**二维非稳态热传导**

**小组成员：马威、田畅、罗凌宇、刘胜禹、**

**王淞、雷荣、李杰**

**指导教师：刘洋**

目录

[一、划分网格： 1](#_Toc23829)

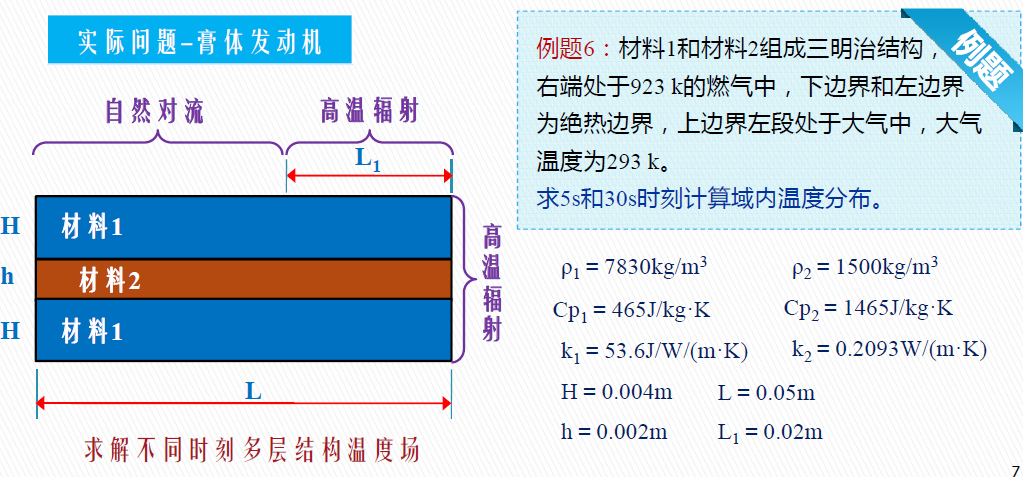
[二、控制离散方程： 2](#_Toc27411)

[三、编程计算（代码见附录） 5](#_Toc24151)

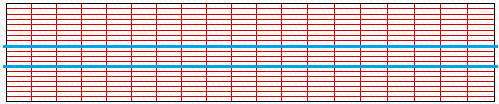
[四、fluent仿真计算 6](#_Toc21090)

[五、 结论分析 6](#_Toc10480)

[六、附录 6](#_Toc21094)



## 一、划分网格：



X方向网格：20；Y方向网格：20 材料1：材料2：材料1=8:4:8

总网格数量：400

**网格尺寸：**

注意有界性条件：方程的各项系数大于0

**显式格式：**





400网格时



40000网格时



**隐式格式：**

有界性条件自动满足

**C-N格式：**







## 二、控制离散方程：

微分形式的控制方程：



积分形式的控制方程：



积分可得：



对右式时间积分加权处理：



两边同除并将面积与体积代入得到：



显式形式场变量系数归一化处理：



系数：







边界条件：

下边界为绝热边界：

，，

左边界为绝热边界：

，，

右边界为辐射换热边界:



上边界右侧：



上边界左侧:



角点的处理:

左上角点:



右上角点:

对于Twall的处理

左下角点:



右下角点:



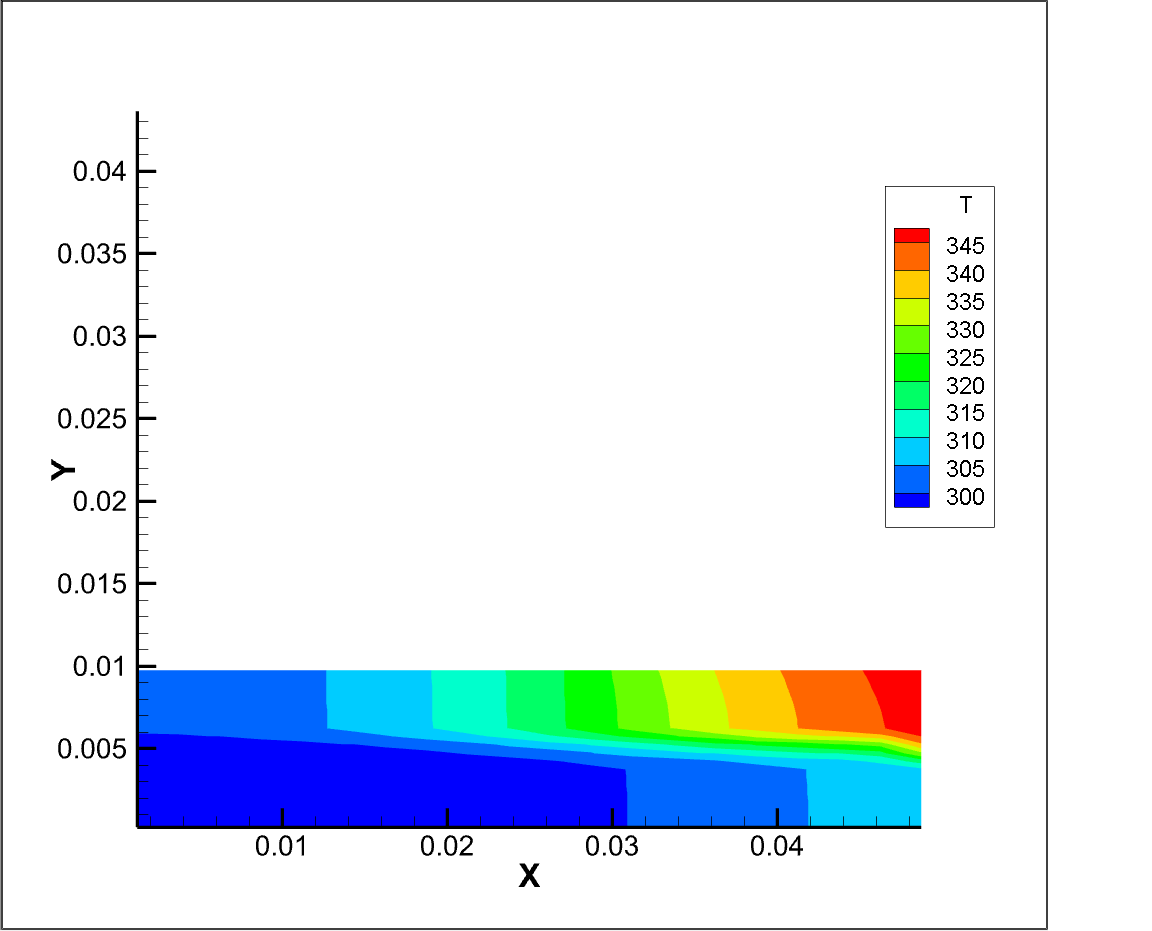
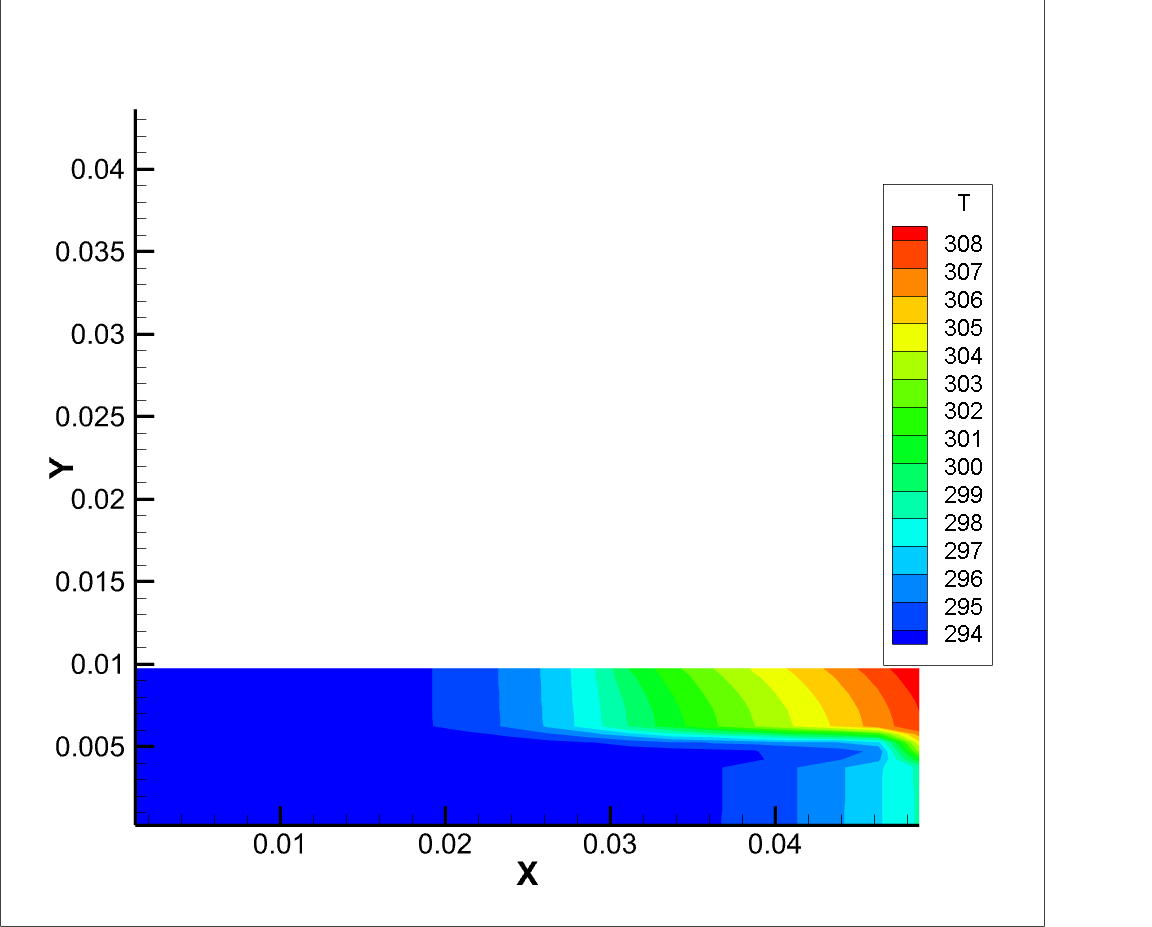
**两材料交界面的处理：**

****

两材料交界面处节点的导热系数****

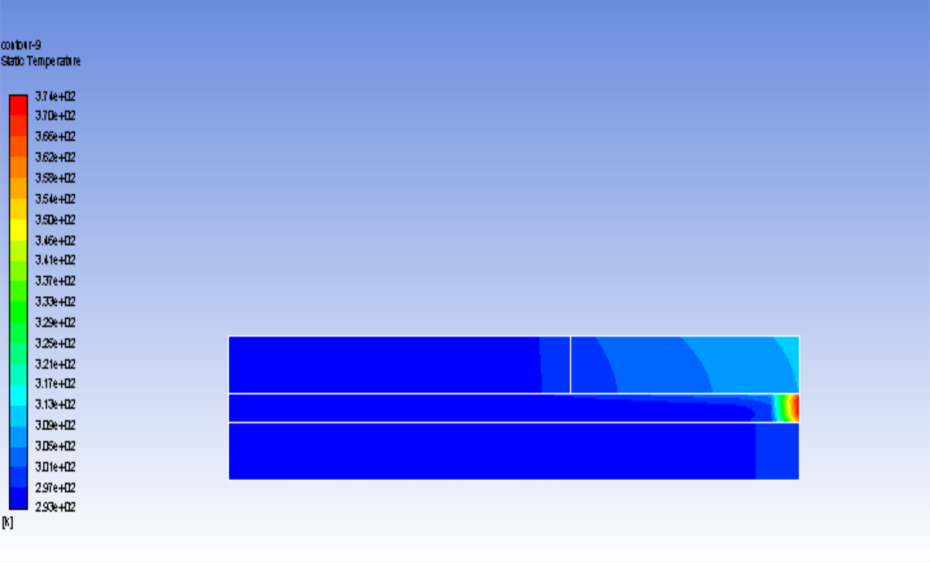
## 三、编程计算（代码见附录）

编程计算结果：

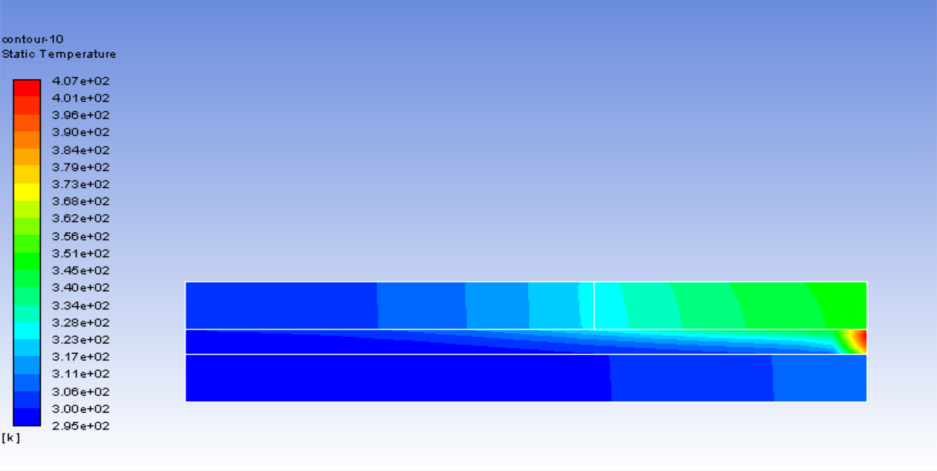


T=5 s温度分布 T=30 s温度分布

## 四、fluent仿真计算



T=5 s温度分布



T=30 s温度分布

## 结论分析

从编程绘制的云图和fluent仿真得到的云图都可以看到，由于高温辐射区域在上边的右半部分和右侧区域，所以每一块材料的温度分布都是从右上往左下依次递减的。而温度最高点出现在材料二的右上角而不是材料一的右上角，可能原因是材料一的热传导系数远大于材料二的热传导系数，且材料二的传导系数较小，温度传导速度慢，导致温度最高。

我们还可以看到编程得到的温度比仿真得到的温度低一些，由于边界条件和网格划分均是相同的，所以很有可能是这两种算法在进行迭代时的迭代误差导致的。

## 六、附录

详细代码

#!usr/bin/env python

# -\*- coding:utf-8 \_\*-

"""

@author:tianchang

@file: 膏体14-10.py

@time: 2019/11/16 14:10-20s

"""

# 添加中断后继续的代码部分

# 二维非稳态导热问题的有限体积数值解法#

from mpl\_toolkits.mplot3d import Axes3D

from scipy import linalg

import matplotlib.pyplot as plt

import numpy as np

# 定义求解函数，形参为网格数目

import os

def 打开tecplot(N\_x\_grid, N\_y\_grid, delta\_t, eff\_1, h, eff\_2):

args = 'cd "内流场大作业代码上边界对流eff\_1={3} eff\_2={5} h={4}"&&"T-2d-Nx={0}-Ny={1}-t={2}.plt"'.format(

N\_x\_grid, N\_y\_grid, (int(5 / delta\_t)) \* 0.004, eff\_1, h, eff\_2)

os.popen(args)

args = 'cd "内流场大作业代码上边界对流eff\_1={3} eff\_2={5} h={4}"&&"T-2d-Nx={0}-Ny={1}-t={2}.plt"'.format(

N\_x\_grid, N\_y\_grid, (int(30 / delta\_t)) \* 0.004, eff\_1, h, eff\_2)

os.popen(args)

def 数据写入文件(

eff\_1,

eff\_2, h,

N\_x\_grid,

N\_y\_grid, k, X,

Y,

T,

a\_p,

a\_n,

a\_s,

a\_w,

a\_e,

sp,

su):

'''# 判断是否存在文件夹，不存在则建立'''

if not os.path.exists(

r'./内流场大作业代码上边界对流eff\_1={0} eff\_2={2} h={1}'.format(eff\_1, h, eff\_2)):

os.mkdir(

r'./内流场大作业代码上边界对流eff\_1={0} eff\_2={2} h={1}'.format(eff\_1, h, eff\_2))

with open(

r'./内流场大作业代码上边界对流eff\_1={3} eff\_2={5} h={4}/T-2d-Nx={0}-Ny={1}-t={2}.plt'.format(

N\_x\_grid, N\_y\_grid, (k + 1) \* 0.004, eff\_1, h, eff\_2), 'w', encoding='UTF-8') as fp1:

fp1.write("VARIABLES = X, Y, T, ap,an,as,aw,ae,sp,su\n") # 按计算节点数输出结果

fp1.write("ZONE I=%d,J=%d, F=POINT,t=\"%.3f\"\n" %

(N\_x\_grid, N\_y\_grid, (k + 1) \* 0.004))

for j in range(N\_y\_grid - 1, -1, -1):

for i in range(0, N\_x\_grid):

fp1.write(

"{:.5f} {:.5f} {:.3f} {:.3f} {:.3f} {:.3f} {:.3f} {:.3f} {:.3f} {:.3f}\n".format(

X[i][j],

Y[i][j],

T[i][j],

a\_p[i][j],

a\_n[i][j],

a\_s[i][j],

a\_w[i][j],

a\_e[i][j],

sp[i][j],

su[i][j]))

def solve(N\_x\_grid, N\_y\_grid, t, eff\_1, eff\_2, h):

'''N\_x\_grid, N\_y\_grid, t, eff, h'''

# 物性参数定义：发动机长度，高温辐射区域长度，两层材料的宽度、热传导系数、密度、比热容，燃气温度，发射率，对流换热系数

L = 0.05

L1 = 0.02

HH = 0.004

hh = 0.002

k1 = 53.6

k2 = 0.2093

k0 = 0.41697

rou1 = 7830

rou2 = 1500

Cp1 = 465

Cp2 = 1465

Tair = 293

Tgas = 923

T0 = 293 # 初始温度

# eff = 0.8 # 假定发射率

# h = 10 # 假定导热系数

sigma = 5.67 \* 10 \*\* -8

# 网格尺寸定义：

delta\_x = L / N\_x\_grid # 0.0025

delta\_y = (2 \* HH + hh) / N\_y\_grid # 0.0005

# 时间步长及迭代次数：

delta\_t = 0.004

cal\_num = int(t / delta\_t)

# 定义求解线性方程组的动态数组：

AA = np.zeros((N\_x\_grid \* N\_y\_grid, N\_x\_grid \* N\_y\_grid))

# print(AA)

CC = np.zeros((N\_x\_grid \* N\_y\_grid))

# 定义初始0矩阵:

a\_w = np.zeros((N\_x\_grid, N\_y\_grid))

a\_e = np.zeros((N\_x\_grid, N\_y\_grid))

a\_n = np.zeros((N\_x\_grid, N\_y\_grid))

a\_s = np.zeros((N\_x\_grid, N\_y\_grid))

sp = np.zeros((N\_x\_grid, N\_y\_grid))

su = np.zeros((N\_x\_grid, N\_y\_grid))

a\_p = np.zeros((N\_x\_grid, N\_y\_grid))

a\_p0 = np.zeros((N\_x\_grid, N\_y\_grid))

T = np.zeros((N\_x\_grid, N\_y\_grid)) X = np.zeros((N\_x\_grid, N\_y\_grid))

Y = np.zeros((N\_x\_grid, N\_y\_grid))

# 离散方程系数计算

for k in range(0, cal\_num):

if k == 0:

for i in range(0, N\_x\_grid):

for j in range(0, N\_y\_grid):

T[i][j] = T0

for i in range(0, N\_x\_grid):

for j in range(0, N\_y\_grid):

# 结点位置确定

Y[i][j] = delta\_y \* (j + 0.5)

X[i][j] = delta\_x \* (i + 0.5)

# 给定初始温度场

# 判断结点在哪一区域，给出a\_e,a\_w,a\_s,a\_n,a\_p相应系数

if j <= 0.4 \* N\_y\_grid - 1 or j >= 0.6 \* N\_y\_grid: # 0——7，12——19

a\_w[i][j] = k1 / delta\_x \* delta\_y # 10.72

a\_e[i][j] = k1 / delta\_x \* delta\_y # 10.72

a\_s[i][j] = k1 / delta\_y \* delta\_x # 268

a\_n[i][j] = k1 / delta\_y \* delta\_x # 268

a\_p[i][j] = rou1 \* Cp1 \* delta\_x \* delta\_y / delta\_t

else: # 8——11

a\_w[i][j] = k2 / delta\_x \* delta\_y

a\_e[i][j] = k2 / delta\_x \* delta\_y

a\_s[i][j] = k2 / delta\_y \* delta\_x

a\_n[i][j] = k2 / delta\_y \* delta\_x

a\_p[i][j] = rou2 \* Cp2 \* delta\_x \* delta\_y / delta\_t

# 交界面处系数

if j == 0.6 \* N\_y\_grid - 1 or j == 0.4 \* N\_y\_grid - 1: # 7，11

a\_n[i][j] = k0 / delta\_y \* delta\_x

if j == 0.6 \* N\_y\_grid or j == 0.4 \* N\_y\_grid: # 8，12

a\_s[i][j] = k0 / delta\_y \* delta\_x

# 边界条件

if i == 0: # 左边界为绝热边界

a\_w[i][j] = 0.0

su[i][j] = 0.0

sp[i][j] = 0.0

if i == N\_x\_grid - 1: # 右边界为辐射换热边界

if j <= 0.4 \* N\_y\_grid - 1 or j >= 0.6 \* N\_y\_grid: # 0——7，12——19

a\_e[i][j] = 0.0

# 将边界结点的温度作为壁面温度

su[i][j] = eff\_1 \* sigma \* \

(Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_y

sp[i][j] = 0.0

else:

a\_e[i][j] = 0.0

# 将边界结点的温度作为壁面温度

su[i][j] = eff\_2 \* sigma \* \

(Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_y

sp[i][j] = 0.0

if j == 0: # 下边界为绝热边界

a\_s[i][j] = 0.0

su[i][j] = 0.0

sp[i][j] = 0.0

if j == N\_y\_grid - 1: # 上边界为混合边界

if i <= N\_x\_grid \* 0.6 - 1: # 上边界左侧为对流换热

a\_n[i][j] = 0.0

su[i][j] = h \* (Tair - T[i][j]) \* delta\_x

sp[i][j] = 0

else: # 改动1： 上边界右侧为辐射换热加对流换热

a\_n[i][j] = 0.0

su[i][j] = eff\_1 \* sigma \* \

(Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_x + h \* (Tair - T[i][j]) \* delta\_x

sp[i][j] = 0.0

# 四个角点系数替换

if i == 0 and j == 0: # 左下角点

su[i][j] = 0.0

if i == 0 and j == N\_y\_grid - 1: # 左上角点

su[i][j] = h \* (Tair - T[i][j]) \* delta\_x

if i == N\_x\_grid - 1 and j == N\_y\_grid - 1: # 右上角点

su[i][j] = eff\_1 \* sigma \* (Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_x + eff\_1 \* sigma \* (

Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_y + h \* (Tair - T[i][j]) \* delta\_x

if i == N\_x\_grid - 1 and j == 0: # 右下角点

su[i][j] = eff\_1 \* sigma \* \

(Tgas \*\* 4 - (T[i][j]) \*\* 4) \* delta\_y

a\_p0[i][j] = a\_p[i][j] - \

(a\_w[i][j] + a\_e[i][j] + a\_s[i][j] + a\_n[i][j] - sp[i][j])

# 将系数给入系数矩阵AA,CC

for i in range(0, N\_x\_grid):

for j in range(0, N\_y\_grid):

AA[i \* N\_y\_grid + j][i \* N\_y\_grid + j] = a\_p[i][j]

CC[i \* N\_y\_grid + j] = su[i][j] + a\_p0[i][j] \* T[i][j]

if j != 0:

CC[i \* N\_y\_grid + j] = CC[i \* N\_y\_grid + j] + \

a\_s[i][j] \* T[i][j - 1]

if j != N\_y\_grid - 1:

CC[i \* N\_y\_grid + j] = CC[i \* N\_y\_grid + j] + \

a\_n[i][j] \* T[i][j + 1]

if i != 0:

CC[i \* N\_y\_grid + j] = CC[i \* N\_y\_grid + j] + \

a\_w[i][j] \* T[i - 1][j]

if i != N\_x\_grid - 1:

CC[i \* N\_y\_grid + j] = CC[i \* N\_y\_grid + j] + \

a\_e[i][j] \* T[i + 1][j]

result = linalg.solve(AA, CC) # 求解温度场

T\_对流求和 = 0

T\_对流求和\_i = 0

for i in range(0, N\_x\_grid):

for j in range(0, N\_y\_grid):

T[i][j] = result[i \* N\_y\_grid + j]

if j == N\_y\_grid - 1: # 上边界为混合边界

if i <= N\_x\_grid \* 0.6 - 1: # 上边界左侧为对流换热

T\_对流求和 += T[i][j]

T\_对流求和\_i += 1

#print (T\_对流求和, T\_对流求和\_i)

delta\_t\_温差 = T\_对流求和 / T\_对流求和\_i - 293

数据写入文件(eff\_1, eff\_2,h,N\_x\_grid,N\_y\_grid,k,X,Y,T,a\_p,a\_n,a\_s,a\_w,a\_e,sp, su)

print('已完成:{:.2f}% t\_温差:{:.3f}'.format(k / cal\_num \* 100, delta\_t\_温差))

打开tecplot(N\_x\_grid, N\_y\_grid, delta\_t, eff\_1, h, eff\_2) # 最后调用

# return (X, Y, T)'''

# 调用函数的主程序

solve(20, 20, 30, 0.8, 0.3, 0.3) # eff\_1=0.3,h=10