Applied Computational Methods in Mechanical Sciences

(ME466)

Assignment 10

Himanshu Kumar

16ME234

October 28, 2019

**Problem Statement:**

A roll of steel is subjected to left end and at right end. If the roll is of length , use:

1. Explicit method
2. Implicit method
3. Crank-Nicholson method

to find the distribution from to , when the initial temperature is . Compare the results based on the schemes. The governing equation is given by:

where

and

**Python Code:**

import time

import matplotlib.pyplot as plt

from mpl\_toolkits import mplot3d

import numpy as np

def disp(x):

for i in range(len(x)):

print(x[i])

def uniform\_grid(lims, n):

rng = lims[1]-lims[0]

delx=rng/n

xm = [0 for i in range(n+1)]

for i in range(n):

xm[i+1] = xm[i]+delx

return(xm,delx)

def matrix\_form\_solve\_plot(lx,nx,ly,ny,err\_lim):

global x,dx,y,dy

x,dx = uniform\_grid(lx,nx)

y,dy = uniform\_grid(ly,ny)

e= 1

c= -2\*(1+( (dx\*dx)/(dy\*dy) ))

w= 1

s= (dx\*dx)/(dy\*dy)

n= (dx\*dx)/(dy\*dy)

lx = len(x)-1

ly = len(y)-1

global t

t = [ [0 for i in range(lx+1)] for j in range(ly+1) ]

err = [ [0 for i in range(lx+1)] for j in range(ly+1) ]

Qi =0

# boundary conditions

# 1. left

for i in range(ly+1):

t[i][0] = 75

# 2. Right

for i in range(ly+1):

t[i][lx] = 100

# 3. Top

for i in range(lx+1):

t[ly][i] = 300

# 4. Bottom

for i in range(lx+1):

t[0][i] = 50

pt = [ [0 for i in range(lx+1)] for j in range(ly+1) ]

for i in range(0,ly+1):

for j in range(0,lx+1):

pt[i][j] = t[i][j]

#gauss run

run = 0

while(1):

for i in range(1,ly):

for j in range(1,lx):

t[i][j] = (Qi - ( s\*t[i-1][j] + n\*t[i+1][j] + e\*t[i][j+1]+ w\*t[i][j-1] ))/(c)

#Error calculation

for i in range(1,ly):

for j in range(1,lx):

try:

err[i][j] = abs((t[i][j]- pt[i][j])/t[i][j])

except:

pass

#finding maximum error:

max\_err = 0

for i in range(1,ly):

for j in range(1,lx):

if(err[i][j]>max\_err):

max\_err = err[i][j]

if(max\_err<err\_lim):

break

#reiterate

run = run+1

for i in range(1,ly):

for j in range(1,lx):

pt[i][j]= t[i][j]

print("\n No. of iterations:",run)

print ("\n CPU time: ", time.process\_time(),'s')

#plotting filled contour

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

fig,ax=plt.subplots(1,1)

cp = ax.contourf(X, Y, t,50,cmap = 'viridis')

fig.colorbar(cp) # Add a colorbar to a plot

ax.set\_title('Filled Contours Plot')

ax.set\_xlabel('x (m)')

ax.set\_ylabel('y (m)')

plt.show()

#plotting line contour

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

fig,ax=plt.subplots(1,1)

cp = ax.contour(X, Y, t,15)

fig.colorbar(cp) # Add a colorbar to a plot

ax.clabel(cp, inline=1, fontsize=7)

ax.set\_title('Filled Contours Plot')

ax.set\_xlabel('x (m)')

ax.set\_ylabel('y (m)')

plt.show()

#plotting line contour

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

fig,ax=plt.subplots(1,1)

cp = ax.contour(X, Y, t,130)

fig.colorbar(cp) # Add a colorbar to a plot

ax.clabel(cp, inline=1, fontsize=7)

ax.set\_title('Filled Contours Plot')

ax.set\_xlabel('x (m)')

ax.set\_ylabel('y (m)')

plt.show()

# #plotting 3D

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

# fig = plt.figure()

# ax = plt.axes(projection='3d')

# ax.contour3D(X, Y, t, 50, cmap='binary')

# ax.set\_xlabel('x')

# ax.set\_ylabel('y')

# ax.set\_zlabel('t')

# ax.set\_title('3D contour')

# plt.show()

mag = 10

nx=48\*mag

ny=60\*mag

err\_lim = 0.001

limx = (0,2.4)

limy = (0,3.0)

matrix\_form\_solve\_plot(limx,nx,limy,ny,err\_lim)

**Results:**

1. **With r=0.7 and :**

CPU time: 0.6875 s