ELEC 4700: Assignment 2 - Finite Difference Method

Table of Contents

Jacob Godin - 100969991

Part 1

Use the Finite Difference Method to solve for the electrostatic potential in the rectangular region L x W using del^2 * V = 0

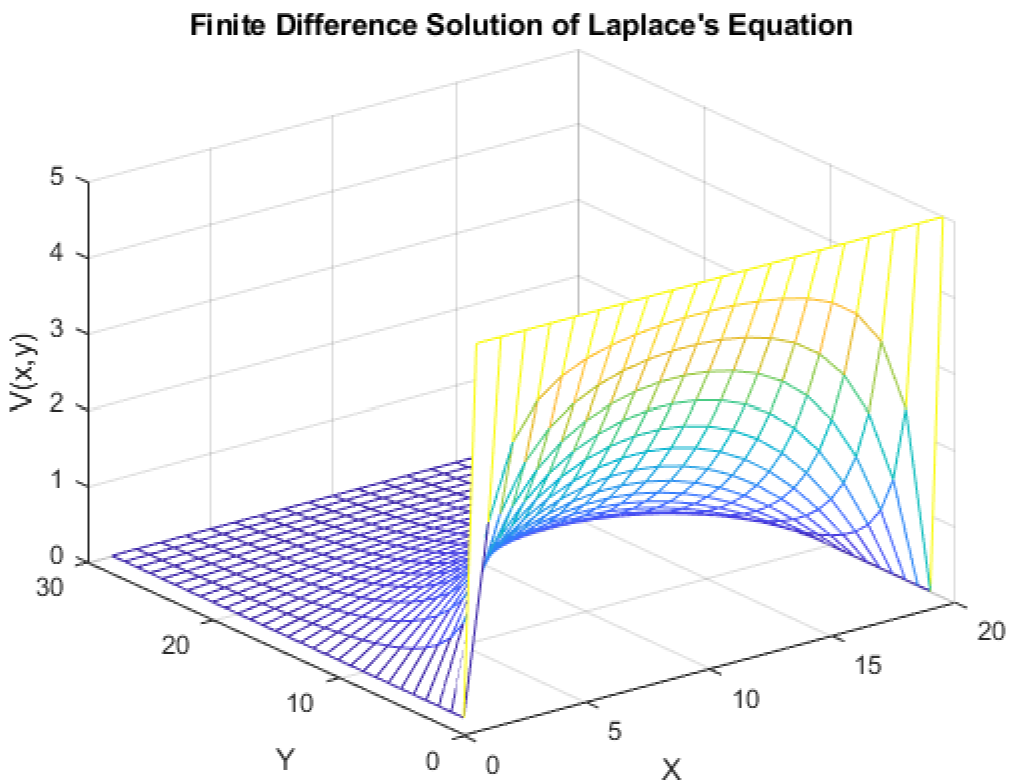
Part 1: a)

V = V0 @ x = 0, V = 0 @ x = L and V = 0 @ y = 0, y = W

```
% The Finite Difference Method will be used to solve Laplace's
 equation in
% 2D space to determine the effect of an electrostatic potential. The
% matrix form GV = F will be used to solve the equation.

clear all; clc;

% Inputs
W = 20;
L = 1.5*W;

nx = L;
ny = W;

V0 = 5;

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

% Surface map
sMap = zeros(nx,ny);

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

```
        if i == 1 % left
            G(n,:) = 0;
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = 0;
        elseif j == 1 % bottom
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = 0;
        elseif j == ny % top
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = 0;
        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

phi_vec = G\B;

% Map values back to Surface Map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap(i,j) = phi_vec(n);
    end
end

figure(1)
mesh(sMap);
xlabel("X");
ylabel("Y");
zlabel("V(x,y)");
title("Finite Difference Solution of Laplace's Equation");
```

```
% As stated in the boundary conditions, the potential of V0 is located
 on
% the left side of the surface and 0V is located on the right
 boundary. The
% top and bottom boundaries are left open. The electrostatic potential
% gradually decreases as away from the left side of V0.
```



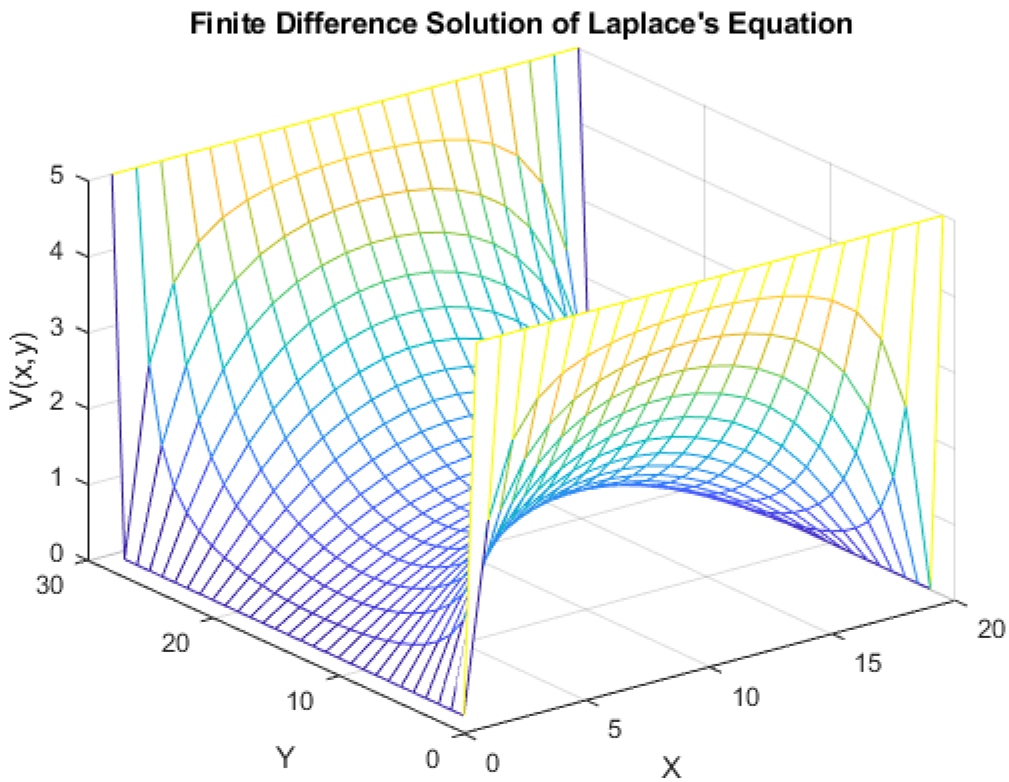
Part 1: b)

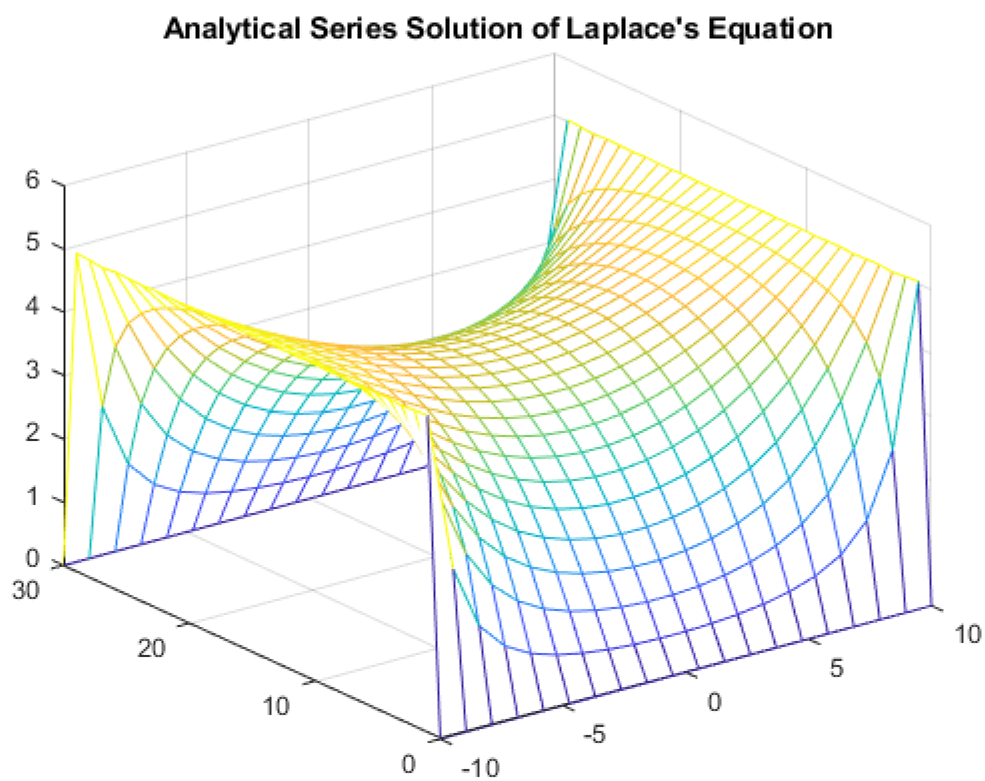
V = V0 @ x = 0, x = L and V = 0 @ y = 0, y = W

```
clear all; clc;

% Inputs
W = 20;
L = 30;

nx = L;
ny = W;

V0 = 5;

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

% Surface map
```

```
sMap = zeros(nx,ny);

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

       if i == 1 % left
           G(n,:) = 0;
           G(n,n) = 1;

           B(n) = V0;
       elseif i == nx % right
           G(n,:) = 0;
           G(n,n) = 1;

           B(n,1) = V0;
       elseif j == 1 % bottom
           G(n,:) = 0;
           G(n,n) = 1;

           B(n,1) = 0;
       elseif j == ny % top
           G(n,:) = 0;
           G(n,n) = 1;

           B(n,1) = 0;
       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

phi_vec = G\B;

% Map values back to Surface Map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap(i,j) = phi_vec(n);
    end
end

figure(2)
mesh(sMap);
```

```
xlabel("X");
ylabel("Y");
zlabel("V(x,y)");
title("Finite Difference Solution of Laplace's Equation");

% Similar to the plot above, this plot features the right side of the
% surface with a potential of V0.

% Analytical Series Solution

sMap2=zeros(L,W);
a=30;
b=10;

x=linspace(-10,10,20);
y=linspace(0,L,L);
[xx,yy]=meshgrid(x,y);

figure(3)
for n=1:2:600
    sMap2 = (sMap2+(cosh(n*pi*xx/a).*sin(n*pi*yy/a))./(n*cosh(n*pi*b/
a)));

    mesh(x,y,(4*V0/pi)*sMap2)
    title("Analytical Series Solution of Laplace's Equation")
    pause(0.01)

end

% The above plot is obtained from solving the analytic series solution
 of
% Laplace's equation. Due to the memory constraints of MATLAB, the n
% variable is only capable of reaching a value of around 600. Anthing
% greater than this value, the solution becomes inaccurate. To obtain
% perfect accuracy, the series must be evaluated to infinity.
 Therefore the
% numeric solution is prefered.
```


Finite Difference Solution of Laplace's Equation
V(x,y)
5
4
3
2
1
0
30
20
10
0
Y
0
5
10
15
20
X

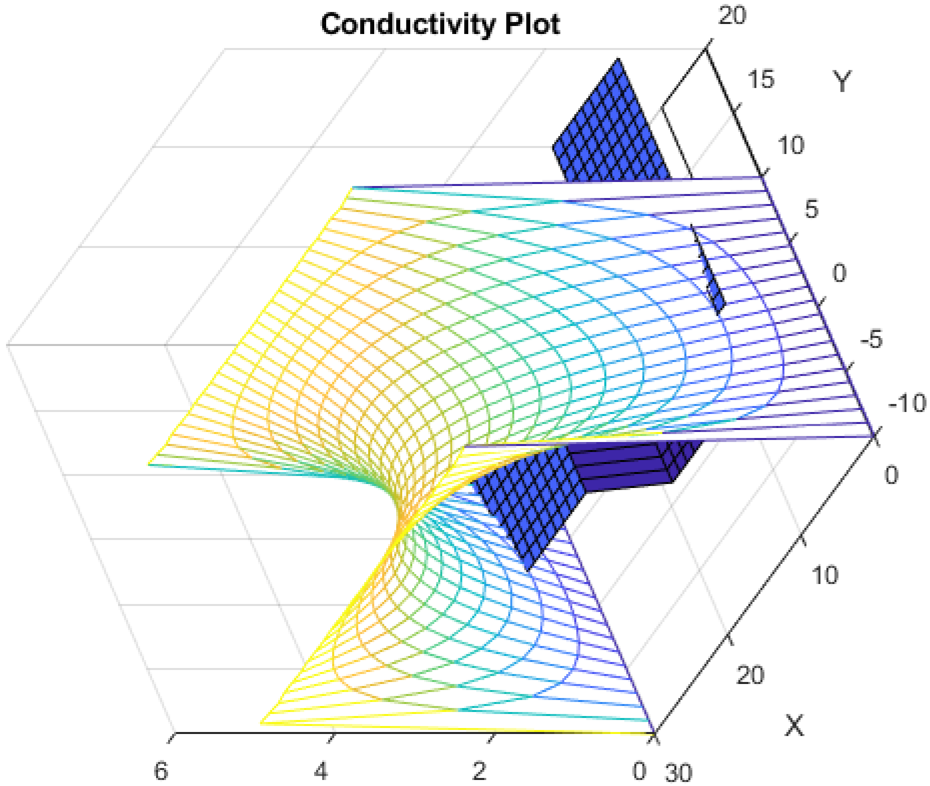

Analytical Series Solution of Laplace's Equation
6
5
4
3
2
1
0
30
20
10
0
-10
-5
0
5
10

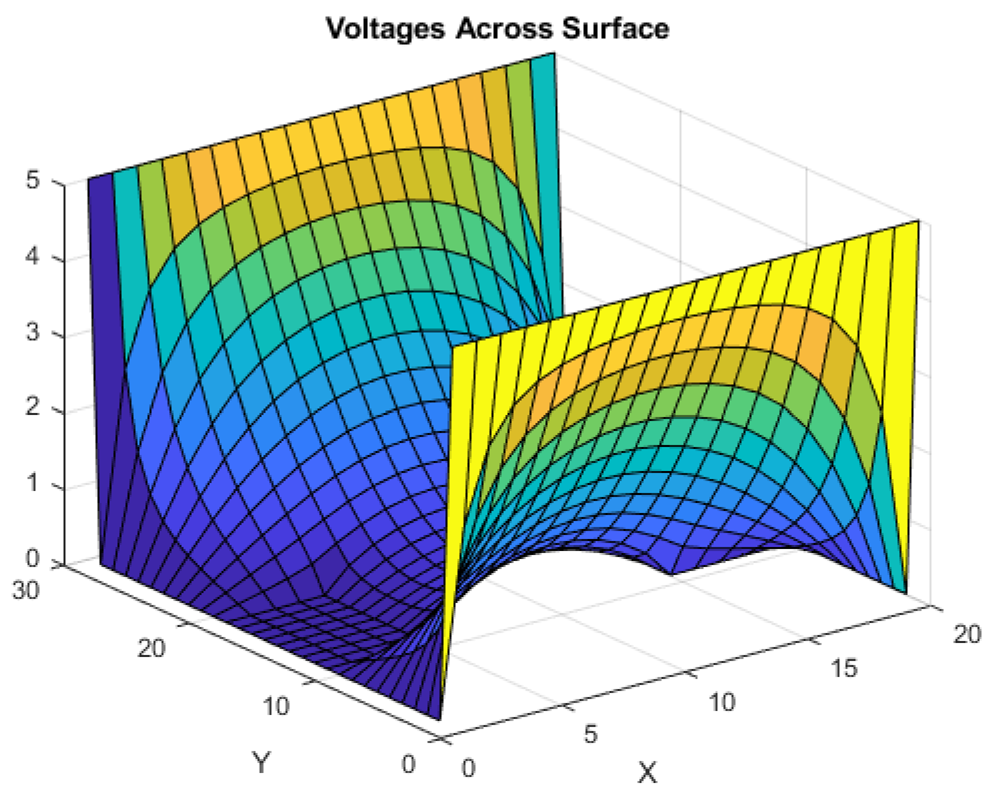
Part 2: a)

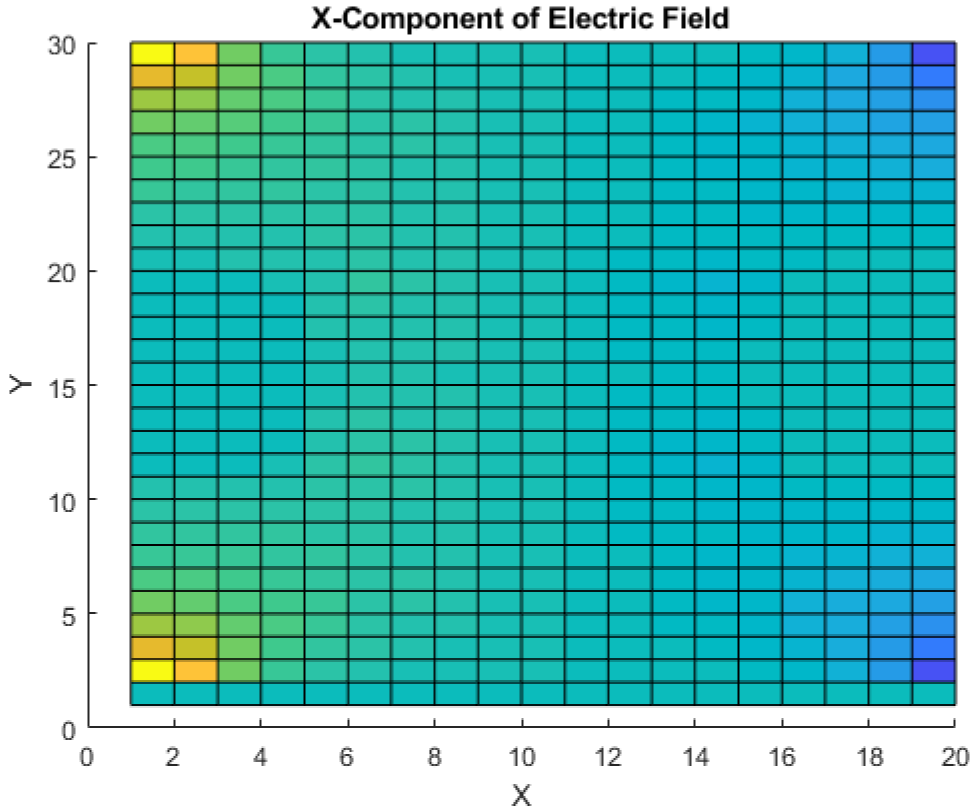
```
% Similar to Part 1, this part describes the current flow in a 2D
 surface
% with a highly resistive barrier.

clear all; clc;

%Inputs
W = 20;
L = 1.5*W;

nx = L;
ny = W;

V0 = 5;

sig1 = 1;
sig2 = 1e-2;

Wb = 6;
Lb = 10;

top_box = [(nx/2)-(Lb/2) 0 Lb Wb];
bottom_box = [(nx/2)-(Lb/2) ny-Wb Lb Wb];

% Populate sigma matrix
sigma = ones(nx,ny);
for i=1:nx
    for j=1:ny
        if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2)))
            sigma(i,j) = sig2;
        end
    end
end

% Conductivity Plot
figure(3)
rectangle('Position',top_box)
rectangle('Position',bottom_box)
surface(sigma)
camroll(90)
title('Conductivity Plot')
xlabel('Y')
ylabel('X')

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

for i=1:nx
    for j=1:ny
```

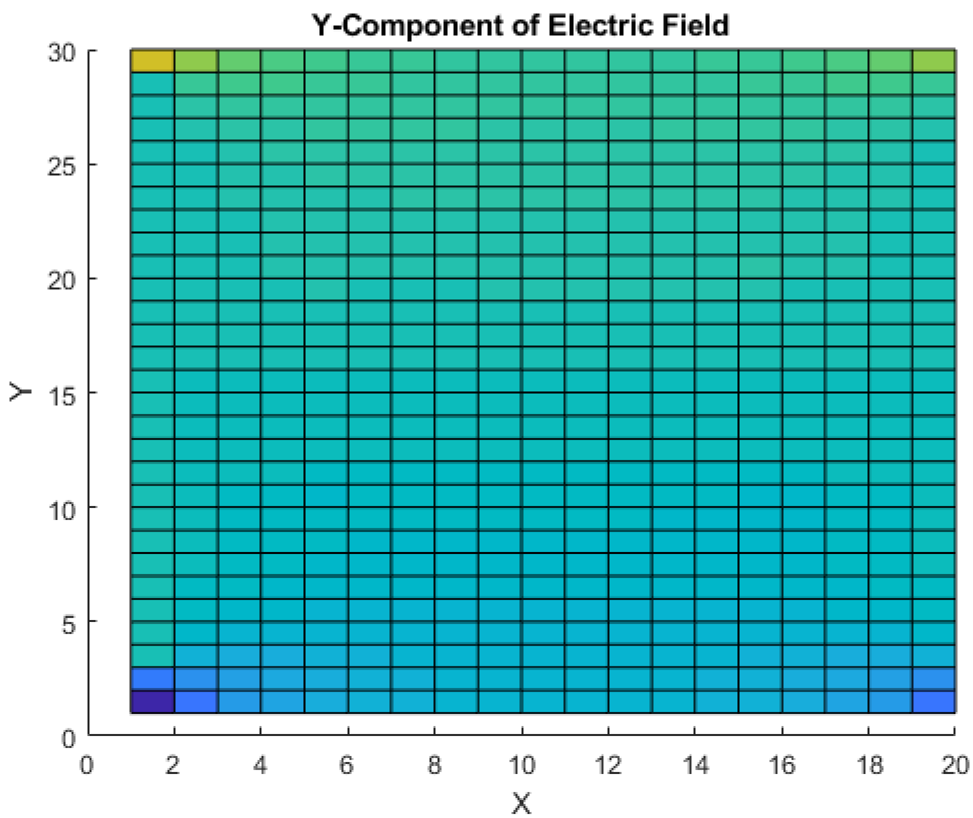
```
        n = j + (i-1)*ny; % Node mapping

        if i == 1 % Left
            G(n,:) = 0;
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary

            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                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) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            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) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
```

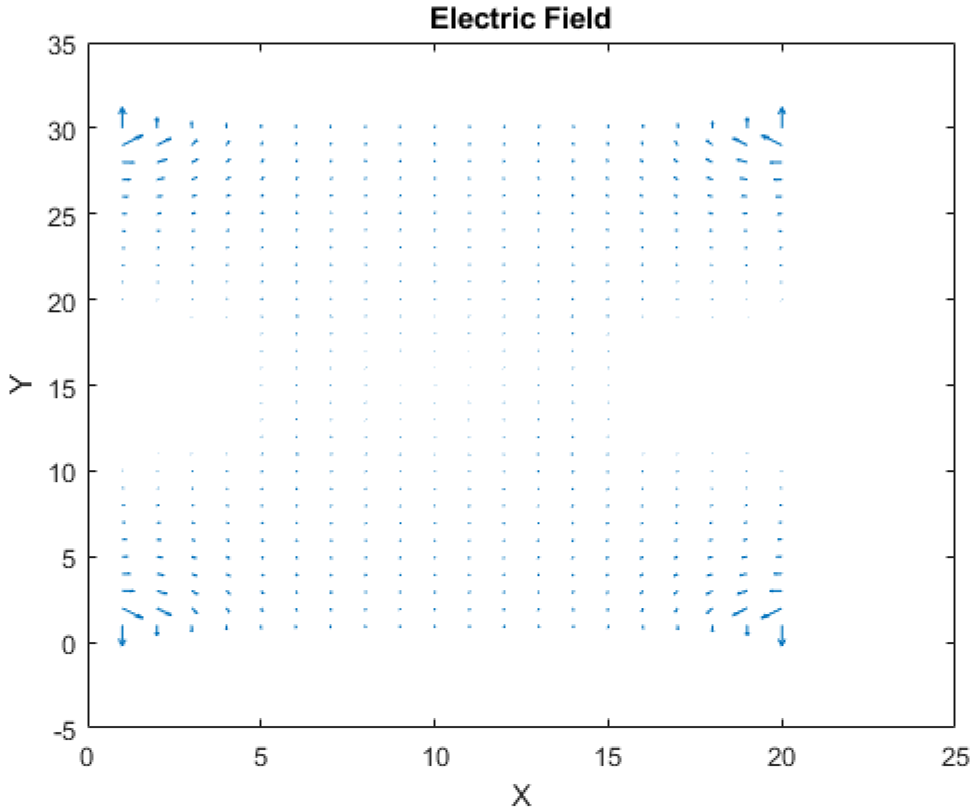
```
sMap3 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap3(i,j) = phi_vec(n);
    end
end

figure(4)
surf(sMap3)
title('Voltages Across Surface')
xlabel('X');
ylabel('Y');

% Electric Field
[Ex, Ey] = gradient(sMap3);

figure(5)
surface(Ex)
title('X-Component of Electric Field')
xlabel('X');
ylabel('Y');

figure(6)
surface(Ey)
title('Y-Component of Electric Field')
xlabel('X');
ylabel('Y');

figure(7)
quiver(Ex,Ey)
title('Electric Field')
xlabel('X');
ylabel('Y');

% Current Density
Jx = Ex.*sigma;
Jy = Ey.*sigma;

figure(8)
quiver(Jx,Jy)
title('Current Density')
xlabel('X');
ylabel('Y');
```

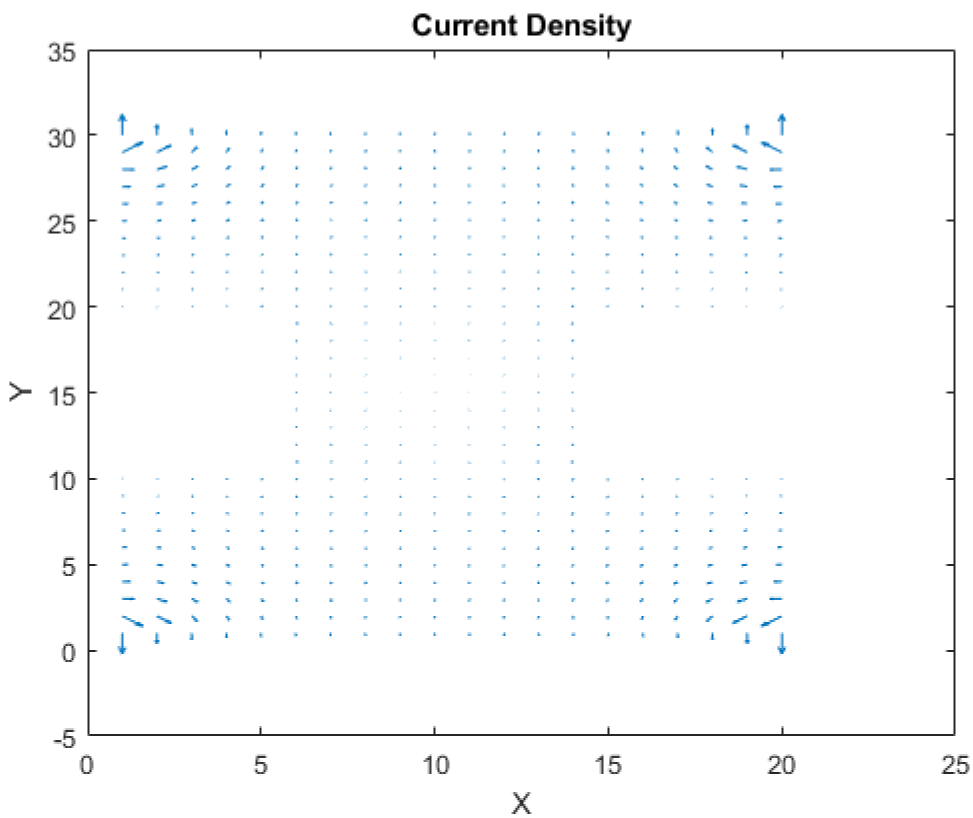

Conductivity Plot
20
15
10
5
0
-5
-10
Y
0
10
20
30
X
6
4
2
0


Voltages Across Surface
5
4
3
2
1
0
30
20
10
0
Y
0
5
10
15
20
X


X-Component of Electric Field
Y
X
0
5
10
15
20
25
30
0
2
4
6
8
10
12
14
16
18
20


Y-Component of Electric Field
Y
X
0
5
10
15
20
25
30
0
2
4
6
8
10
12
14
16
18
20


Electric Field
35
30
25
20
15
10
5
0
-5
0
5
10
15
20
25
Y
X


Current Density
35
30
25
20
15
10
5
0
-5
0
5
10
15
20
25
Y
X

Part 2: c)

Create different bottle necks (4) and graph current mesh

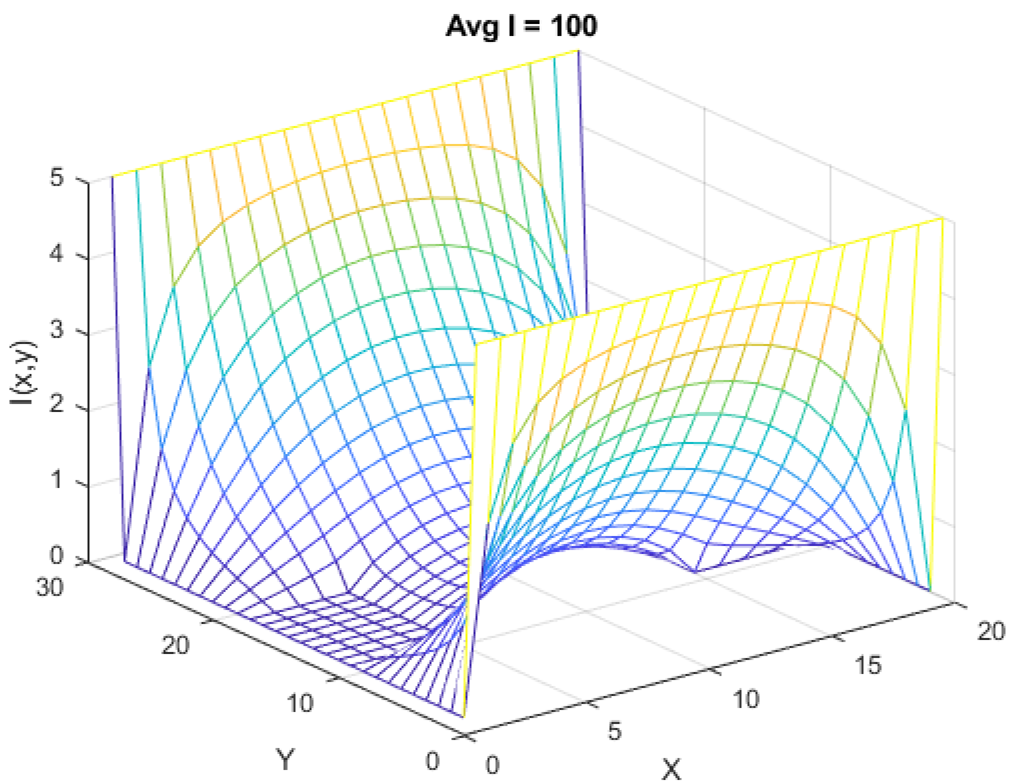
```
% Current in Bottle Neck
R = sigma;
for i=1:nx
    for j=1:ny
        if sigma(i,j) == sig2
            R(i,j) = 1/sig2;
        end
    end
end

current = sMap3./R;
C0 = sum(current(1,:));
C1 = sum(current(L,:));
C = (C0 + C1)/2;
figure(9)
mesh(current);
xlabel('X');
ylabel('Y');
zlabel('I(x,y)');
title(['Avg I = ', num2str(C)]);

% Adjust bottle neck
Wb = 8;
Lb = 10;

top_box = [(nx/2)-(Lb/2) 0 Lb Wb];
bottom_box = [(nx/2)-(Lb/2) ny-Wb Lb Wb];

% Populate sigma matrix
sigma = ones(nx,ny);
for i=1:nx
    for j=1:ny
        if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2)))
            sigma(i,j) = sig2;
        end
    end
end

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

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

        if i == 1 % Left
            G(n,:) = 0;
```

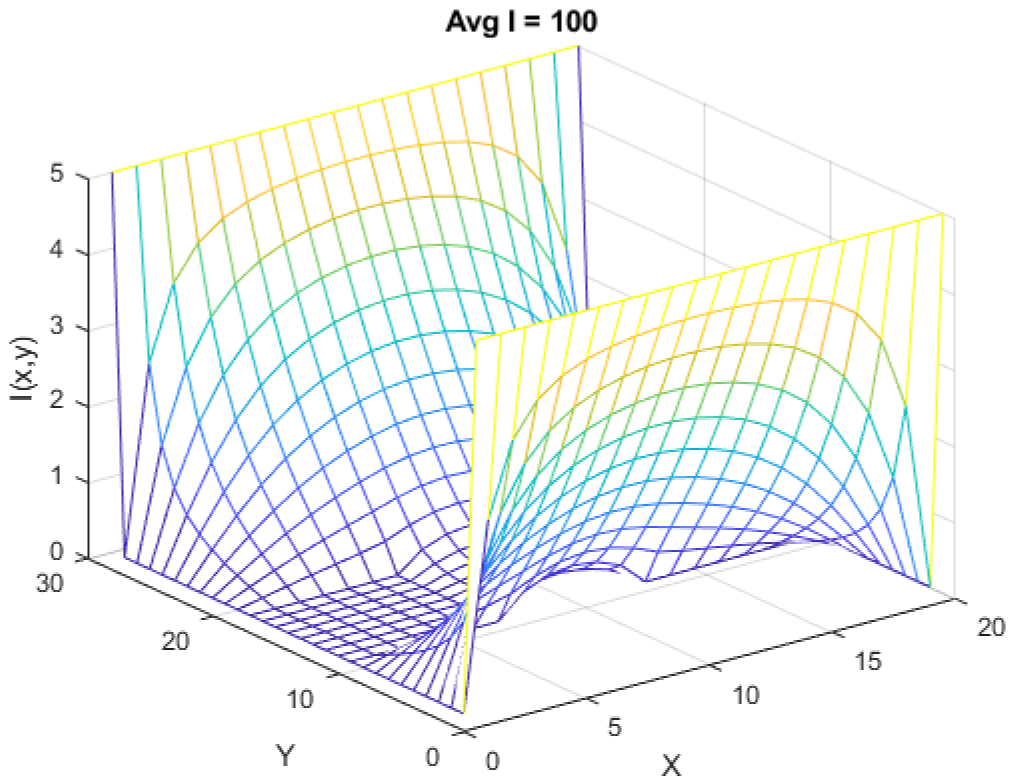
```
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary

            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                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) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            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) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
sMap4 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
```

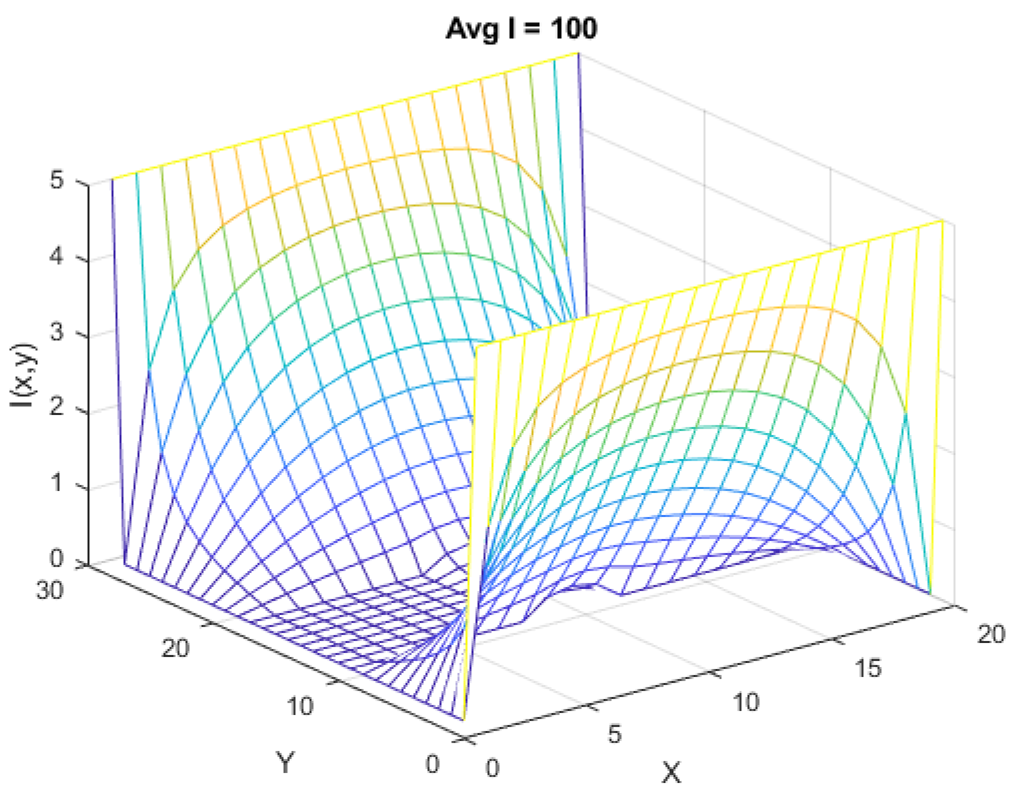
```
        for j=1:ny
            n = j + (i-1)*ny;
            sMap4(i,j) = phi_vec(n);
        end
    end

    R = sigma;
    for i=1:nx
        for j=1:ny
            if sigma(i,j) == sig2
                R(i,j) = 1/sig2;
            end
        end
    end

    current = sMap4./R;
    figure(10)
    mesh(current);
    xlabel('X');
    ylabel('Y');
    zlabel('I(x,y)');
    title(['Avg I = ', num2str(C)]);

    % Adjust bottle neck
    Wb = 9;
    Lb = 10;

    top_box = [(nx/2)-(Lb/2) 0 Lb Wb];
    bottom_box = [(nx/2)-(Lb/2) ny-Wb Lb Wb];

    % Populate sigma matrix
    sigma = ones(nx,ny);
    for i=1:nx
        for j=1:ny
            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
     bottom_box(4) || j > bottom_box(2)))
                sigma(i,j) = sig2;
            end
        end
    end

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

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

            if i == 1 % Left
                G(n,:) = 0;
                G(n,n) = 1;

                B(n) = V0;
```

```
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary

            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                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) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            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) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
sMap5 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap5(i,j) = phi_vec(n);
```

```
    end
end

R = sigma;
for i=1:nx
    for j=1:ny
        if sigma(i,j) == sig2
            R(i,j) = 1/sig2;
        end
    end
end

current = sMap5./R;
figure(11)
mesh(current);
xlabel('X');
ylabel('Y');
zlabel('I(x,y)');
title(['Avg I = ', num2str(C)]);

% Decreasing the width of the bottleneck seems to have no affect on
 the
% total current in the surface, as each iteration of different
 bottleneck
% width produces the same total current.
```




Avg I = 100
I(x,y)
0
1
2
3
4
5
30
20
10
0
Y
0
5
10
15
20
X


Avg I = 100
I(x,y)
0
1
2
3
4
5
30
20
10
0
Y
0
5
10
15
20
X

Part 2: d)

Change sigma, calculate current, graph mesh current

```
% Adjust bottle neck
Wb = 5;
Lb = 10;

top_box = [(nx/2)-(Lb/2) 0 Lb Wb];
bottom_box = [(nx/2)-(Lb/2) ny-Wb Lb Wb];

% Populate sigma matrix
sigma = ones(nx,ny);
sig2 = 1e-3;
sig1 = 1;
for i=1:nx
    for j=1:ny
        if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2)))
            sigma(i,j) = sig2;
        end
    end
end

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

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

        if i == 1 % Left
            G(n,:) = 0;
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary
```

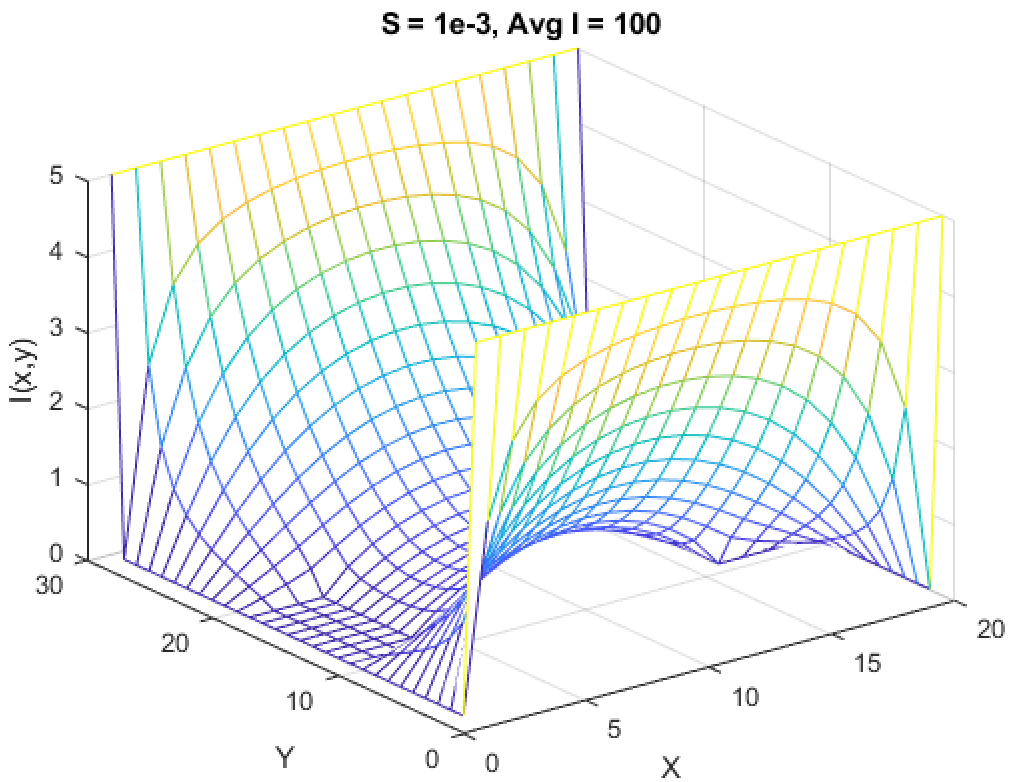
```
            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                nxm = j + (i-2)*ny;
                nxp = j +(i)*ny;
                nym = j-1 + (i-1)*ny;
                nyp = j+1 + (i-1)*ny;

                G(n,n) = -4;
                G(n, nxm) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            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) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
sMap5 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap5(i,j) = phi_vec(n);
    end
end

R = sigma;
for i=1:nx
    for j=1:ny
        if sigma(i,j) == sig2
            R(i,j) = 1/sig2;
        end
    end
end

current = sMap5./R;
figure(12)
mesh(current);
xlabel('X');
```

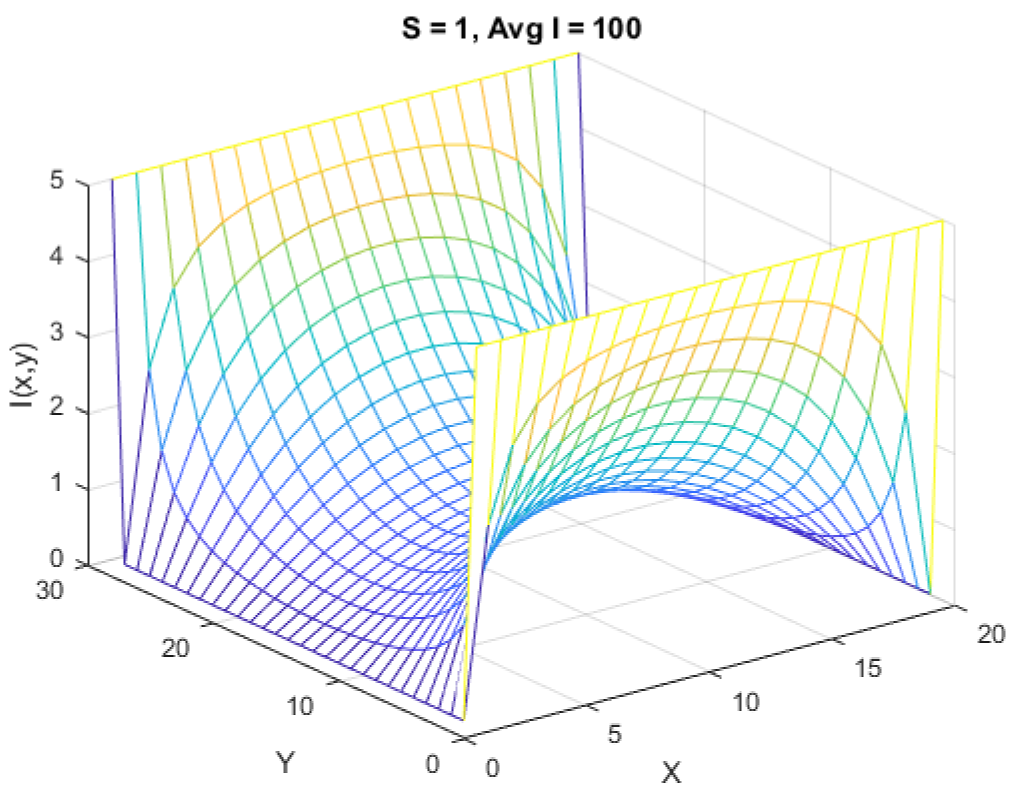
```
ylabel('Y');
zlabel('I(x,y)');
title(['S = 1e-3, Avg I = ', num2str(C)]);

% Populate sigma matrix
sigma = ones(nx,ny);
sig2 = 1;
sig1 = 1;
for i=1:nx
    for j=1:ny
        if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2)))
            sigma(i,j) = sig2;
        end
    end
end

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

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

        if i == 1 % Left
            G(n,:) = 0;
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary

            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                nxm = j + (i-2)*ny;
                nxp = j +(i)*ny;
                nym = j-1 + (i-1)*ny;
                nyp = j+1 + (i-1)*ny;
```

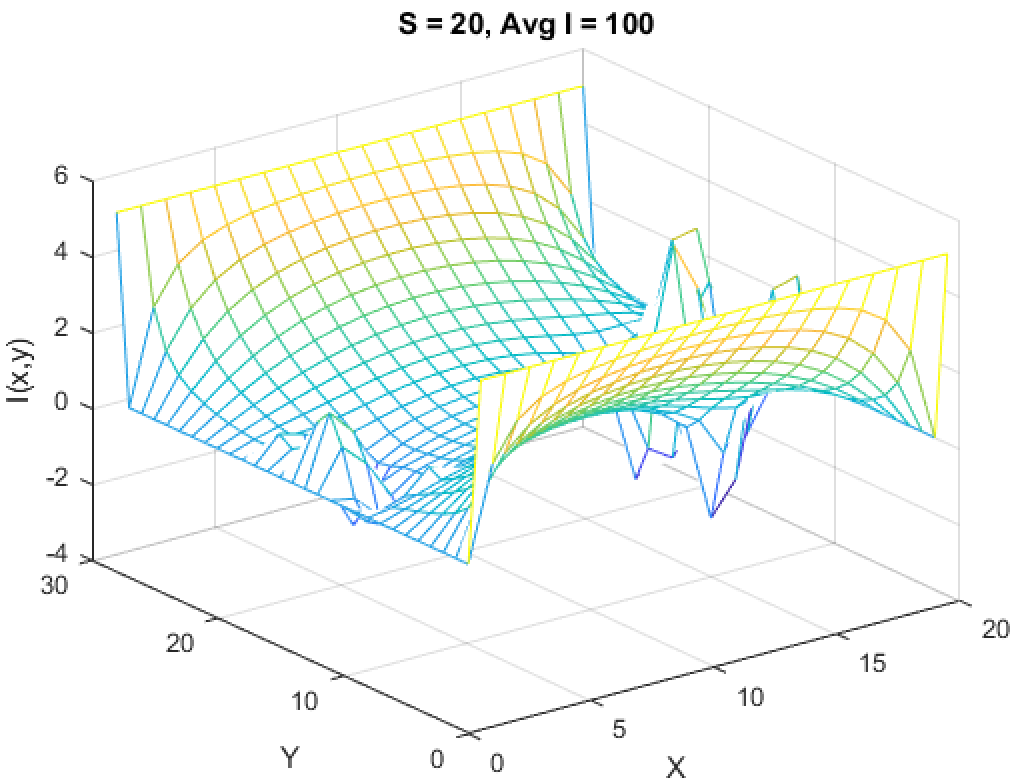
```
                G(n,n) = -4;
                G(n, nxm) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            else
                nxm = j + (i-2)*ny;
                nxp = j +(i)*ny;
                nym = j-1 + (i-1)*ny;
                nyp = j+1 + (i-1)*ny;

                G(n,n) = -4;
                G(n, nxm) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
sMap5 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap5(i,j) = phi_vec(n);
    end
end

R = sigma;
for i=1:nx
    for j=1:ny
        if sigma(i,j) == sig2
            R(i,j) = 1/sig2;
        end
    end
end

current = sMap5./R;
figure(13)
mesh(current);
xlabel('X');
ylabel('Y');
zlabel('I(x,y)');
title(['S = 1, Avg I = ', num2str(C)]);

% Populate sigma matrix
sigma = ones(nx,ny);
sig2 = 2;
sig1 = 1;
```

```
for i=1:nx
    for j=1:ny
        if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2)))
            sigma(i,j) = sig2;
        end
    end
end

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

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

        if i == 1 % Left
            G(n,:) = 0;
            G(n,n) = 1;

            B(n) = V0;
        elseif i == nx % Right
            G(n,:) = 0;
            G(n,n) = 1;

            B(n,1) = V0;
        elseif j == 1 % Bottom
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        elseif j == ny % Top
                G(n,:) = 0;
                G(n,n) = 1;

                B(n,1) = 0;
        else % Not along any boundary

            if (i > top_box(1) && i < top_box(1)+top_box(3) && (j <
 bottom_box(4) || j > bottom_box(2))) % Inside box
                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) = sig2;
                G(n, nxp) = sig2;
                G(n, nym) = sig2;
                G(n, nyp) = sig2;
            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) = sig1;
                G(n, nxp) = sig1;
                G(n, nym) = sig1;
                G(n, nyp) = sig1;
            end

        end
    end
end

phi_vec = G\B;
sMap5 = zeros(nx,ny);

% Map voltages back on to surface map
for i=1:nx
    for j=1:ny
        n = j + (i-1)*ny;
        sMap5(i,j) = phi_vec(n);
    end
end

R = sigma;
for i=1:nx
    for j=1:ny
        if sigma(i,j) == sig2
            R(i,j) = 1/sig2;
        end
    end
end

current = sMap5./R;
figure(14)
mesh(current);
xlabel('X');
ylabel('Y');
zlabel('I(x,y)');
title(['S = 20, Avg I = ', num2str(C)]);

% Increasing the conductivity in the barrier region increases the
 total
% current in that region. However, in every iteration of different
% conductivity values, the average current does not change.
```


S = 1e-3, Avg I = 100
I(x,y)
0
1
2
3
4
5
30
20
10
0
Y
0
5
10
15
20
X


S = 1, Avg I = 100
I(x,y)
0
1
2
3
4
5
30
20
10
0
Y
0
5
10
15
20
X



Published with MATLAB® R2017b