Megan McEwen
Student # 101003509
23 February 2020

Assignment 2 Report

1.0 Introduction

The purpose of this report is to use Matlab to explore how Laplace's equation can be used to solve electrostatic potential and current flow problems using the Finite Difference method. Part 1 considers electrostatic potential problems using a homogeneous resistor network with constant conductivity, placed in a rectangular region of size L*W. Part 2 considers current flow problems using the same region, but with contacts inserted to create a "bottleneck" with different conductivity. Both parts use the Finite Difference method to solve the Laplace equation, using the matrix form of the problem GV = F.

2.0 Part 1: Finite Difference Method Analysis in Simple Box

This section uses Laplace to solve $\nabla^2 V = 0$, using the matrix form of GV = F. There are two cases examined: a 1D simple case and a 2D case. The 2D case is additionally compared to an analytical series solution, to see how the two different approaches compare. Both cases are solved for a simple box with dimensions L*W, where L and W are set with a 3:2 ratio. Initially, for Part 1, these are set with L = 30 and W = 20.

2.1 1D Case

The first case is a simple case with V = V0 at x = 0 and V = 0 at x = L. This is treated as a 1D case with the top and bottom boundary conditions also set at 0.
To create the initial G matrix, a sparse matrix is created as shown:

```
%create initial matrices
G = sparse(L*W,L*W);
F = zeros(L*W,1);
```

Next, each coordinate inside the box is set up using a for loop. Within the loop, a mapping equation is used to set up the n value, with four specific local mapping equations for the points surrounding n:

```
    %mapping equation
    n = y + (x-1) * W;

    %local mapping
    nxm = y+(x-2)*W;
    nxp = y+(x)*W;
    nym = (y-1)+(x-1)*W;
    nyp = (y+1)+(x-1)*W;
```

Each boundary condition is then checked. If the point is on the x boundaries, the F matrix is set up and the G values are set to 0 or 1. If the point is on the y boundary, the G(n,n) point is set to -3 with surrounding points set to 1. Finally, if the point is not on any boundary, the G(n,n) value is set to -4 with surrounding points set to 1.

```
if(x ==1) %left x boundary
            G(n,:) = 0;
            G(n,n) = 1;
            F(n) = V0;
        elseif (x==L) %right x boundary
            G(n,:) = 0;
            G(n,n) = 1;
            F(n) = 0;
        elseif (y ==1) %bottom y boundary
           G(n,:) = 0;
           G(n,nxm) = 1;
           G(n,n) = -3;
           G(n,nxp) = 1;
           G(n,nyp) = 1;
        elseif(y ==W) %top y boundary
           G(n,:) = 0;
           G(n,nxm) = 1;
           G(n,n) = -3;
           G(n,nxp) = 1;
           G(n,nyp) = 1;
        else %all interior points
           G(n,:)= 0;
           G(n,nxm) = 1;
           G(n,n) = -4;
           G(n,nxp) = 1;
           G(n,nym) = 1;
           G(n,nyp) = 1;
        end
```

Next, the GV = F formula is rearranged to solve for V. The mapping equation is then used again to populate the V matrix solution.

```
V = G\F; %use G matrix to solve for V

%populate V matrix solution
for (x = 1:L)
    for (y = 1:W)
        n = y + (x-1) * W;
        VMatrix(x,y) = V(n);
    end
end
```

Finally, the surf() function is used to plot the final solution. The final solution for the first case is shown below:

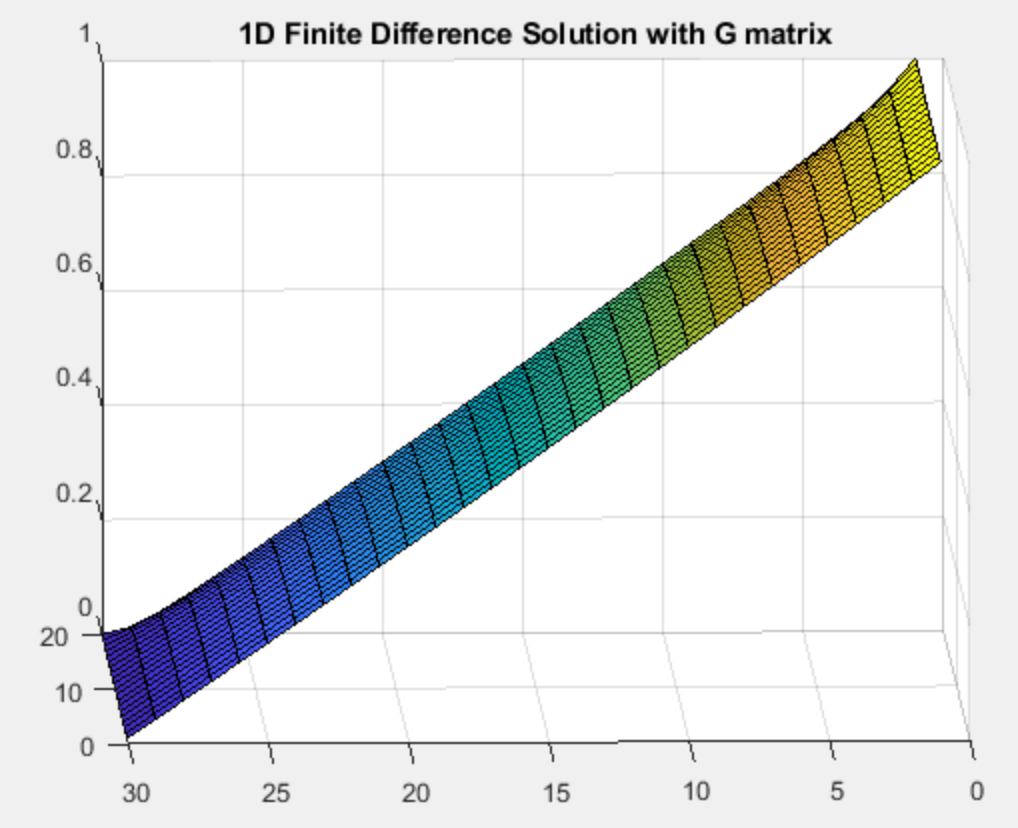

Figure 1: 1D Solution with x = 0, x = V0, y = 0.

2.2 2D Case with Analytical Solution

This case uses a very similar setup to the first case, but uses a different setup for the G matrix and different boundary conditions, with V = V0 at x = 0 and x = L, and V = 0 at y = 0 and y = W. The code for the 2D case G matrix setup can be seen below:

```
if(x ==1)
    G(n,n) = 1;
    F(n) = 1;
elseif (x==L)
    G(n,n) = 1;
    F(n) = 1;
elseif (y ==1)
    G(n,n) = 1;
elseif(y ==W)
    G(n,n) = 1;
else
   G(n,nxm) = 1;
   G(n,n) = -4;
   G(n,nxp) = 1;
   G(n,nym) = 1;
   G(n,nyp) = 1;
end
```

The solution for this case, plotted with surf(), can be seen in Figure 2:

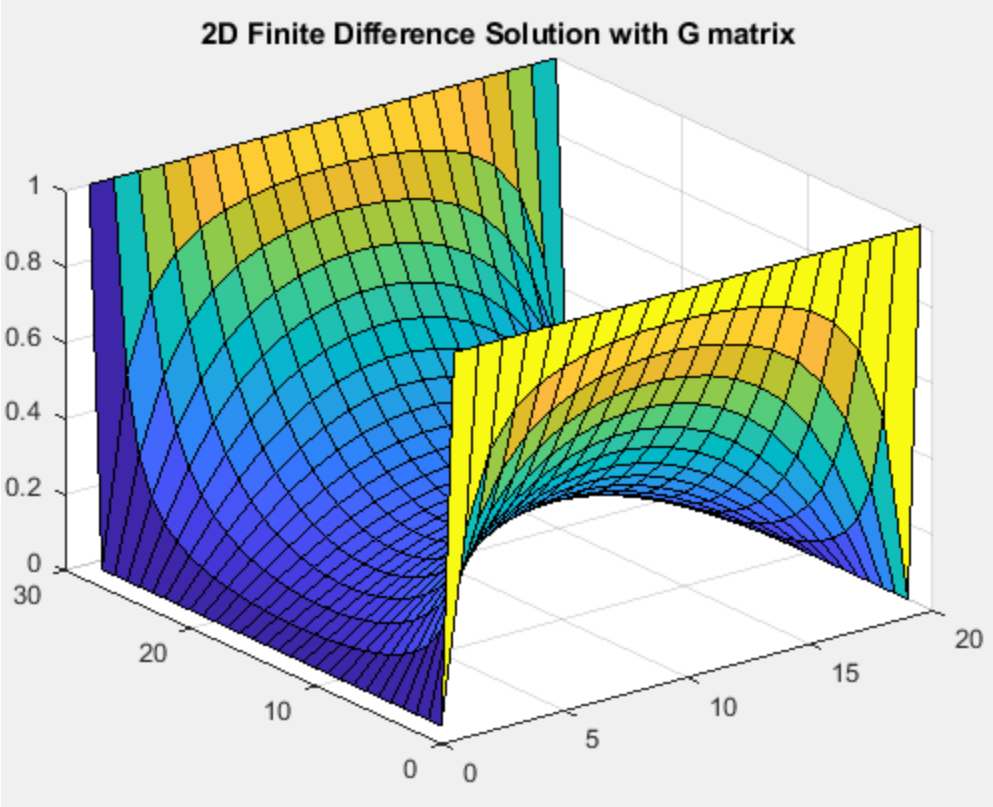

Figure 2: 2D Solution with G matrix

The same scenario is then solved again, this time using the analytical series shown:

$$V(x,y) = \frac{4V_0}{\pi} \sum_{n=1,3,5\ldots}^{\infty} \frac{1}{n} \frac{\cosh\left(\frac{n\pi x}{a}\right)}{\cosh\left(\frac{n\pi b}{a}\right)} \sin\left(\frac{n\pi y}{a}\right)$$

The code to set up the analytical solution is shown below:

```
%analytical solution
a = L;
b = W/2;
x_a = linspace(-b,b,L);
y_a = linspace(0,a,W);

[X,Y] = meshgrid(x_a,y_a);
sum = zeros(size(X));

        for (n = 1:2:iter)
            %use analytical equation to find sum
            sum = sum +
((1/n)*((cosh((n*pi*X)/a))./(cosh((n*pi*b)/a))).*(sin((n*pi*Y)/a)));

            surf(sum); %watch analytical solution converge
        end
       V_A = ((4*V0)/pi)*sum;
```

The solution can be run for any number of iterations; in this example, it runs for 100 iterations before converging. The final converged solution can be seen in Figure 3:

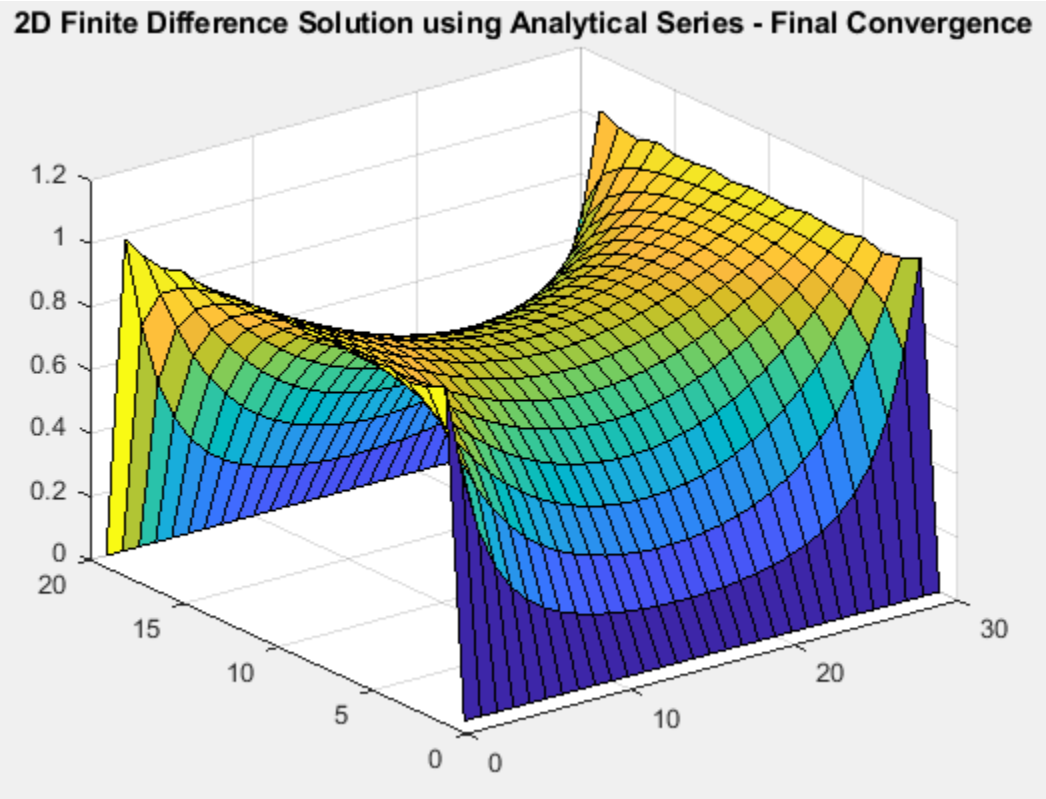

Figure 3: Analytical Solution to 2D Scenario

Comparing the two results, the analytical and numerical solutions have the same general shape. As more iterations are completed, the analytical solution more closely resembles the numerical solution. If the numerical solution is given a tighter meshing, it becomes more similar to the analytical solution shape. The analytical series solution can continue indefinitely, but converges close to the numerical solution in ~100 iterations. The numerical solution gives a clear solution with tighter mesh spacing, but requires more time and processing to converge than the analytical solution. However, if the analytical solution is not given enough iterations, its solution can appear incorrect.

3.0 Part 2: Finite Difference Method Analysis with Bottleneck

This section focuses on solving for current flow in the box using $\nabla (\sigma_{x,y} \nabla V) = 0$. The box contains two distinct regions: a large region with conductivity = 1, and a bottleneck created with contacts that have a lower conductivity = $1e^{-2}$. The contacts are shown in the diagram in Figure 4:

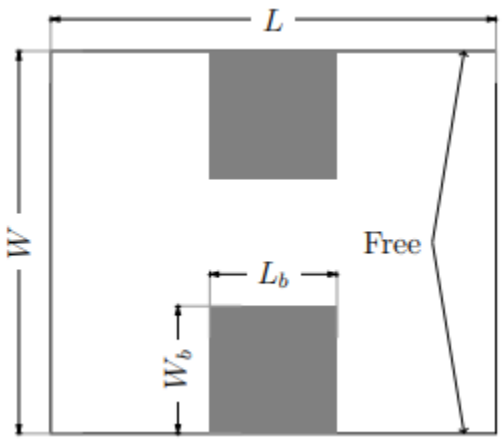

Figure 4: Region for Part 2

3.1 Current Flow with Contacts in Box

To set up the problem in Matlab, the contact regions are defined and the conductivity is set for each region:

```
contactLeft = (L/2) - (Lb/2);
contactRight = (L/2) + (Lb/2);
contactBottom = Wb;
contactTop = W - Wb;

%set up sigma for contacts
sigma(:,:) = sigmaOutside;
sigma(contactTop:W, contactLeft:contactRight) = sigmaInside;
sigma(1:contactBottom, contactLeft:contactRight) = sigmaInside;
```

The sigma plot showing the contacts is seen in Figure 5:

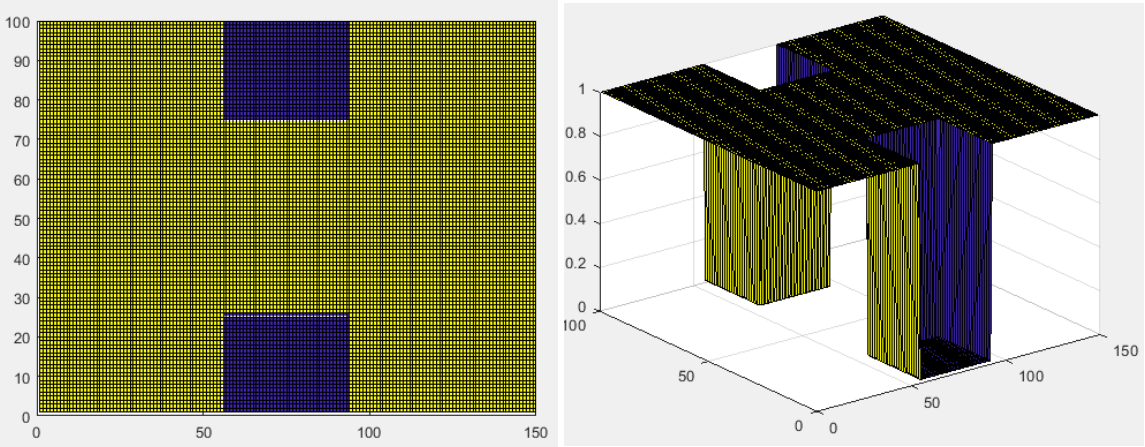

Figure 5: Contacts with Low Conductivity

Next, the G matrix is set up as in Part 1. To accommodate the effects of the contacts, the sigma values are included in the calculation for each n point using the code below:

```
%G matrix
for x=1:L
    for y=1:W

        %mapping equation
        n = y +(x-1)*W;

        if(x==1)

            G(n,n) = 1;
            F(n)= 1;

        elseif(x==L)

            G(n,n) = 1;
            F(n)=0;

        elseif(y == 1)

            %local mapping
            nyp = y+1+(x-1)*W;
            nxp = y+(x)*W;
            nxm = y+(x-2)*W;

            sig_yp =(sigma(y,x)+sigma(y+1,x))/2;
            sig_xp=(sigma(y,x)+ sigma(y,x+1))/2;
            sig_xm =(sigma(y,x)+ sigma(y,x-1))/2;

            G(n,n)= -(sig_yp+sig_xp+sig_xm);
            G(n,nyp)= sig_yp;
            G(n,nxp)=sig_xp;
            G(n,nxm)= sig_xm;

        elseif(y==W)
            %local mapping
            nxp = y+(x)*W;
            nxm = y+(x-2)*W;
            nym = y-1+(x-1)*W;

            sig_xp=(sigma(y,x)+ sigma(y,x+1))/2;
            sig_xm =(sigma(y,x)+ sigma(y,x-1))/2;
            sig_ym =(sigma(y,x)+ sigma(y-1,x))/2;

            G(n,n)=-(sig_ym+sig_xp+sig_xm);
            G(n,nym)=sig_ym;
            G(n,nxp)=sig_xp;
            G(n,nxm)=sig_xm;

        else
            %local mapping
            nyp = y+1+(x-1)*W;
            nxp = y+(x)*W;
            nxm = y+(x-2)*W;
            nym = y-1+(x-1)*W;
```

```
            sig_yp =(sigma(y,x)+sigma(y+1,x))/2;
            sig_xp=(sigma(y,x)+ sigma(y,x+1))/2;
            sig_xm =(sigma(y,x)+ sigma(y,x-1))/2;
            sig_ym =(sigma(y,x)+ sigma(y-1,x))/2;

            G(n,n)=-(sig_yp+sig_ym+sig_xp+sig_xm);
            G(n,nyp)= sig_yp;
            G(n,nym)= sig_ym;
            G(n,nxp)= sig_xp;
            G(n,nxm)= sig_xm;
        end
    end
end
```

Next, the V matrix is solved for and the mapping equation is used to generate the final V solution:

```
V = G\F';

VMatrix = zeros(W,L);

%populate V matrix solution
for (x = 1:L)
    for (y = 1:W)
        n = y + (x-1)*W;
        VMatrix(y,x) = V(n);
    end
end
```

Finally, the electric and current density plots are generated:

```
%solve for E
[Ex, Ey] = gradient(-VMatrix);
E_magnitude = sqrt(Ex.^2 + Ey.^2);

%solve for J
Jx = sigma.*Ex;
Jy = sigma.*Ey;
```

The final plots showing V(x,y), E(x,y), and J(x,y) are shown in the figures below:

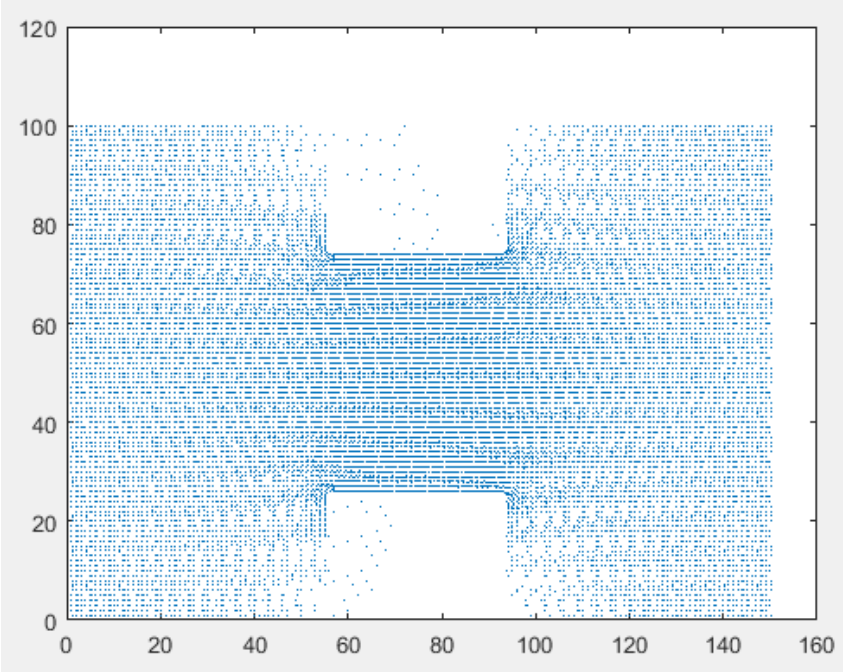

Figure 6: J(x,y) Plot

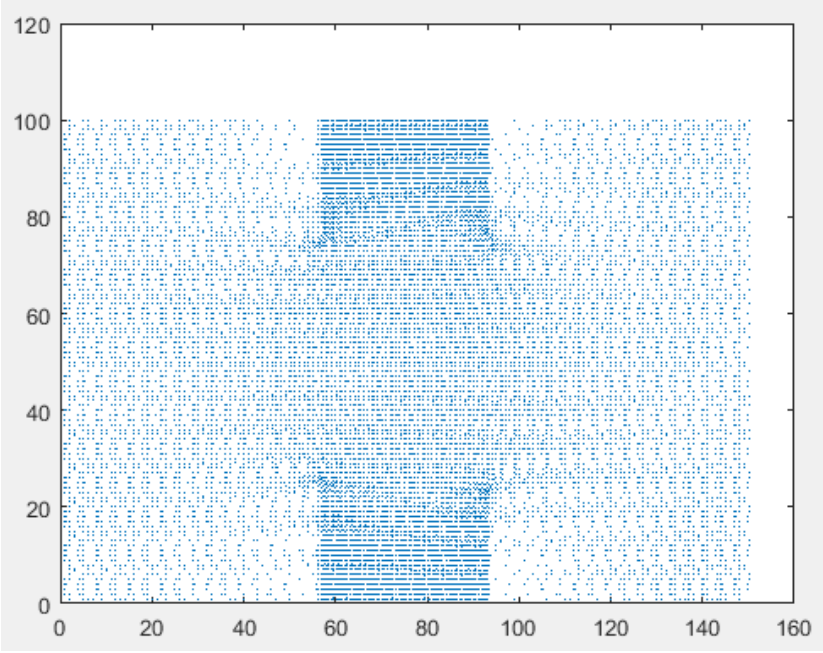

Figure 7: E(x,y) Plot

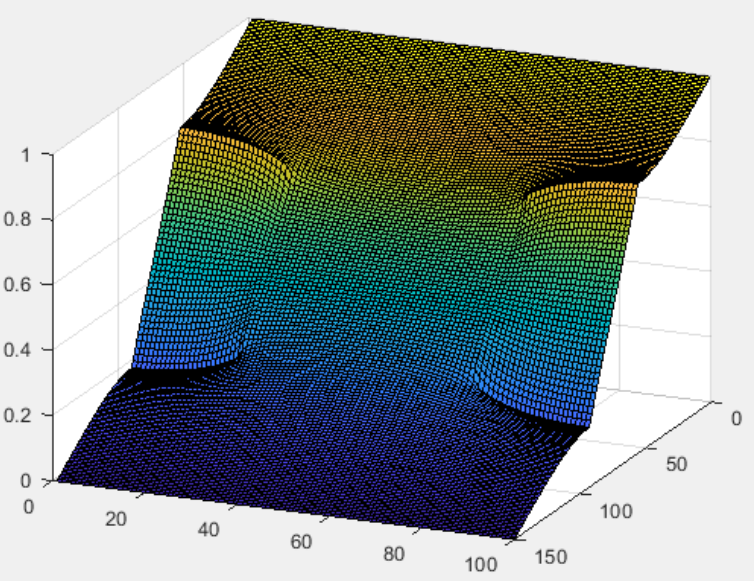

Figure 8: V(x,y) Plot

3.2 Effect of Varying Mesh Density on Current

The same code is used with a modification that loops through repeatedly, slowly increasing the mesh size of the simulation. As the mesh increases, the current density of the box also generally increases, as shown in Figure 9. Note that the bottleneck and conductivity both remain constant in this simulation:

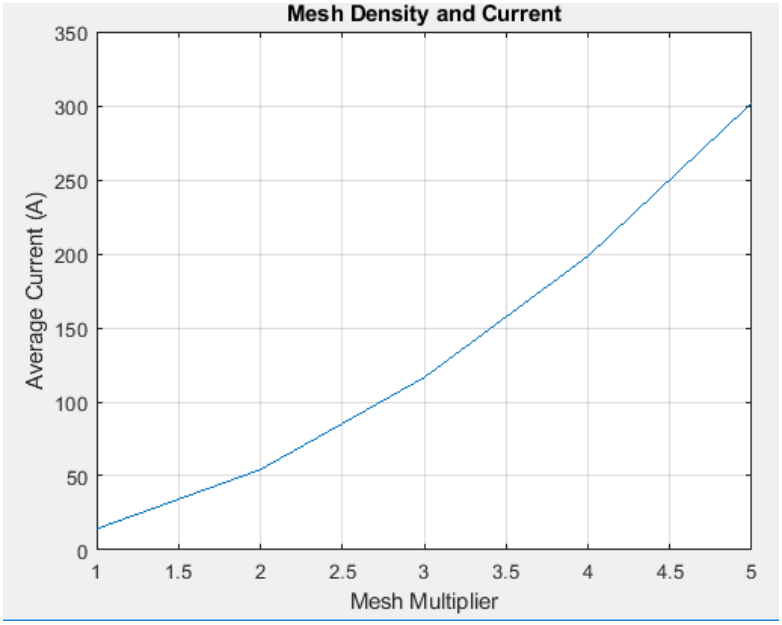

Figure 9: Mesh Density vs Current Density

3.3 Effect of Bottleneck Size on Current

The size of the bottleneck also impacts the current density. The same code from Part a is repeated, but with a modification that narrows the bottleneck with each iteration and plots the corresponding current density. This plot is shown in Figure 10:

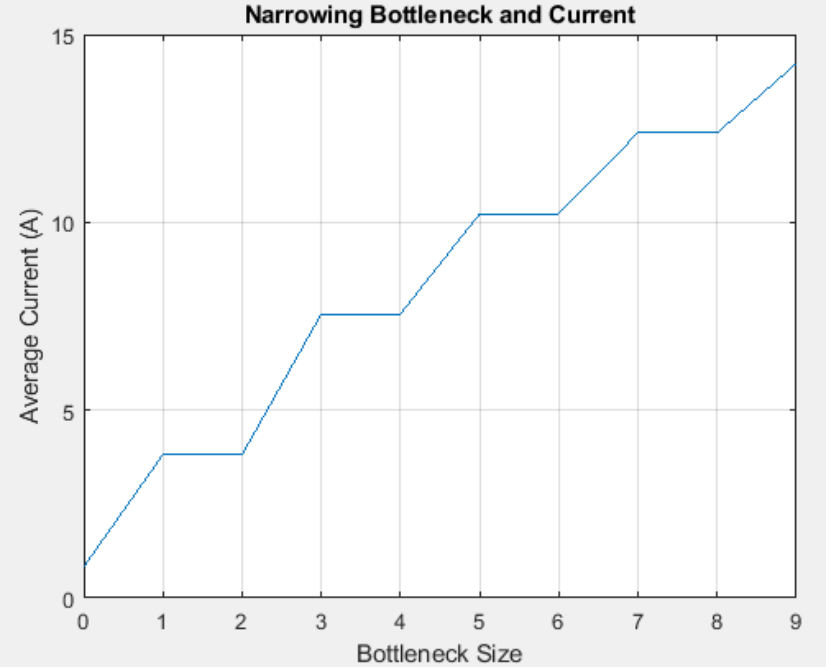

Figure 10: Effect of Bottleneck on Current Density

Note that as the bottleneck decreases in size, the average current density also decreases. This is due to Ohm's Law (V = IR) – as the resistance increases in the gap of the bottleneck due to it narrowing, the law results in the current decreasing (since the voltage remains fixed).

3.4 Effect of Varying Conductivity on Current

Finally, the effect of the conductivity of the contacts on the current density is examined. In this scenario, the sigma value of the overall box remains constant at sigma = 1, while the conductivity of the contacts is iteratively increased using a loop. The result of increasing conductivity of the contacts on the current density can be seen in Figure 11:

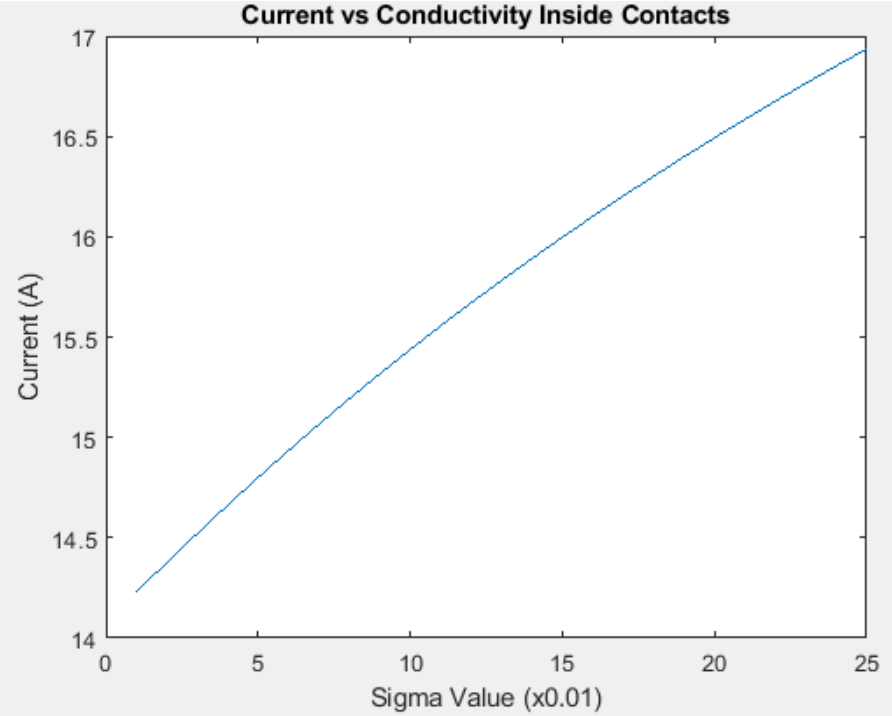

Figure 11: Conductivity vs Current Density

As the conductivity of the contacts is increased, the overall current density also increases. This makes sense, as increased conductivity allows more current to flow, increasing the overall current density.