Parallel Plate Capacitor Partial Differential

Erin Goeke

May 9, 2017

1 Introduction

This homework set uses the relaxation method and a computer to model the potential in a parallel plate capacitor system. The code used can be found at the end. I will be explaining when I decide if an answer is good enough and testing the theory that as the plates get closer together, the potential outside the plates gets flatter.

2 My Idea of Convergence

To explain my method of testing convergence I will be using an initialArray that is 100x100, with plates 40 pixels apart, 10 pixels long, with a potential magnitude of 10. My first step is to increase the tolerance until I cannot find a visible difference. The below two graphs are the results of these two commands, respectively.

```
>> o = relax(array, 'paraPlate', .00001, [10 10 10]);
>> o2 = relax(array, 'paraPlate', .000001, [10 10 10]);
```

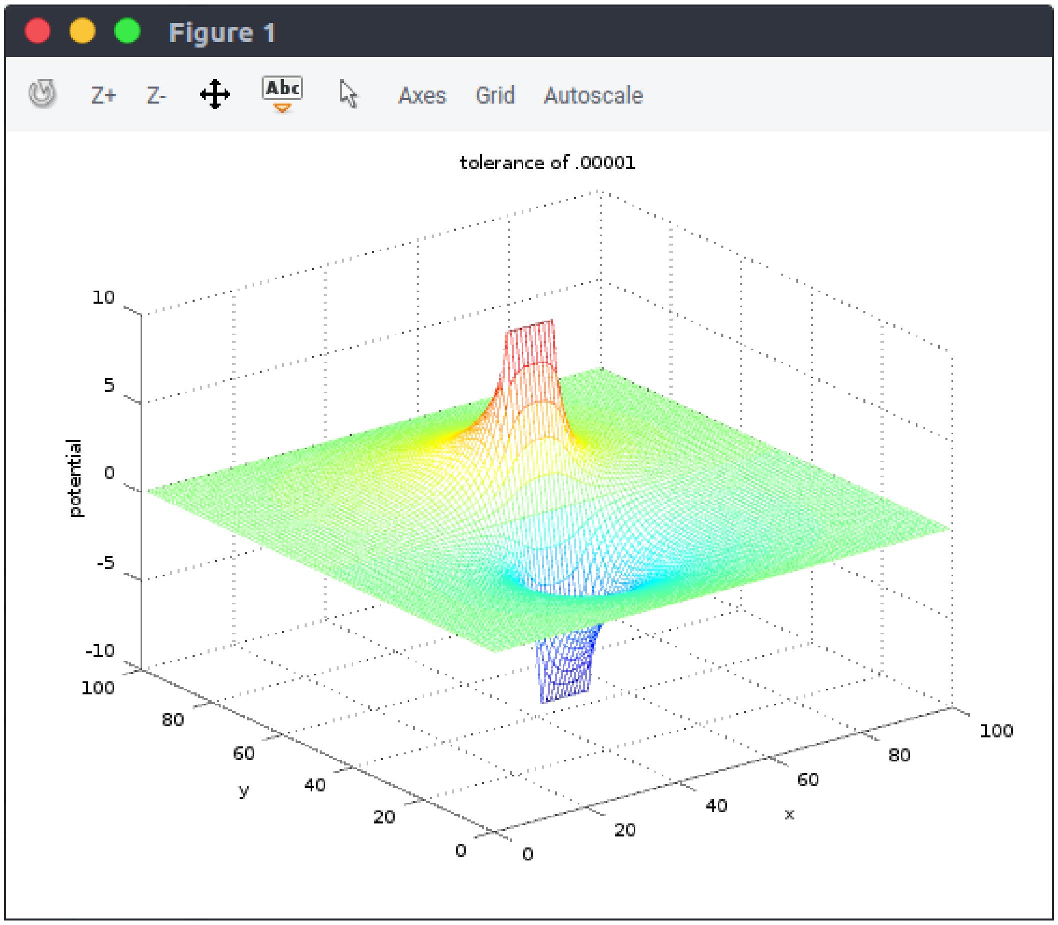
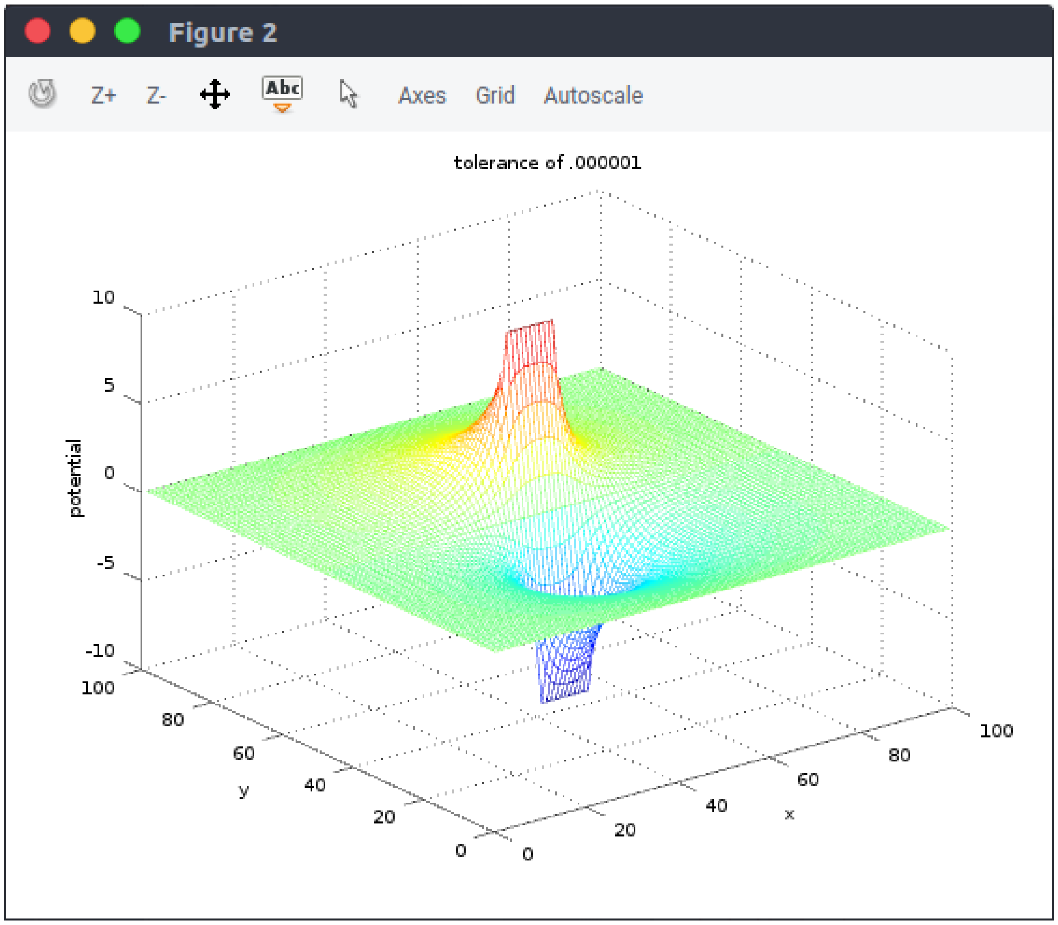
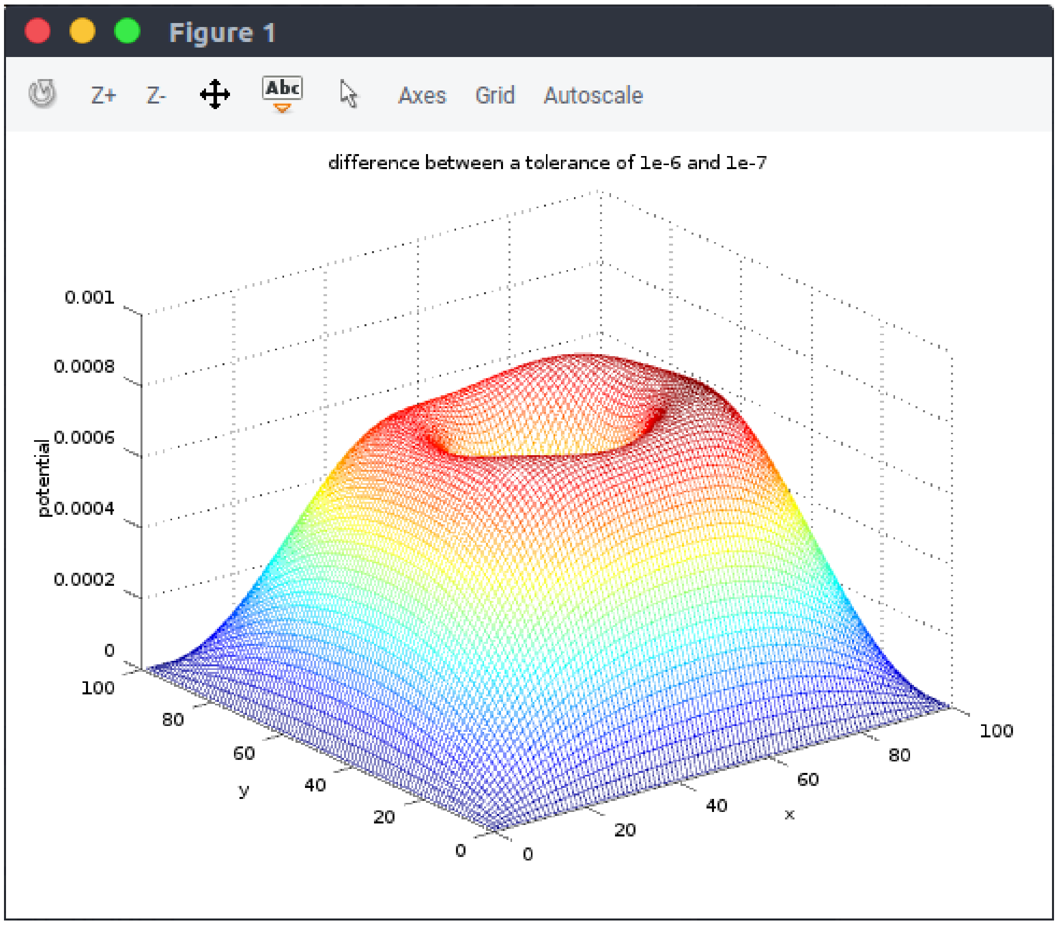


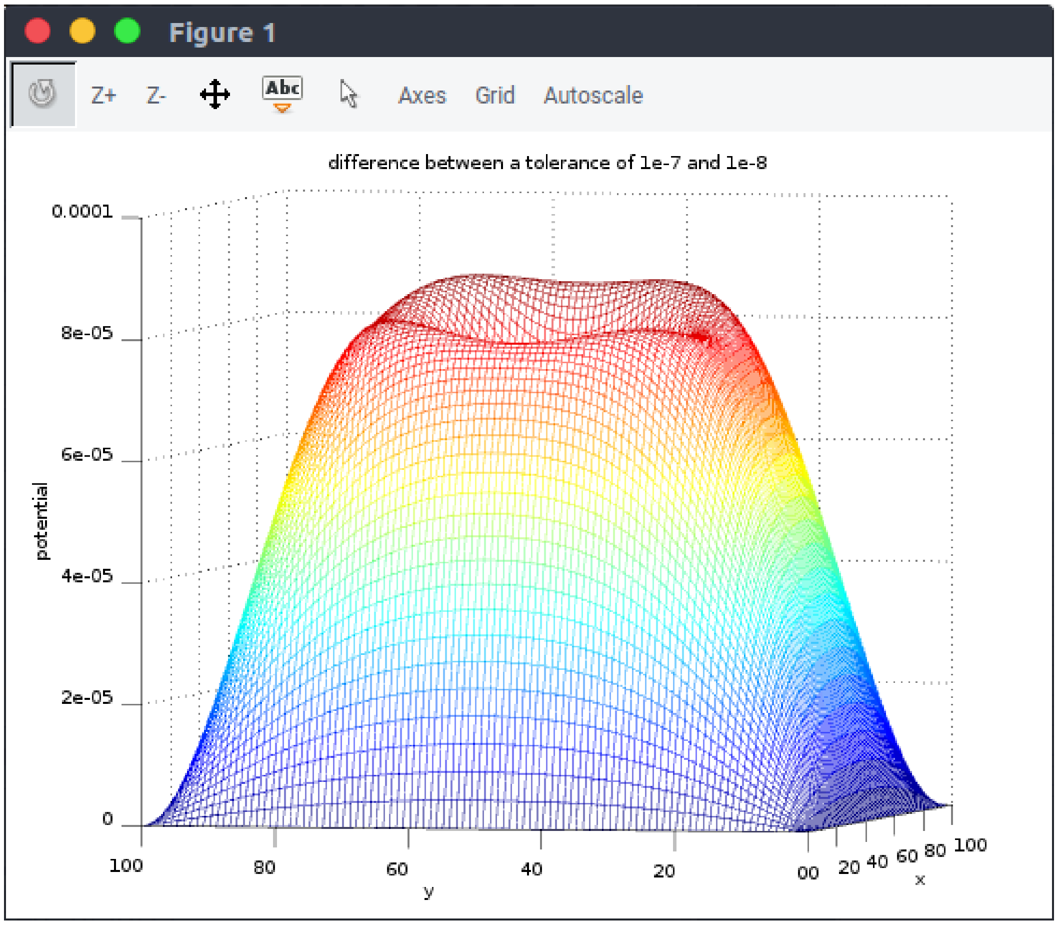
Figure 1
Z+
Z-
Axes
Grid
Autoscale
tolerance of .00001
potential
10
5
0
-5
-10
y
100
80
60
40
20
0
x
0
20
40
60
80
100



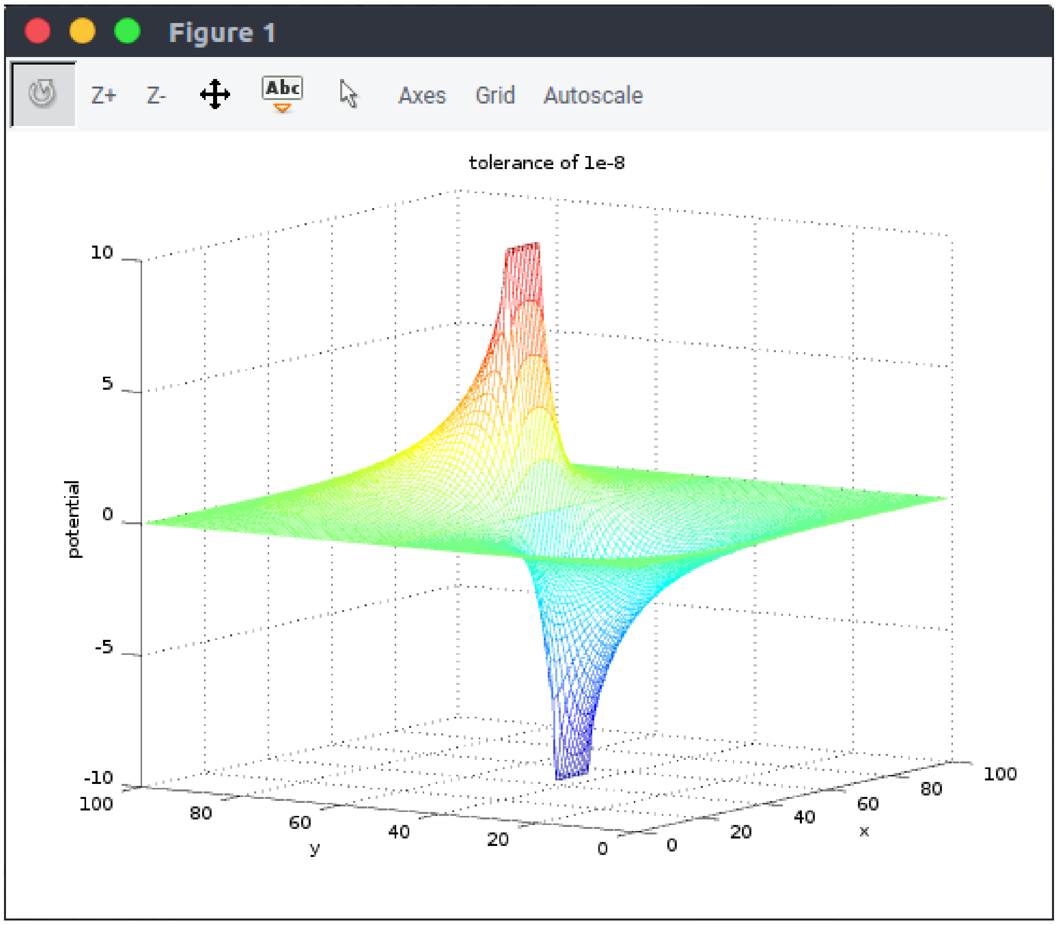
I can't really tell the difference between the two. However, just to make sure I'm going to find the difference in outputs between the tolerance .000001 and the tolerance 10 times smaller.



Hmmm... this looks not so good. I would like my answer to be accurate within at least .0001. Lets see the error between a tolerance of 1e-7 and 1e-8.

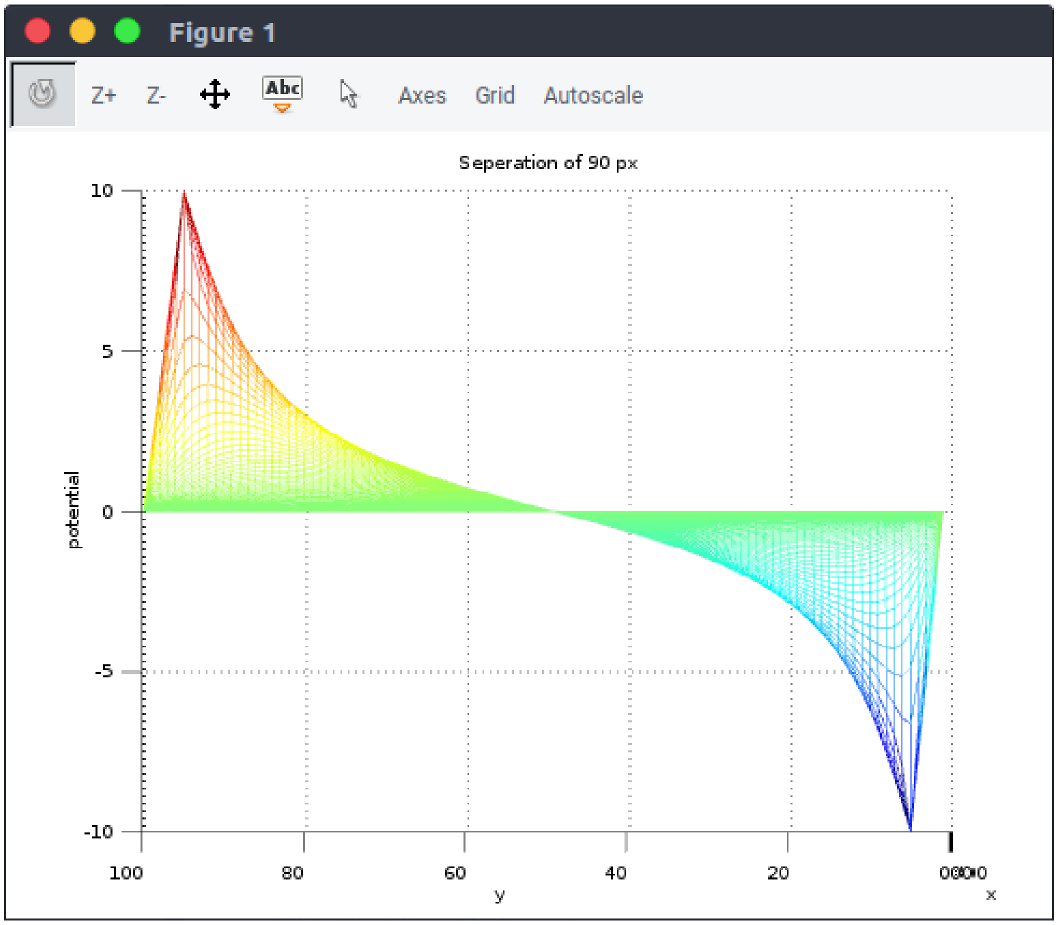


Much better. I think a tolerance of 1e-8 is an acceptable for this particular solution.

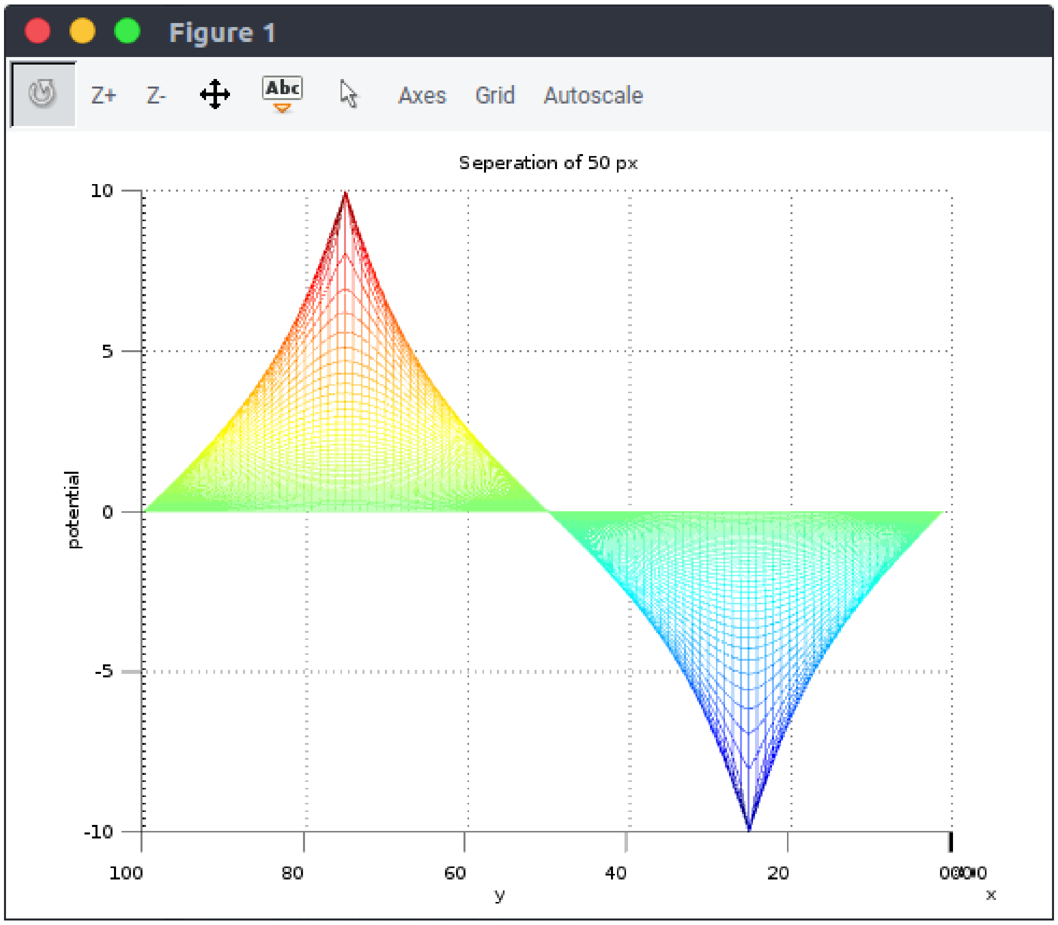


3 Testing the Theory

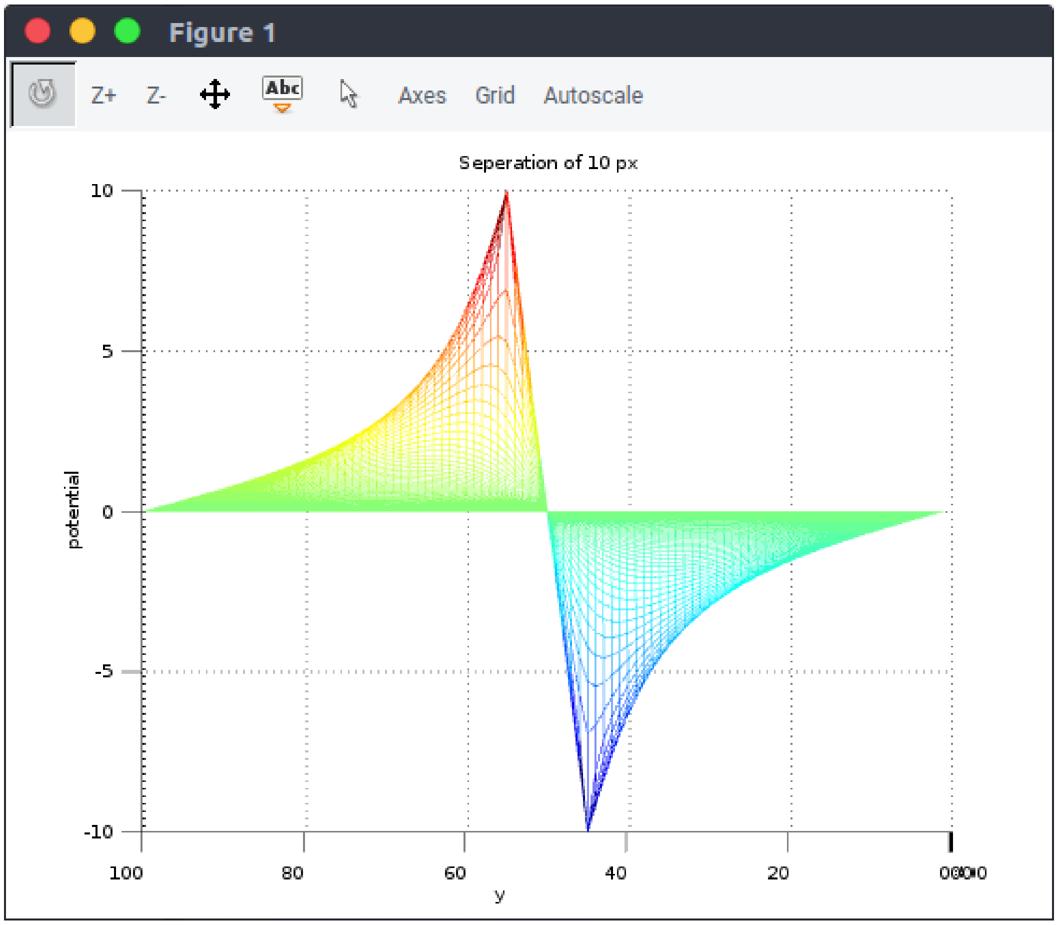
The theory goes that the potential around the parallel plates becomes flatter as the plates move closer together. To test this I have plotted parallel plates which are 10 pixels across on a 100x100 plane at varying separations: 90 pixels, 50 pixels, 10 pixels, 3 pixels, and 1 pixel. All of these answers are relaxed to within .0001 using the method shown in the previous section



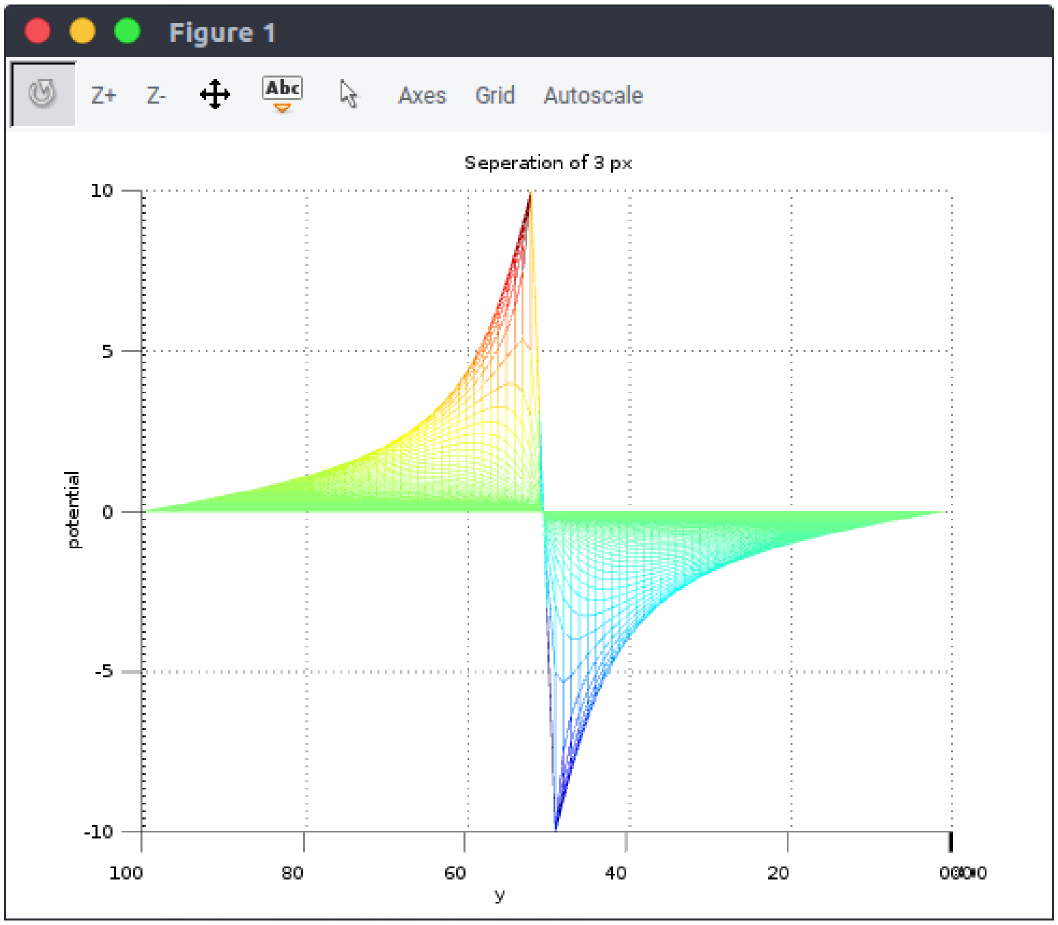
Towards the center of the graph it looks like the potential is comparatively flat as it switches over from negative to positive.



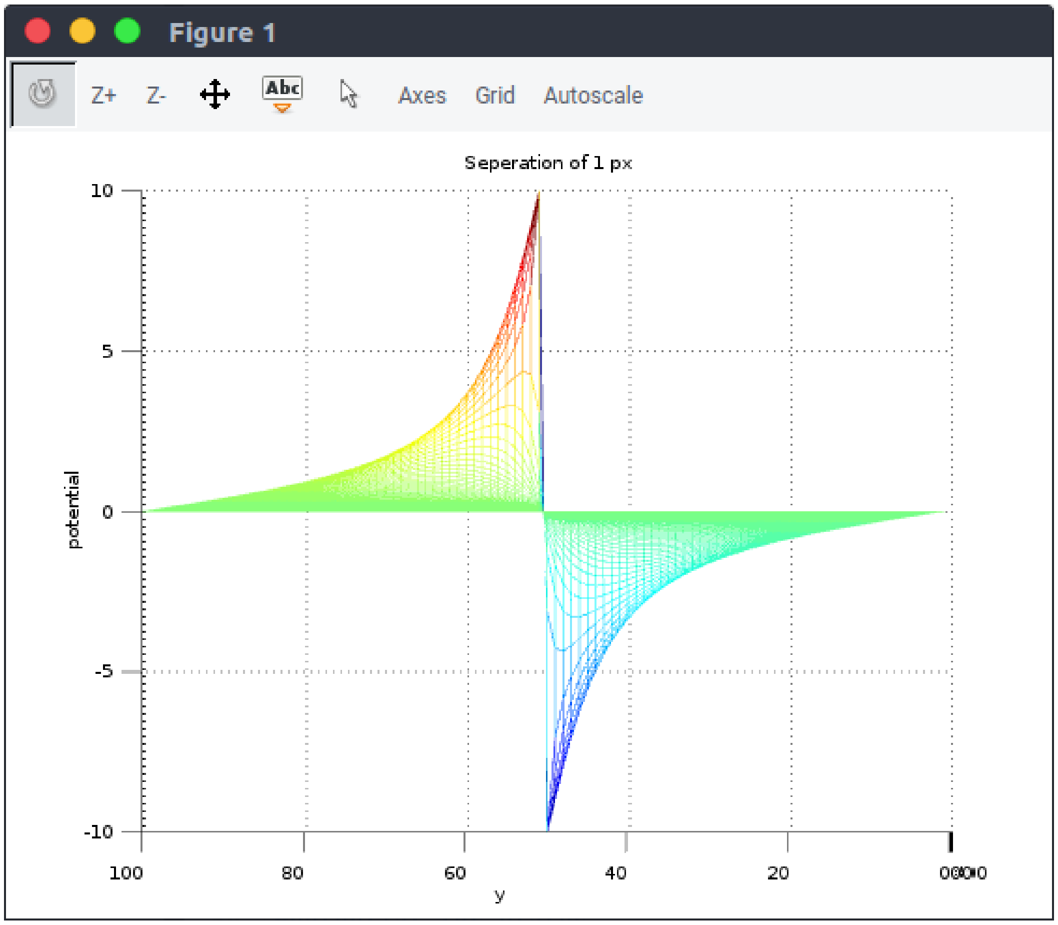
Again, even though the plates are much closer it doesn't look like the potential is gaining any flatter regions. They are 50 pixels apart though.



Now that the plates are getting much closer it looks like the potential is getting flatter in certain regions. But honestly, I still would not call this flat.



Wow, even though the distance has only changed by 7 pixels, towards the edges of the graph the curvature of the potential is noticeably flatter!



Again, this graph is flatter than the one before it and fits in with the trend that the closer the plates the quicker the potential drops off into a flat(ish) region.

4 Matlab Code

4.1 paraPlate.m

```

%% Author: eggoeke <eggoeke@nephele>
%% Created: 2017-05-09

function [output] = paraPlate (array, p)
```

```
s=size(array);
%plate seperation
sep = p(1);
%plate size
siz = p(2);
%potential for plates
potential = p(3);
%centering the dimensions for plates
dim = ones(1,2);
dim(1) = s(1)/2 - siz/2;
dim(2) = s(1)/2 + siz/2;
for i=1:2
   if(mod(dim(i),1) != 0)
     dim(i) = round(dim(i));
    endif
endfor
%centering the two rows the plates are on
rows = ones(1,2);
rows(1) = s(1)/2 + sep/2;
rows(2) = s(1)/2 - sep/2;
for i=1:2
   if(mod(rows(i),1) != 0)
     rows(i) = round(rows(i));
    endif
endfor
output=array;
output(1,:)=0;
output(:,1)=0;
output(s(1),:)=0;
output(:,s(2))=0;
output(rows(1),dim(1):dim(2))= potential;
output(rows(2),dim(1):dim(2)) = -potential;

endfunction
```

4.2 relax.m

```matlab
%Written on 5/3/17 by the MathPhys class to do
   Relaxation Method
output=feval(boundaryFunct,initialArray,p);
[nrows,ncolumns]=size(initialArray);
test=tol+1;
    copy1=output;
    copy2=output;
    copy3=output;
    copy4=output;

while test>tol
    copy1(1:nrows-1,:)=output(2:nrows,:); %shifts up
    copy2(2:nrows,:)=output(1:nrows-1,:); %shifts down
    copy3(:,1:ncolumns-1)=output(:,2:ncolumns); %shifts
        left
    copy4(:,2:ncolumns)=output(:,1:ncolumns-1); %shifts
        right
    newArray=feval(boundaryFunct,(copy1+copy2+copy3+
       copy4)/4,p);%avg shifted matrices, returns avg
       of middle
    test=max(max(abs(newArray-output))); %absolute
       value of differences between arrays and find max
    output=newArray;
end
```