#===================================

# Crank\_Nicolson

#===================================

import math

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

#========================= function definition =========================

def TDMA(a,b,c,r,s):

n=len(a)

# forward substitution

c[0]=c[0]/b[0]

r[0]=r[0]/b[0]

for i in range(1,n):

c[i]=c[i]/(b[i]-a[i]\*c[i-1])

r[i]=(r[i]-a[i]\*r[i-1])/(b[i]-a[i]\*c[i-1])

# backward substitution

s[n-1]=r[n-1]

for i in reversed(range(n-1)):

s[i]=r[i]-s[i+1]\*c[i]

#============================= main program ============================

m=40 # number of horizontal divisions

n=40 # number of vertical divisions

l=1. # length of the domain

h=1. # height of the domain

dx=l/m

dy=h/n

dt=5 # time step

alpha=23.1\*1e-6 # diffusivity of the iron

cr1=1e-3 # convergence criteria for steady state condition

cr2=1e-3 # convergence criteria

T1=60 # left hand side temperature

T2=20 # right hand side temperature

T3=10 # bottom side temperature

T4=30 # top side temperature

Fo\_x=alpha\*dt/dx\*\*2 # Fourier number

Fo\_y=alpha\*dt/dy\*\*2 # Fourier number

# initial condition

T=np.zeros((m+1,n+1));T\_old=np.zeros((m+1,n+1));TT=np.zeros((m+1,n+1))

# matrix definition

a=np.zeros(m-1);b=np.zeros(m-1);c=np.zeros(m-1);r=np.zeros(m-1);s=np.zeros(m-1)

# boundary conditions

for i in range(m+1):

T[i,0]=T3

T[i,n]=T4

for j in range(n+1):

T[0,j]=T1

T[m,j]=T2

#============================== main loop ==============================

t=0

err1=1

while err1>cr1:

t=t+dt

T\_old[:,:]=T[:,:]

k=0

err2=1

while err2>cr2:

k=k+1

TT[:,:]=T[:,:]

for j in range(1,n):

for i in range(1,m):

a[i-1]=-0.5\*Fo\_x

b[i-1]=1+Fo\_x+Fo\_y

c[i-1]=-0.5\*Fo\_x

r[i-1]=T\_old[i,j] +0.5\*Fo\_y\*( T[i,j+1]+T[i,j-1] ) +0.5\*Fo\_x\*( T\_old[i+1,j]-2\*T\_old[i,j]+T\_old[i-1,j] ) +0.5\*Fo\_y\*( T\_old[i,j+1]-2\*T\_old[i,j]+T\_old[i,j-1] )

i=1

r[i-1]=r[i-1]-a[i-1]\*T[i-1,j]

i=m-1

r[i-1]=r[i-1]-c[i-1]\*T[i+1,j]

TDMA(a,b,c,r,s)

for i in range(1,m):

T[i,j]=s[i-1]

err2=0

for i in range(1,m):

for j in range(1,n):

err2=err2+(T[i,j]-TT[i,j])\*\*2/((m-1)\*(n-1))

err2=math.sqrt(err2)

err1=0

for i in range(1,m):

for j in range(1,n):

err1=err1+(T[i,j]-T\_old[i,j])\*\*2/((m-1)\*(n-1))

err1=math.sqrt(err1)

# printing time and second norm of the error

print('{:10d}{:15.2e}'.format(t,err1))

#=============================== results ===============================

print('\n=================================================================\n')

print(' calculation completed\n')

print(' mesh:{:8d}\*{:2d}'.format(m,n))

print(' x Fourier number:{:8.3f}'.format(Fo\_x))

print(' y Fourier number:{:8.3f}'.format(Fo\_y))

print(' time until reaching steady state condition:{:10.2f}s\n'.format(t))

# generating the grid

Y,X=np.meshgrid(np.linspace(0,h,n+1),np.linspace(0,l,m+1))

# temperature contour

fig1=plt.figure(1)

plt.contourf(X,Y,T,50,cmap='jet')

plt.axes().set\_aspect('equal')

plt.xlabel('x(m)');plt.ylabel('y(m)')

# temperature surface

fig2=plt.figure(2)

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

ax.plot\_surface(X,Y,T,cmap='jet')

ax.set\_xlabel('x(m)');ax.set\_ylabel('y(m)');ax.set\_zlabel('T($^oC$)')

plt.show()