Numerical Solutions of the Poisson Equation for Electric Potential

Paul Simmerling, *Electrical Engineering & Physics*, University of Connecticut

***Abstract*—Within this paper we will derive the finite difference method for the numerical solutions Gauss's Law of Electromagnetism. We also looked at several various charge distributions for the parallel plate capacitor to show the versatility of the numerical solver.**

I. Introduction

ELECTROMAGNETISM is perhaps the most influential physical phenomenon for the modern world. It is the basis on which all electronics function and without understanding its inner workings, devices such as transistors, integrated circuits, and ultimately phones and cars would never work. In electronic device theory, the most important parameter is the electric potential, voltage, within the circuit and components. Knowing the behavior of the voltage allows us to understand the response characteristics of the circuit.

Knowing how important the voltage is, we want to be able to find its behavior for any arbitrary charge distribution. The behavior of the electric potential can be derived using Maxwells equations, specifically Gauss's Law.

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon} \tag{1}$$

By definition, the potential is defined as the line integral of the vector under consideration,

$$-\int \mathbf{E} \cdot \mathbf{dl} = V \tag{2}$$

From equation 2, we can invert this relation and plug it back into Gauss's Law to derive the second order Poisson equation for Electromagnetism.

$$-\nabla V = \mathbf{E} \tag{3}$$

$$\nabla \cdot (-\nabla V) = \frac{\rho}{\varepsilon} \tag{4}$$

$$-\nabla^2 V = \frac{\rho}{\varepsilon} \tag{5}$$

Equation 5 results in a second order partial differential equation that can be solved to find the electric potential for any arbitrary charge distribution. Within this report we will use the one dimensional variant of the Poisson equation to find the behavior of the voltage. First we will find it around a single wire, then we will look at how various charge distributions affect the fields around a parallel plate capacitor.

II. Single Wire

A. Analytic Derivation

We will begin by using the Poisson equation with $\rho = 0$. For this, we will assume that the wire is infinitely long. An accurate solution of the problem presented is beyond the scope of this paper however our approximation will work when close to the wire and far from the ends.

$$\nabla^2 V = 0 \Rightarrow \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) V = 0 \tag{6}$$

Because the wire is infinitely long the potential along the y-axis will be isotropic, $V(x, y) = V(x, y + \Delta y)$.

$$\frac{\partial^2}{\partial x^2} V = 0 \tag{7}$$

$$V(x, y) = C_1 x + C_2 \tag{8}$$

Using the initial conditions $V(x = 0) = 100V, V(x = L) = 0$,

$$V(x) = 100V - \frac{100V}{L} x \tag{9}$$

Thus, we would expect the potential to drop off roughly linearly.

B. Numerical Solution

To simulate the Poisson equation, we need to discretize the equation first. This can be done using the definition to find the finite difference method.

$$\frac{\partial^2}{\partial x^2}V(x,y) \approx \frac{V(x+\Delta x,y)+V(x-\Delta x,y)-2V(x}{(\Delta x)^2} \tag{10}$$

$$\frac{\partial^2}{\partial y^2}V(x,y) \approx \frac{V(x,y+\Delta y)+V(x,y-\Delta y)-2V(x}{(\Delta y)^2} \tag{11}$$

This can be easily discretized,

$$V_{i,j} = \frac{1}{4}\left[V_{i+1,j}+V_{i-1,j}+V_{i,j+1}+V_{i,j-1}\right] \tag{12}$$

with,

$$x = x_0 + i\Delta,\ y = y_0 + j\Delta,$$
$$i,j = 0,\ldots,N_{max-1}$$

We can now write a Python code to numerically solve this equation.

```
def poisson(V, Niter=\
  10000, tol=1.e-9):
  NXmax, NYmax = V.shape
  Vp = V.copy()
  for iter in range(Niter):
    for i in range(1,NXmax-2):
      for j in range(1,NYmax-2):
        if abs(V[i,j]) != 100:
          V[i,j] = (V[i+1,j]+\
            V[i-1,j]+V[i,j+1]\
            +V[i,j-1])/4
    if sum((V-Vp)**2)/V.size < tol:
      print(iter)
      break
    Vp = V.copy()
```

This function takes in a numpy array that holds the initial conditions and iterates through each position and uses the update rule (equation 12). This update rule needs to be done multiple times as it relies on values that may not have been calculated yet. We can repeat this update rule for each position until we reach the desired precision. This code ignores the edges as we assume that there will be boundary conditions.

Specifically, for our single wire.

```
Nmax = 40
V = zeros((Nmax, Nmax), float)
V[0,1:(Nmax-1)] = 100#V
poisson(V)
```

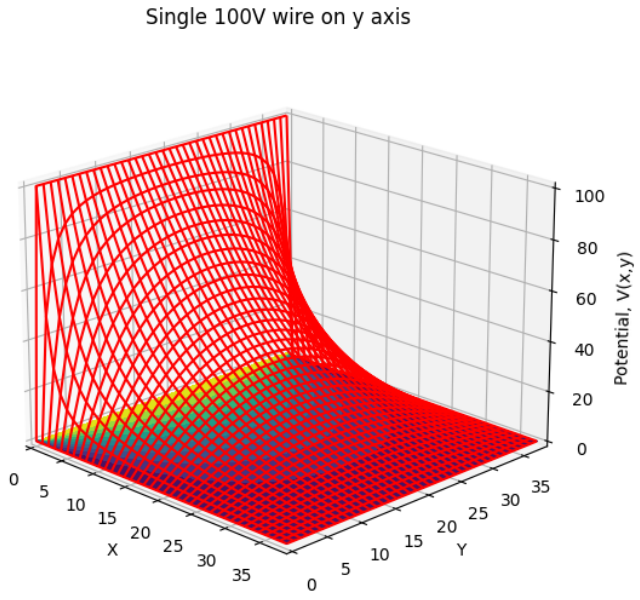

Fig. 1. Numerical solution of the Poisson Equation for a single 100V wire.

Here we set the y-axis to 100V and the boundaries to 0V. A graph of the voltage generated using this code can be seen in figure 1. From this figure, we see that our analytic solution is not accurate. While it is roughly linear near the wire, we can see that it is very non-linear everywhere else.

III. Parallel Plate Capacitor

One of the uses for these numerical solutions is finding the behavior in parallel plate capacitors. Our first test will be two parallel plate capacitors held at 100V and -100V. The python code can be seen below.

```
L, w = 150, 60#For linear regime => d<<W->
d = 15
xlb, xub, yp1, yp2 = floor((L-w)/2), \
  ceil((L+w)/2), floor((L-d)/2), \
  ceil((L+d)/2)
V = zeros((L, L), float)
V[xlb:xub, yp1] =  100#V
V[xlb:xub, yp2] = -100#V
poisson(V)
```

A graph of the resultant potential can be seen in figure 2. From figure 2, we can see that the largest gradient is within the two parallel plates. This aligns with the common assumption that most of the electric field stays within the two plates.

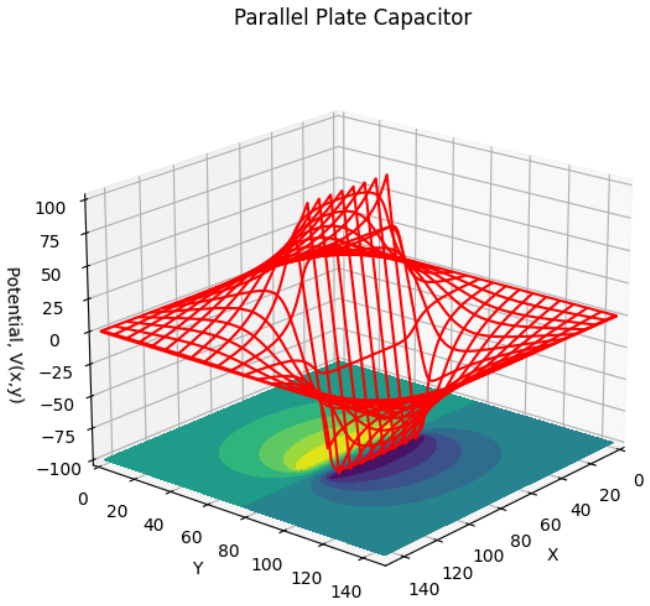

Fig. 2. Numerical solution of the Poisson Equation parallel plate capacitors at 100V and -100V.

A. Non-linear Charge Densities

We also tested the full Gauss's Law equation where $\rho \neq 0$. This gives us the discretization see in equations 13, 14, and 15.

$$V_{i,j} = \frac{1}{4}\left[V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}\right] + \pi\rho(i\Delta, j\Delta)\Delta^2 \quad (13)$$

And when programming, the equation can be split.

$$V_{i,j} = \frac{1}{4}\left[V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}\right], \ \rho(i,j) = 0 \quad (14)$$

$$V_{i,j} = \rho(i,j), \ \rho(i,j) \neq 0 \quad (15)$$

Writing equations 14 and 15 into python code, we get what is seen below.

```
def poisson(V, rho=None, \
    Niter=10000, tol=1.e-9):
  NXmax, NYmax = V.shape
  if rho is None:
    rho = zeros(V.shape, float)
  Vp = V.copy()
  for iter in range(Niter):
    for i in range(1,NXmax-2):
      for j in range(1,NYmax-2):
        if rho[i,j] != 0:
          V[i,j] = rho[i,j]
        else:
          V[i,j] = (V[i+1,j]+\
            V[i-1,j]+V[i,j+1]\
            +V[i,j-1])/4
  if sum((V-Vp)**2)/V.size < tol:
    print(iter)
    break
  Vp = V.copy()
```

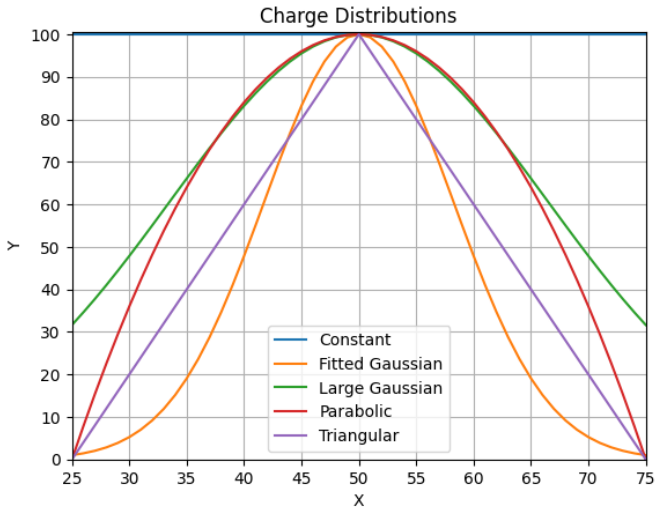

Fig. 3. Charge densities tested.

We can test this code with several charge distributions as see in figure 3. The code for generating the resultant potentials can be seen in the appendix.

IV. Conclusion

Numerical methods to find the voltage surrounding charge densities can be easily done using python and numpy. As shown above, cases that can not be easily analytically solved can be found using numerical methods.

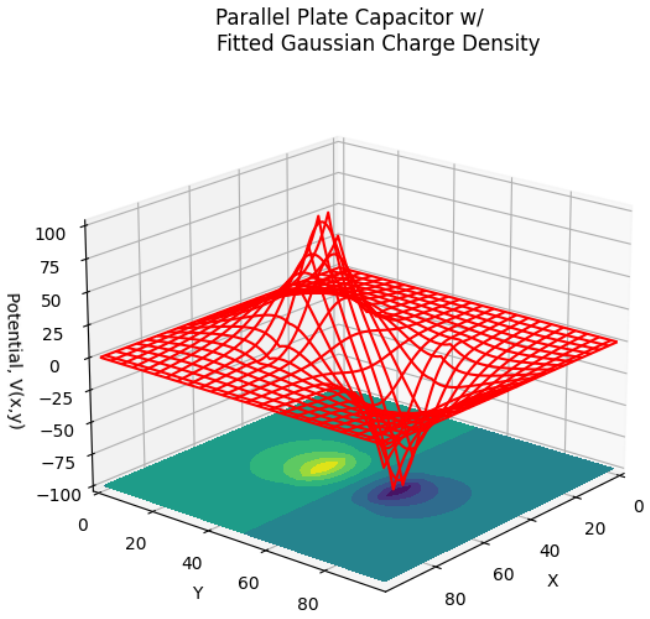

Fig. 4. Numerical solution of the Poisson Equation parallel plate capacitors with fitted Gaussian charge density (edges have 1% of center).

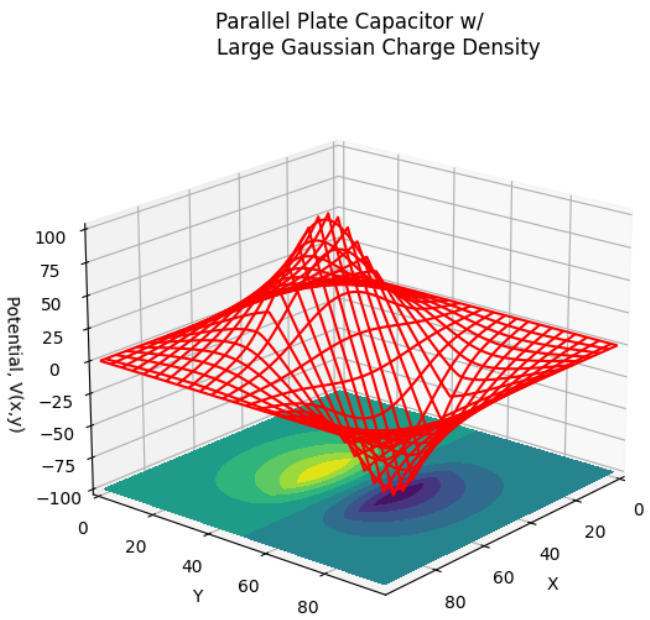

Fig. 5. Numerical solution of the Poisson Equation parallel plate capacitors with large Gaussian charge density ($\sigma = 2\sigma_{fitted}$).

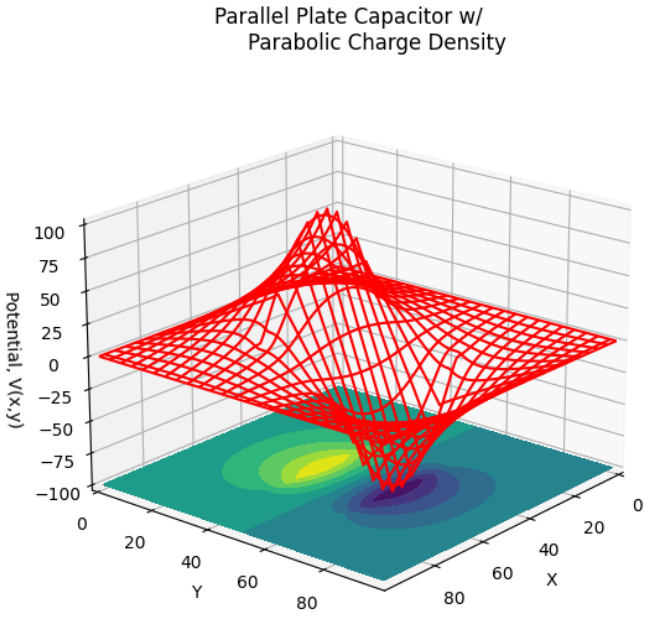

Fig. 6. Numerical solution of the Poisson Equation parallel plate capacitors with parabolic charge distribution.

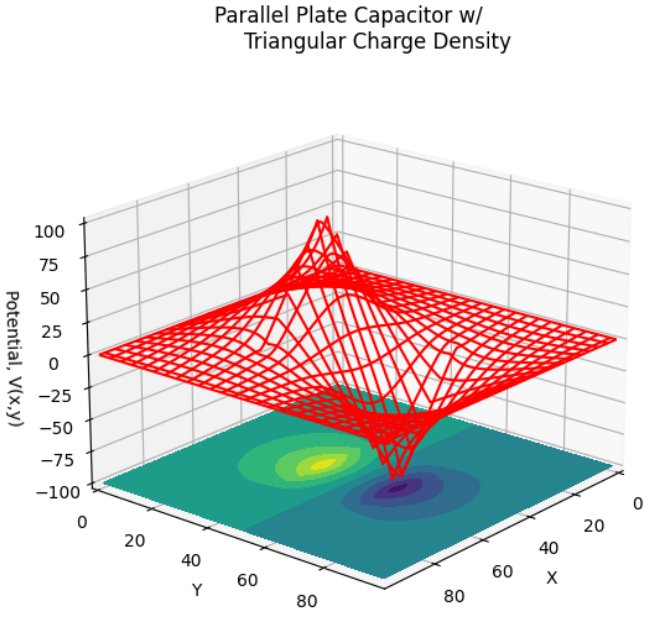

Fig. 7. Numerical solution of the Poisson Equation parallel plate capacitors with triangular change distribution.

APPENDIX
CODE FOR PYTHON3

```
from numpy import zeros, min, max, dot, \
    sum, linspace, arange, ones
from math import floor, ceil, exp, log, sqrt, pi, sin
import matplotlib.pylab as p
from mpl_toolkits.mplot3d import Axes3D

def plot(V, title, save_name, elev=20., azim=30.):
    #Get X,Y range
    NXmax, NYmax = V.shape
    x = range(0, NXmax-1, 1)
    y = range(0, NYmax-1, 1)
    X, Y = p.meshgrid(x,y)#Make grid
    def functz(V):#Make Z for plotting
        z = V[X,Y]
        return z
    Z = functz(V)
    fig = p.figure()#Make new fig
    ax = Axes3D(fig)
    #Plot fewer lines
    xstride, ystride = floor(max(x)/20), floor(max(y)/20)
    ax.plot_wireframe(X, Y, Z, rstride=xstride, \
        cstride=ystride, color = 'r')
    ax.set_xlabel('X'); ax.set_ylabel('Y')
    ax.set_zlabel('Potential, V(x,y)')
    ax.set_xlim(0,max(x)); ax.set_ylim(0,max(y))
    ax.set_title(title)
    ax.contourf(X,Y,Z,11,stride=25,offset=min(Z))
    ax.view_init(elev=elev, azim=azim)
    p.savefig('figures/%s.png'%save_name,bbox_inches='tight')
    print("Finished %s (%s.png)" % (title, save_name))

def poisson(V, rho=None, Niter=10000, tol=1.e-9):
    print("Initializing")
    NXmax, NYmax = V.shape#Get shape
    if rho is None:#make rho if nothing is passed
        rho = zeros(V.shape, float)
    print("Working hard, wait for figure while I count to %d" % \
        (floor((Niter-1)/10)*10))
    Vp = V.copy()
    for iter in range(Niter):#repeat many times
        if iter%10 == 0: print(iter)
        for i in range(1,NXmax-2):#x axis, skip BC
            for j in range(1,NYmax-2):#y axis, skip BC
                if rho[i,j] != 0:#If rho!=0 use rho eqn
                    V[i,j] = rho[i,j]
                elif abs(V[i,j]) != 100:#don't modify if IC
                    V[i,j] = (V[i+1,j]+V[i-1,j]+\
```

```
                        V[i,j+1]+V[i,j-1])/4#rho=0 eqn
        if sum((V-Vp)**2)/V.size < tol:#Check to see in tol
            print(iter)
            break
        Vp = V.copy()#make copy for tol checking

def gaussian(k, A=100, mu=0, sig=1):#Make gaussian change dist
    for i in range(len(k)):
        k[i] = A*exp(-((i-len(k)/2-mu+1/2)/sig)**2)
    return k

def parab(k, A=100):#Make parabolic charge dist
    for i in range(len(k)):
        k[i] = -i*(i-len(k)+1)/(len(k)/2-1/2)**2*A
    return k

def linear(k, A=100):#Make triangular charge dist
    for i in range(len(k)):
        k[i] = A*i/(len(k)/2)
        if i > len(k)/2:
            k[i] = A*(len(k)-i-1)/(len(k)/2-1/2)
        else:
            k[i] = A*i/(len(k)/2-1/2)
    return k

if __name__ == "__main__":
    print("Case Study 4")
    Nmax = 40#range of plot
    V = zeros((Nmax, Nmax), float)#make all 0, also BC=0
    V[0,1:(Nmax-1)] = 100 #Volts, IC-> could use rho but won't
    poisson(V)
    plot(V, "Single 100V wire on y axis", "cs4_1", azim=-45.)

    ang,el=40,20#set viewing angle
    L, w = 150, 60#For linear regime => d<<W->d~w/10
    d = 15
    xlb, xub, yp1, yp2 = floor((L-w)/2), ceil((L+w)/2), \
        floor((L-d)/2), ceil((L+d)/2)#set pos for plates
    V = zeros((L, L), float)
    V[xlb:xub, yp1] =  100 #Volts, IC-> could use rho but won't
    V[xlb:xub, yp2] = -100 #Volts, IC-> could use rho but won't
    poisson(V)
    plot(V, "Parallel Plate Capacitor", "cs4_2", elev=el, azim=ang)

    A, L = 100, 100 #set amplitude and range
    s = 25/sqrt(-log(0.01))#sigma for gauss, 0.01*MAX at ends
    V = zeros((L, L), float)#make initial grid+BC
    rho = zeros(V.shape,float)#Make rho
    rho[25:76,37] = gaussian(rho[25:76,37], A, 0, s)#set plate1
    rho[25:76,63] = -rho[25:76,37]#set plate2
```

```
poisson(V,rho)
plot(V, """Parallel Plate Capacitor w/
    Fitted Gaussian Charge Density""", "cs4_3", elev=el,azim=ang)

V = zeros((L, L), float)#make initial grid+BC
rho = zeros(V.shape,float)#Make rho
rho[25:76,37] = gaussian(rho[25:76,37], A, 0, 2*s)#set plate1
rho[25:76,63] = -rho[25:76,37]#set plate2
poisson(V,rho)
plot(V, """Parallel Plate Capacitor w/
    Large Gaussian Charge Density""", "cs4_4", elev=el,azim=ang)

V = zeros((L, L), float)#make initial grid+BC
rho = zeros(V.shape,float)#Make rho
rho[25:76,37] = parab(rho[25:76,37], A)#set plate1
rho[25:76,63] = -rho[25:76,37]#set plate2
poisson(V,rho)
plot(V, """Parallel Plate Capacitor w/
    Parabolic Charge Density""", "cs4_5", elev=el, azim=ang)

V = zeros((L, L), float)#make initial grid+BC
rho = zeros(V.shape,float)#Make rho
rho[25:76,37] = linear(rho[25:76,37], A)#set plate1
rho[25:76,63] = -rho[25:76,37]#set plate2
poisson(V,rho)
plot(V,"""Parallel Plate Capacitor w/
    Triangular Charge Density""", "cs4_6", elev=el,azim=ang)

#Make plot of charge densities
cs = ones(51,float)*100#Normal parallel plate
fg = gaussian(zeros(51,float), A, 0, s)#fitted gaus
lg = gaussian(zeros(51,float), A, 0, 2*s)#large gaus
pb = parab(zeros(51,float), A)#parab
tg = linear(zeros(51,float), A)#triang
fig = p.figure()#make fig
ax = p.gca()
p.plot(arange(25,76),cs,arange(25,76),fg,arange(25,76),\
    lg,arange(25,76),pb,arange(25,76),tg)
ax.set_xlabel('X'); ax.set_ylabel('Y')
ax.set_xlim(25,75); ax.set_ylim(0,100.5)
p.xticks(arange(25, 76, 5)); p.yticks(arange(0, 101, 10))
ax.set_title("Charge Distributions")
p.grid(True)
p.legend(["Constant","Fitted Gaussian", "Large Gaussian", \
    "Parabolic", "Triangular"])
p.savefig('figures/cs4_cd.png', bbox_inches='tight')

#Tried sep of variables just to see soln
L = 40
sC = pi/L
```

```
C1 = 100/(1-exp(-2*sC*L))
V = zeros((L, L), float)
for x in range(L):
    for y in range(L):
        V[x,y]=(C1*exp(-sC*x)+(100-C1)*exp(sC*x))*sin(sC*y)
plot(V, """Analytic Sep
 Vars for Single 100V
 wire on y axis""", "cs4_1a", azim=-45.)
```