny; %Map 2D geometry to 1D vector

        % V=1 @ x=0 BC
        if i == 1
```

```
            G(n,n) = 1;
            B(n) = 1;

        % V=1 @x=L BC
        elseif i == nx
            G(n,n) = 1;
            B(n) = 1;

        %V=0 @ y=0 BC
        elseif j == 1
           G(n,n) = 1;

        %V=0 @ y=W BC
        elseif j == ny
            G(n,n) = 1;

        %Matrix elemenets for inner nodes
        else
            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

%Solve Matrix equation to find V
V = G\B';

%Map solution back to 2D space
Vmap = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny;
        Vmap(i,j) = V(n);
    end
end

%Plot potential
x = 0:meshspace:L;
figure(1)
plot(x,Vmap(:,floor(ny/2)));
title('Potential vs x-position')
xlabel('x')
ylabel('Potential')
figure(2)
[X, Y] = meshgrid(0:meshspace:L,0:meshspace:W);
```

```
surf(X',Y',Vmap)
colorbar
xlabel('x')
ylabel('y')
title('Potential vs Position')
```





This code was now used to compare the numerical solution to the analytical solution at several different mesh sizes. These tests showed that as the meshing was smaller and smaller, the numerical solution appered smoother and better fit the analytical solution. However, as the mesh sized is decreased the simulation time begins to rise quite quickly and the accuraccy does not increase by much. Therefore, there comes a point where increasing the mesh density is overkill. From these experiments I found that a mesh size of 0.05 struck a good balance between accuracy and simulation time. As the size was decreased from 0.5 the simulation time began to rise very quickly while the accuracy only increased slightly. Both the analytical method and numerical method have some inherent error. The analytical method is expressed as an infinite series. In order to perform a calculation, this series must be truncated at some point. This error can be reduced by including more and more terms of the series but this will increase computiation time. The numerical solution has errors due to the fact that space is discretized and the differential operators have to approximated as discrete matrix operators. This error can be reduced by decreasing the mesh size but this will increase computation time. The analytical solution has the advatantage that it is a closed form expression making it fast to evaluate and accurate since it is an exact solution to the problem. The disadvantage of the analytical solution is that it only applies to this scenario and it is very difficult and usually impossible to derive solutions for other scenarios. The advantages of the numericale solution is that it works for a variety of geometries and scenarios ad it is relativly easy to implement. The disadvantage of the numerical solution is that it is slower than the analytical solution and it is less accurate than the analytical solution.

Question 2

In this part of the lab MATLAB code was written to solve Laplace equation in a rectangular region of high conductivity but containing two low conductivity boxes the form a bottleneck in the centre. This complicates laplaces equation because now the coefficents in front of the partial differentials vary depending on the conductivity of the material and the G matrix must be modified. In order to this, it is useful to think

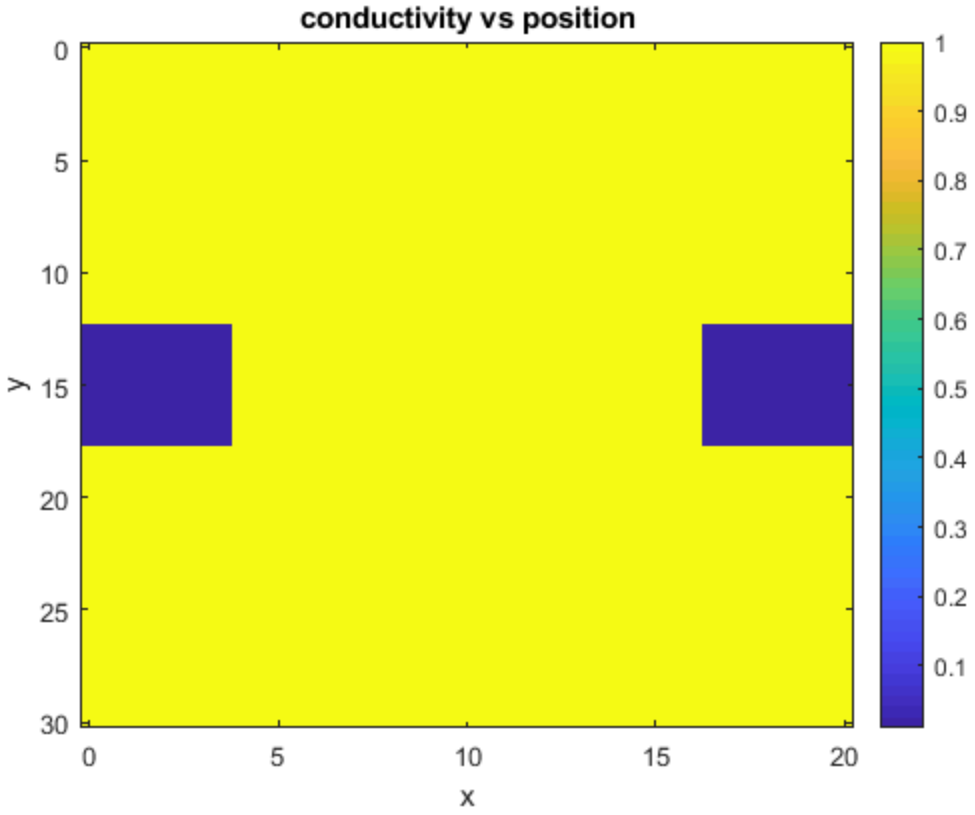
of each node as onnected to its four neighbouring nodes with a resistance equal to the conductivity of the material (average conductivity if nodes lie in seperate materials). This approach makes it easy to write the equations for each node and combine them into the G matrix. the boundary at x=0 is V=1 and x=L is V=0. The other two boundarires are absorbing boundaries. The code for this simulation is shown below.

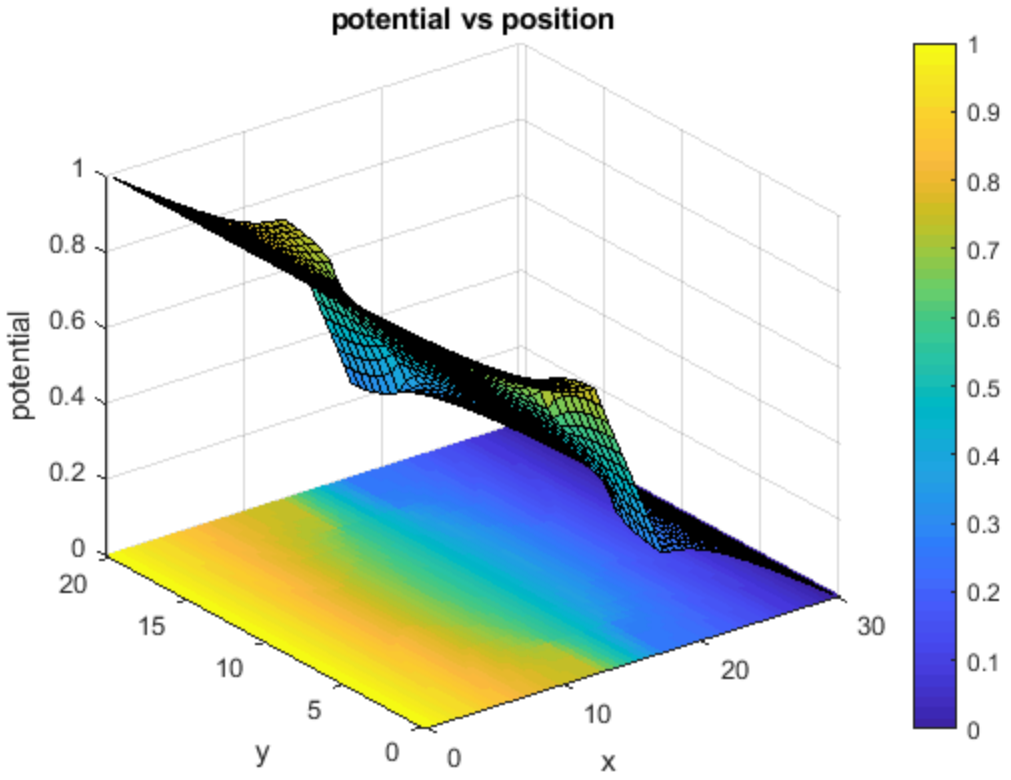
```
L = 30;
W = 20;
Lb = 6;
Wb = 4;
meshspace = 0.5;
nx = floor(L/meshspace + 1);
ny = floor(W/meshspace + 1);

cond1 = 1; % Backgground conductivity
cond2 = 1e-2; %Conductivity of 2 boxes


%This section creates a conductivity map witch assigns a conductivity
 to
%each node in the mesh. This will be used to create the G matrix
condMap = zeros(nx,ny);

for i = 1:nx
   for j = 1:ny
       if (i-1)>0.5*(L-Lb)/meshspace&&(i-1)<0.5*(L+Lb)/
meshspace&&((j-1)<Wb/meshspace||(j-1)>(W-Wb)/meshspace)
           condMap(i,j) = cond2;
       else
           condMap(i,j) = cond1;
       end
   end
end

figure(1)
imagesc([0 W],[0 L],condMap);
colorbar
xlabel('x')
ylabel('y')
title('conductivity vs position')

G = sparse(nx*ny);
B = zeros(1,nx*ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny;

        %V=1 @ x=0 BC
        if i == 1
            G(n,n) = 1;
            B(n) = 1;

        %V=0 @ x=L BC
        elseif i == nx
            G(n,n) = 1;
```

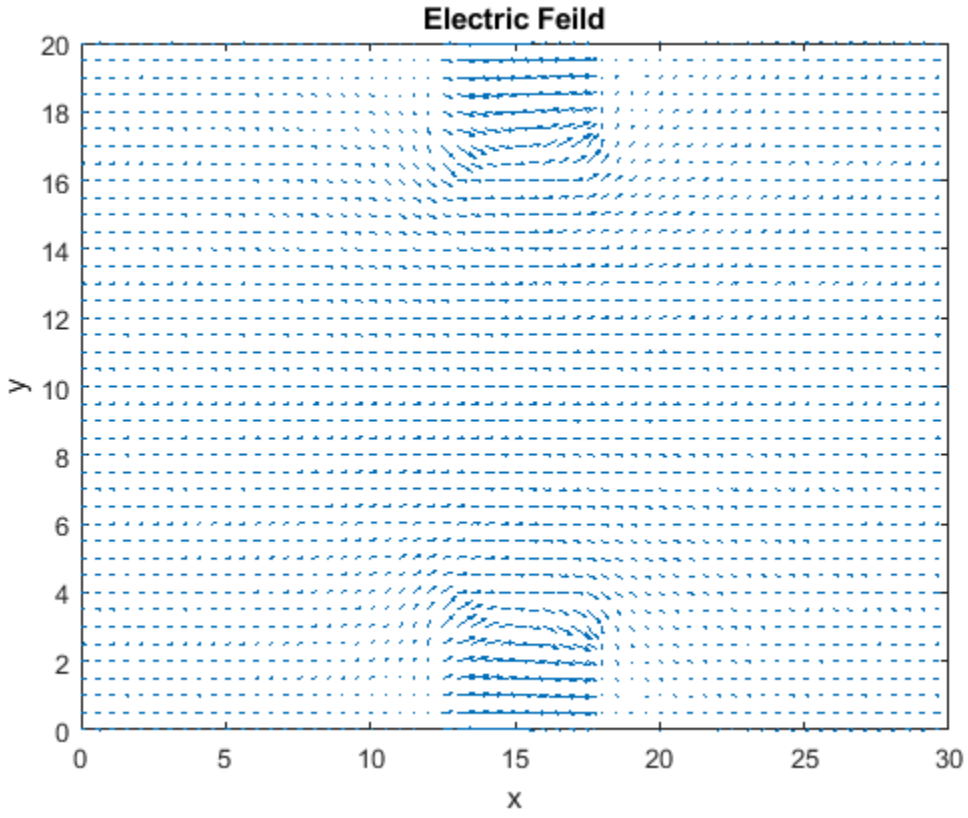
```
%Absorbing BC @ y=0
elseif j == 1
    nxm = j + (i-2)*ny;
    nxp = j + i*ny;
    nyp = j+1 + (i-1)*ny;

    %Resistor Values from 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;

    %node equations from resistor values
    G(n,n) = -(rxm + rxp + ryp);
    G(n,nxm) = rxm;
    G(n,nxp) = rxp;
    G(n,nyp) = ryp;

 %Absorbing BC @ y=W
elseif j == ny
    nxm = j + (i-2)*ny;
    nxp = j + i*ny;
    nym = j-1 + (i-1)*ny;

    %Resistor Values from conduction map
    rxm = (condMap(i,j) + condMap(i-1,j))/2;
    rxp = (condMap(i,j) + condMap(i+1,j))/2;
    rym = (condMap(i,j) + condMap(i,j-1))/2;

    %node equations from resistor values
    G(n,n) = -(rxm + rxp + rym);
    G(n,nxm) = rxm;
    G(n,nxp) = rxp;
    G(n,nym) = rym;

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

    %Resistor Values from 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;

    %node equations from resistor values
    G(n,n) = -(rxm + rxp + rym + ryp);
    G(n,nxm) = rxm;
    G(n,nxp) = rxp;
    G(n,nym) = rym;
    G(n,nyp) = ryp;
```

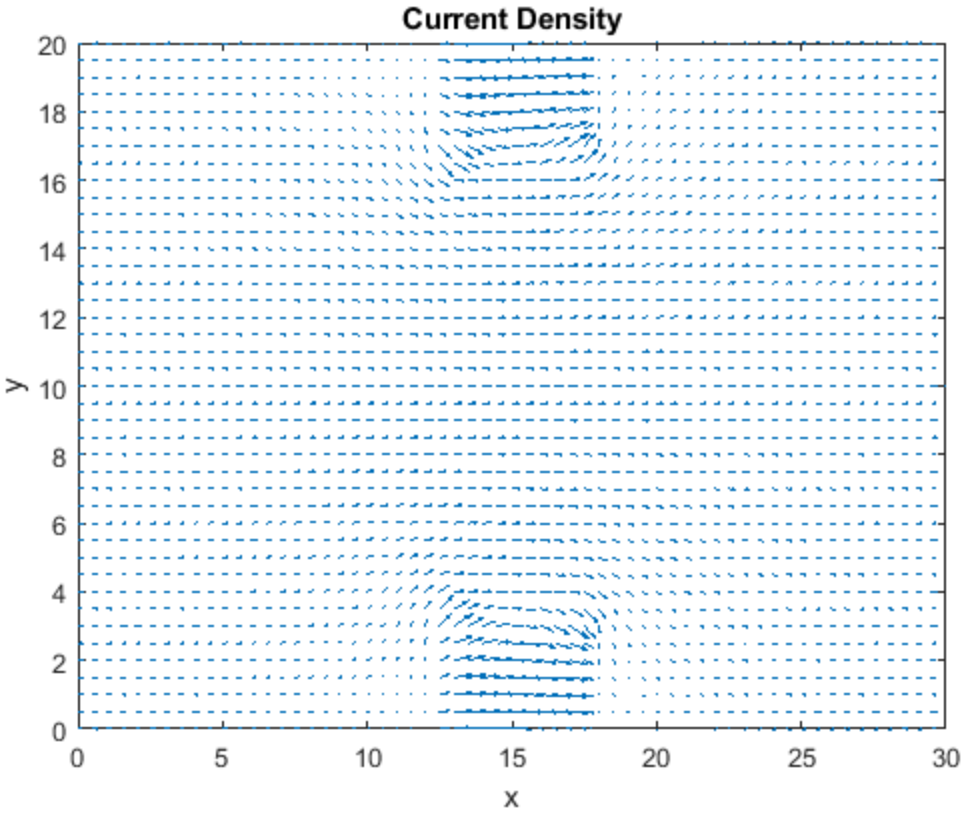
```
        end
    end
end

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

[X, Y] = meshgrid(0:meshspace:L,0:meshspace:W);
figure(2)
surf(X',Y',Vmap)
colorbar
hold on
imagesc([0 L],[0 W],Vmap')
xlabel('x')
ylabel('y')
zlabel('potential')
title('potential vs position')
hold off

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

figure(3)
quiver(X',Y',Ex,Ey)
xlim([0 30])
ylim([0 20])
xlabel('x')
ylabel('y')
title('Electric Feild')
Jx = condMap.*Ex;
Jy = condMap.*Ey;

figure(4)
quiver(X',Y',Ex,Ey)
xlim([0 30])
ylim([0 20])
xlim([0 30])
ylim([0 20])
xlabel('x')
ylabel('y')
title('Current Density')

%Currents at each terminal. Should be equal.
I1 = sum(Jx(1,:))
I2 = sum(Jx(nx,:))
```

```
I1 =

    0.5651

I2 =

    0.5651
```



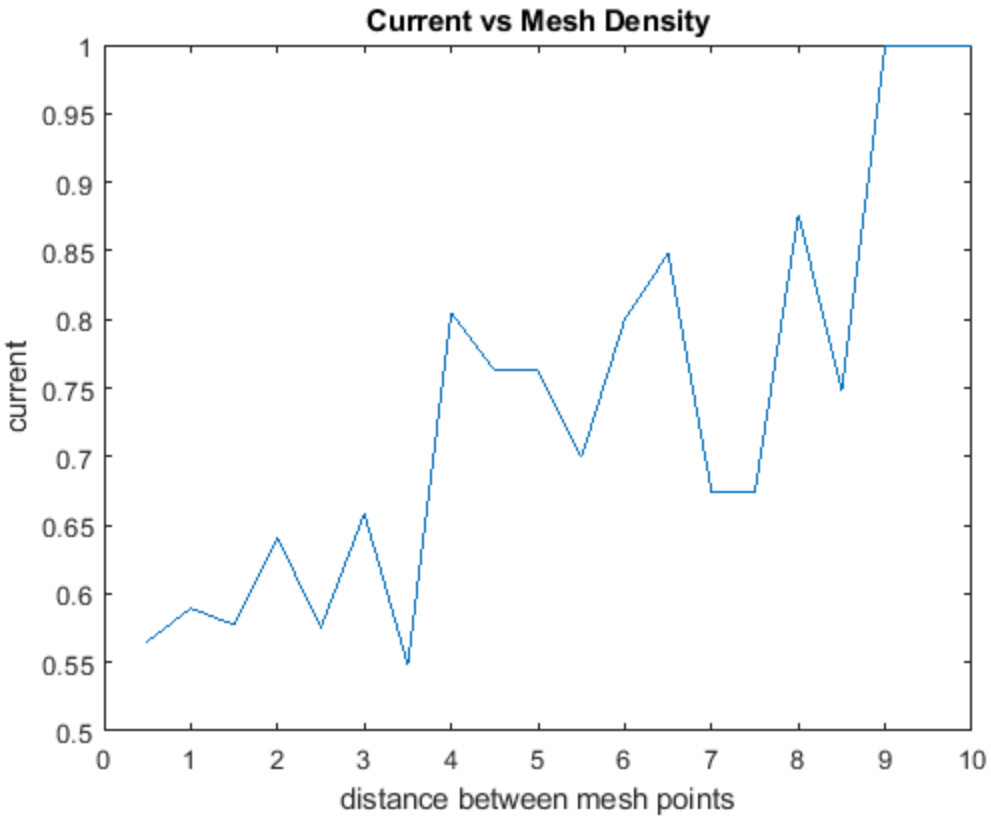

potential vs position
potential
1
0.8
0.6
0.4
0.2
0
20
15
10
5
0
y
0
10
20
30
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Electric Feild
y
20
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
x



The results of this simulation show that the current at each terminal (x=0 and x=L) are equal to eachother. This is the expected result for Kirchoff's current law. Now the effects of mesh size, bottleneck size, and background conductivity on the overall current will be explored. In ioder to this, I took all the code shown above that is neccessary for calculation of current and put it in a function called Bottleneck(meshspace,cond2,Wb). This function takes the distance between mesh points, box conductivity, and width of the boxes as inputs and returns the current. The code for this function is almost identicle to the code shown above. This function was used to generate plots as shown below.

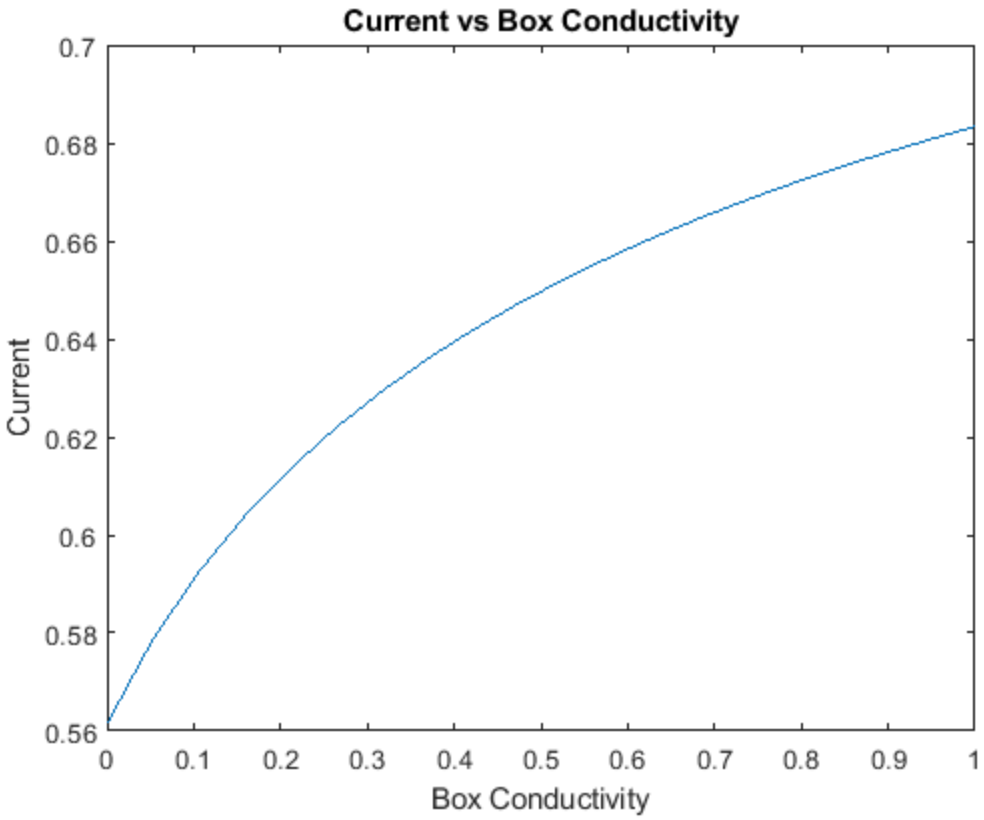
```
meshspace = linspace(0.5, 10, 20);
for i =1:20
    I(i) = Bottleneck(meshspace(i),0.01,4); %currents for plotting
end
plot(meshspace,I)
xlabel('distance between mesh points')
ylabel('current')
title('Current vs Mesh Density')
```



This plot shows the general trend of the current decreasing with mesh size but stabalizing at a value a little over 0.5. It also shows that at a low mesh density the current values vary significantly due to the innacuracy of the calculation.

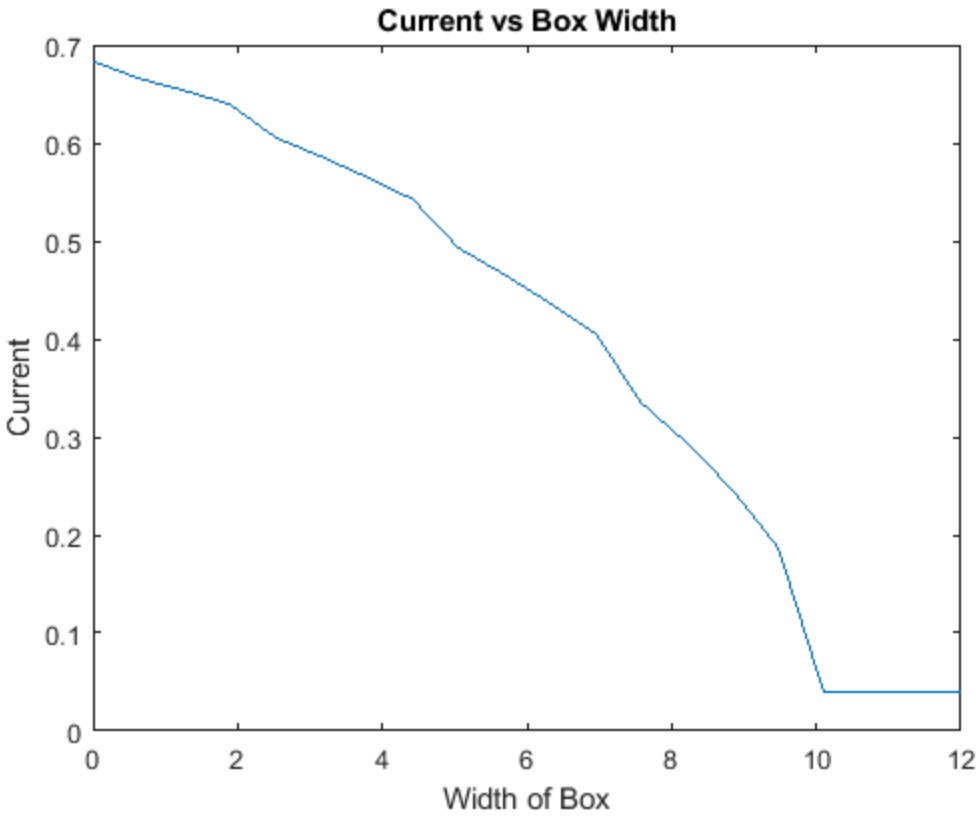
```
cond2 = linspace(0, 1, 20);
for i =1:20
    I(i) = Bottleneck(0.5,cond2(i),4); %currents for plotting
end
plot(cond2,I)
xlabel('Box Conductivity')
ylabel('Current')
title('Current vs Box Conductivity')
```

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



This plot shows the effect of varying the conductivity inside the boxes. When the conductivity is zero that is equivalent to the boxes being a perfect insulator and when it is 1 it is equialent to the boxes having the same conductivity as the background. As expected increasing the conductivity in the box increases the total current as the current would be able to flow more easily inside the box.

```
Wb = linspace(0, 12, 20);
for i =1:20
    I(i) = Bottleneck(0.5,0.01,Wb(i)); %currents for plotting
end
plot(Wb,I)
xlabel('Width of Box')
ylabel('Current')
title('Current vs Box Width')
```



This plot shows the effect of varying the width of the low conductivity box on the overall current. When the width is zero that is equivalent to having no box present and when it is 10 it is equivalent to havng one box that is the width of the region. As expected, as the current decreases as the box width is increased, and then stays constant after it is increased past 10.

Published with MATLAB® R2018a