Assignment 2

Table of Contents

Question 1: Electrostatic Potential 1
Part 1 - 1D Case 1
Part 2 - 2D Case 2

Matthew Lazarus 100962142

Question 1: Electrostatic Potential

In this question, a grid is set up, with specific boundary conditions. Using the maxtrix form of the problem (GV=F), the electrostatic potential within the region is found and plotted.

```
% Clear all previous variables, figures, etc, to ensure that the
 workspace
% is clean.
clear all
clearvars
clearvars -GLOBAL
close all
```

Part 1 - 1D Case

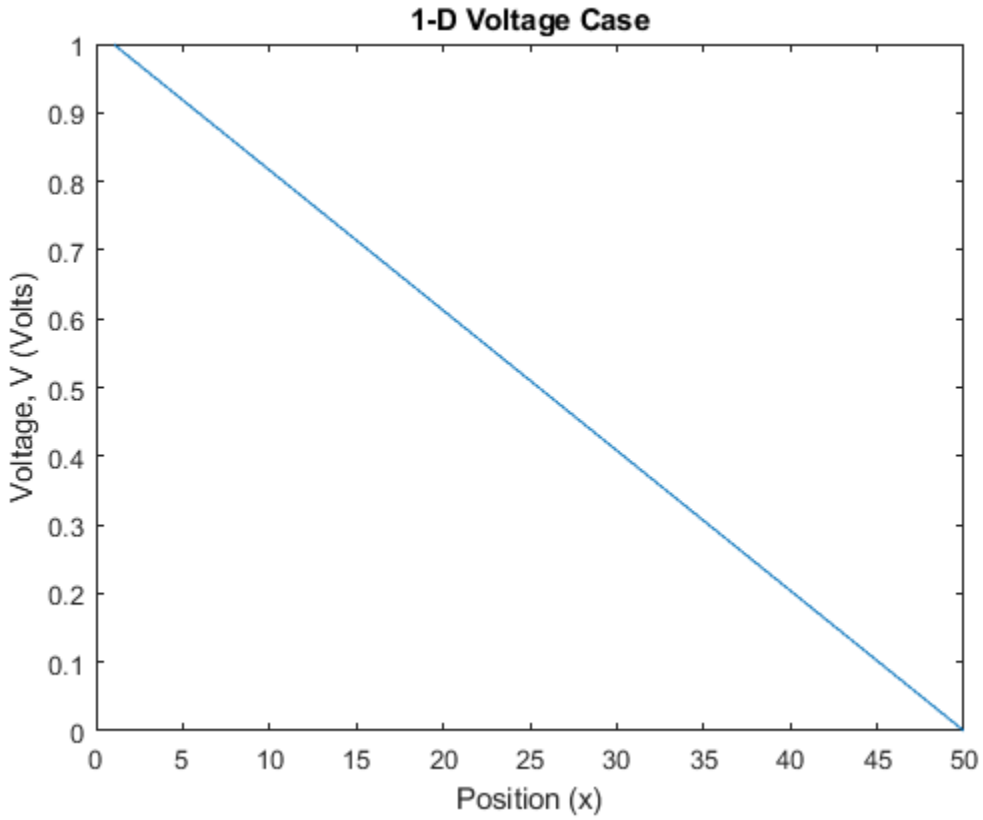
```
% Set the length of the grid, initialize the G and B matrix.
L = 50;
G = sparse(L,L);
B= zeros(L,1);
B(1,1)=1;

% Set the diagonal of the G matrix to 1. This value will be
 overwritten
% later if it is not a boundary condition.
for count = 1:L
    G(count,count)=1;
end
```

Loop through, if not a boundary case, set the gradient based on $(V_{x-1} - 2V_x + V_{x+1})/\triangle^2 = 0$

```
for col = 1:L
    if(col~=1 &&col~=L)
        n = col;
        nyBefore = n-1;
        nyAfter = n+1;
        G(n,n) = -2;
        G(n, nyBefore) =1;
        G(n, nyAfter)=1;
    end
end
V=G\B;
```

Seen in the figure below is the 1D solution to the electrostatic problem. As is expected, the potential linearly decreases across the grid.

```
figure;
plot(1:L,V)
xlabel('Position (x)')
ylabel('Voltage, V (Volts)')
title('1-D Voltage Case')
```



Part 2 - 2D Case

Set the length and width of the grid, initialize the G and B matrix.

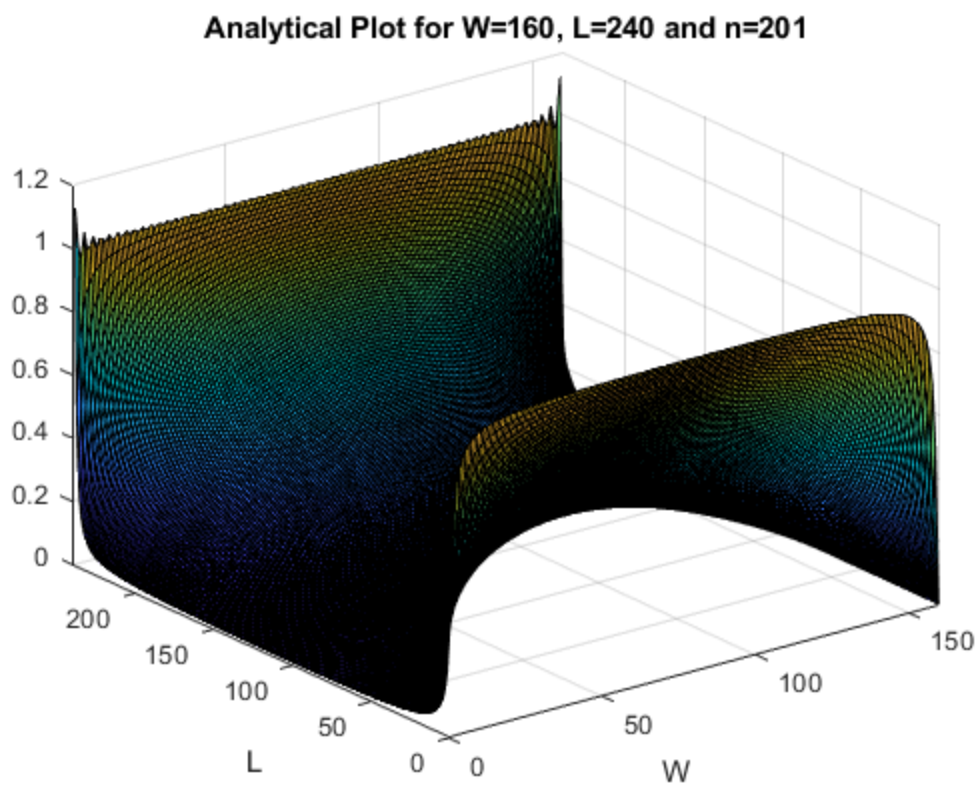
```
L = 240;
W = 160;
G = sparse(L*W,L*W);
B=zeros(L*W,1);
B(1:W,1)=1;
B(L*W-W:L*W,1)=1;

% Set the diagonal of the G matrix to 1. This value will be
 overwritten
% later if it is not a boundary condition.
for count = 1:L*W
    G(count,count)=1;
end
```

Loop through rows and columns, if not a boundary case, set the gradient based on $(V_{x-1} - 2V_x + V_{x+1})/\triangle^2 + (V_{y-1} - 2V_y + V_{y+1})/\triangle^2 = 0$

```
for col = 1:L
    if(col~=1 &&col~=L)
        for row = 1:W
            if(count~=1 && row ~=1)
                n = row + (col -1)*W;
                nyBefore = n-1;
                nyAfter = n+1;
                nxBefore = row+(col-2)*W;
                nxAfter = row+col*W;
                G(n,n) = -4;
                G(n, nyBefore) =1;
                G(n, nyAfter)=1;
                G(n, nxBefore)=1;
                G(n, nxAfter) =1;
            end
        end
    end
end
V=G\B;

%Map the voltage to original grid.
voltMap = zeros(L,W);
vAnalytical = zeros(L,W);
for cols = 1:L
    for rows = 1:W
        n= rows+(cols-1)*W;
        voltMap(cols,rows)=V(n);
        r =linspace(1,201,101);
        sum = 0;
        a=W;
        b=L/2;
        for rCount = 1:length(r)
            sum = sum + (1/r(rCount))* (cosh(r(rCount)*pi*(cols-b)/a)/
cosh(r(rCount)...
                *pi*b/a))*sin(r(rCount)*pi*rows/a);
        end
        vAnalytical(cols,rows) = 4*sum/pi;
    end
end
```

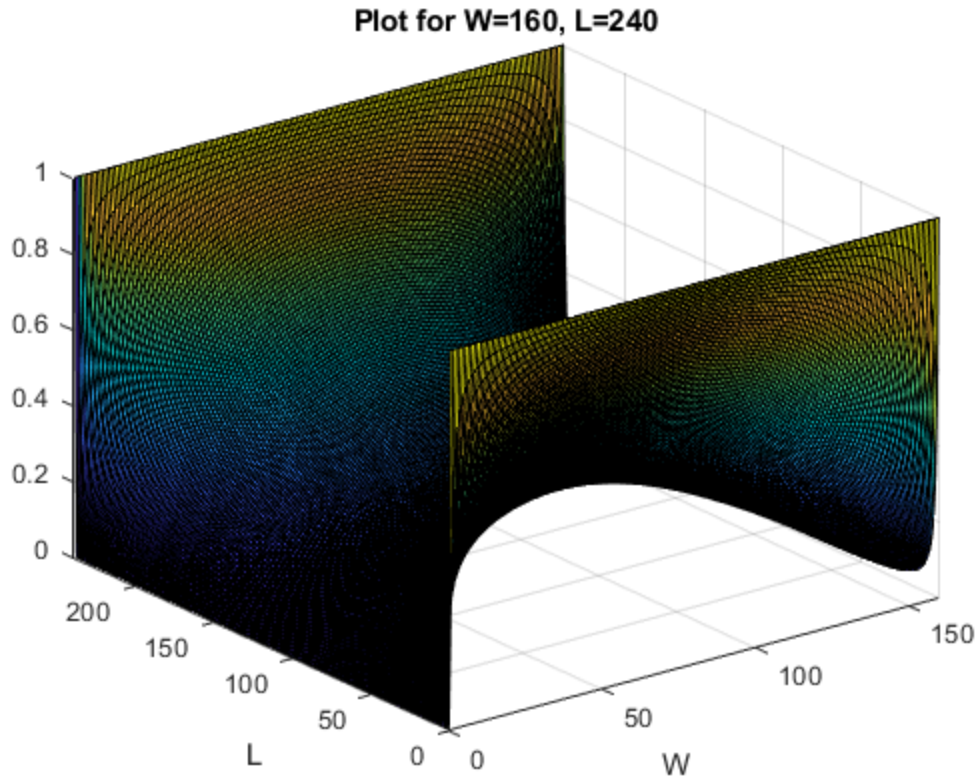
Seen in the figure below is the analytical solution to the potential problem. Over a number of trials, it was seen that as n increases (i.e. r above), the precision of the solution increases. The maximum value of n = 201 was seen to be a suitable end value. For small values of n, the solution generated is significantly different than the expected solution.

```
figure;
surf(1:W,1:L,vAnalytical);
xlabel('W');
ylabel('L');
title(['Analytical Plot for W=',num2str(W), ', L=', num2str(L), ' and
 n=', num2str(2*length(r)-1)]);
```



Seen in the figure below is the solution to the electrostatic potential problem. After trying a number of different mesh sizes, it is clear that as the size of the grid increases, the accuracy of the solution increases.

```
figure;
surf(1:W,1:L,voltMap);
xlabel('W');
ylabel('L');
title(['Plot for W=',num2str(W), ', L=', num2str(L)]);
```



Comparing the two figures above, it can be seen that the analytical solution and the numerical solution are very similar. The main advantage of the analytical solution is that is exact. However, to achieve this precision, a large n value needs to be used, resulting in a long computation time. The numerical solution may not be able to achieve the same precision, but it can be computed significantly faster.

Published with MATLAB® R2018a