```
In [10]: """
    ...: Created on Fri Mar  6 23:09:40 2020
    ...:
    ...: @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 > 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
    ...: #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 = (   0.03 + (T_0/2+T_1/2)*5*10**-5 + (T_0/2+T_1/2)**2*2*10**-7
)
    ...: lamb_2 = (   0.035 + (T_1/2+T_s/2)*5*10**-5 + (T_1/2+T_s/2)**2*3*10**-7
)
    ...:
    ...: """
    ...:
    ...: 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 = 0.15 #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 = 0.8 #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 > 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 > 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 > 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([[-0.96152301,  0.27396839, -0.02036721],
       [-0.20260896, -0.65710253,  0.7260619 ],
       [-0.18553466, -0.7022518 , -0.68732765]])
     fun: array([ 1.48975232e-09,  2.37491804e-10, -2.79840151e-10])
 message: 'The solution converged.'
    nfev: 18
     qtf: array([4.40429758e-08, 1.96781622e-08, 6.38110268e-09])
       r: array([-5.80760832e+00,  4.79717450e+03, -1.46089898e+02, -2.78354113e+04,
        2.93379225e+03,  2.62462226e+03])
  status: 1
 success: True
       x: array([67.03746609,  0.16326516,  0.23512724])
>>>>>>>>>>> When the q is  285  :
  T_s =  67.03746608972276
  D_1 =  0.1632651591349287
  D_2 =  0.23512723718565534

solution :     fjac: array([[-0.54478703, -0.49951986, -0.67356292],
       [-0.2563694 , -0.66555485,  0.70093614],
       [ 0.79842459, -0.55454184, -0.23452404]])
     fun: array([-1.24202870e-11,  1.48443746e-09, -1.51479185e-09])
 message: 'The solution converged.'
    nfev: 15
     qtf: array([ 1.36361432e-07, -9.62300408e-07, -2.23232986e-07])
       r: array([ 1.83785516e+00,  4.94640263e+03,  3.65388080e+02, -1.04226160e+04,
       -4.66045967e+02,  5.98074632e+02])
  status: 1
 success: True
       x: array([1.80483596e+02, 1.71476799e-01, 3.12679967e-01])
>>>>>>>>>>> When the T_s is  45  :
  q =  180.48359589170963
  D_1 =  0.17147679877331837
  D_2 =  0.31267996669429376

>>>>>>>>>>> When the blanket is 1 cm thick :
  delta =  0.09
  delta_1 =  0.02
  delta_2 =  0.07

solution :     fjac: array([[-0.5795571 , -0.58508689, -0.56726264],
       [-0.15027128, -0.60742216,  0.78003645],
       [ 0.800957  , -0.53731895, -0.26411407]])
     fun: array([-4.91382934e-10,  7.37259143e-11, -4.39513315e-10])
 message: 'The solution converged.'
    nfev: 9
     qtf: array([ 9.67597585e-08, -4.13514898e-08, -6.62531876e-08])
       r: array([ 1.72622371,  0.48426458, -4.28562794, -1.87088172, -0.85235182,
        6.36126317])
  status: 1
 success: True
       x: array([166.03742783, 314.9286749 ,  42.21567007])
>>>>>>>>>>> When the delta_1 and delta_2 are  0.02 0.07  :
  delta_1 =  0.02
```

```
  delta_2 =  0.07
       q  =  166.03742782727693
      T_1 =  314.92867489714
      T_s =  42.21567006579727

structure delta_1 delta_2 q T_1 T_s
1 0.02 0.05 195.07 297.77 48.38
2 0.02 0.06 178.94 307.4 44.94
3 0.02 0.07 166.04 314.93 42.22
4 0.02 0.08 155.46 320.98 40.01
5 0.02 0.09 146.63 325.95 38.19
```

In [11]: