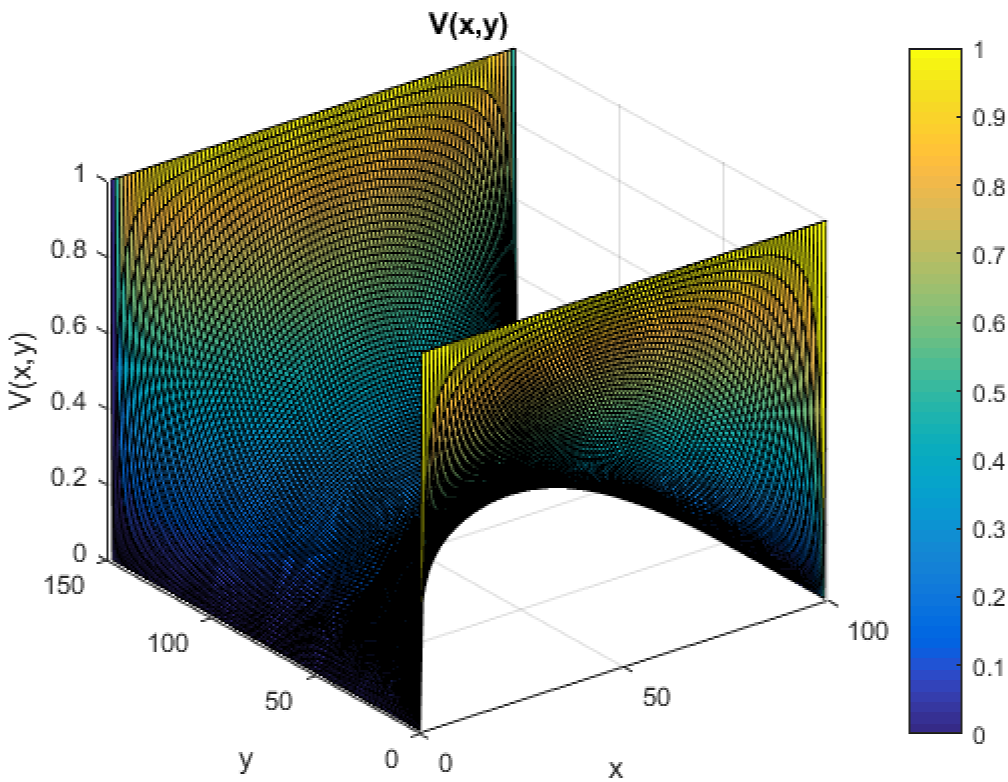
Table of Contents


```
% Assignment 2 Finite Difference Method
% Chantel Lepage 100999893
close all
clear all
```

Part 1 A

```
L=150;
W=100;


G=sparse(L*W,L*W);
V=zeros(L*W,1);

for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        nxm = j+(i-2)*W;
        nxp = j+(i)*W;
        nym = (j-1)+(i-1)*W;
        nyp = (j+1)+(i-1)*W;
        if i==1  %left edge
            G(n,n)=1;
            V(n)=1;
        elseif i==L %right edge
            G(n,n)=1;
            V(n)=1;
        elseif j==W %top edge
            G(n,n)=-3;
        elseif j==1 %bottom edge
            G(n,n)=-3;
        else   %inside parts
            G(n,n) = -4;
            G(n,nxm)= 1;
            G(n,nxp) = 1;
            G(n,nym) = 1;
            G(n,nyp) = 1;
        end
```

```
    end
end

phiVec = G\V;

phi=zeros(L,W);

for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        phi(i,j)= phiVec(n);
    end
end

figure(1)
surf(phi)
colorbar
xlabel('x');
ylabel('y');
zlabel('V(x,y)');
title('V(x,y)');
```

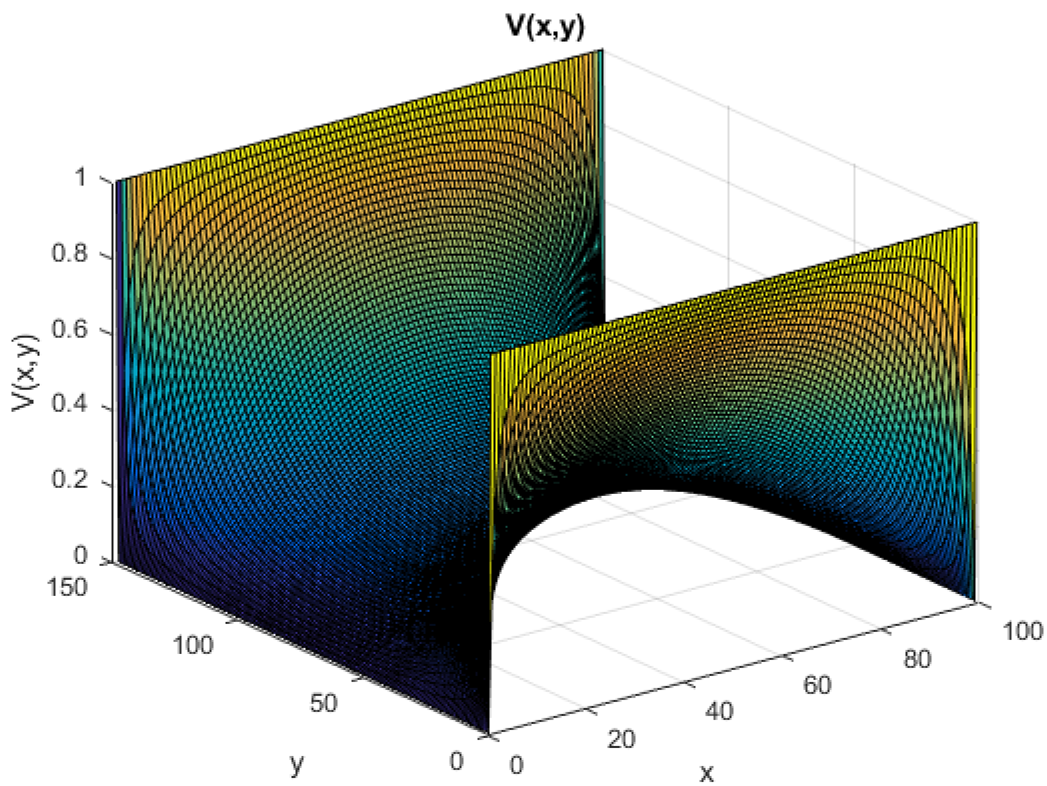


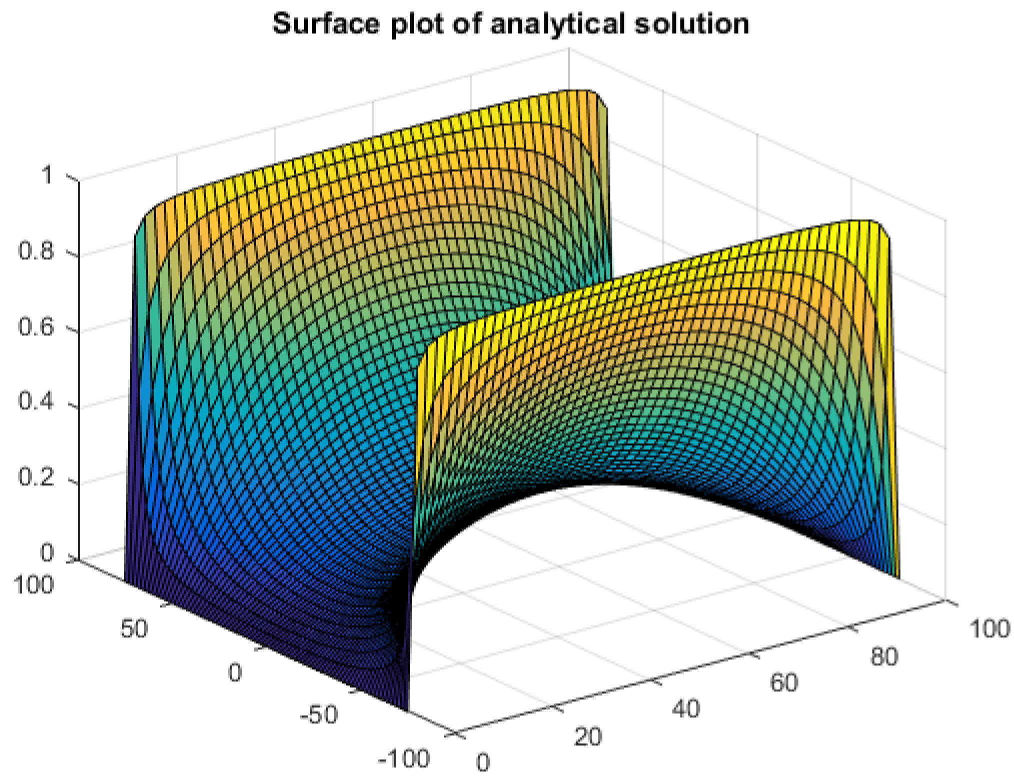
Part 1 B

```
L=150;
W=L*2/3;
```

```
G=sparse(L*W,L*W);
V=zeros(L*W,1);

for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        nxm = j+(i-2)*W;
        nxp = j+(i)*W;
        nym = (j-1)+(i-1)*W;
        nyp = (j+1)+(i-1)*W;
        if i==1  %left edge
            G(n,n)=1;
            V(n)=1;
        elseif i==L %right edge
            G(n,n)=1;
            V(n)=0;
        elseif j==W %top edge
            G(n,n)=-3;
            G(n,nxp)=1;
            G(n,nxm)=1;
            G(n,nym)=1;
        elseif j==1 %bottom edge
            G(n,n)=-3;
            G(n,nxp) = 1;
            G(n,nyp) = 1;
            G(n,nxm) = 1;
        else   %inside parts
            G(n,n) = -4;
            G(n,nxm)= 1;
            G(n,nxp) = 1;
            G(n,nym) = 1;
            G(n,nyp) = 1;
        end
    end
end

for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        phi(i,j)= phiVec(n);
    end
end

figure(2)
surf(phi)
pause(0.01)
xlabel('x');
ylabel('y');
zlabel('V(x,y)');
title('V(x,y)');


[x,y]=meshgrid(-75:2:75,0:2:100);
a=100;
```

```
b=75;
V=0;

for k=1:100 %using k instead of n to aviod confusion
    if rem(k,2)==1
        V=V+(4/pi)*(cosh(k*pi*x/a).*sin(k*pi*y/a)) ./ (k*cosh(k*pi*b/
a));
        figure(3)
        surf(y,x,V)
        title('Surface plot of analytical solution')
        pause(0.01)
    end
end
```





Part 1 Conclusions

For part a the linear output shows us that the voltage changes linearly in x and constant in y. These results make sense as the conduction was distributed evenly in this example. With the numerical solutions the advatage with it is that it runs very quickly; however, it can take up a lot of space due to the sizes of matrices thtat are being used. With analyitcal soloutions it does not require large amounts of space; however, they can take more time to run.

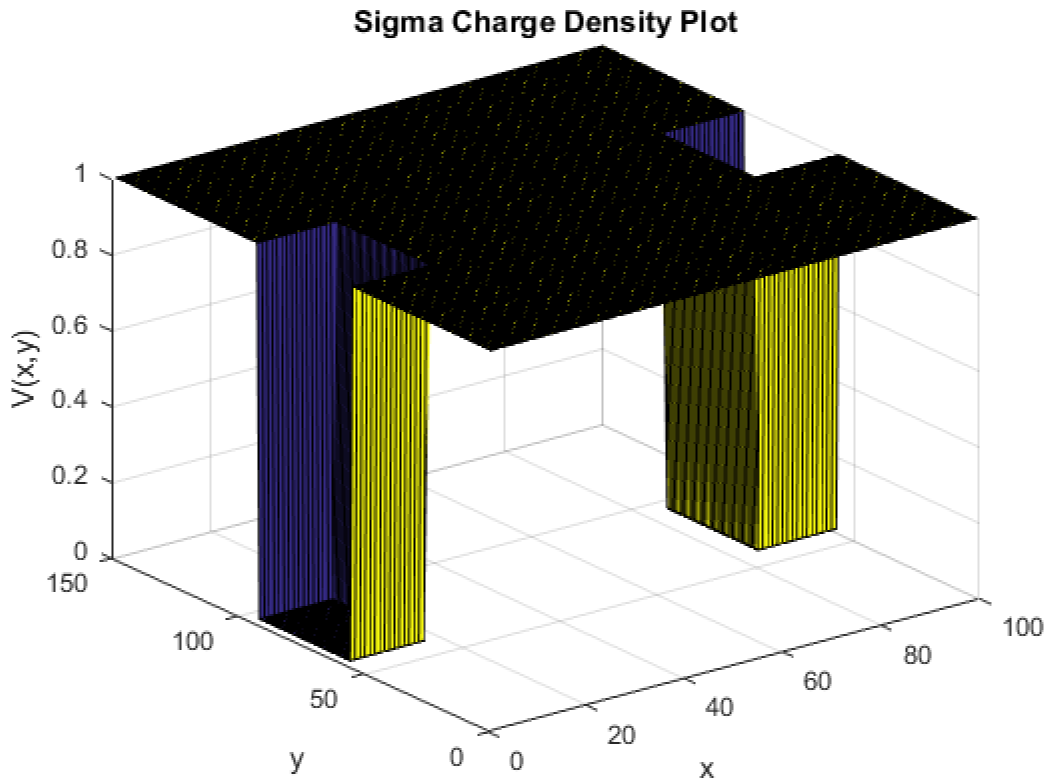
Part 2 A

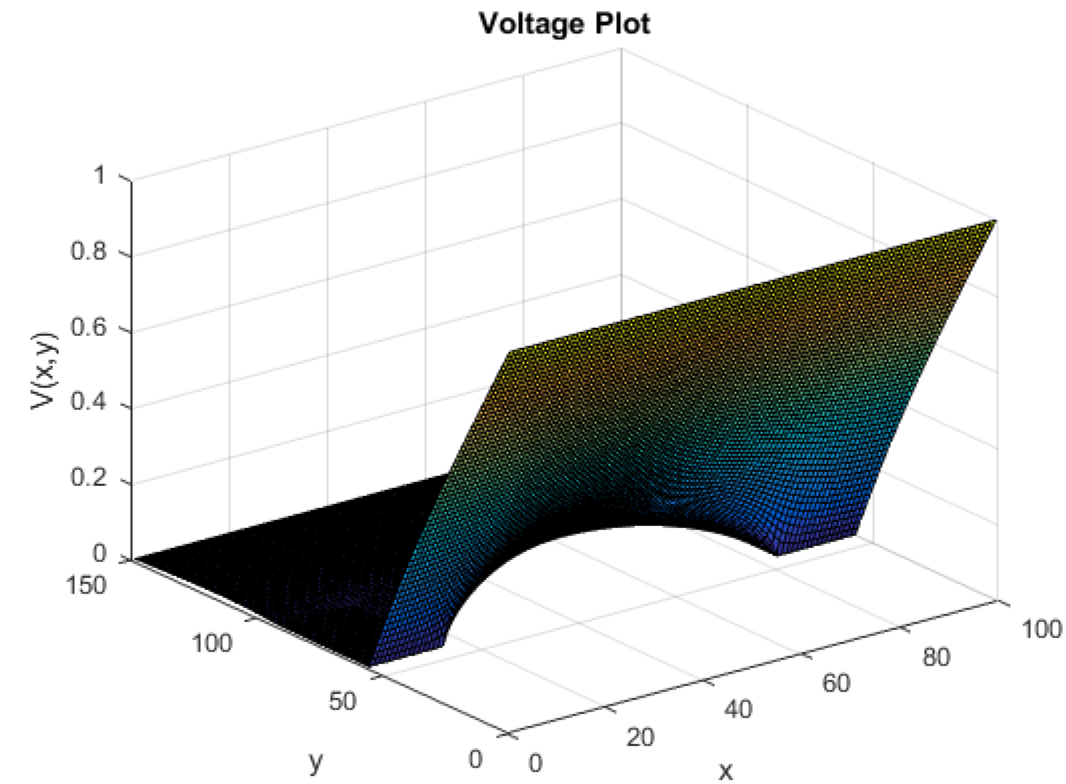
```
L=150;
W=100;

G=sparse(L*W,L*W);
V=zeros(L*W,1);

sigmaOut=1;
sigmaIn=1e-2;

midX = L/2;
midY = W/2;
boxL = L/4;
boxW = W*2/3;
leftEdge = midX - boxL/2;
rightEdge = midX + boxL/2;
```

```
topEdge = midY + boxW/2;
bottomEdge = midY - boxW/2;



for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        nxm = j+(i-2)*W;
        nxp = j+(i)*W;
        nym = (j-1)+(i-1)*W;
        nyp = (j+1)+(i-1)*W;
          if i == 1
            G(n,n) = 1;
            V(n) = 1;
            sigmaMap(i,j) = sigmaOut;
        elseif i == L
            G(n,n) = 1;
            V(n) = 0;
            sigmaMap(i,j) = sigmaOut;
        elseif (j == W)
            G(n,n) = -3;
            if(i>leftEdge && i<rightEdge)
                G(n,nxm) = sigmaIn;
                G(n,nxp) = sigmaIn;
                G(n,nym) = sigmaIn;
                sigmaMap(i,j) = sigmaIn;
            else
                G(n,nxm) = sigmaOut;
                G(n,nxp) = sigmaOut;
                G(n,nym) = sigmaOut;
                sigmaMap(i,j) = sigmaOut;
            end
        elseif (j == 1)
            G(n,n) = -3;
            if(i>leftEdge && i<rightEdge)
                G(n,nxm) = sigmaIn;
                G(n,nxp) = sigmaIn;
                G(n,nyp) = sigmaIn;
                sigmaMap(i,j) = sigmaIn;
            else
                G(n,nxm) = sigmaOut;
                G(n,nxp) = sigmaOut;
                G(n,nyp) = sigmaOut;
                sigmaMap(i,j) = sigmaOut;
            end
        else
            G(n,n) = -4;
            if( (j>topEdge || j<bottomEdge) && i>leftEdge &&
 i<rightEdge)
                G(n,nxp) = sigmaIn;
                G(n,nxm) = sigmaIn;
                G(n,nyp) = sigmaIn;
                G(n,nym) = sigmaIn;
```

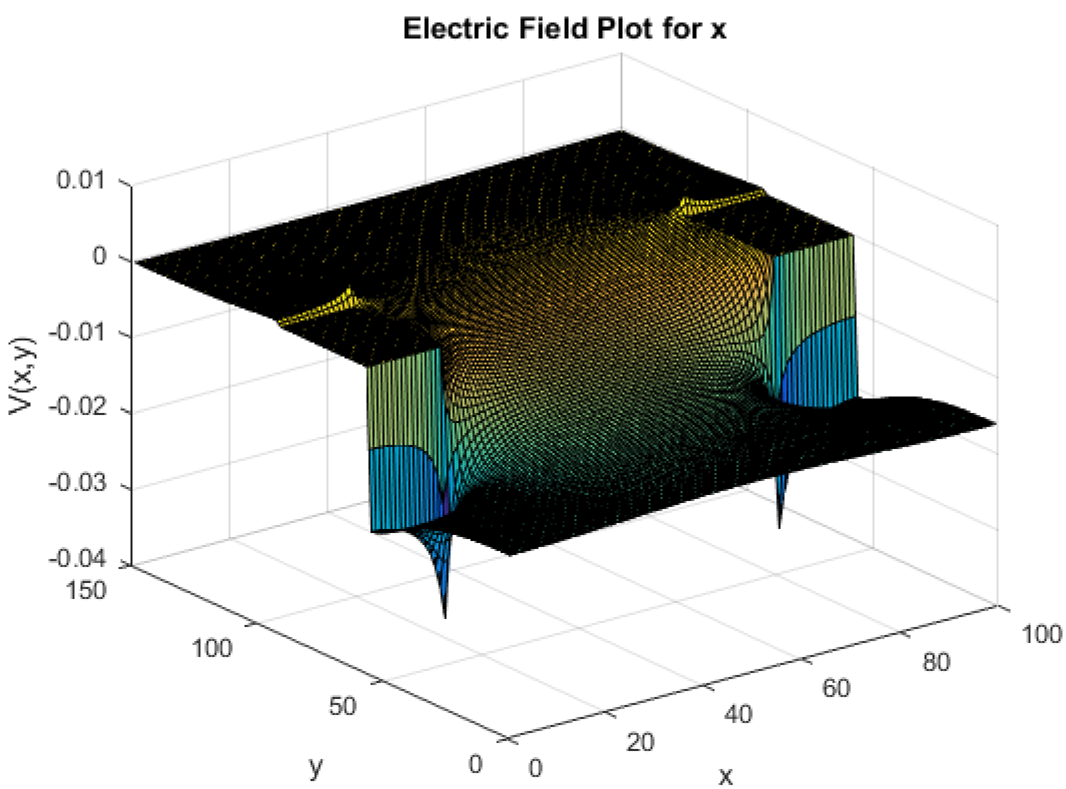
```
                sigmaMap(i,j) = sigmaIn;
            else
                G(n,nxp) = sigmaOut;
                G(n,nxm) = sigmaOut;
                G(n,nyp) = sigmaOut;
                G(n,nym) = sigmaOut;
                sigmaMap(i,j) = sigmaOut;
            end
        end
    end
end

phiVec = G\V;
phi=zeros(L,W);

for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        phi(i,j)= phiVec(n);
    end
end

[Ey,Ex] = gradient(phi);
E = gradient(phi);
J = sigmaMap.* E;

figure(4)
surf(sigmaMap)
xlabel('x');
ylabel('y');
zlabel('V(x,y)')
title('Sigma Charge Density Plot');

figure(5)
surf(phi)
xlabel('x');
ylabel('y');
zlabel('V(x,y)')
title('Voltage Plot');

figure(6)
surf(Ex)
xlabel('x');
ylabel('y');
zlabel('V(x,y)')
title('Electric Field Plot for x');

figure(7)
surf(Ey)
xlabel('x');
ylabel('y');
zlabel('V(x,y)')
title('Electric Field Plot for y');
```

```
figure(8)
surf(J)
xlabel('x');
ylabel('y');
zlabel('V(x,y)')
title('Current Density');
```

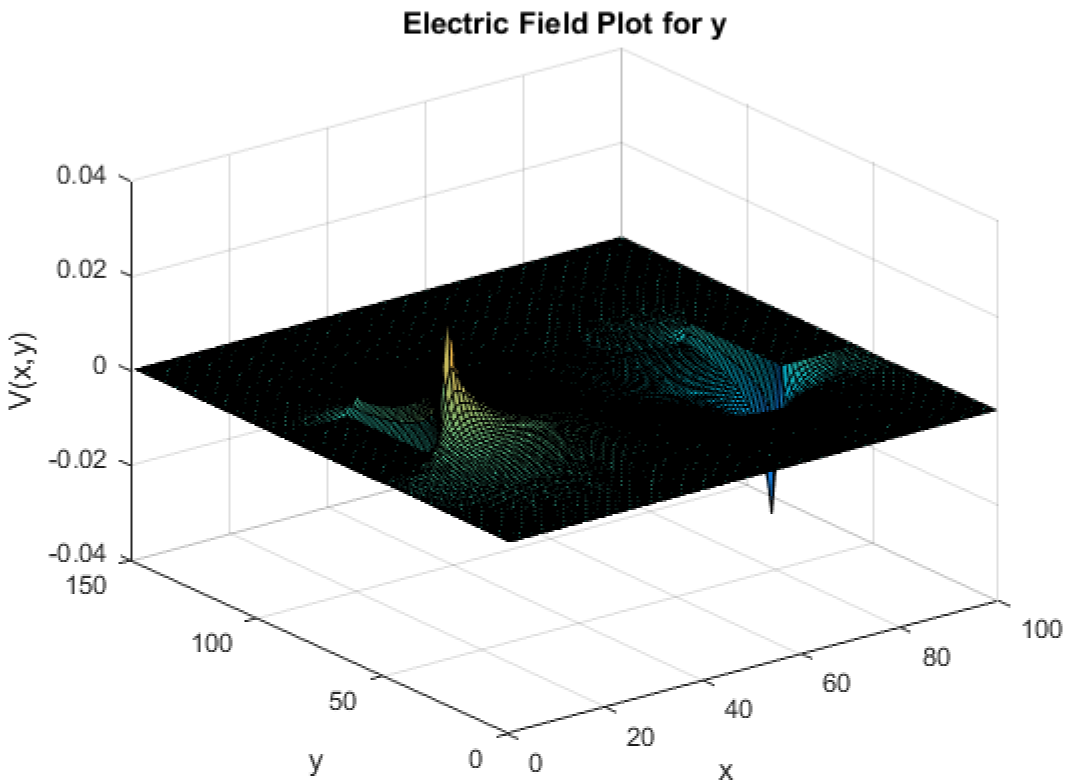




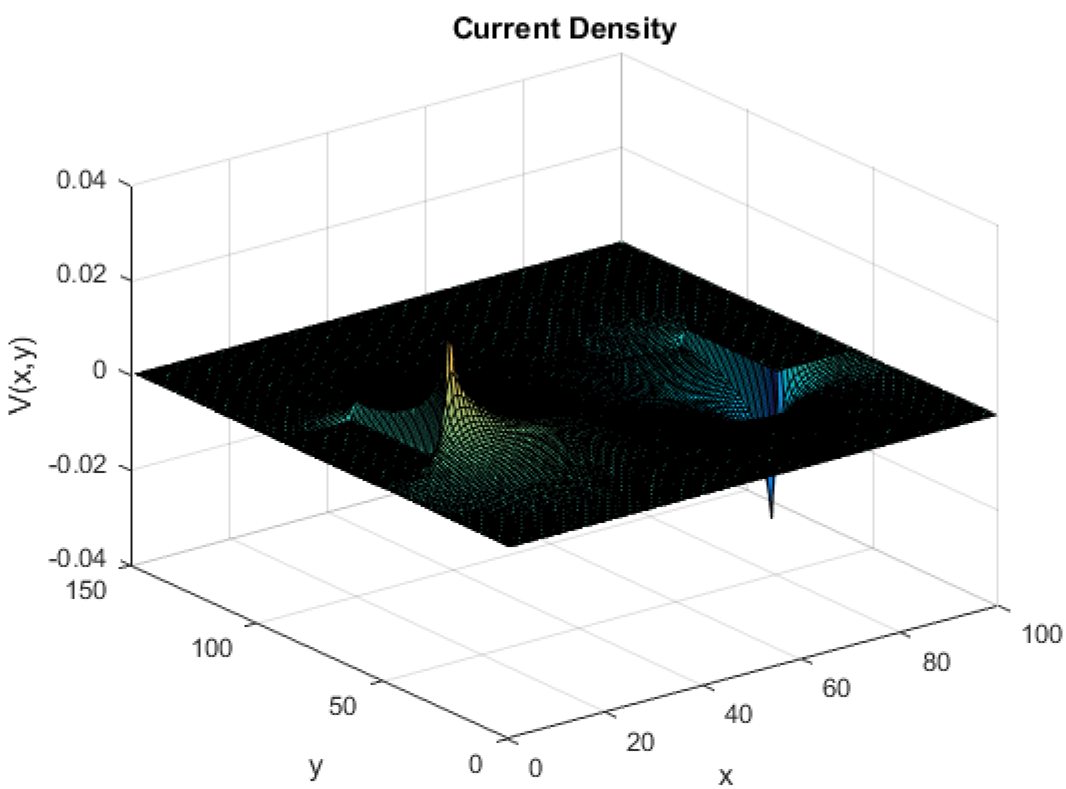
Voltage Plot
V(x,y)
1
0.8
0.6
0.4
0.2
0
150
100
50
0
y
0
20
40
60
80
100
x



Electric Field Plot for x
V(x,y)
0.01
0
-0.01
-0.02
-0.03
-0.04
150
100
50
0
y
0
20
40
60
80
100
x



Electric Field Plot for y
V(x,y)
0.04
0.02
0
-0.02
-0.04
150
100
50
0
y
0
20
40
60
80
100
x



Current Density
V(x,y)
0.04
0.02
0
-0.02
-0.04
150
100
50
0
y
0
20
40
60
80
100
x

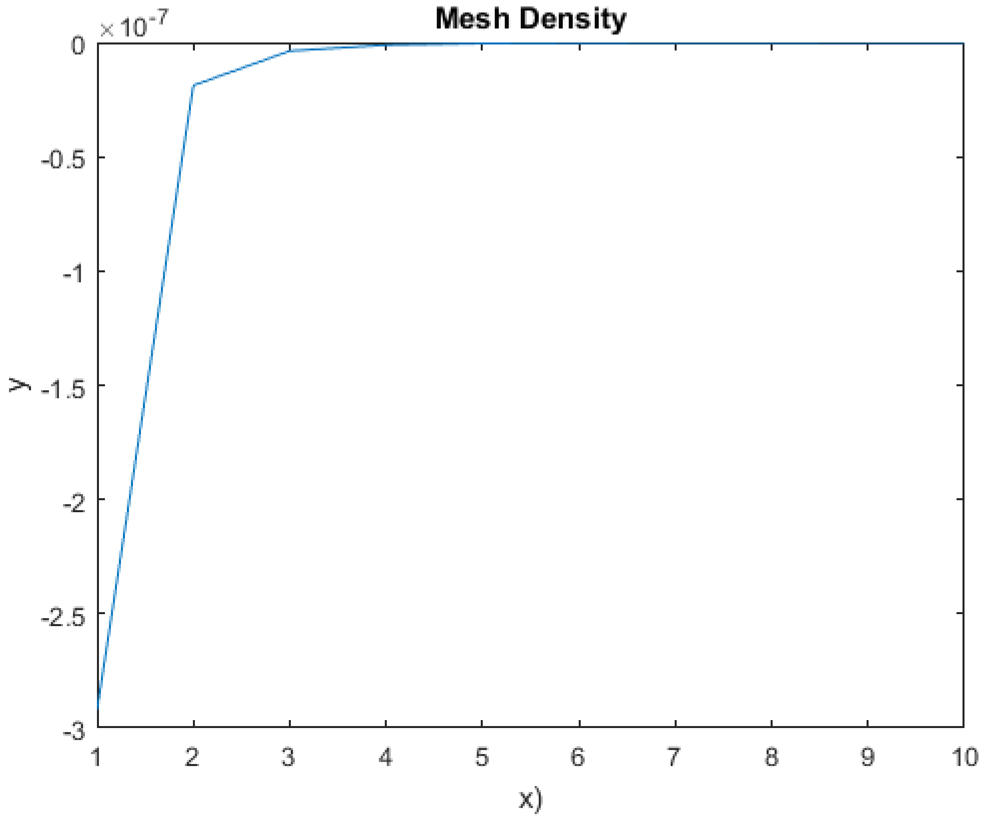
Part 2 B

```
I = zeros(1,10);


for k =1:10
    L = 30;
    W = 20;
    G = sparse(L*W,L*W);
    V= zeros(L*W,1);

    fprintf('%i\n',k)

    scale=k;

    scaleA = 1/scale;
    scaleB = scaleA^2;
    scaleL = L*scale;
    scaleW = W*scale;

    sigmaMap=zeros(scaleL,scaleW);

    sigmaOut = 1;
    sigmaIn = 1e-2;

    midX = L/2;
    midY = W/2;
    boxL = L/4;
    boxW = W*2/3;
    leftEdge = midX - boxL/2;
    rightEdge = midX + boxL/2;
    topEdge = midY + boxW/2;
    bottomEdge = midY - boxW/2;

    for i = 1:scaleL
        for j = 1:scaleW
            n = j + (i-1)*scaleW;
            nxm = j+(i-2)*scaleW;
            nxp = j+i*scaleW;
            nyp = j+1+ (i-1)*scaleW;
            nym = j-1+ (i-1)*scaleW;
            if i == 1
                G(n,n) = 1/scaleB;
                V(n) = 1;
                sigmaMap(i,j) = sigmaOut;
            elseif i == scaleL
                G(n,n) = 1/scaleB;
                V(n) = 0;
                sigmaMap(i,j) = sigmaOut;
            elseif (j == scaleW)
                G(n,n) = -3/scaleB;
                if(i/scale>leftEdge && i/scale<rightEdge)
                    G(n,nxm) = sigmaIn/scaleB;
```

```
                    G(n,nxp) = sigmaIn/scaleB;
                    G(n,nym) = sigmaIn/scaleB;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxm) = sigmaOut/scaleB;
                    G(n,nxp) = sigmaOut/scaleB;
                    G(n,nym) = sigmaOut/scaleB;
                    sigmaMap(i,j) = sigmaOut;
                end
            elseif (j == 1)
                G(n,n) = -3/scaleB;
                if(i/scale>leftEdge && i/scale<rightEdge)
                    G(n,nxm) = sigmaIn/scaleB;
                    G(n,nxp) = sigmaIn/scaleB;
                    G(n,nyp) = sigmaIn/scaleB;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxm) = sigmaOut/scaleB;
                    G(n,nxp) = sigmaOut/scaleB;
                    G(n,nyp) = sigmaOut/scaleB;
                    sigmaMap(i,j) = sigmaOut;
                end
            else
                G(n,n) = -4/scaleB;
                if( (j/scale>topEdge || j/scale<bottomEdge) && i/
scale>leftEdge && i/scale<rightEdge)
                    G(n,nxp) = sigmaIn/scaleB;
                    G(n,nxm) = sigmaIn/scaleB;
                    G(n,nyp) = sigmaIn/scaleB;
                    G(n,nym) = sigmaIn/scaleB;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxp) = sigmaOut/scaleB;
                    G(n,nxm) = sigmaOut/scaleB;
                    G(n,nyp) = sigmaOut/scaleB;
                    G(n,nym) = sigmaOut/scaleB;
                    sigmaMap(i,j) = sigmaOut;
                end
            end
        end
    end

    phiVec = G\V;
    phi=zeros(scaleL,scaleW);

    for i=1:scaleL
        for j=1:scaleW
            n=j+(i-1)*scaleW;
            phi(i,j)= phiVec(n);
        end
    end

    [Ey,Ex] = gradient(phi);
    E = gradient(phi);
```

```
    J = sigmaMap.* E;

    region = L*W;
    I(k)= (sum(sum(J))/(scaleL*scaleW))/region;
end

fprintf('done\n')

x=linspace(1,10,10);

figure(9)
plot(x,I);
title('Mesh Density')
xlabel('x)')
ylabel('y')
```

1
2
3
4
5
6
7
8
9
10
done



Part 2 C

```
I = zeros(1,10);

for k =1:10

    bottle=k;

    L=150;
    W=100;

    G=sparse(L*W,L*W);
    V=zeros(L*W,1);

    sigmaOut=1;
    sigmaIn=1e-2;

    midX = L/2;
    midY = W/2;

    boxW = W*2/3;
    spaceW = W - boxW;
    boxL = L/4;
    boxW = spaceW/bottle;
```

```
    leftEdge = midX - boxL/2;
    rightEdge = midX + boxL/2;
    topEdge = midY + boxW/2;
    bottomEdge = midY - boxW/2;



    for i=1:L
        for j=1:W
            n=j+(i-1)*W;
            nxm = j+(i-2)*W;
            nxp = j+(i)*W;
            nym = (j-1)+(i-1)*W;
            nyp = (j+1)+(i-1)*W;
              if i == 1
                G(n,n) = 1;
                V(n) = 1;
                sigmaMap(i,j) = sigmaOut;
            elseif i == L
                G(n,n) = 1;
                V(n) = 0;
                sigmaMap(i,j) = sigmaOut;
            elseif (j == W)
                G(n,n) = -3;
                if(i>leftEdge && i<rightEdge)
                    G(n,nxm) = sigmaIn;
                    G(n,nxp) = sigmaIn;
                    G(n,nym) = sigmaIn;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxm) = sigmaOut;
                    G(n,nxp) = sigmaOut;
                    G(n,nym) = sigmaOut;
                    sigmaMap(i,j) = sigmaOut;
                end
            elseif (j == 1)
                G(n,n) = -3;
                if(i>leftEdge && i<rightEdge)
                    G(n,nxm) = sigmaIn;
                    G(n,nxp) = sigmaIn;
                    G(n,nyp) = sigmaIn;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxm) = sigmaOut;
                    G(n,nxp) = sigmaOut;
                    G(n,nyp) = sigmaOut;
                    sigmaMap(i,j) = sigmaOut;
                end
            else
                G(n,n) = -4;
                if( (j>topEdge || j<bottomEdge) && i>leftEdge &&
i<rightEdge)
                    G(n,nxp) = sigmaIn;
                    G(n,nxm) = sigmaIn;
```

```
                    G(n,nyp) = sigmaIn;
                    G(n,nym) = sigmaIn;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxp) = sigmaOut;
                    G(n,nxm) = sigmaOut;
                    G(n,nyp) = sigmaOut;
                    G(n,nym) = sigmaOut;
                    sigmaMap(i,j) = sigmaOut;
                end
            end
        end
    end

    phiVec = G\V;
    phi=zeros(L,W);

    for i=1:L
        for j=1:W
            n=j+(i-1)*W;
            phi(i,j)= phiVec(n);
        end
    end

    [Ey,Ex] = gradient(phi);
    E = gradient(phi);
    J = sigmaMap.* E;

    region = L*W;
    I(k)= (sum(sum(J))/(L*W))/region;
end

x = 1./linspace(1,10,10);

figure(10)
plot(x,I);
title('Current for Bottl-neck')
xlabel('x)')
ylabel('y')
```

Matrix dimensions must agree.

Error in assign_2_finite_difference_method (line 501)
J = sigmaMap. E;*

Part 2 D

```
I = zeros(1,10);

for k =1:10
    sigma(k) = 1/(k);
    L=150;
    W=100;
```

```
G=sparse(L*W,L*W);
V=zeros(L*W,1);

sigmaOut=1;
sigmaIn=sigma(k);

midX = L/2;
midY = W/2;
boxL = L/4;
boxW = W*2/3;
leftEdge = midX - boxL/2;
rightEdge = midX + boxL/2;
topEdge = midY + boxW/2;
bottomEdge = midY - boxW/2;



for i=1:L
    for j=1:W
        n=j+(i-1)*W;
        nxm = j+(i-2)*W;
        nxp = j+(i)*W;
        nym = (j-1)+(i-1)*W;
        nyp = (j+1)+(i-1)*W;
          if i == 1
            G(n,n) = 1;
            V(n) = 1;
            sigmaMap(i,j) = sigmaOut;
        elseif i == L
            G(n,n) = 1;
            V(n) = 0;
            sigmaMap(i,j) = sigmaOut;
        elseif (j == W)
            G(n,n) = -3;
            if(i>leftEdge && i<rightEdge)
                G(n,nxm) = sigmaIn;
                G(n,nxp) = sigmaIn;
                G(n,nym) = sigmaIn;
                sigmaMap(i,j) = sigmaIn;
            else
                G(n,nxm) = sigmaOut;
                G(n,nxp) = sigmaOut;
                G(n,nym) = sigmaOut;
                sigmaMap(i,j) = sigmaOut;
            end
        elseif (j == 1)
            G(n,n) = -3;
            if(i>leftEdge && i<rightEdge)
                G(n,nxm) = sigmaIn;
                G(n,nxp) = sigmaIn;
                G(n,nyp) = sigmaIn;
                sigmaMap(i,j) = sigmaIn;
            else
```

```
                    G(n,nxm) = sigmaOut;
                    G(n,nxp) = sigmaOut;
                    G(n,nyp) = sigmaOut;
                    sigmaMap(i,j) = sigmaOut;
                end
            else
                G(n,n) = -4;
                if( (j>topEdge || j<bottomEdge) && i>leftEdge &&
 i<rightEdge)
                    G(n,nxp) = sigmaIn;
                    G(n,nxm) = sigmaIn;
                    G(n,nyp) = sigmaIn;
                    G(n,nym) = sigmaIn;
                    sigmaMap(i,j) = sigmaIn;
                else
                    G(n,nxp) = sigmaOut;
                    G(n,nxm) = sigmaOut;
                    G(n,nyp) = sigmaOut;
                    G(n,nym) = sigmaOut;
                    sigmaMap(i,j) = sigmaOut;
                end
            end
        end
    end

    phiVec = G\V;
    phi=zeros(L,W);

    for i=1:L
        for j=1:W
            n=j+(i-1)*W;
            phi(i,j)= phiVec(n);
        end
    end

    [Ey,Ex] = gradient(phi);
    E = gradient(phi);
    J = -sigmaMap.* E;

    region = L*W;
    I(k)= (sum(sum(J))/(L*W))/region;
end


figure (11)
    plot(sigma,I);
    title('Sigma Charge Density')
    xlabel('x)')
    ylabel('y')
```

Published with MATLAB® R2016a