Table of Contents

Ben Linton 100969769 I received help from my colleagues when completing this assignment.

Part 1 Electrostatic Potential in a 1-D Rectangular Region

In part 1 the student computed the electrostatic potential of two different set-ups in a rectangular region. The student used the Finite Difference method and the matrix formulation of the problem (GV=F) for his methodology.

The finite difference method is a method of replacing derivatives in an equation with 'differences' and in the case of electrostatic potential....

Electrostatic Potential equation : $$\frac{\partial^2}{\partial x^2}V + \frac{\partial^2}{\partial y^2}V$$

Finite Difference formulation : $$\frac{V_{x-1,y} - 2V_{x,y} + V_{x+1,y}}{(\Delta x)^2} + \frac{V_{x,y-1} - 2V_{x,y} + V_{x,y+1}}{(\Delta y)^2} = 0$$

The first set of conditions analysed is: V = V0 at x=0 and V=0 at x=L. The bottom and top were not at fixed voltages but their first derivative was set to 0.

```
clear all;
close all;

W = 2; % width
L = 3; % length
V0=1; % voltage
dx=0.05; % mesh density
dy=0.05; % mesh density
nx=L/dx; % mesh points
ny=W/dy; % mesh points
```

The coefficients of the finite difference formulation can be calculated. Next the G matrix is built using the function mapCoords which converts x,y positions into locations in the G matrix.

```
VXY =-2*(1/dx^2 +1/dy^2); % Coefficient of middle term
```

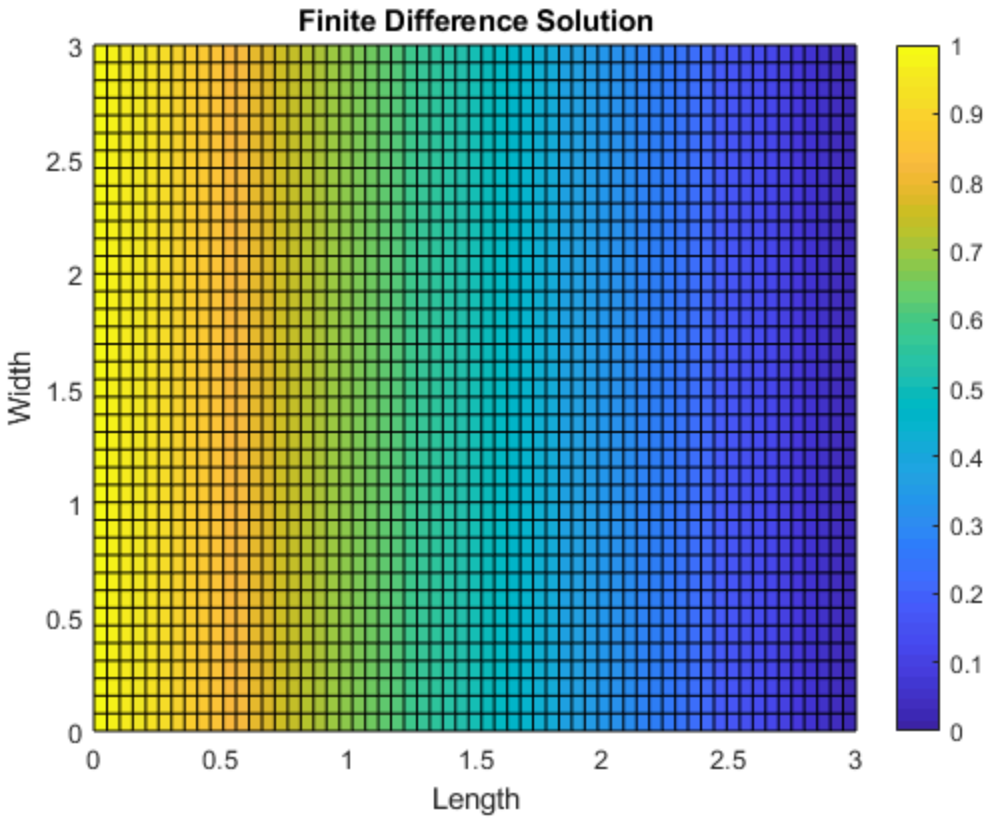
```
Vx = 1/dx^2; % coefficient of x term
Vy = 1/dy^2; % coefficient of y term
G=zeros(nx*ny,nx*ny);

for n = 2:(nx-1)
    for j=2:(ny-1)
         loc=mapCoords(n,j,nx);
         G(loc,loc)=VXY;
         G(loc,mapCoords(n-1,j,nx))=Vx;
         G(loc,mapCoords(n+1,j,nx))=Vx;
         G(loc,mapCoords(n,j-1,nx))=Vy;
         G(loc,mapCoords(n,j+1,nx))=Vy;
    end
end
```

Now we create the F part of the GV=F equation. This part enforces the boundary conditions.

```
F = zeros(nx*ny,1);
for j=1:ny
    loc=mapCoords(1,j,nx);
    G(loc,loc)=1;
    F(loc)=V0;
    loc=mapCoords(nx,j,nx);
    G(loc,loc)=1;
end

for n=2:(nx-1)
    loc=mapCoords(n,1,nx);
    G(loc,loc)=1;
    G(loc,mapCoords(n,2,nx))=-1;
    loc=mapCoords(n,ny,nx);
    G(loc,loc)=1;
    G(loc,mapCoords(n,ny-1,nx))=-1;
end
```

The solution is generated using matrix left division to solve for the 'V' solution. The solution is then reshaped into a 2-D iamge.

```
solution = G\F;
solution = reshape(solution,[],ny)'; % Turns a column vector into a
 matrix, the elements populate down the column until the n'th row and
 then next column starts

figure(1);
X=linspace(0,L,nx);
Y=linspace(0,L,ny);
[X1,Y1]=meshgrid(X,Y);
surf(X1,Y1,solution)
xlabel('Length');
ylabel('Width');
title('Finite Difference Solution');
view(0,90);
colorbar
```



This result seems to make sense according to our initial conditions set-up. One can see a linear reduction in voltage along the x - direction and a constant voltage across the y direction.

Part 1 Electrostatic Potential in a 2-D Rectangular Region

In part 1b the student computed the electrostatic potential of a rectangular region using two methods. First the FD method and second an analytical method.

The boundary conditions are: V = V0 at x=0, x=L and V=0 at y=0, y=W.

Definition of constants

```
clear all

w = 2;
l = 3;
V0=1;
dx=0.1;
dy=0.1;
nx=l/dx;
ny=w/dy;
```

This is identical to part 1a

```
VXY =-2*(1/dx^2 +1/dy^2);
```

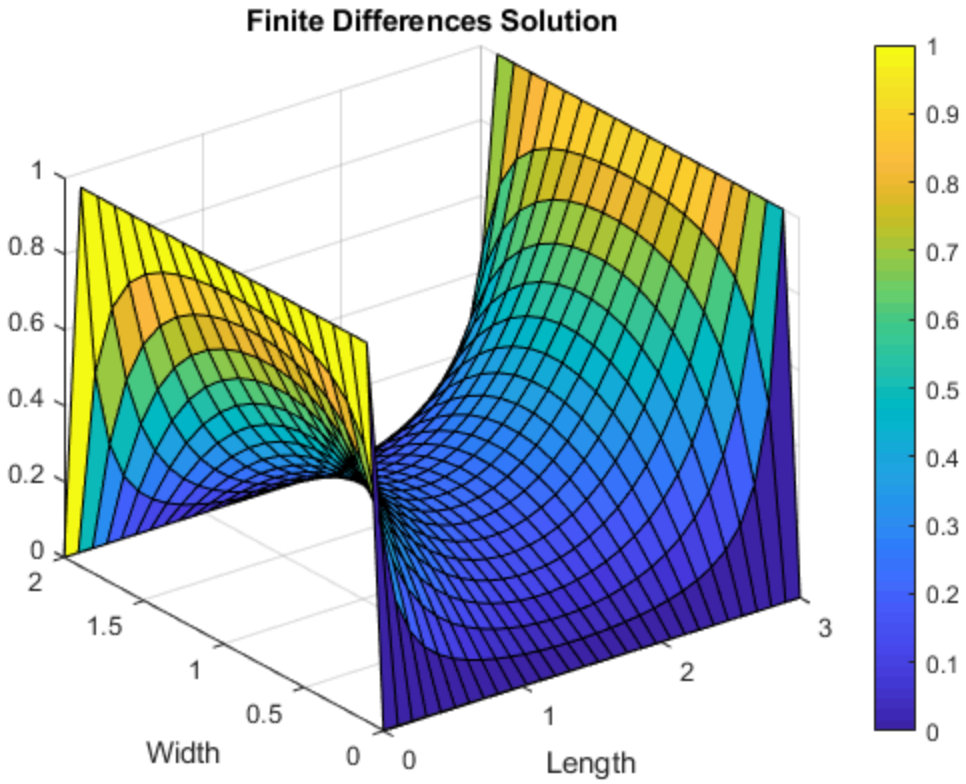
```
Vx = 1/dx^2;
Vy = 1/dy^2;

G1=zeros(nx*ny,nx*ny);

for n = 2:(nx-1)
    for j=2:(ny-1)
         loc=mapCoords(n,j,nx);
         G1(loc,loc)=VXY;
         G1(loc,mapCoords(n,j-1,nx))=Vy;
         G1(loc,mapCoords(n,j+1,nx))=Vy;
         G1(loc,mapCoords(n-1,j,nx))=Vx;
         G1(loc,mapCoords(n+1,j,nx))=Vx;
    end
end
```

In part 1b we have additional boundary conditions that must be accounted for. Process is similar to part 1a

```
F1=zeros(nx*ny,1);
for n=1:nx
    loc = mapCoords(n,1,nx);
    G1(loc,loc) = 1;
    loc = mapCoords(n,ny,nx);
    G1(loc,loc) = 1;
end

for j=1:ny
    loc = mapCoords(1,j,nx);
    G1(loc,loc) = 1;
    F1(loc) = V0;
    loc = mapCoords(nx,j,nx);
    G1(loc,loc) = 1;
    F1(loc) = V0;
end

F1(mapCoords(1,1,nx)) = 0;
F1(mapCoords(1,ny,nx)) = 0;
F1(mapCoords(nx,1,nx)) = 0;
F1(mapCoords(nx,ny,nx)) = 0;
```

Similar to part 1a

```
solution = G1\F1;
solution = reshape(solution,[],ny)';

figure(2);
X=linspace(0,l,nx);
Y=linspace(0,w,ny);
[X1,Y1]=meshgrid(X,Y);
surf(X1,Y1,solution)
xlabel('Length');
ylabel('Width');
title('Finite Differences Solution');
colorbar
```



This result makes sense according to our boundary conditions. The x-plane sides are at a voltage of 1 and the y-plane sides are voltage of 0. The electric potential decreases as you move towards the center but does not reach 0.

Analytical Solution

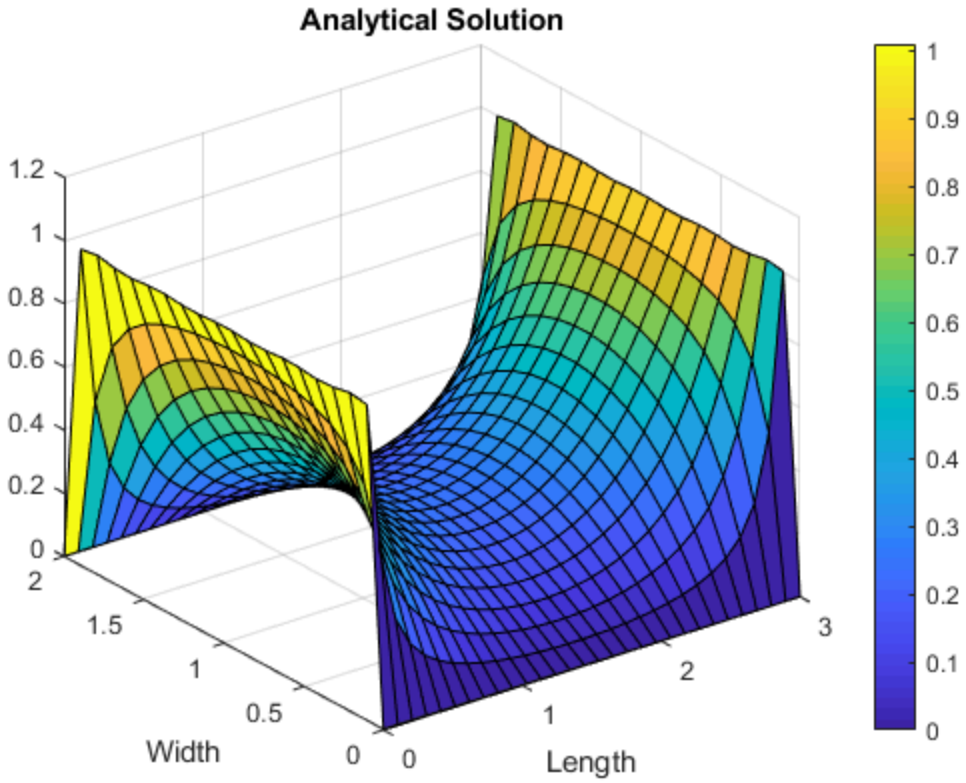
We will now compare our numerical FD solution to the analytical solution generated using the equation in the assignment 2 document provided to the students.

```
n = 100;
Asol=zeros(ny,nx);
X=repmat(linspace(-l/2,l/2,nx),ny,1);
Y=repmat(linspace(0,w,ny),nx,1)';
Difference =zeros(n,1);
```

The infinite series will be calculated and on each iteration a term will be added such tha the series will approach the FD solution. The difference between the FD and analytic solution will be recorded.

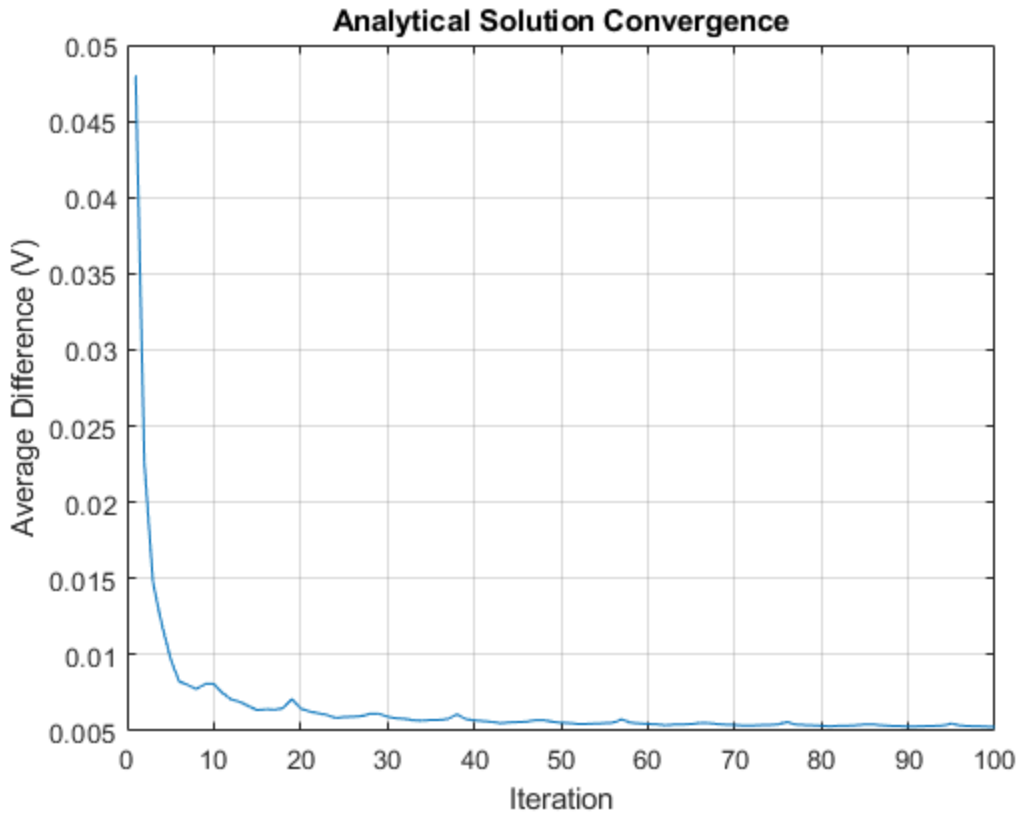
```
for t=1:n
    m=2*t-1;
   Asol=Asol + 1./m.*cosh(m.*pi.*X./w)./cosh(m.*pi.*(l./2)./
w).*sin(m.*pi.*Y./w);
   Difference(t) = mean(mean(abs(Asol.*4.*V0./pi -solution)));
end
Asol=Asol.*4.*V0./pi;
```

```
figure(3);
surf(linspace(0,l,nx),linspace(0,w,ny),Asol);
xlabel('Length');
ylabel('Width');
title('Analytical Solution')
colorbar
```



Upon visual inspection the two solutions seem similar, this is as expected.

```
figure(4);
plot(1:n,Difference);
xlabel('Iteration');
ylabel('Average Difference (V)');
title('Analytical Solution Convergence');
grid on;
```



Through measurement of the difference between the analytical and FD solution, we can see that they are not identical but very similar. As more terms are calculated in the infinite series equation, the similarity of the two solutions increases and the average differece will decrease.

Part 1 Summary

The finite difference method of modelling electrostatic potential is useful for 1D and 2D problems that require a uniform mesh and are in a steady state. FD is a simple and transparent formulation and is not overly complex to program and debug. Finite difference does suffer in more complex problems as the meshing struggles to do non rectangular geometry's and it is difficult to implement non uniform meshing in a program.

The analytical solution method provides insight on the dependance of variables in the formula but is generally only useful for simple problems. Complex problems should not be solved using analytical techniques.

It is useful to plot a movie of the convergence of the analytical and FD solution to determine when to stop analytical series. Past a certain threshold, the solutions begin to approach eachother and the increase of accuracy due to each computation becomes less. At this point the analytical series can be stopped.

Part 2 Finite Difference Method for Current Flow in a Rectangular Region

The finite difference method was used to solve the current flow, electric field and electro static potential in a non uniform conductivity system. The boundary conditions were set such that V=1 on the left side of the x - plane and V =0 on the right side of x -plane. The top and bottom were set to be free.

Since the conducitivity is not uniform the equation for the electrostatic potential becomes:

$$\frac{\partial}{\partial x}(\sigma_x(x,y)\frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(\sigma_y(x,y)\frac{\partial}{\partial y})$$

Much of the methodology was implemented accoridng to week 4's lecture slides, specifically slide 26 - 36.

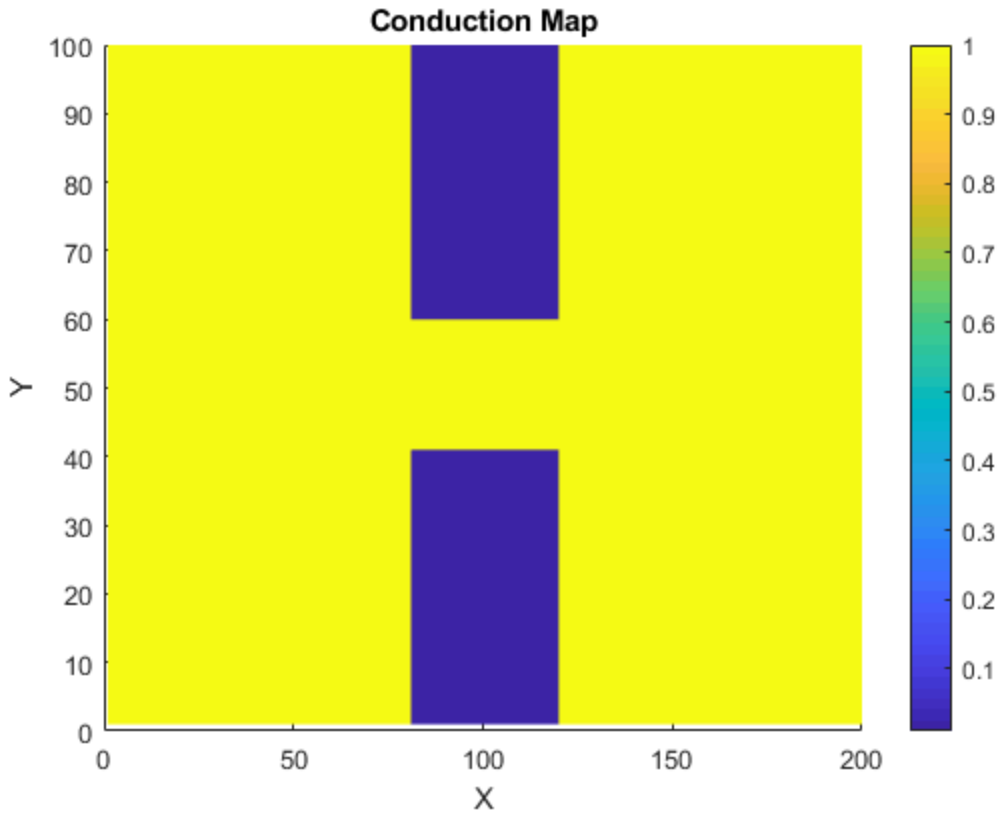
```
    clear all
    close all

    nx=200;
    ny=100;
    sigma=1e-2; % conductivity
    cMap=ones(ny,nx); % Initial conductivity map without low
 conductivity regions.
```

Creation of box co-ordinates.

```
    boxes(1,1)=nx/2;
    boxes(1,2)=0.1*nx;
    boxes(1,3)=0.2*nx +1;
    boxes(1,4)=0.2*nx;
    boxes(2,1)=nx/2;
    boxes(2,2)=0.4*nx;
    boxes(2,3)=0.2*nx +1;
    boxes(2,4)=0.2*nx;
```

This for loop adjusts the conductivity map and places the lower conductivity boxes in it.

```
    for k=1:length(boxes(:,1))
        cMap=boxMaker(cMap,boxes(k,:),sigma,nx,ny);
    end

    figure(1)
    surf(cMap,'edgecolor','none')
    colorbar
    title('Conduction Map')
    xlabel('X')
    ylabel('Y')
    view(0,90)
```



This is the conductivity map of the region. Everywhere is a conductivity of 1 except inside the boxes where it is 1E-2.

This next section is mostly adapted from the slides in week 4. Pg. 35 The treatement of resistors is similar to the treatment of different conductivity's and only the orientation of geometry needed to be adjusted for the assignment situation.

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

for i = 1:ny
        for j = 1:nx
            n = i + (j - 1) * ny;
            if j==1
                G(n,:) = 0;
                G(n,n)=1;
                B(n)=1;
            elseif j==nx
                G(n,:)=0;
                G(n,n)=1;
            elseif i==1
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nyp = i + 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
```

```
                ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                G(n, n) = -(rxm+rxp+ryp);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nyp) = ryp;
            elseif i==ny
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nym = i - 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;

                G(n, n) = -(rxm + rxp + rym);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nym) = rym;
            else
                nxm = i + (j-2)*ny;
                nxp = i + (j)*ny;
                nym = i-1 + (j-1)*ny;
                nyp = i+1 + (j-1)*ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;
                ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                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
```

The solution is calculated using matrix left division. Then it is mapped from a 1-D column into a 2-D image.

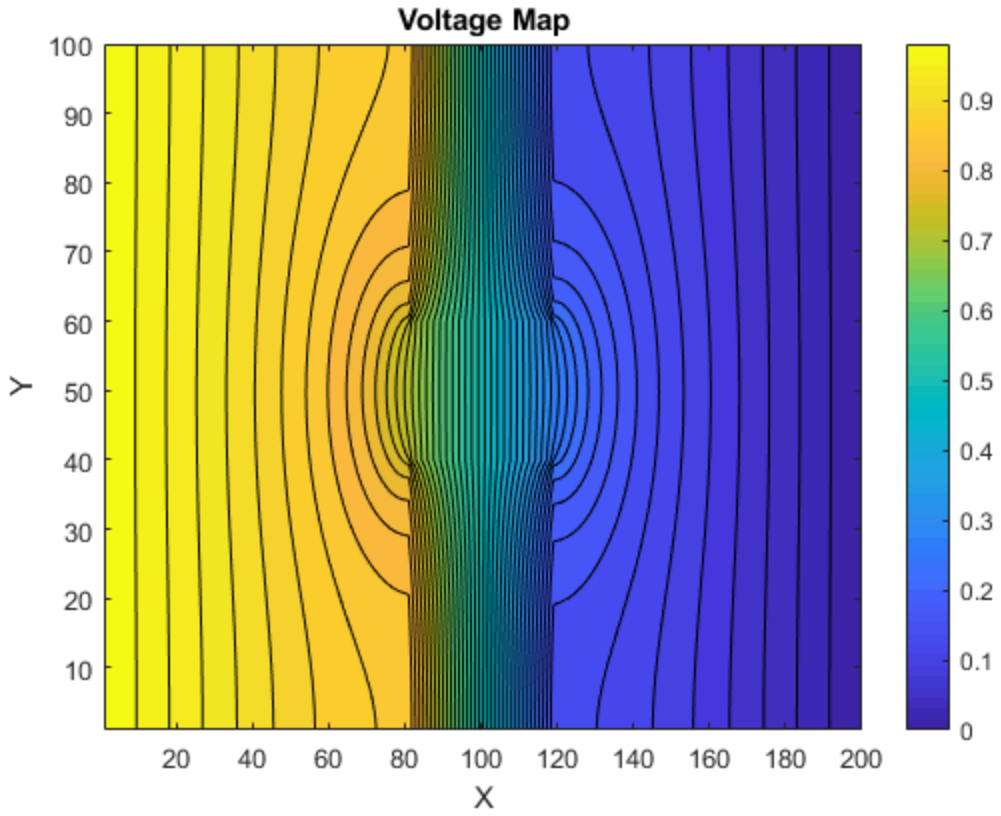
```
V = G\B';
Vmap = zeros(ny,nx);
for i = 1:ny
    for j = 1:nx
        n = i + (j - 1) * ny;
        Vmap(i, j) = V(n);
    end
end
```

Instead of using the process in the slides to calculate the gradient of the electrostatic potential, the Matlab function 'gradient' is used. Then the current flow can be calculated by multiplying the electric field by the electrostatic potential.

```
[Ex,Ey]=gradient(Vmap); % Calculates the gradient in one line
```

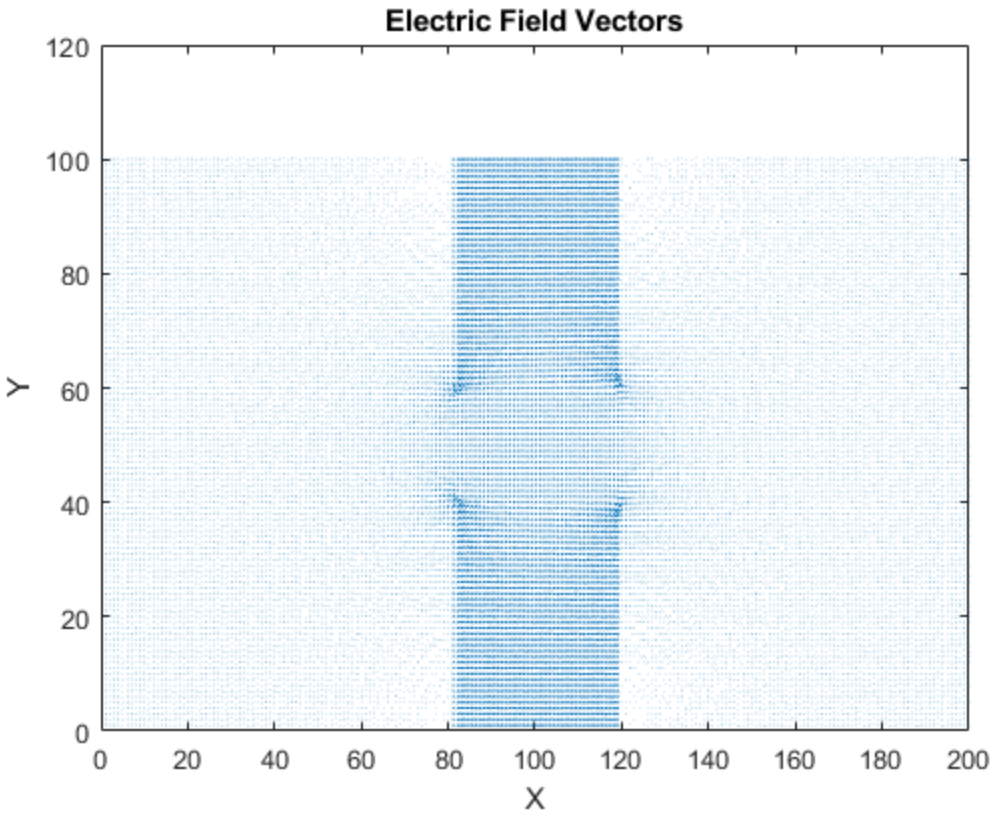
```
Ex=-Ex; % electric field
Ey=-Ey;% electric field
Jx=Ex.*cMap; % current flow
Jy=Ey.*cMap; % current flow
```

```
figure(2)
contourf(Vmap,50)
colorbar
title('Voltage Map')
xlabel('X')
ylabel('Y')
```



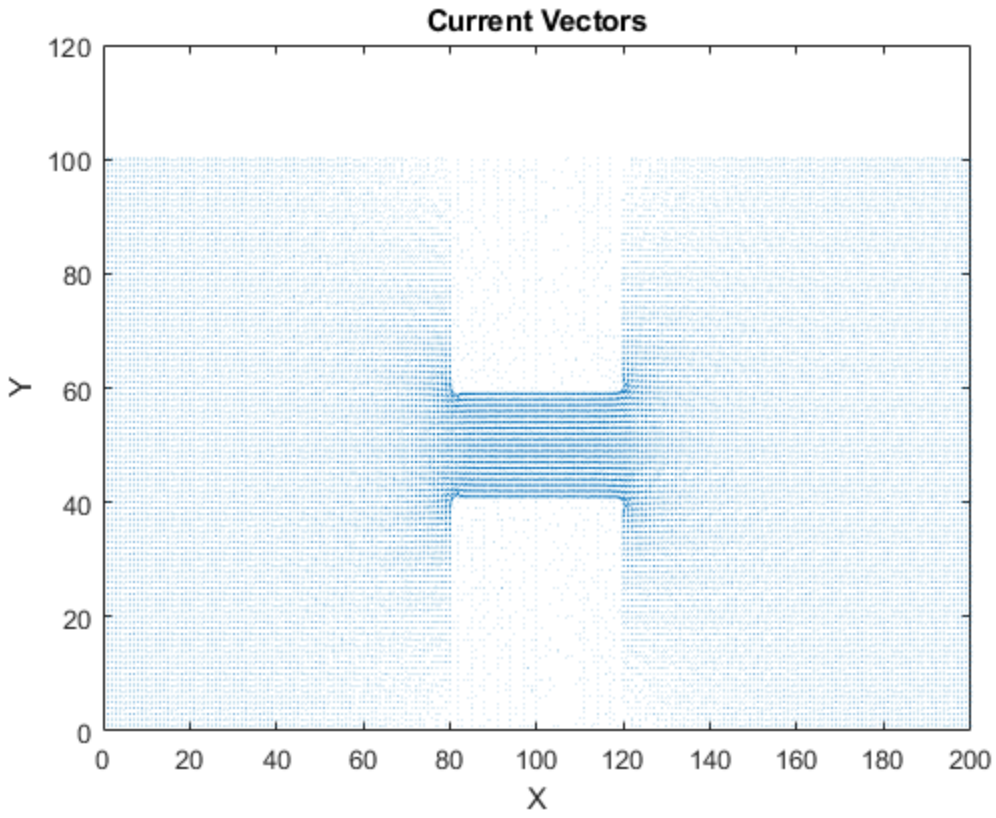
This figure illustrates the electrostatic potential in the system. The countours represent the equipotential lines where the voltage is equal.

```
figure(3)
quiver(Ex,Ey)
ylim = ([0 100]);
title('Electric Field Vectors')
xlabel('X')
ylabel('Y')
view(0,90)
```



This figure illustrates the electric field vectors. The field is strongest in the regions of lower conducvity. This has similarities to a parallel plate capcitor, as there are two high conduvity regions seperated by a lower conducutivity region.

```
figure(4)
quiver(Jx,Jy)
title('Current Vectors')
xlabel('X')
ylabel('Y')
view(0,90)
```



This figure illustrates the current flow in the system. There is negligible current flow inside of the boxes but the region between two boxes has a large flow of current. This makes sense since the 'bottleneck' must move curren through a smaller space.

```
Jl=sum(Jx(1:ny,1));
Jr=sum(Jx(1:ny,nx));
```

The current flow in the system is:

```
fprintf('The left current is %d A \n',Jl)
fprintf('The right curren is %d A \n',Jr)
```

```
The left current is 2.341614e-01 A
The right curren is 2.341614e-01 A
```

Investigating the mesh density.

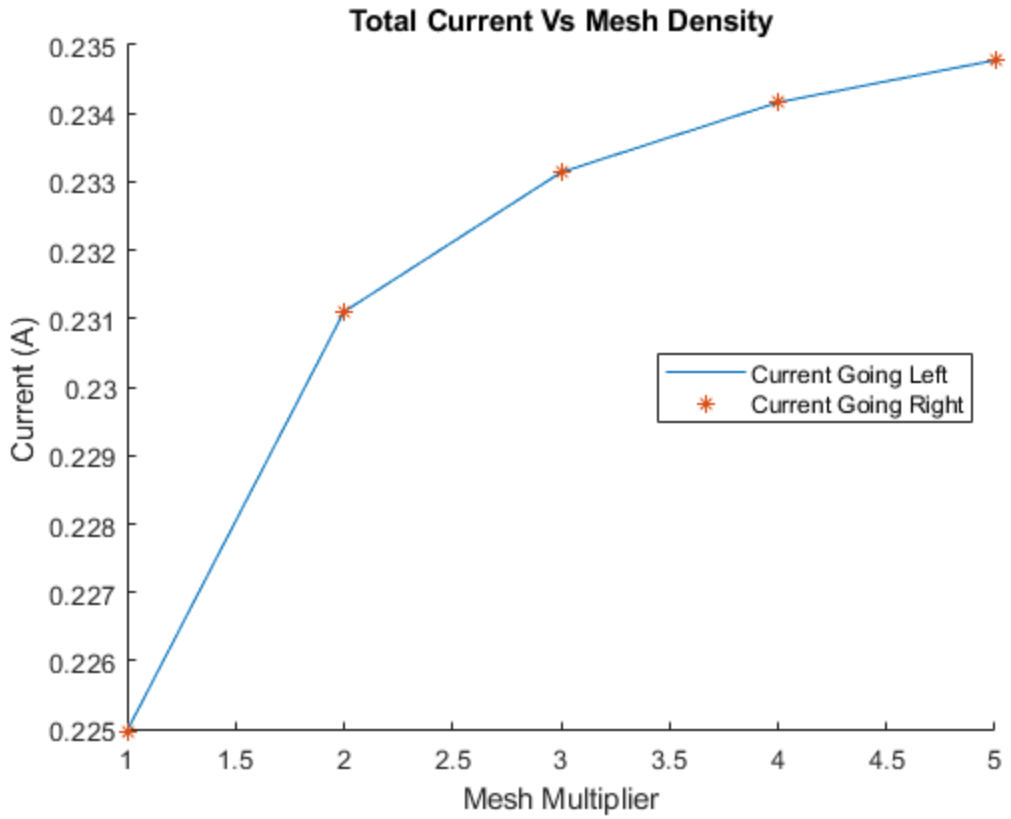
```
clear all
close all
```

I will iterate through 5 different mesh densities, increasing the density each time.

```
for u=1:5
        sigma=1e-2;
        nx=50*u;
```

```
    ny=25*u;
    cMap=ones(ny,nx);

    boxes(1,1)=nx/2;
    boxes(1,2)=0.1*nx;
    boxes(1,3)=0.2*nx +1;
    boxes(1,4)=0.2*nx;
    boxes(2,1)=nx/2;
    boxes(2,2)=0.4*nx;
    boxes(2,3)=0.2*nx +1;
    boxes(2,4)=0.2*nx;

    for k=1:length(boxes(:,1))
        cMap=boxMaker(cMap,boxes(k,:),sigma,nx,ny);
    end
G = sparse(nx*ny);
B = zeros(1,nx*ny);

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

            if j==1
                G(n,:) = 0;
                G(n,n)=1;
                B(n)=1;
            elseif j==nx
                G(n,:)=0;
                G(n,n)=1;
            elseif i==1
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nyp = i + 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                G(n, n) = -(rxm+rxp+ryp);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nyp) = ryp;
            elseif i==ny
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nym = i - 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;

                G(n, n) = -(rxm + rxp + rym);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
```

```
                    G(n, nym) = rym;
                    else
                    nxm = i + (j-2)*ny;
                    nxp = i + (j)*ny;
                    nym = i-1 + (j-1)*ny;
                    nyp = i+1 + (j-1)*ny;

                    rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                    rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                    rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;
                    ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                    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\B';
        Vmap = zeros(ny,nx);
        for i = 1:ny
            for j = 1:nx
                n = i + (j - 1) * ny;
                Vmap(i, j) = V(n);
            end
        end
        [Ex,Ey]=gradient(Vmap);
        Jx=abs(Ex.*cMap);
        Jy=abs(Ey.*cMap);
        J_l(u)=sum(Jx(1:ny,1)); % Sum of the current in the system.
        J_r(u)=sum(Jx(1:ny,nx));% Sum of the current in the system.
end

figure(5)
hold on
plot(J_l,'-')
plot(J_r,'*')
title('Total Current Vs Mesh Density')
legend('Current Going Left','Current Going Right')
legend('Location','east')
xlabel('Mesh Multiplier')
ylabel('Current (A)')
hold off
```



This graph illustrates the total current change for differing mesh densities. As the mesh density increases the total current of the system increases slightly but looks like it eventually plateaus. This makes sense because if the mesh density is too coarse it cannot capture all of the system effects and will miss some system current. As it becomes dense enough it will properly capture the total current which should be a constant value.

Investigating the conductivity.

The conductivity will steadily be increased over time.

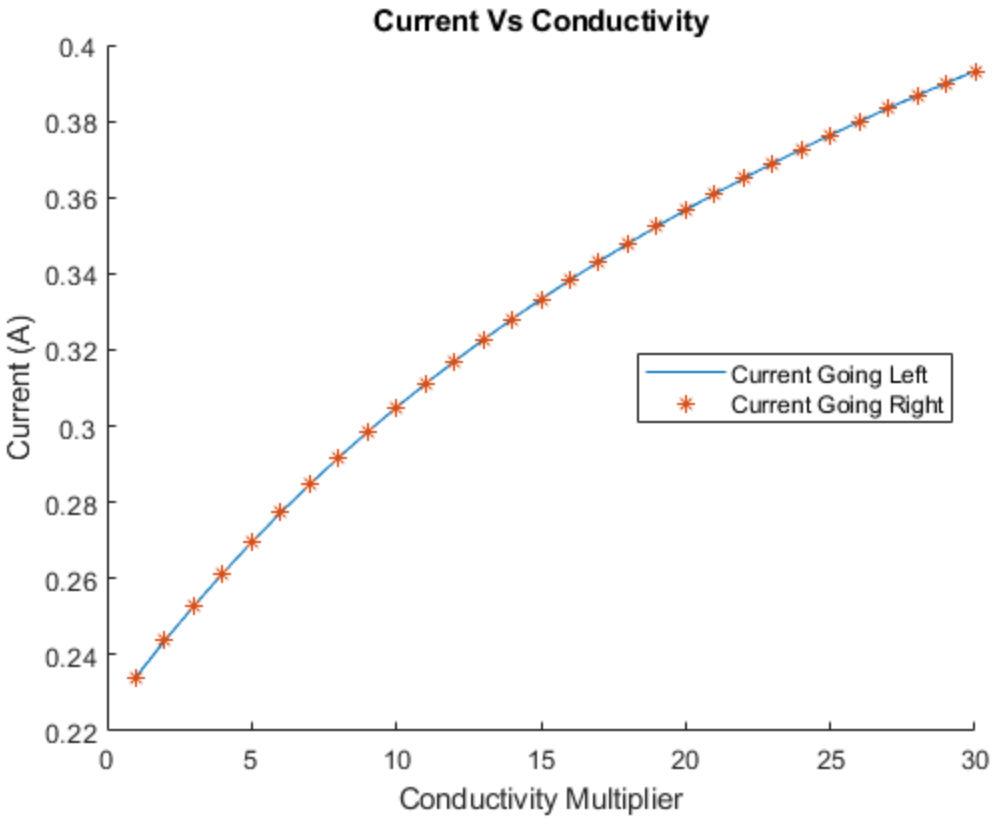
```
clear all
close all
```

I will iterate through different conductivities increasing it each time.

```
for u=1:30
        sigma=u*1e-2;
        nx=200;
        ny=100;
        cMap=ones(ny,nx);

         boxes(1,1)=nx/2;
        boxes(1,2)=0.1*nx;
        boxes(1,3)=0.2*nx +1;
        boxes(1,4)=0.2*nx;
        boxes(2,1)=nx/2;
        boxes(2,2)=0.4*nx;
```

```
    boxes(2,3)=0.2*nx +1;
    boxes(2,4)=0.2*nx;

    for k=1:length(boxes(:,1))
        cMap=boxMaker(cMap,boxes(k,:),sigma,nx,ny);
    end
G = sparse(nx*ny);
B = zeros(1,nx*ny);

for i = 1:ny
        for j = 1:nx
            n = i + (j - 1) * ny;
            if j==1
                G(n,:) = 0;
                G(n,n)=1;
                B(n)=1;
            elseif j==nx
                G(n,:)=0;
                G(n,n)=1;
            elseif i==1
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nyp = i + 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                G(n, n) = -(rxm+rxp+ryp);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nyp) = ryp;
            elseif i==ny
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nym = i - 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;

                G(n, n) = -(rxm + rxp + rym);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nym) = rym;
                else
                nxm = i + (j-2)*ny;
                nxp = i + (j)*ny;
                nym = i-1 + (j-1)*ny;
                nyp = i+1 + (j-1)*ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;
```

```
                    ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                    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\B';
        Vmap = zeros(ny,nx);
        for i = 1:ny
            for j = 1:nx
                n = i + (j - 1) * ny;
                Vmap(i, j) = V(n);
            end
        end
        [Ex,Ey]=gradient(Vmap); % Calculates the gradient in one
 simple function

        Jx=abs(Ex.*cMap);
        Jy=abs(Ey.*cMap);
        J_l(u)=sum(Jx(1:ny,1));
        J_r(u)=sum(Jx(1:ny,nx));
end

figure(6)
hold on
plot(J_l,'-')
plot(J_r,'*')
title('Current Vs Conductivity')
legend('Current Going Left','Current Going Right')
legend('Location','east')
xlabel('Conductivity Multiplier')
ylabel('Current (A)')
hold off
```



This figure illustrates the total system current change with varying conductivities of the boxes. As the conductivity of the boxes increases, there is more current flow in the system. This makes sense because there is less resistance in a system with higher conductivity and the same geometry. The total current is proportional to the conductivity.

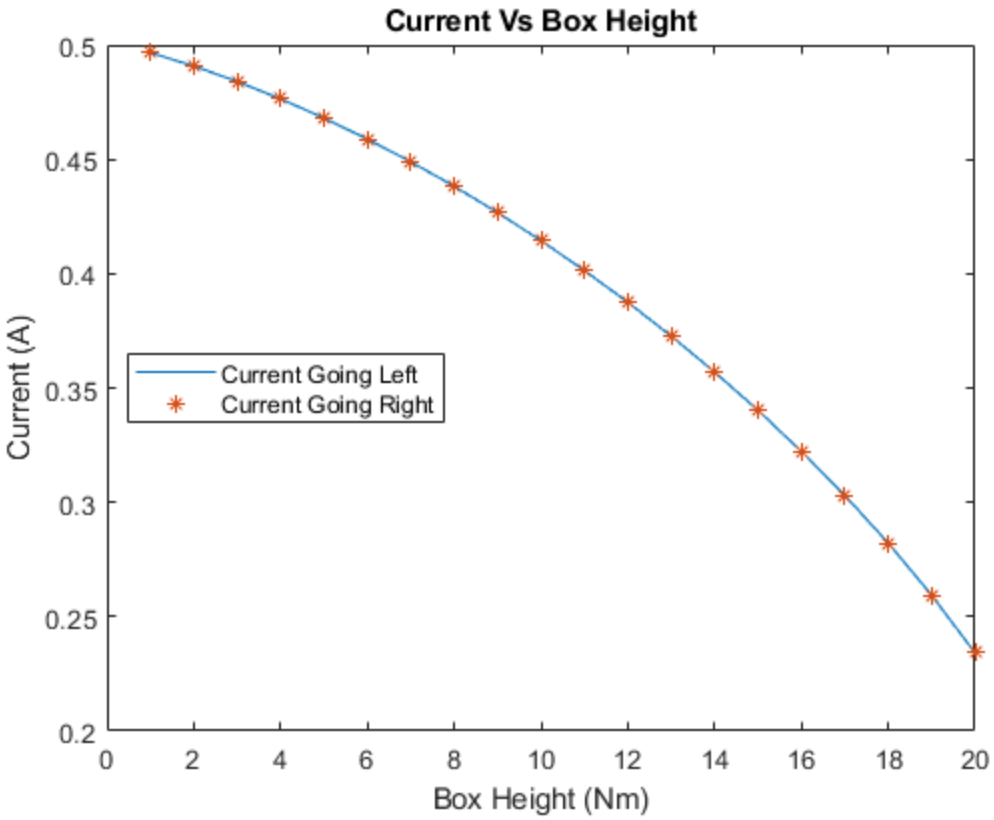
Investigating Narrowing the Bottle Neck

The bottle beck will start off minimal and increase with each iteration. The height of the boxes increases each time.

```
clear all
close all

for u=2:2:40
        sigma=1e-2;
        nx=200;
        ny=100;
        cMap=ones(ny,nx);


        boxes(1,1)=nx/2;
        boxes(1,2)=round(u/2);
        boxes(1,3)=u+1;
        boxes(1,4)=0.2*nx;

        boxes(2,1)=nx/2;
```

```
    boxes(2,2)=100-round(u/2);
    boxes(2,3)=u+1;
    boxes(2,4)=0.2*nx;

    for k=1:length(boxes(:,1))
        cMap=boxMaker(cMap,boxes(k,:),sigma,nx,ny);
    end

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

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

            if j==1
                G(n,:) = 0;
                G(n,n)=1;
                B(n)=1;
            elseif j==nx
                G(n,:)=0;
                G(n,n)=1;
            elseif i==1
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nyp = i + 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                G(n, n) = -(rxm+rxp+ryp);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nyp) = ryp;
            elseif i==ny
                nxm = i + (j - 2) * ny;
                nxp = i + (j) * ny;
                nym = i - 1 + (j - 1) * ny;

                rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;

                G(n, n) = -(rxm + rxp + rym);
                G(n, nxm) = rxm;
                G(n, nxp) = rxp;
                G(n, nym) = rym;
                else
                nxm = i + (j-2)*ny;
                nxp = i + (j)*ny;
                nym = i-1 + (j-1)*ny;
                nyp = i+1 + (j-1)*ny;
```

```
                    rxm = (cMap(i, j) + cMap(i , j-1)) / 2.0;
                    rxp = (cMap(i, j) + cMap(i , j+1)) / 2.0;
                    rym = (cMap(i, j) + cMap(i-1, j )) / 2.0;
                    ryp = (cMap(i, j) + cMap(i+1, j)) / 2.0;

                    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\B';
            Vmap = zeros(ny,nx);
            for i = 1:ny
                for j = 1:nx
                    n = i + (j - 1) * ny;

                    Vmap(i, j) = V(n);
                end
            end

            [Ex,Ey]=gradient(Vmap);

            Jx=abs(Ex.*cMap);
            Jy=abs(Ey.*cMap);
            J_l(round((u-1)/2))=sum(Jx(1:ny,1));
            J_r(round((u-1)/2))=sum(Jx(1:ny,nx));

    end
    figure(7)
    plot(J_l,'-')
    hold on
    plot(J_r,'*')

    title('Current Vs Box Height')
    legend('Current Going Left','Current Going Right')
    legend('Location','west')
    xlabel('Box Height (Nm)')
    ylabel('Current (A)')
    hold off
```



This figure illustrates the change in total system current relating to the size of the bottleneck. The bottleneck starts off very small, meaning there is almost no bottle neck. As we increase the bottle neck the total current in the system begins to decrease. This makes sense because as the bottle neck worsens the conducitivity of the system decreases.

Published with MATLAB® R2018b