```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: """
   ...: Created on 20200408
   ...:
   ...: @author: weililai
   ...:
   ...: #一、外表面换热系数
   ...: W = 0.8
   ...: alpha_r = 5.669*epsilon/(T_s-T_a)*(((273+T_s)/100)**4-((273+T_a)/100)**4) #辐
射换热系数
   ...: if W == 0:
   ...:     alpha_c = 26.4/(297+0.5(T_s+T_a))**0.5*((T_s-T_a)/D_2)**0.25 #无风时的对流
换热系数
   ...: elif W*D_2 > 0.8: #上一版本，20200405版中错写为W > 0.8，并按此条件计算，这一版更
正！！！！！！
   ...:     alpha_c = 4.53*W**0.805/D_2**0.195 #风速大于0.8时的对流换热系数
   ...: else:
   ...:     alpha_c = 0.008/D_2+4.2*W**0.618/D_2**0.382 #有风，但风速小于0.8时的对流换热
系数
   ...:
   ...: alpha_s = alpha_r + alpha_c #外表面换热系数应为辐射换热系数与对流换热系数之和
   ...: #二、一堆传热方程：
   ...: #1.保温层外表面与环境 的传热
   ...: q = (T_s-T_a)*math.pi*D_2*alpha_s
   ...: #2.内外保温层界面处-保温层外表面 的传热
   ...: q = (T_1-T_s)/np.log(D_2/D_1)*2*math.pi*lamb_2
   ...: #3.管道壁面-内外保温层界面处 的传热
   ...: q = (T_0-T_1)/np.log(D_1/D_0)*2*math.pi*lamb_1
   ...: #三、两种保温材料的导热系数方程
   ...: lamb_1 = (    k1_0 + (T_0/2+T_1/2)*k1_1 + (T_0/2+T_1/2)**2*k1_2       )
   ...: lamb_2 = (    k2_0 + (T_1/2+T_s/2)*k2_1 + (T_1/2+T_s/2)**2*k2_2       )
   ...: """
   ...:
   ...: from scipy.optimize import root,fsolve
   ...: import numpy as np
   ...: import math
   ...: from matplotlib import pyplot as plt
   ...:
   ...: #设定（除温度单位为摄氏度外，其他单位都按SI国际单位制）
   ...: D_0 = 0.3 #管径
   ...: T_0 = 400 #介质温度（近似为管壁温度）
   ...: T_a = 20 #环境温度
   ...: T_1 = 350 #两种保温材料界面处温度，不得高于0.9倍外层材料最高使用温度，建议0.8倍
   ...: q_max = 285 #最大热损（W/m）；线热损和面热损转换： q=pi*D_2*Q
   ...: T_s_max = 45 #最高表面温度：有时就是防烫伤温度
   ...: epsilon = 0.25 #镀锌钢板的黑度，其他外护层材料参考GB50264-2013的5.8.9
   ...: W = 1 #风速
   ...: k1_0 = 0.03 #内层材料导热系数方程多项式0次项系数
   ...: k1_1 = 5*10**-5 #内层材料导热系数方程多项式1次项系数
   ...: k1_2 = 2*10**-7 #内层材料导热系数方程多项式2次项系数
   ...: k2_0 = 0.035 #外层材料导热系数方程多项式0次项系数
   ...: k2_1 = 5*10**-5 #外层材料导热系数方程多项式1次项系数
   ...: k2_2 = 3*10**-7 #外层材料导热系数方程多项式2次项系数
   ...:
```

```
   ...: #传热方程组设立，由热损q求解 T_s、D_1、D_2，外表面换热系数的选取方法参照标准
《GB50264-2013》
   ...: def GB50264_q_to_T_s(x):
   ...:     alpha_r = 5.669*epsilon/(x[0]-T_a)*(((273+x[0])/100)**4-((273+T_a)/
100)**4) #辐射换热系数
   ...:     if W == 0:
   ...:         alpha_c = 26.4/(297+0.5(x[0]+T_a))**0.5*((x[0]-T_a)/x[0])**0.25 #无风
时的对流换热系数
   ...:     elif W*x[2] > 0.8:
   ...:         alpha_c = 4.53*W**0.805/x[2]**0.195 #风速大于0.8时的对流换热系数
   ...:     else:
   ...:         alpha_c = 0.008/x[2]+4.2*W**0.618/x[2]**0.382 #有风，但风速小于0.8时的
对流换热系数
   ...:     alpha_s = alpha_r + alpha_c #外表面换热系数应为辐射换热系数与对流换热系数之和
   ...:     return np.array([(x[0]-T_a)*math.pi*x[2]*alpha_s-q,
   ...:                      (T_1-x[0])/np.log(x[2]/x[1])*2*math.pi*(k2_0 +
(T_1/2+x[0]/2)*k2_1 + (T_1/2+x[0]/2)**2*k2_2)-q,
   ...:                      (T_0-T_1)/np.log(x[1]/D_0)*2*math.pi*(k1_0 +
(T_0/2+T_1/2)*k1_1 + (T_0/2+T_1/2)**2*k1_2)-q])
   ...:
   ...:
   ...: #传热方程组设立，由表面温度T_s求解 q、D_1、D_2，外表面换热系数的选取方法参照标准
《GB50264-2013》
   ...: def GB50264_T_s_to_q(x):
   ...:     alpha_r = 5.669*epsilon/(T_s-T_a)*(((273+T_s)/100)**4-((273+T_a)/100)**4)
#辐射换热系数
   ...:     if W == 0:
   ...:         alpha_c = 26.4/(297+0.5(T_s+T_a))**0.5*((T_s-T_a)/T_s)**0.25 #无风时的
对流换热系数
   ...:     elif W*x[2] > 0.8:
   ...:         alpha_c = 4.53*W**0.805/x[2]**0.195 #风速大于0.8时的对流换热系数
   ...:     else:
   ...:         alpha_c = 0.008/x[2]+4.2*W**0.618/x[2]**0.382 #有风，但风速小于0.8时的
对流换热系数
   ...:     alpha_s = alpha_r + alpha_c #外表面换热系数应为辐射换热系数与对流换热系数之和
   ...:     return np.array([(T_s-T_a)*math.pi*x[2]*alpha_s-x[0],
   ...:                      (T_1-T_s)/np.log(x[2]/x[1])*2*math.pi*(k2_0 +
(T_1/2+T_s/2)*k2_1 + (T_1/2+T_s/2)**2*k2_2)-x[0],
   ...:                      (T_0-T_1)/np.log(x[1]/D_0)*2*math.pi*(k1_0 +
(T_0/2+T_1/2)*k1_1 + (T_0/2+T_1/2)**2*k1_2)-x[0]])
   ...:
   ...:
   ...: #传热方程组设立，由D_1、D_2求解 q、T_1、T_s，外表面换热系数的选取方法参照标准
《GB50264-2013》
   ...: def GB50264_D_to_T_s(x):
   ...:     alpha_r = 5.669*epsilon/(x[2]-T_a)*(((273+x[2])/100)**4-((273+T_a)/
100)**4) #辐射换热系数
   ...:     if W == 0:
   ...:         alpha_c = 26.4/(297+0.5(x[2]+T_a))**0.5*((x[2]-T_a)/x[2])**0.25 #无风
时的对流换热系数
   ...:     elif W*D_2 > 0.8:
   ...:         alpha_c = 4.53*W**0.805/D_2**0.195 #风速大于0.8时的对流换热系数
   ...:     else:
   ...:         alpha_c = 0.008/D_2+4.2*W**0.618/D_2**0.382 #有风，但风速小于0.8时的对
流换热系数
   ...:     alpha_s = alpha_r + alpha_c #外表面换热系数应为辐射换热系数与对流换热系数之和
   ...:     return np.array([(x[2]-T_a)*math.pi*D_2*alpha_s-x[0],
   ...:                      (x[1]-x[2])/np.log(D_2/D_1)*2*math.pi*(k2_0 + (x[1]/
```

```
2+x[2]/2)*k2_1 + (x[1]/2+x[2]/2)**2*k2_2)-x[0],
   ...:                             (T_0-x[1])/np.log(D_1/D_0)*2*math.pi*(k1_0 +
(T_0/2+x[1]/2)*k1_1 + (T_0/2+x[1]/2)**2*k1_2)-x[0]])
   ...:
   ...:
   ...:
   ...: #传热方程组求解，由最大热损q_max求解 T_s、D_1、D_2
   ...: q = q_max
   ...: sol_root1 = root(GB50264_q_to_T_s,[60,D_0+0.02,D_0+0.04])
   ...: sol_fsolve1 = fsolve(GB50264_q_to_T_s,[60,D_0+0.02,D_0+0.04])
   ...: T_s = sol_fsolve1[0]
   ...: D_1 = sol_fsolve1[1]
   ...: D_2 = sol_fsolve1[2]
   ...: print(" ")
   ...: print("solution :",sol_root1)
   ...: print(">>>>>>>>>>> When the q is ",q," :")
   ...: print("  T_s = ",T_s)
   ...: print("  D_1 = ",D_1)
   ...: print("  D_2 = ",D_2)
   ...:
   ...: #当由热损q求解 出的T_s>T_s_max时，重新由表面温度T_s_max求解 q、D_1、D_2
   ...: if T_s > T_s_max:
   ...:     T_s=T_s_max
   ...:     sol_root2 = root(GB50264_T_s_to_q,[q,D_1,D_2])
   ...:     sol_fsolve2 = fsolve(GB50264_T_s_to_q,[q,D_1,D_2])
   ...:     q = sol_fsolve2[0]
   ...:     D_1 = sol_fsolve2[1]
   ...:     D_2 = sol_fsolve2[2]
   ...:     print(" ")
   ...:     print("solution :",sol_root2)
   ...:     print(">>>>>>>>>>> When the T_s is ",T_s_max," :")
   ...:     print("  q = ",q)
   ...:     print("  D_1 = ",D_1)
   ...:     print("  D_2 = ",D_2)
   ...:
   ...:
   ...: #毡材厚度为1cm时，可行的实际保温层厚度
   ...: delta = round(D_2/2-D_0/2+0.005,2)
   ...: delta_1 = round(D_1/2-D_0/2+0.005,2)
   ...: delta_2 = round(delta - delta_1,2)
   ...: print(" ")
   ...: print(">>>>>>>>>>> When the blanket is 1 cm thick :")
   ...: print("  delta = ",delta)
   ...: print("  delta_1 = ",delta_1)
   ...: print("  delta_2 = ",delta_2)
   ...:
   ...: #传热方程组求解，由D_1、D_2求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*delta
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: print(" ")
   ...: print("solution :",sol_root3)
   ...: print(">>>>>>>>>>> When the delta_1 and delta_2 are ",delta_1,delta_2," :")
   ...: print("  delta_1 = ",delta_1)
   ...: print("  delta_2 = ",delta_2)
   ...: print("       q  = ",q)
   ...: print("      T_1 = ",T_1)
```

```
   ...: print("       T_s = ",T_s)
   ...:
   ...:
   ...:
   ...: #以列表的形式展现相近结构方案的热损、界面温度、表面温度计算结果
   ...: #传热方程组求解，由D_1、D_2-0.04求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*(delta-0.02)
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: print(" ")
   ...: print("structure delta_1 delta_2 q T_1 T_s")
   ...: #print("solution :",sol_root3)
   ...: print(1,round(delta_1,2),round(delta_2-0.02,2),round(q,
2),round(T_1,2),round(T_s,2))
   ...:
   ...: #传热方程组求解，由D_1、D_2-0.02求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*(delta-0.01)
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: #print("solution :",sol_root3)
   ...: print(2,round(delta_1,2),round(delta_2-0.01,2),round(q,
2),round(T_1,2),round(T_s,2))
   ...:
   ...: #传热方程组求解，由D_1、D_2求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*delta
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: #print("solution :",sol_root3)
   ...: print(3,round(delta_1,2),round(delta_2,2),round(q,2),round(T_1,2),round(T_s,
2))
   ...:
   ...: #传热方程组求解，由D_1、D_2+0.02求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*(delta +0.01)
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: #print("solution :",sol_root3)
   ...: print(4,round(delta_1,2),round(delta_2+0.01,2),round(q,
2),round(T_1,2),round(T_s,2))
   ...:
   ...: #传热方程组求解，由D_1、D_2+0.04求解 q、T_1、T_s
   ...: D_1 = D_0 + 2*delta_1
   ...: D_2 = D_0 + 2*(delta +0.02)
   ...: sol_root3 = root(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: sol_fsolve3 = fsolve(GB50264_D_to_T_s,[q,T_1,T_s])
   ...: q = sol_fsolve3[0]
```

```
   ...: T_1 = sol_fsolve3[1]
   ...: T_s = sol_fsolve3[2]
   ...: #print("solution :",sol_root3)
   ...: print(5,round(delta_1,2),round(delta_2+0.02,2),round(q,
2),round(T_1,2),round(T_s,2))
   ...:

solution :     fjac: array([[-9.29272434e-01,  3.69395077e-01,  1.43601268e-04],
       [-3.62472524e-01, -9.11932416e-01,  1.92335483e-01],
       [-7.11787352e-02, -1.78680011e-01, -9.81329222e-01]])
     fun: array([ 1.17539685e-08,  8.47752517e-08, -5.23630206e-09])
 message: 'The solution converged.'
    nfev: 21
     qtf: array([-1.41064450e-06,  5.64672542e-06,  7.39444275e-07])
       r: array([-9.72402915e+00,  3.57525948e+04, -5.89650649e+02, -9.16773485e+04,
        6.36746330e+02,  1.96689705e+02])
  status: 1
 success: True
       x: array([46.33417775,  0.32653032,  0.47725037])
>>>>>>>>>>> When the q is  285  :
  T_s =  46.33417775190708
  D_1 =  0.3265303182703386
  D_2 =  0.47725037255656816

solution :     fjac: array([[-0.57776483, -0.55722356, -0.59639727],
       [-0.29626613, -0.53768639,  0.78937933],
       [ 0.76053546, -0.63276792, -0.14556982]])
     fun: array([-5.68434189e-14,  5.17275112e-12, -5.00222086e-12])
 message: 'The solution converged.'
    nfev: 10
     qtf: array([ 4.47514626e-09, -1.15104880e-07, -4.25616660e-08])
       r: array([ 1.73144701e+00,  4.48289799e+03,  5.65064202e+02, -8.73937605e+03,
        6.28694388e+02,  1.19411744e+03])
  status: 1
 success: True
       x: array([273.92179621,   0.32765131,   0.48675689])
>>>>>>>>>>> When the T_s is  45  :
  q =  273.92179620570613
  D_1 =  0.3276513059012274
  D_2 =  0.4867568859181716

>>>>>>>>>>> When the blanket is 1 cm thick :
  delta =  0.1
  delta_1 =  0.02
  delta_2 =  0.08

solution :     fjac: array([[-0.57899374, -0.58153036, -0.57147939],
       [-0.20475995, -0.57474253,  0.79230322],
       [ 0.78920188, -0.5757547 , -0.2136982 ]])
     fun: array([-6.99742486e-11,  9.03810360e-12, -6.26982910e-11])
 message: 'The solution converged.'
    nfev: 9
     qtf: array([ 2.29600201e-08, -1.12214497e-08, -1.57496837e-08])
       r: array([ 1.72761931,  1.21597393, -6.38174598, -3.45015954, -1.98991132,
        9.35221617])
  status: 1
 success: True
       x: array([257.92376171, 331.53596942,  43.14867778])
>>>>>>>>>>> When the delta_1 and delta_2 are  0.02 0.08  :
  delta_1 =  0.02
  delta_2 =  0.08
       q  =  257.9237617149663
```

```
      T_1 =  331.5359694212691
      T_s =  43.14867777896561

structure delta_1 delta_2 q T_1 T_s
1 0.02 0.06 303.62 317.96 48.71
2 0.02 0.07 278.46 325.5 45.65
3 0.02 0.08 257.92 331.54 43.15
4 0.02 0.09 240.84 336.48 41.08
5 0.02 0.1 226.39 340.62 39.33
```

```
In [2]:
```