ELEC 4700

Assignment 2

Finite Difference Method

Carleton University

Ottawa, Ontario Canada

Name: Eda Oritseserundede

CUID: 100993421

Due Date: Sunday, February 28th, 2021.

Introduction

In this assignment, a rectangular region was examined using the finite difference method. This method is used to solve laplace's electrostatic problems, through modelling the region as a mesh of resistances and voltages.

First, the voltage in a rectangular region was modelled and examined using the numerical and analytical series techniques. Furthermore, in the second experiment, rectangular bottlenecks constraints were added to the region with high resistivity. In addition, this helped the examinations of a current flowing device encountering the rectangular bottlenecks.

Question 1

Simple cases where;

$$V = Vo \ @ \ x = 0 \ and \ V = 0 \ @ \ x = L$$

was examined as the boundaries for the finite difference method. The following figures were derived from this experiment:

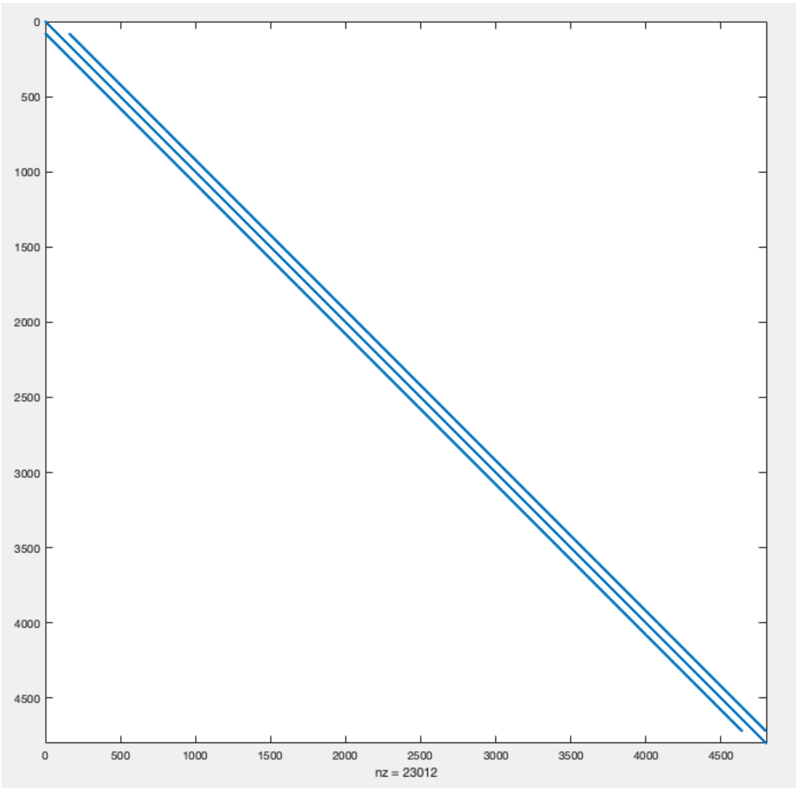

Figure 1: Plot showing the Sparsity of the G-matrix

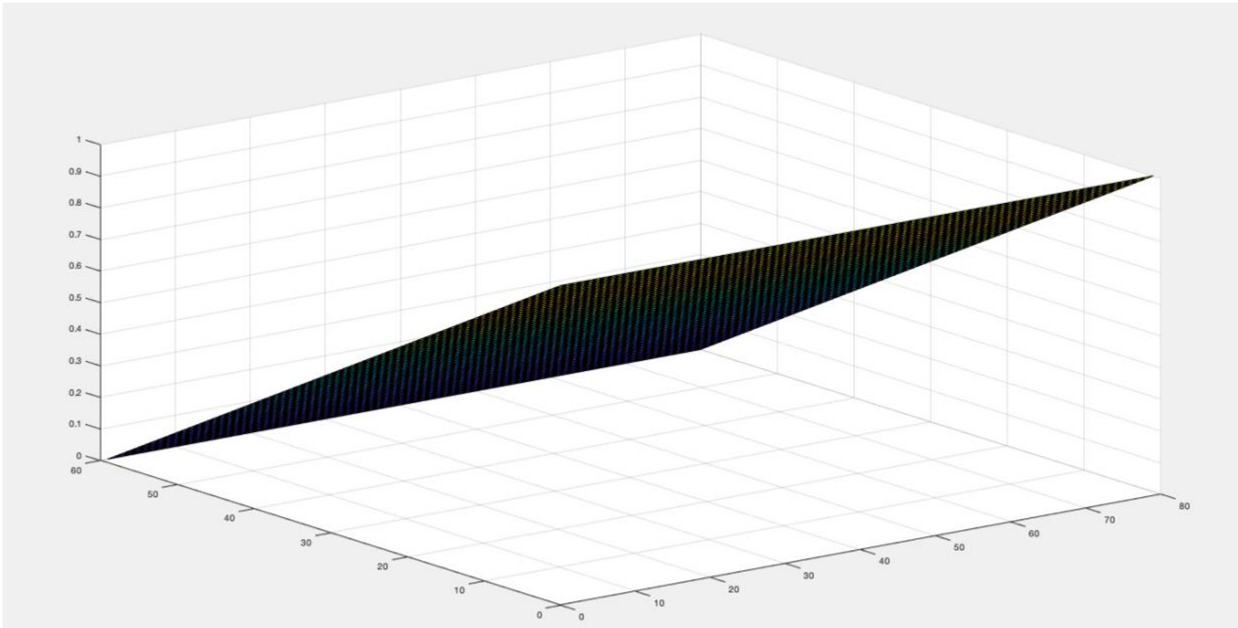

Figure 2: The voltage model of the fixed boundary conditions

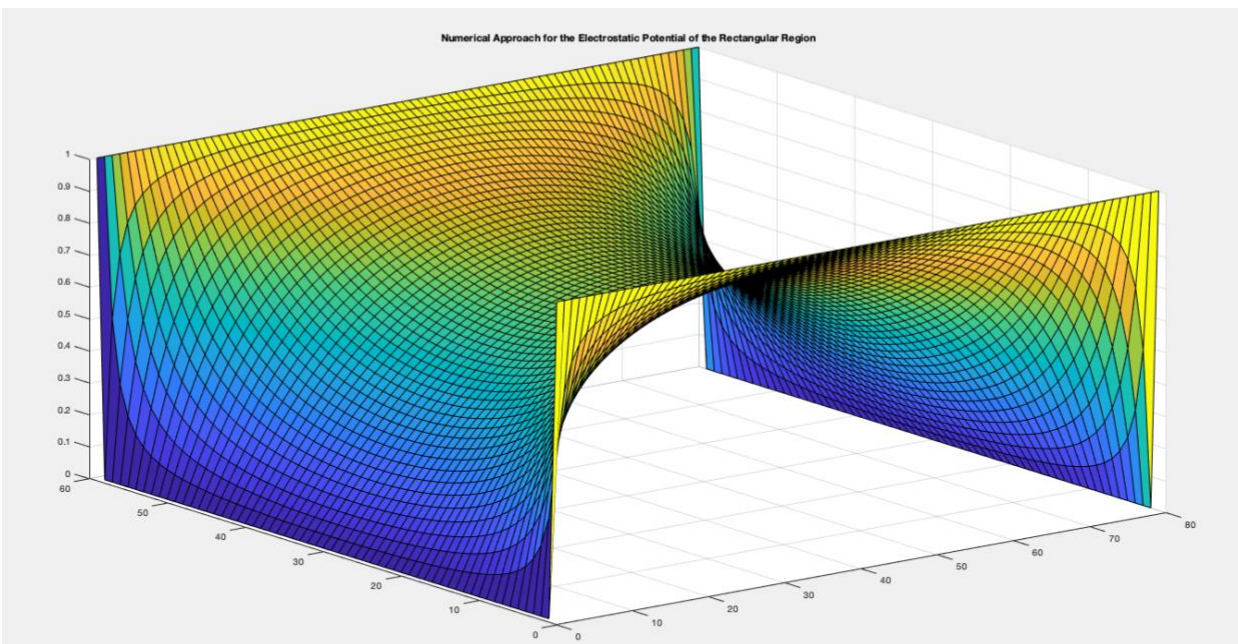

Figure 3: Electrostatic potential of the rectangular region
using the meshing and numerical approach

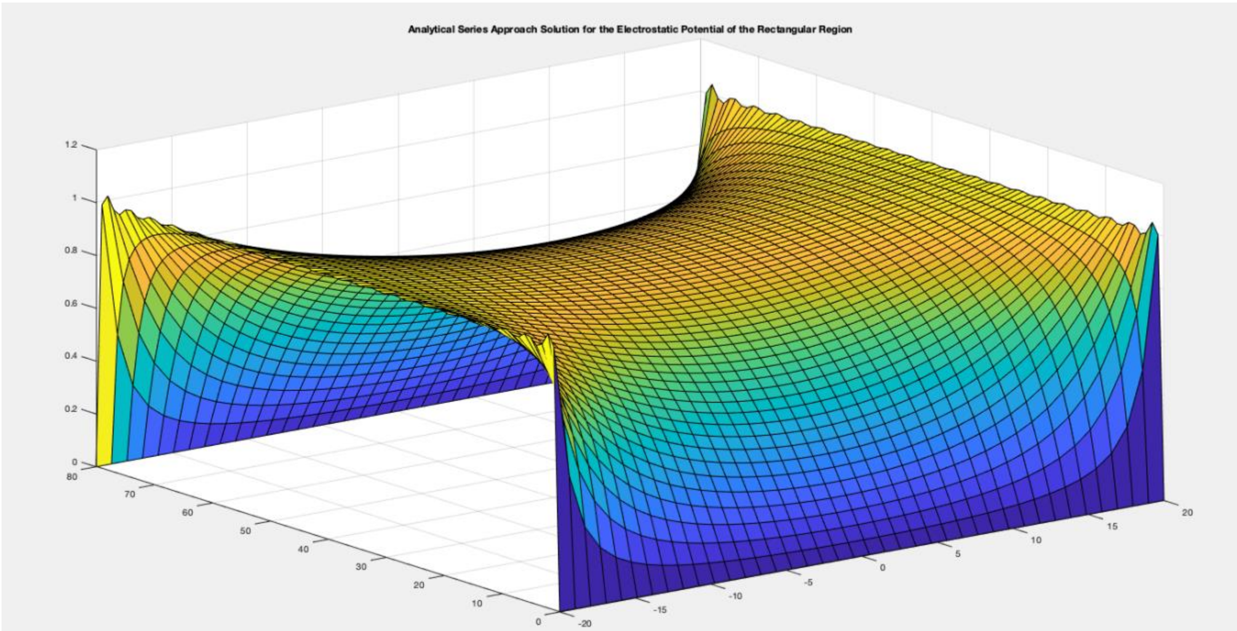

Figure 4: Electrostatic potential of the rectangular region
using the analytical series approach method

Figure’s 3 and 4 produced similar results, however they both have their advantages and disadvantages. The Numerical method uses a smaller space step which gives a more accurate result, however in this model the Numerical method works on a lot of approximations which contributes to the overall accuracy of the model.

Furthermore, in the Analytical approach we are working with an exact equation for our model, which means there are no approximations when compared to the Numerical approach. But the disadvantages of this method, is that because we are working in series, it becomes more difficult to get an accurate stop to the summation process since we are dealing with an equation.

In conclusion, the meshing method was best fit for dealing with our model approximations, however, when dealing with summations to infinity the Analytical method proves more accurate.

Question 2

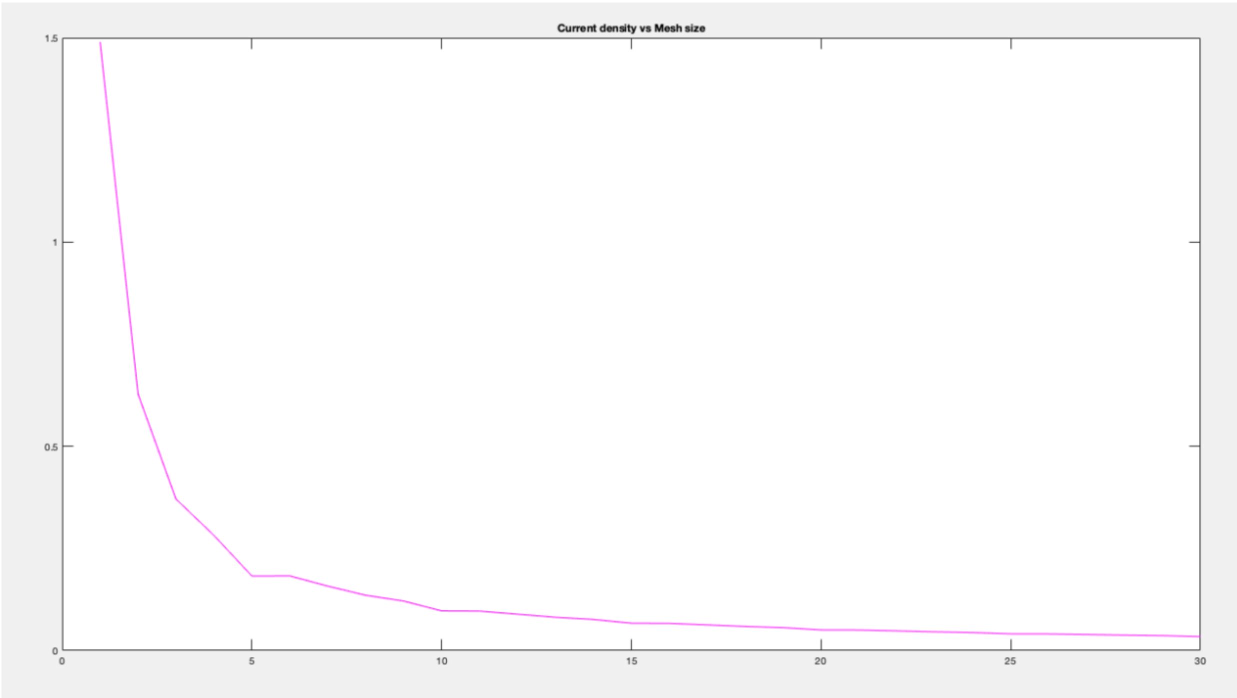

Figure 5: Current Density Vs Variation in Mesh size

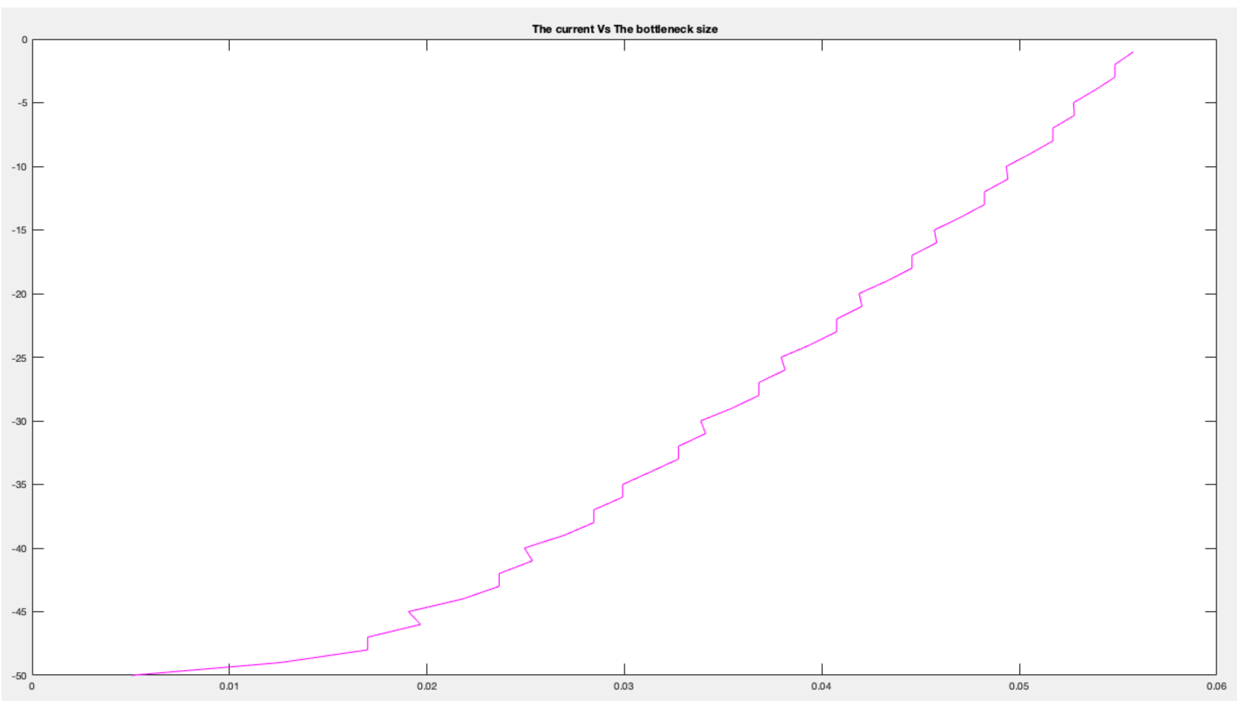

Figure 6: Current Density Vs Variation in Bottleneck width

Examination: From Figure’6 when the bottleneck width is increased the current gives a zig zag distribution, it also increases as well.

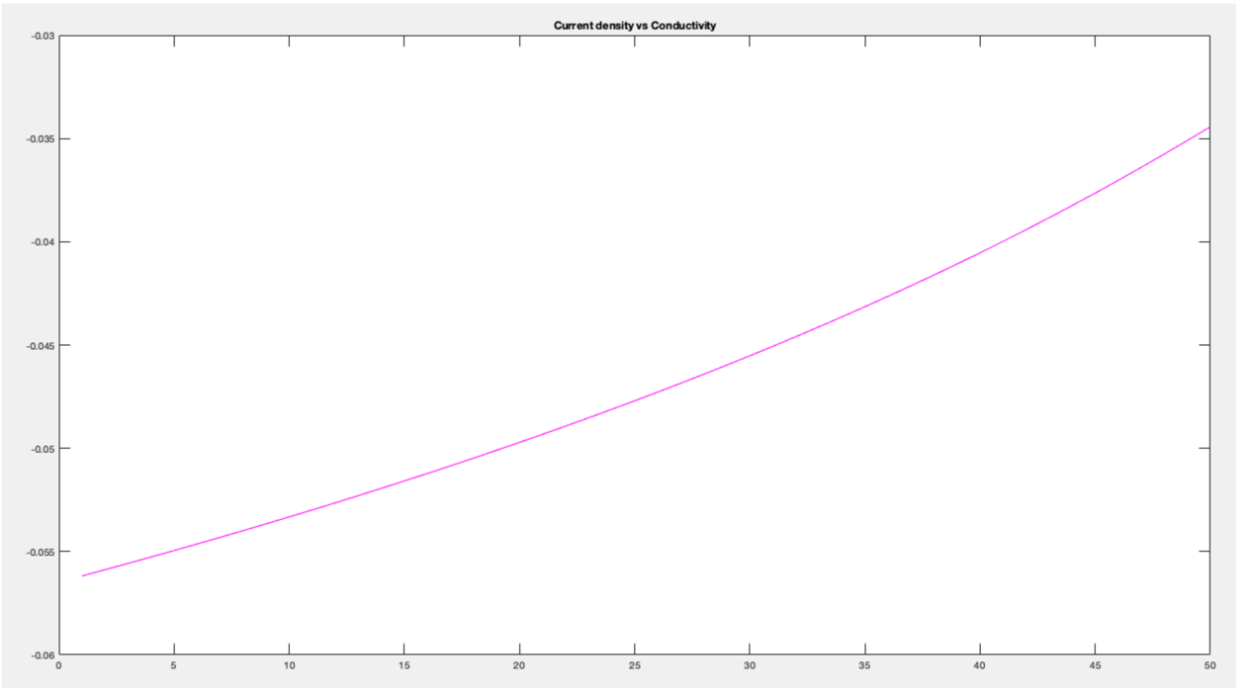

Figure 7: Current Density Vs Variation in Conductivity

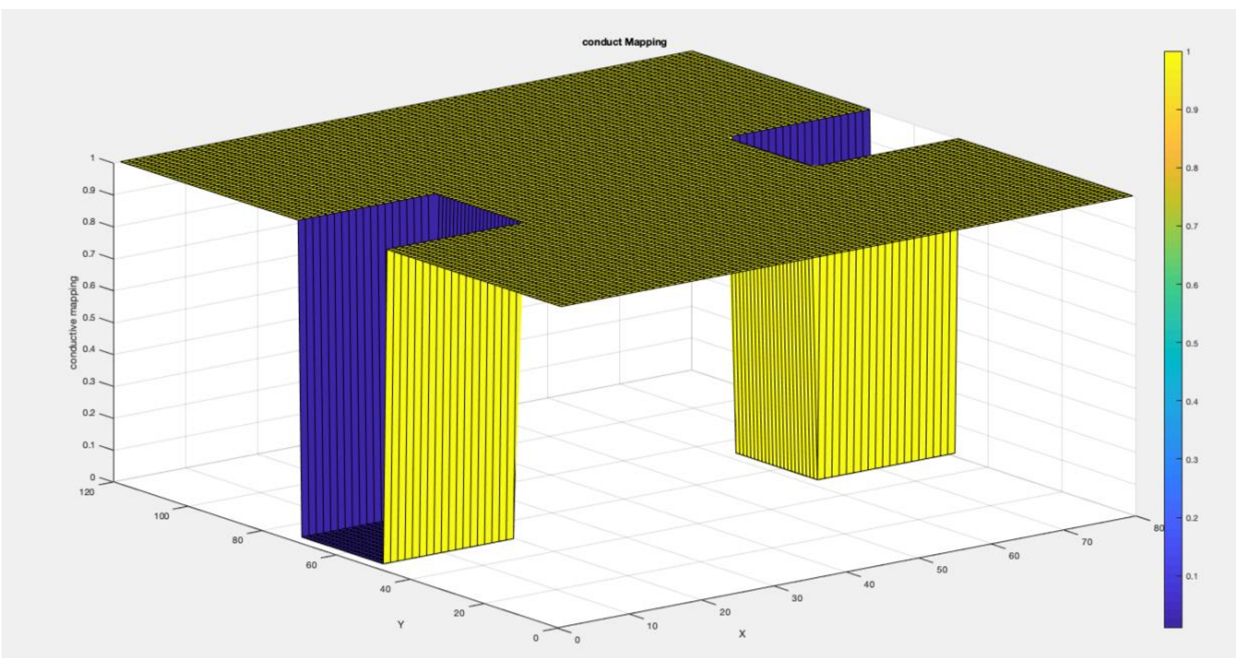

Figure 8: Conductivity Mapping

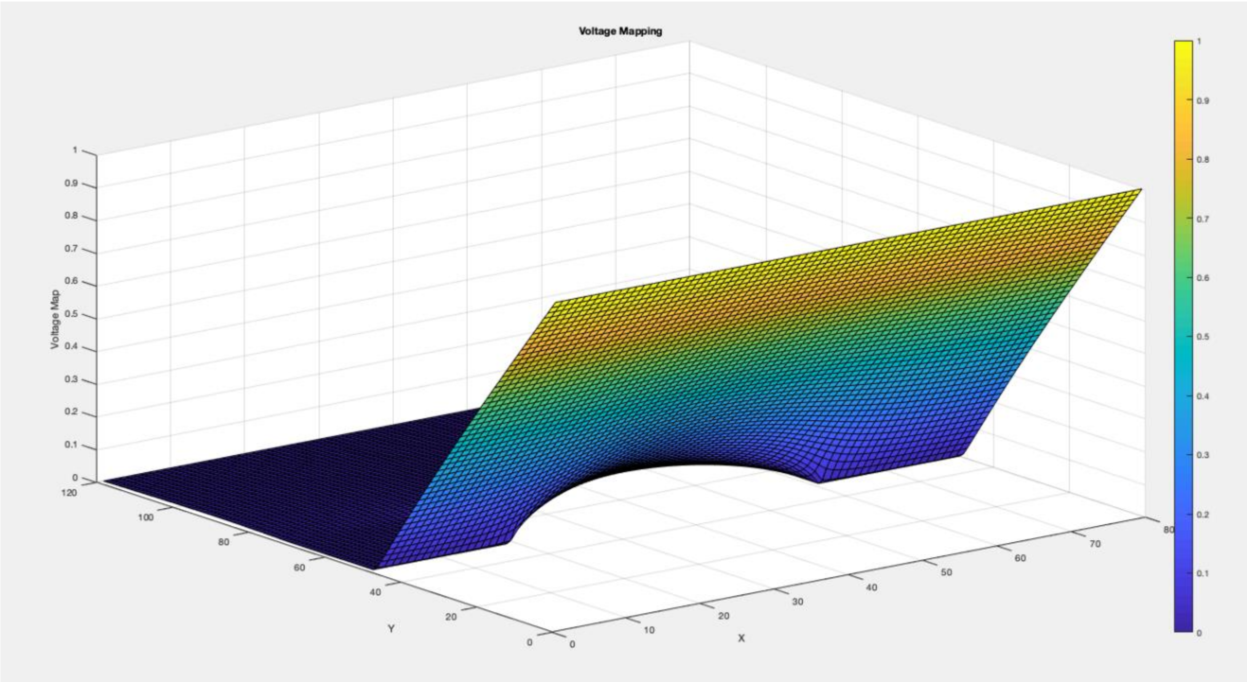

Figure 9: Voltage Mapping

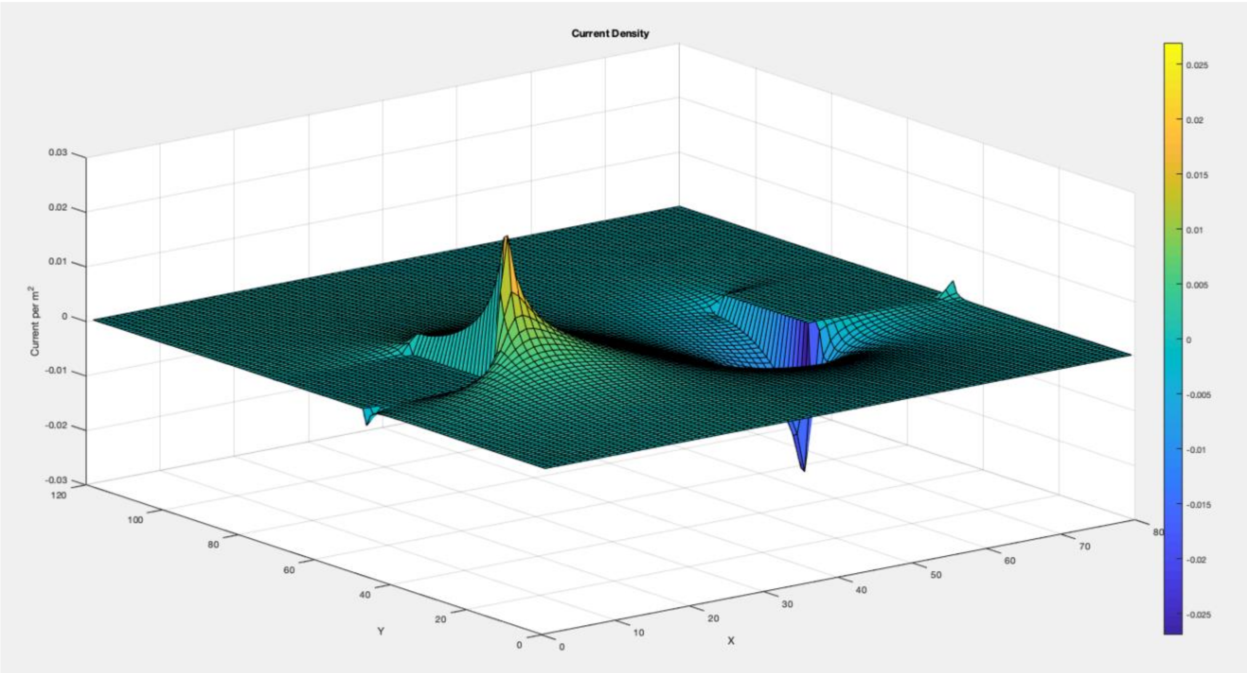

Figure 10: Plot of Current Density in Rectangular Region

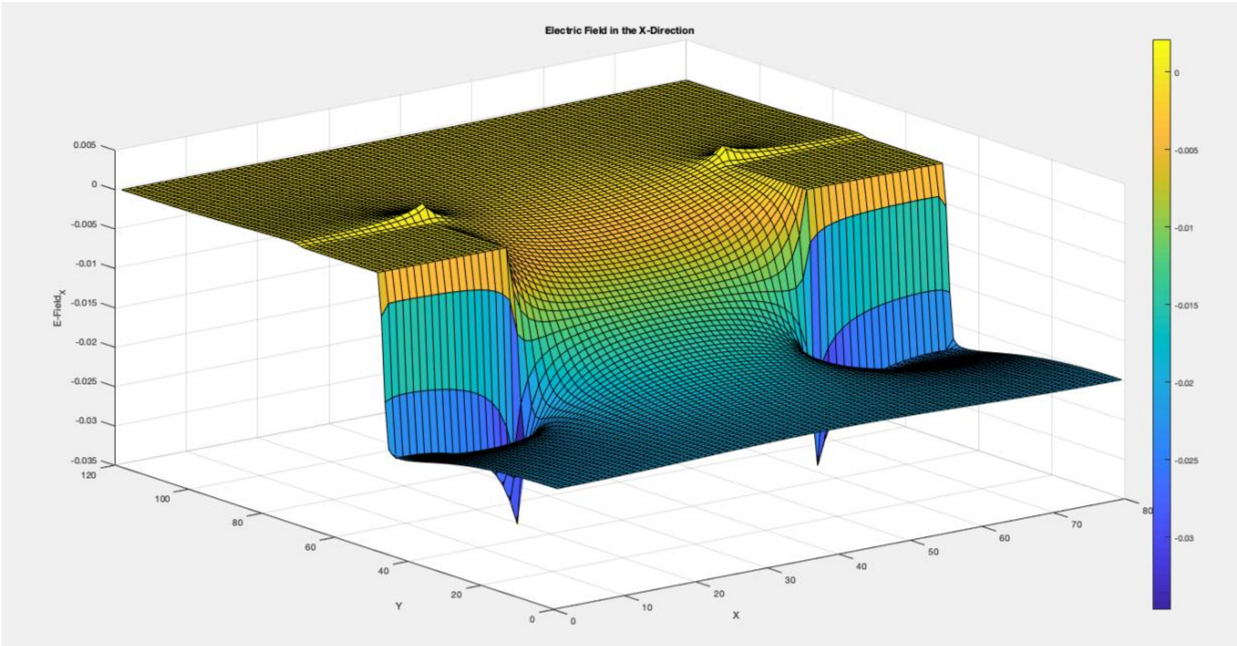

Figure 11: Electric Field flow in the X-direction of the Rectangular Region

Examination: From Figure 11, we notice the regions with high and low voltages. This was caused by an increase in voltage potential which in turn would generate an Electric field which would likely increase or decrease the voltage potential of the region.

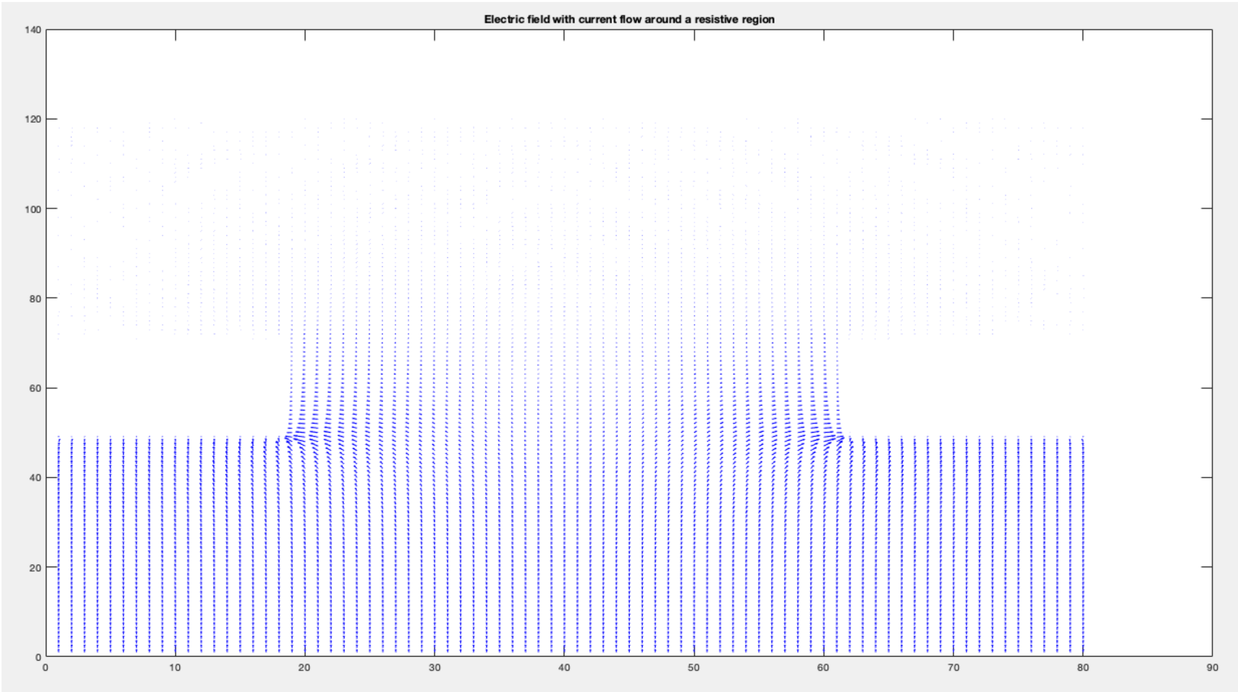

Figure 12: A Quiver plot of the Current Density

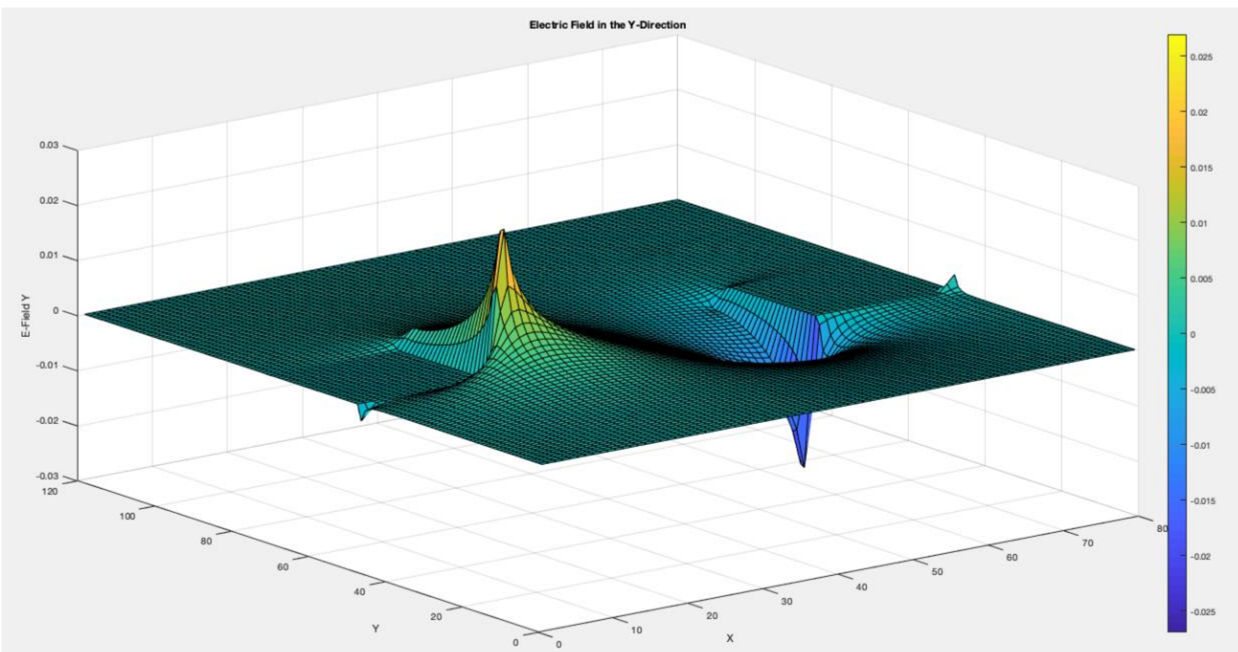

Figure 13: Electric Field in the Y-direction

Appendix

```
% ELEC 4700
% Name: Oritseserundede Eda
% Student Number: 100993421
% Assignment 2
% Question 1
% The potential for the rectangular region with isolated conducting sides
% using the finite difference method for solving the electrostatic
% potential for the width and length of the 1D rectangular region
% Using a 3/2 ratio for plots

clc
clear
set(0,'DefaultFigureWindowStyle','docked')

% The width and length of rectangular region using a 3/2 ratio
width = 60;
length = 80;

% Filling the G and V matrices with the width and length
G = sparse((width * length), (width * length));
V = zeros(1, (width * length));
v0 = 1;

for i = 1:width
    for j = 1:length
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        if (i == 1)
            G(n, :) = 0;
            G(n, n) = 1;
            V(1, n) = 1;
        elseif (i == width)
            G(n, :) = 0;
            G(n, n) = 1;

        elseif (j == 1 && i > 1 && i < width)
            G(n, n) = -1;
            G(n, nyp) = 1;

        elseif (j == length && i > 1 && i < width)
```

```
            G(n, n) = -1;
            G(n, nym) = 1;

        else
            G(n, n) = -4;
            G(n, nxm) = 1;
            G(n, nxp) = 1;
            G(n, nym) = 1;
            G(n, nyp) = 1;
        end
    end
end

figure (1);
spy(G);
m_sol = G\V';
figure (2);

surface_1 = zeros(width, length);

for i = 1:width

    for j = 1:length
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        surface_1(i, j) = m_sol(n);
    end
end

surf(surface_1);

% Question 1(b)
% solving for V = Vo at x equal to zero and the length of the rectangular
% region and for V = 0 at y equal to zero and the width of the rectangular
% region. With the solution done through a bunch of mesh sizes to the
% analytic series solution

G2 = sparse((width * length), (width * length));
V2 = zeros(1, (width * length));
vo = 1;

for i = 1:width
    for j = 1:length
```

```
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        if i == 1
            G2(n, :) = 0;
            G2(n, n) = 1;
            V2(1, n) = vo;
        elseif i == width
            G2(n, :) = 0;
            G2(n, n) = 1;
            V2(1, n) = vo;
        elseif j == 1
            G2(n, :) = 0;
            G2(n, n) = 1;
        elseif j == length
            G2(n, :) = 0;
            G2(n, n) = 1;
        else
            G2(n, :) = 0;
            G2(n, n) = -4;
            G2(n, nxm) = 1;
            G2(n, nxp) = 1;
            G2(n, nym) = 1;
            G2(n, nyp) = 1;
        end
    end
end

solution2 = G2\V2';
figure (3);
surface_2 = zeros(width, length);

for i = 1:width

    for j = 1:length
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        surface_2(i, j) = solution2(n);
    end
end
```

```
surf(surface_2);
title("Numerical Approach for the Electrostatic Potential of the Rectangular Region");

region = zeros(80, 40);
a = 80;
b = 20;

x = linspace(-20,20,40);
y = linspace(0,80,80);

[mesh_x, mesh_y] = meshgrid(x, y);

for n = 1:2:200
    region =  (region + (4 * v0/pi).*(cosh((n * pi * mesh_x)/a) .* sin((n * pi * mesh_y)/a)) ./ (n *
cosh((n * pi * b)/a)));
    figure(4);
    surf(x, y, region);
    title("Analytical Series Approach Solution for the Electrostatic Potential of the Rectangular
Region");
    pause(0.01);
end

% Question 2
% In this question, the finite difference method is also used to solve the
% current flow in the rectangular region
clc
clear
set(0,'DefaultFigureWindowStyle','docked')
width = 120;
length = 80;
x_num = 80;
y_num = 100;

G = sparse((width * length), (width * length));
V = zeros(1, (width * length));
vo = 1;
% The conducting regions outside and inside the boxes, with the bottlenecks
% per region constrained by the 3/2 ratio of the width and length of our
% rectangular region.
conduct_out = 1;
conduct_in = 1e-2;
bn_1 = [(width * 0.4), (width * 0.6),  length, (length * 0.75)];
bn_2 = [(width * 0.4), (width * 0.6), 0, (length * 0.25)];
conduct_map = ones(width, length);

for i = 1:width
```

```
    for j = 1:length
        if(i > bn_1(1) && i < bn_1(2) && ((j < bn_2(4)) || (j > bn_1(4))))
            conduct_map(i,j) = 1e-2;
        end
    end
end

% Mapping the conduct of the region
figure(5);
surf(conduct_map);
colorbar
title('conduct Mapping');
xlabel('X')
ylabel('Y')
zlabel('conductive mapping')

% Boundary conditions for the G-matrix
for i = 1:width
    for j = 1:length
        % Boundary locations
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        % indexes to be filled in the boundaries
        index1 = (i == 1);
        index2 = (i == width);
        index3 = (j == 1 && i > 1 && i < width);
        index4 = (i == bn_1(1));
        index5 = (i == bn_1(2));
        index6 = (i > bn_1(1) && i < bn_1(2));
        index7 = (j == length && i > 1 && i < width);
        index8 = (i == bn_1(2));
        index9 = (i > bn_1(1) && i < bn_1(2));
        index10 = (i == bn_1(1) && ((j < bn_2(4)) || (j > bn_1(4))));
        index11 = (i == bn_1(2) && ((j < bn_2(4)) || (j > bn_1(4))));
        index12 = (i > bn_1(1) && i < bn_1(2) && ((j < bn_2(4)) || (j > bn_1(4))));

        if (index1)
            G(n, :) = 0;
            G(n, n) = 1;
            V(1, n) = 1;
        elseif (index2)
            G(n, :) = 0;
```

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

elseif (index3)

    if (index4)
        G(n, n) = -3;
        G(n, nyp) = conduct_in;
        G(n, nxp) = conduct_in;
        G(n, nxm) = conduct_out;

    elseif (index5)
        G(n, n) = -3;
        G(n, nyp) = conduct_in;
        G(n, nxp) = conduct_out;
        G(n, nxm) = conduct_in;

    elseif (index6)
        G(n, n) = -3;
        G(n, nyp) = conduct_in;
        G(n, nxp) = conduct_in;
        G(n, nxm) = conduct_in;
    else
        G(n, n) = -3;
        G(n, nyp) = conduct_out;
        G(n, nxp) = conduct_out;
        G(n, nxm) = conduct_out;
    end

elseif (index7)

    if (index4)
        G(n, n) = -3;
        G(n, nym) = conduct_in;
        G(n, nxp) = conduct_in;
        G(n, nxm) = conduct_out;

    elseif (index8)
        G(n, n) = -3;
        G(n, nym) = conduct_in;
        G(n, nxp) = conduct_out;
        G(n, nxm) = conduct_in;

    elseif (index9)
        G(n, n) = -3;
        G(n, nym) = conduct_in;
        G(n, nxp) = conduct_in;
```

```
            G(n, nxm) = conduct_in;
        else
            G(n, n) = -3;
            G(n, nym) = conduct_out;
            G(n, nxp) = conduct_out;
            G(n, nxm) = conduct_out;
        end

    else

        if (index10)
            G(n, n) = -4;
            G(n, nyp) = conduct_in;
            G(n, nym) = conduct_in;
            G(n, nxp) = conduct_in;
            G(n, nxm) = conduct_out;

        elseif (index11)
            G(n, n) = -4;
            G(n, nyp) = conduct_in;
            G(n, nym) = conduct_in;
            G(n, nxp) = conduct_out;
            G(n, nxm) = conduct_in;

        elseif (index12)
            G(n, n) = -4;
            G(n, nyp) = conduct_in;
            G(n, nym) = conduct_in;
            G(n, nxp) = conduct_in;
            G(n, nxm) = conduct_in;
        else
            G(n, n) = -4;
            G(n, nyp) = conduct_out;
            G(n, nym) = conduct_out;
            G(n, nxp) = conduct_out;
            G(n, nxm) = conduct_out;
        end

    end
  end
end

solution1 = G\V';
surface_2 = zeros(width, length);

% Mapping the solution vector to a matrix, using width and length
```

```
for i = 1:width
    for j = 1:length
        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;
        surface_2(i, j) = solution1(n);
    end
end

figure (6);
surf(surface_2);
colorbar
title('Voltage Mapping');
xlabel('X')
ylabel('Y')
zlabel('Voltage Map')

[E_y, E_x] = gradient(surface_2);
J = conduct_map.*gradient(surface_2);
Jx = conduct_map.*(-E_y);
Jy = conduct_map.*(-E_x);

% The current density using the Quiver plot
figure(7)
quiver (Jx,Jy,'m');
title('Resistive Bottlenecks and Current flow')

% Plot for Current Density
figure (8)
surf(J)
colorbar
title('Current Density');
xlabel('X')
ylabel('Y')
zlabel('Current per m^2')

% Plot of Electric Field in the Y-direction
figure (9)
surf (E_y)
colorbar
title('Electric Field in the Y-Direction');
xlabel('X')
ylabel('Y')
zlabel('E-Field Y')
```

```
% Plot of electric field in the X-direction
figure (10)
surf(E_x)
colorbar
title('Electric Field in the X-Direction')
xlabel('X')
ylabel('Y')
zlabel('E-Field_X')

% Plot of the E-field(x,y)
E_field = sqrt(E_y.^2 + E_x.^2);
figure (11)
surf(E_field)

% The Electric Field using the Quiver plot
figure (12)
quiver (-E_y, -E_x, 'b');
title('Electric field with current flow around a resistive region')

% Calculating the current density and mesh size
clc
set(0,'DefaultFigureWindowStyle','docked')
clear

num = 30;
% Using width and length ratio as provided in the assignment instructions
width = 2;
length = 3;

curr_den = [];

for num = 1:num
   width = 3*num;
   length = 2*num;
   V0 = 5;
   G = sparse(length*width,length*width);
   solution1 = zeros(length*width,1);
   conduct_out = 1;
   conduct_in = 1e-2;
   conduct = conduct_out.*ones(length,width);

   for i = 1:width
      for j = 1:length
         if((i <= 0.8*width && i >= (0.3*width) && j <= (0.3*length)) || (i <= (0.8*width) && i
>= (0.3*width) && j >= (0.8*length)))
```

```
            conduct(j,i) = conduct_in;
        end
    end
end

for i = 1:width
    for j = 1:length

        n = j + (i - 1) * length;
        nxm = j + ((i-1) - 1) * length;
        nxp = j + ((i+1) - 1) * length;
        nym = (j-1) + (i - 1) * length;
        nyp = (j+1) + (i - 1) * length;

        if(i == 1)
            solution1(n,1) = V0;
            G(n,n) = 1;
        elseif(i == width)
            solution1(n,1) = 0;
            G(n,n) = 1;
        elseif(j == 1)

            G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j+1,i))/2));
            G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
            G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
            G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;

            solution1(n,1) = 0;
        elseif(j == length)
            G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2));
            G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
            G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
            G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;
            solution1(n,1) = 0;
        else
            G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2)+((conduct(j,i) + conduct(j+1,i))/2));
            G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
            G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
            G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;
            G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;
            solution1(n,1) = 0;
        end
    end
```

```
    end

    V = G\solution1;

    for i = 1:width
        for j = 1:length
            n = (i-1)*length+j;
            surface_2(j,i) = V(n,1);
        end
    end

    [E_x2,E_y2] = gradient(surface_2);
    J_xdir = conduct.*(-E_x2);
    J_ydir = conduct.*(-E_y2);
    curr_den(num) = mean(mean((((J_xdir.^2)+(J_ydir.^2)).^0.5)));
end

% The current density Vs Mesh size
figure(13)
plot(1:num,curr_den,'m')
title('Current density vs Mesh size')

clear
num = 50;
curr_den = [];
for num = 1:num
    width = 90;
    length = 60;
    V0 = 5;
    G = sparse(length*width,length*width);
    solution1 = zeros(length*width,1);
    conduct_out = 1;
    conduct_in = 0.01;
    conduct = conduct_out.*ones(length,width);

    for i = 1:width
        for j = 1:length
            if((i <= 0.8*width && i >= 0.3*width && j <= 0.01*num*length) || (i <= (1-
num*0.01)*length && i >= 0.25*width && j >= (1-num*0.01)*length))
                conduct(j,i) = conduct_in;
            end
        end
    end

    for i = 1:width
        for j = 1:length
```

```
            n = j + (i - 1) * length;
            nxm = j + ((i-1) - 1) * length;
            nxp = j + ((i+1) - 1) * length;
            nym = (j-1) + (i - 1) * length;
            nyp = (j+1) + (i - 1) * length;
            if(i == 1)
                solution1(n,1) = V0;
                G(n,n) = 1;
            elseif(i == width)
                solution1(n,1) = 0;
                G(n,n) = 1;
            elseif(j == 1)
                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j+1,i))/2));
                G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;

                solution1(n,1) = 0;
            elseif(j == length)
                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2));
                G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;

                solution1(n,1) = 0;
            else
                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2)+((conduct(j,i) + conduct(j+1,i))/2));
                G(n,nxm) = ((conduct(j,i) + conduct(j,i-1))/2);
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;
                G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;
                solution1(n,1) = 0;
            end
        end
    end

    V = G\solution1;
    for i = 1:width
        for j = 1:length
            n = j + (i - 1) * length;
            nxm = j + ((i-1) - 1) * length;
            nxp = j + ((i+1) - 1) * length;
            nym = (j-1) + (i - 1) * length;
```

```
            nyp = (j+1) + (i - 1) * length;
            surface_2(j,i) = V(n,1);
        end
    end

    [E_x2,E_y2] = gradient(surface_2);
    J_xdir = conduct.*(-E_x2);
    J_ydir = conduct.*(-E_y2);
    curr_den(num) = mean(mean((((J_xdir.^2)+(J_ydir.^2)).^0.5)));
end

% Plot of the current density vs the bottleneck size
figure(14)
plot(curr_den,(-1)*(1:num), 'm')
title('The current Vs The bottleneck size')

clear
num = 50;
curr_den = [];

for num = 1:num

    width = 90;
    length = 60;
    V0 = 5;
    G = sparse(length*width,length*width);
    solution1 = zeros(length*width,1);
    conduct_out = 1;
    conduct_in = 1.02-num*0.02;
    conduct = conduct_out.*ones(length,width);

    for i = 1:width
        for j = 1:length
            if((i <= 0.8*width && i >= 0.3*width && j <= 0.3*length) || (i <= 0.8*width && i >=
0.3*width && j >= 0.8*length))
                conduct(j,i) = conduct_in;
            end
        end
    end

    for i = 1:width
        for j = 1:length
            n = j + (i - 1) * length;
            nxm = j + ((i-1) - 1) * length;
            nxp = j + ((i+1) - 1) * length;
            nym = (j-1) + (i - 1) * length;
```

```
            nyp = (j+1) + (i - 1) * length;

            if(i == 1)
                solution1(n,1) = V0;
                G(n,n) = 1;
            elseif(i == width)
                solution1(n,1) = 0;
                G(n,n) = 1;
            elseif(j == 1)

                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j+1,i))/2));
                G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;

                solution1(n,1) = 0;
            elseif(j == length)

                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2));
                G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;

                solution1(n,1) = 0;
            else

                G(n,n) = -(((conduct(j,i) + conduct(j,i-1))/2)+((conduct(j,i) +
conduct(j,i+1))/2)+((conduct(j,i) + conduct(j-1,i))/2)+((conduct(j,i) + conduct(j+1,i))/2));
                G(n,nxm) = (conduct(j,i) + conduct(j,i-1))/2;
                G(n,nxp) = (conduct(j,i) + conduct(j,i+1))/2;
                G(n,nym) = (conduct(j,i) + conduct(j-1,i))/2;
                G(n,nyp) = (conduct(j,i) + conduct(j+1,i))/2;
                solution1(n,1) = 0;

            end
        end
    end

    V = G\solution1;

    for i = 1:width
        for j = 1:length
            n = j + (i - 1) * length;
            nxm = j + ((i-1) - 1) * length;
```

```
            nxp = j + ((i+1) - 1) * length;
            nym = (j-1) + (i - 1) * length;
            nyp = (j+1) + (i - 1) * length;
            surface_2(j,i) = V(n,1);
        end
    end

    [E_x2,E_y2] = gradient(surface_2);
    J_xdir = conduct.*(-E_x2);
    J_ydir = conduct.*(-E_y2);
    curr_den(num) = mean(mean((((J_xdir.^2)+(J_ydir.^2)).^0.5)));
end

% Current density vs Conductivity plot
figure(15)
plot(1:num,(-1)*curr_den,'m')
title('Current density vs Conductivity')
```