Code:

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# variables are hard-coded; any of these can be modified

numxbins = 32

numybins = 32

xmin = 0

ymin = 0

stepsize = 1

error = .001

imax = 10000

# iterator function

def iterateAvg(v):

# create a new array to store new values

newv = np.copy(v)

maxdiff = 0.0

for i in range(1, numybins-1):

for j in range(1, numxbins-1):

newv[i][j] = (v[i-1][j] + v[i+1][j] + v[i][j-1] + v[i][j+1])/4.0

diff = abs(newv[i][j] - v[i][j])

if diff > maxdiff:

maxdiff = diff

return newv, maxdiff

# plotting function

def PlotSolution(xmin, ymin, numxbins, numybins, stepsize, v):

x = np.linspace(xmin, stepsize\*numxbins, numxbins, np.float32)

y = np.linspace(ymin, stepsize\*numybins, numybins, np.float32)

fig1 = plt.figure()

ax = fig1.gca(projection='3d')

X, Y = np.meshgrid(x, y)

print X

print Y

surf = ax.plot\_surface(X, Y, v, rstride=1, cstride=1, cmap='cool', linewidth=0, antialiased=True)

plt.xlabel("X")

plt.ylabel("Y")

fig2 = plt.figure()

cs = plt.contourf(X, Y, v, 32, rstride=1, cstride=1,cmap='cool')

plt.colorbar()

plt.xlabel("X")

plt.ylabel("Y")

plt.show()

# main method

v = np.zeros((numxbins, numybins))

# Boundary conditions along edges; can be set to constants or functions

def xMinFunc(xval):

return 0

def xMaxFunc(xval):

return 0

def yMaxFunc(yval):

return 100

def yMinFunc(yval):

return 100

# Populate v with edge conditions along boundaries

x = np.linspace(xmin, stepsize\*numxbins, numxbins, np.float32)

y = np.linspace(ymin, stepsize\*numybins, numybins, np.float32)

f = np.vectorize(xMinFunc)

v[0] = f(x)

f = np.vectorize(xMaxFunc)

v[numxbins-1] = f(x)

f = np.vectorize(yMinFunc)

v[:, 0] = f(y)

f = np.vectorize(yMaxFunc)

v[:, numybins-1] = f(y)

# initialize curdiff to a value greater than desired error

curdiff = error + 1

count = 0

while curdiff > error:

v, curdiff = iterateAvg(v)

count += 1

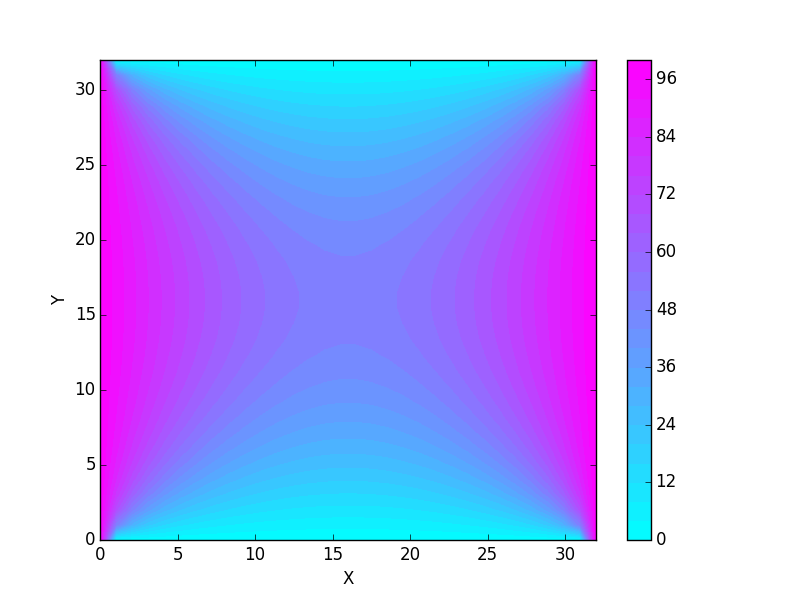
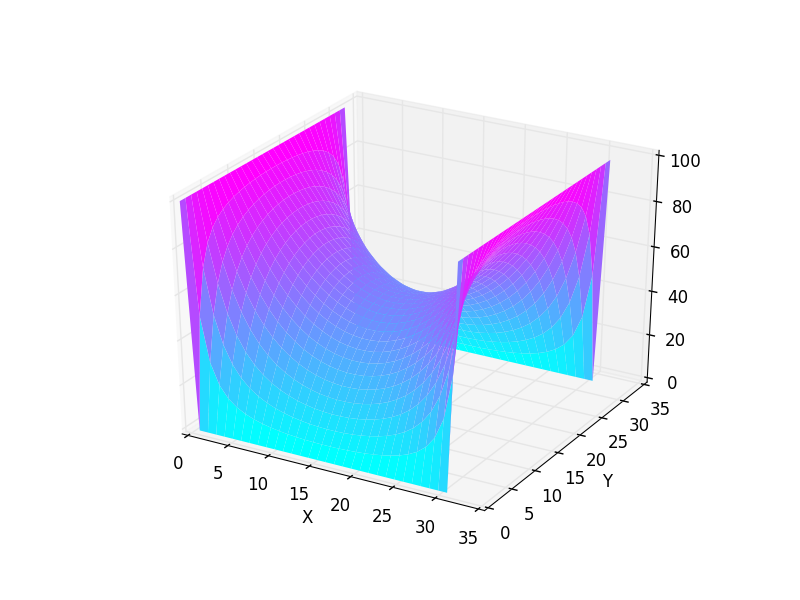
# terminate if function does not converge within desired number of iterations

if count > imax:

break

print count

PlotSolution(xmin, ymin, numxbins, numybins, stepsize, v)

Sample output (1175 iterations, error < .001): 

Arbitrary functions can also be assigned to boundary conditions!

For example, setting the boundary conditions to:

def xMinFunc(xval):

return xval \* xval

def xMaxFunc(xval):

return 100

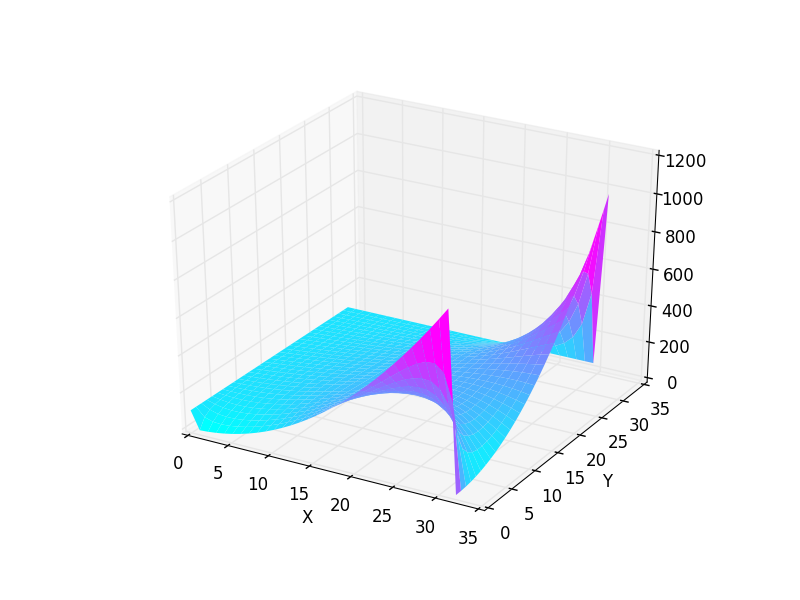
def yMaxFunc(yval):

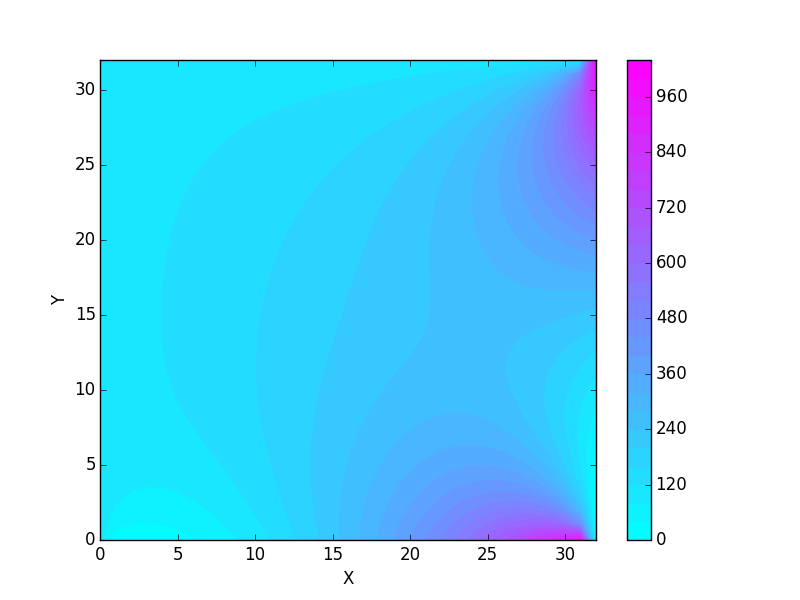
return yval\*yval

def yMinFunc(yval):

return 100

Yields the following (1445 iterations, error < .001):





This is a representation of a two dimensional surface in a three-dimensional space. The size of the physical system being modeled depends on both the step size and the number of bins. The total distance along each axis is the product of these two quantities. An error of less than .001 is easily achieved within the sample maximum iteration parameter of 10000; both samples here converged in under 1500 iterations. Decreasing the maximum error to .1 reduces the number of iterations performed by about a factor of 4 (to ~240 iterations). Both of these examples were performed with a sample size of 32 along each axis, and a step size of 1; increasing the sample size will increase computational complexity as well. Boundary conditions can be set to either constants or functions.