```
%Miranda Heredia
%100996160
clear
close all
```

Question 2 - Part B

Solving the system using incremental sizes of mesh

```
for meshsize = 20:10:200

    %meshsize is replacing Length
    Width = (3/2)*meshsize;
    G = sparse(meshsize*Width);
    F = zeros(1, meshsize*Width);

    sigMap = zeros(Width, meshsize);
    sigOut = 1;
    sigIn = 1e-2;

    %Bottleneck parameters
    %Different approach than in Part A
    Box = [meshsize*2/5 meshsize*3/5 Width*2/5 Width*3/5];

    %Populate the G matrix
    for x=1: meshsize
        for y = 1:Width
             n =  y +(x-1)*Width;    %Mapping equation FD
        %Local mapping of the nodes around (x,y)
        nxm = y+(x-2)*Width;
        nxp = y+(x)*Width;
        nym = (y-1)+(x-1)*Width;
        nyp = (y+1)+(x-1)*Width;

        %Boundaries
        if x==1     %Left BC
            G(n, :) = 0;
            G(n,n) = 1;
            F(n) = 1;
        elseif x==meshsize    %Right BC
            G(n, :) = 0;
            G(n,n) = 1;
            F(n) =0;
        elseif y==Width     %Upper BC
            if x > Box(1) && x < Box(2)


                G(n, n) = -3;
                G(n, n+1) = sigIn;
                G(n, n+Width) = sigIn;
                G(n, n-Width) = sigIn;
```

```
        else

            G(n, n) = -3;
            G(n, n+1) = sigOut;
            G(n, n+Width) = sigOut;
            G(n, n-Width) = sigOut;

        end

    elseif y==1         %Lower BC
         if x > Box(1) && x < Box(2)
            G(n, n) = -3;
            G(n, n+1) = sigIn;
            G(n, n+Width) = sigIn;
            G(n, n-Width) = sigIn;

        else

            G(n, n) = -3;
            G(n, n+1) = sigOut;
            G(n, n+Width) = sigOut;
            G(n, n-Width) = sigOut;
        end
    else
        if  x > Box(1) && x < Box(2) && (y < Box(3)||y > Box(4))
        %Laplacian Equation in Differences
            G(n,n) = -4;
            G(n,nxm)= sigIn;
            G(n,nxp) = sigIn;
            G(n,nym) = sigIn;
            G(n,nyp) = sigIn;

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

            end
        end
    end
end

%checking for bottleneck paraemters but using meshsize
%meshsize replaces length because it is getting incremented
for Length = 1 : meshsize

    for Width = 1 : Width

       %Checkig boundary conditions if inside the bottleneck
        if (Length >= Box(1)) && (Length <= Box(2))
           sigMap(Width, Length) = sigIn;
```

```
            else

                sigMap(Width, Length) = sigOut;

            end
            %Specific case for inside the bottleneck
            if (Length >= Box(1)) && (Length <= Box(2)) && (Width >=
Box(3)) && (Width <= Box(4))

                sigMap(Width, Length) = sigOut;

            end
        end
    end

  SolV = G\F';
  SolVmatrix = zeros(Width, meshsize,1);

    for i = 1:meshsize

        for j = 1:Width

            n = j + (i-1)*Width;
            SolVmatrix(j,i) = SolV(n);

        end
    end
    %electric field found using gradient of voltage
   [Ex, Ey] = gradient(SolVmatrix);

   %current desntiy is sigma times electric field
   J_x = sigMap.*Ex;
   J_y = sigMap.*Ey;
   J = sqrt(J_x.^2 + J_y.^2);

  %plotting current density vs mesh size
   figure(1)
   hold on

   if meshsize == 20

       %Must keep track of old and new current to plot
       Curr = sum(J, 1);
       TotalC = sum(Curr);
       Curr_old = TotalC;
       plot([meshsize, meshsize], [Curr_old, TotalC])

   end
   if meshsize > 20

       Curr_old = TotalC;
       Curr = sum(J, 2);
       TotalC = sum(Curr);
       plot([meshsize-10, meshsize], [Curr_old, TotalC])
```

```
        xlabel("Meshsize")
        ylabel("Current Density")

    end

    title("Mesh Density")

end

%From the plot, the current increases as the mesh density increases.
```

Published with MATLAB® R2019b