赛区评阅编号（由赛区组委会填写）：

2018 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》(以下简称为“竞赛章程和参赛规则”, 可从全国大学生数学建模竞赛网站下载)。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛章程和参赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D 中选择一项填写）：A

我们的报名参赛队号（12 位数字全国统一编号）：201809002232

参赛学校（完整的学校全称，不含院系名）：上海交通大学

参赛队员 (打印并签名)：1. 钱文涛

2. 盛志耀

3. 王思极

指导教师或指导教师组负责人 (打印并签名)：尚建辉老师

日期：2018 年 9 月 14 日

(此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。)

赛区评阅编号（由赛区组委会填写）：

2018 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号（由赛区组委会填写）：

全国评阅随机编号（由全国组委会填写）：

（此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页，即电子版论文的第一页为标题、摘要和关键词页。）

高温工作温度分布模型

摘要

人们在进行高温作业时需要穿着专业的高温工作服以防危险。通常的高温工作服由三层织物材料构成，本文将研究高温工作服的隔层材料的导热性质，建立数学模型模拟来确定假人皮肤外侧的温度变化情况。

针对问题一，首先建立系统的热传导方程，系统包括：人体、高温工作服以及外界热源。再考虑到人体表面接触高温工作服，高温工作服同时与外界热源接触，所以不妨将我们的方程简化成为一维热传导方程。

在建立完一维热传导方程之后，结合已知的边界条件，就可以通过求解微分方程得到系统的温度分布。再将已经求解得到的温度分布与“附件 2. 假人皮肤外侧的测量温度”进行比对，如果对比结果直接满足，那可以继续进行问题二、三的求解；反之，则需要对建立的模型进行修正。

最后经过对比，发现数据的拟合程度确实存在一定的出入，故考虑进行修正。修正一：在工作服表面和 75 度热源之间有一层薄空气层，记作 V 层；修正二：人体内环境和工作服之间存在人体组织，记作 0 层。

修正之后，再度求解微分方程。最终发现此时的修正结果与附件所给数据符合得很好。

针对问题二，结合已经求解得到的温度分布模型，修改初始条件为：环境温度为 $65°C$、IV 层的厚度为 $5.5mm$，确定出合适的 II 层厚度，使得确保工作 60 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。

针对问题三，结合已经求解得到的温度分布模型，修改初始条件为：环境温度为 $80°C$，确定出合适的 II 层和 IV 层厚度，使得确保工作 60 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。

最后再总体考虑所以材料的参数进行微修正，并做一定的回归分析，控制模型的误差。

关键字：热传导方程　温度分布　误差分析

一、问题的提出

实际生活中从事高温环境工作的人员需要穿着专业的高温工作服以防被烧伤。那么根据现有材料的相关参数模拟得出合适的服装设计就十分的重要。另外，热传导方程是一个我们熟知的偏微分方程，它具体描述了一个区域内的温度如何随时间变化。

二、问题重述

现根据附件所提供的专用服装材料的参数值以及假人皮肤外侧的测量温度数据，建立数学模型，解决以下问题:

1. 利用附件 problem1.xlsx 中的相关材料参数对附件中的家人皮肤测量温度进行模拟，建立模型，计算温度分布，并生成温度分布文件。
2. 利用在问题一中提出的模型，当环境温度为 $65°C$、IV 层的厚度为 $5.5mm$ 时，确定 II 层的最优厚度，确保工作 60 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。
3. 利用在问题一种提出的模型，当环境温度为 $80°C$ 时，确定 II 层和 IV 层的最优厚度，确保工作 30 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。

三、模型的假设

- 从一维情况考虑。
- 不考虑热辐射和对流。
- 假设 75 度是一个稳定热源的温度，它和外层织物存在一个空气层，那一层的初始温度是 75 度，充当了衣物和 75 度热源之间的缓冲层。
- 人的皮肤和内部温度存在一层组织充当热传导材料。
- 人体组织和所有衣物以及衣物和人体间的空气初始温度都是 $37°C$。
- 所有过程满足热传导方程。

四、符号说明

符号	意义
u	温度 (°C)
s	厚度 (cm)
t	时间 (second)
T_1	人体内部温度 (37°C)
T_2	理想外部热源温度 (75°C)
k	热扩散率 (m^2/s))
ρ	密度 (kg/m^2))
C	比热 $(J/(kg \cdot^\circ C))$
λ	热传导率 $(W/(m \cdot^\circ C)$

另外，方便起见，我们把衣物和外部热源之间的空气层、IV 层、III，II 层、I 层、人体组织层分别记作第 0 层、第 1 层、第 2 层、第 3 层、第 4 层、第 5 层。

五、问题分析

5.1 问题一分析

针对问题一，首先建立系统的热传导方程，系统包括：人体、高温工作服以及外界热源。再考虑到人体表面接触高温工作服，高温工作服同时与外界热源接触，大多数接触面的弧度不大，所以不妨将我们的方程简化成为一维热传导方程。

在建立完一维热传导方程之后，结合已知的边界条件，就可以通过求解微分方程得到系统的温度分布。再将已经求解得到的温度分布与“附件 2. 假人皮肤外侧的测量温度”进行比对，如果对比结果直接满足，那可以继续进行问题二、三的求解；反之，则需要对建立的模型进行修正。

最后经过对比，发现数据的拟合程度确实存在一定的出入，故考虑进行修正。修正一：在工作服表面吸附有一层薄空气层，记作 V 层；修正二：人体内环境和工作服之间存在人体组织，记作 0 层。

修正之后，再度求解微分方程。最终发现此时的修正结果与附件所给数据符合得很好。

5.2 问题二分析

结合已经求解得到的温度分布模型，修改初始条件为：环境温度为 $65°C$、IV 层的厚度为 $5.5mm$，确定出合适的 II 层厚度，使得确保工作 60 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。

5.3 问题三分析

结合已经求解得到的温度分布模型，修改初始条件为：环境温度为 80℃，确定出合适的 II 层和 IV 层的厚度，使得确保工作 60 分钟时，假人皮肤外侧温度不超过 $47°C$，且超过 $44°C$ 的时间不超过 5 分钟。

六、模型描述

首先，我们根据热扩散率的公式 $k = \frac{\lambda}{C\rho}$ 可以计算得到各层的热扩散系数 k，得到下表格:

专用服装材料的参数值					
分层	密度 (kg/m^3)	比热 $(J/(kg \cdot^\circ C))$	热传导率 $(W/(m \cdot^\circ C))$	厚度 (mm)	热扩散系数 (m^2/s)
I 层	300	1377	0.082	0.6	1.985×10^{-7}
II 层	862	2100	0.37	0.6-25	2.044×10^{-7}
III 层	74.2	1726	0.045	3.6	3.5137×10^{-7}
IV 层	1.18	1005	0.028	0.6-6.4	2.3611×10^{-5}

表 1　专用服装材料的参数值表

根据热传导方程:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial^2 s} \tag{1}$$

我们通过换元，令 $s = \sqrt{k}x$, 即 $x = sk^{-\frac{1}{2}}$，称 x 为等效位移，厚度经过这个换元之后成为等效厚度，得到:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x} \tag{2}$$

把第 0、1、2、3、4、5 层的等效厚度分别记为 $d_0, d_1, d_2, d_3, d_4, d_5$, 它们的总等效厚度记为 a, 把第 0 层的等效厚度和 a 的比例记作 α。我们假定人体和所有衣物以及他们中间的空气层的初始温度为 37 度，而第 0 层 (外部空气) 的初始温度为 75 度，可得初始条件如下:

$$u(x,0) = \begin{cases} T_1 & 0 < x < \alpha a \\ T_2 & \alpha a < x < a \end{cases} \tag{3}$$

以及边界条件如下:

$$\begin{cases} u(0,t) = T_1 \\ u(a,t) = T_2 \end{cases} \tag{4}$$

根据微分方程理论，方程 (1) 的通解具有如下形式：

$$\begin{cases} (c_1\sin(\omega x)+c_2\cos(\omega x))e^{-\omega^2 t} \\ d_1 x+d_2 \\ (c_1 e^{\omega x}+c_2 e^{-\omega x})e^{\omega^2 t} \end{cases} \tag{5}$$

根据边界条件，选择 $u(x,t)$ 的形式为：

$$u(x,t)=\frac{T_2-T_1}{a}x+T_1+\sum_{n=1}^{\infty}A_n\sin(\frac{n\pi}{a}x)e^{-\frac{n^2\pi^2}{a^2}t} \tag{6}$$

代入边界条件，边界条件显然成立, 对于初始条件，我们有：

$$\sum_{n=1}^{\infty}A_n\sin(\frac{n\pi}{a}x)e^{-\frac{n^2\pi^2}{a^2}t}=\begin{cases} \frac{T_1-T_2}{a}x & (0\le x\le \alpha a) \\ (T_2-T_1)(1-\frac{x}{a}) & (\alpha a\le x\le a) \end{cases} \tag{7}$$

$$\int_0^a \sin(\frac{n\pi}{a}x)(\sum_{n=1}^{\infty}A_n\sin(\frac{n\pi}{a}x))dx=\frac{a}{2}A_n \tag{8}$$

$$\frac{a}{2}A_n=\int_0^{\alpha a}\sin(\frac{n\pi}{a}x)\frac{T_1-T_2}{a}xdx+\int_{\alpha a}^{a}\sin(\frac{n\pi}{a}x)(T_2-T_1)(1-\frac{x}{a})dx \tag{9}$$

通过积分运算，我们可以计算出 A_n 的值:

$$A_n=2\frac{T_2-T_1}{n\pi}\cos(n\pi\alpha) \tag{10}$$

而温度分布为函数为：

$$u(x,t)=\frac{T_2-T_1}{a}x+T_1+\sum_{n=1}^{\infty}2\frac{T_2-T_1}{n\pi}\cos(n\pi\alpha)\sin(\frac{n\pi}{a}x)e^{-\frac{n^2\pi^2}{a^2}t} \tag{11}$$

可以看出，随着时间增加，温度分布将趋于一条直线，这很符合我们常理。 接下来，我们开始利用 python 程序来验证我们的想法。利用人工智能相关模块 pytorch 并秉承并行化计算的思想，我们设计了高效的计算 $u(x,t)$ 的函数，并且只对无穷级数的前 1000 项和 10000 项作求和，发现这两种方式下得到的误差结果没有任何区别，故取前 1000 项是一个合理的近似。

我们不知道第 0 层 (外部空气)) 和第 5 层 (人体组织) 的等效厚度，但是我们从附件 2 中得知，在 75 度的外部环境中，皮肤表面的平衡温度为 48.0799 度。我们先计算了第 1、2、3、4 层的总等效厚度 $d_1+d_2+d_3+d_4$，为 $21.72022s^{1/2}$，根据平衡状态下温度分布为一条直线 ($T=\frac{75-47}{a}x+37$) 的性质，得到以下的方程：

$$d_5=\alpha a-21.72022 \tag{12}$$

$$\frac{75-37}{a}d_5+37=48.0799 \tag{13}$$

d_0,a 可以根据 α 确定。同时 d_5 等于 $(1-\alpha)a$，也可以确定。所以我们把 α 作为一个自变量来计算温度分布，和 $x = d_0$(皮肤表面) 处的温度变化情况。经过一段时间的调整，我们找到了一个 α 最优解：

$$\begin{cases} \alpha = 0.75 \\ a = 47.380508849 \\ d_0 = 13.815158896 \end{cases} \tag{14}$$

我们计算了这种情况下，在 $x = d_0$，也就是皮肤表面的温度时间曲线。为了分析和实际值的差距，我们计算了以下两个量（以摄氏度为单位）：

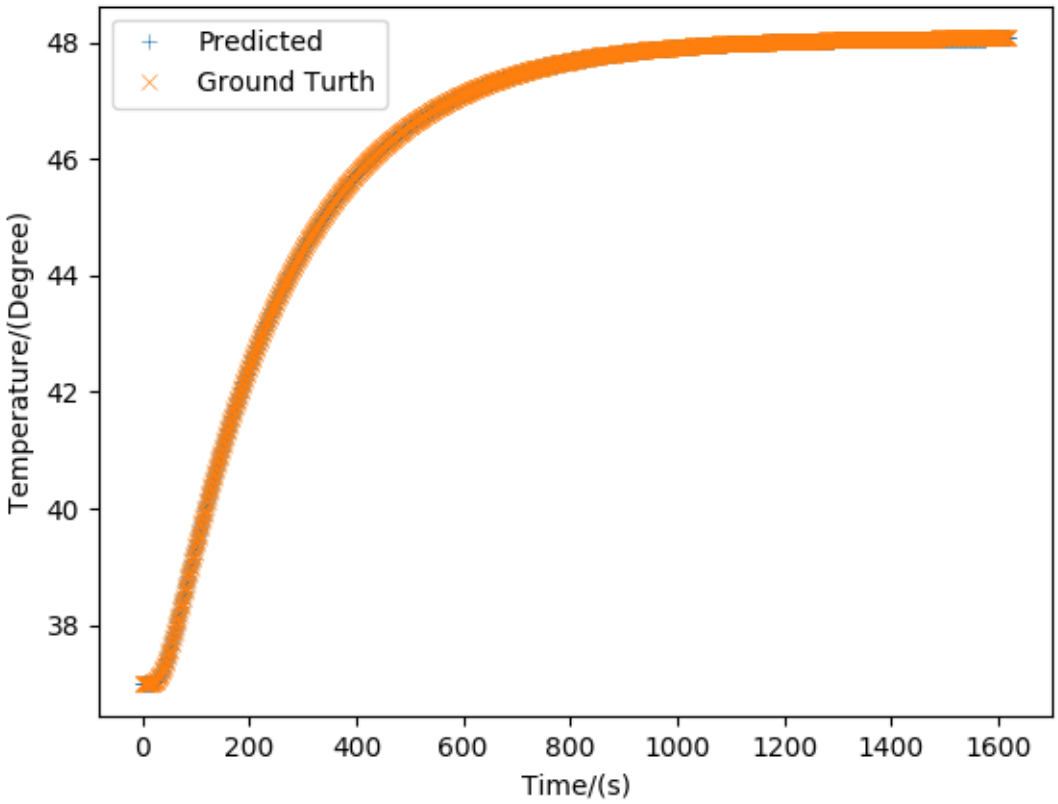
相关系数 公式如下：

$$cor(x,y) = \frac{\sum_{i=1}^{N}(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \hat{x})^2 \sum_{i=1}^{N}(y_i - \hat{y})^2}} \tag{15}$$

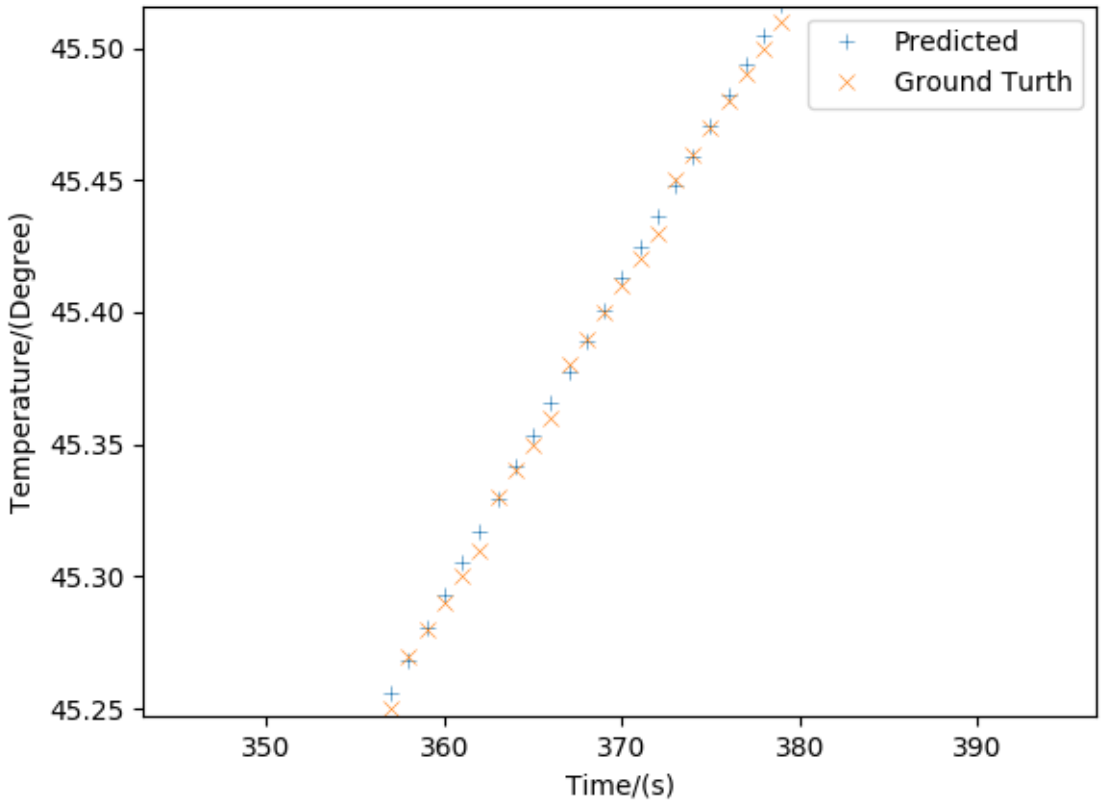
误差平方的平均的开根号 公式如下：

$$\Delta(x,y) = \left(\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2\right)^{\frac{1}{2}} \tag{16}$$

这个函数很好地模拟了温度的分布，它的预测值和实际值的相关系数为 0.999999361557，误差平方的平均的开根号为 0.00576161546633。下面是我们的函数预测值和实际温度的对比图。



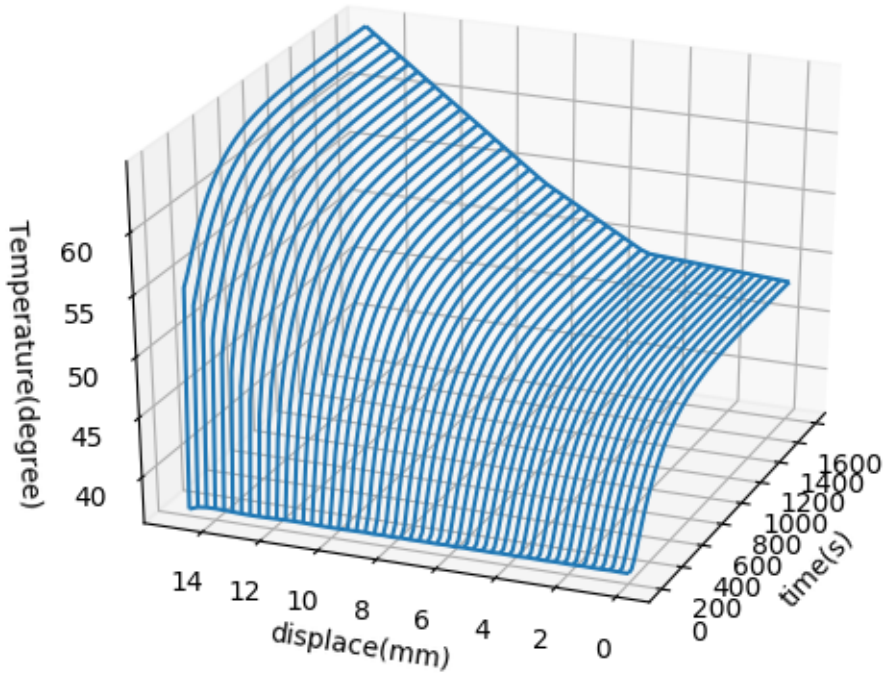
这种情况下难以看到它们的差别，接下来是放大后的的对比图：



我们的模型非常成功地预测了人体表面温度随时间的变化，所以我们可以很有自信地利用它来解决 3 个问题。

七、问题一

模型的提出和推导已经完成，这里解释我们计算温度随时间和位移变化的关系的过程。我们把第 1、2、3、4 层的所占的实际厚度 (15.2mm) 分成 500 等分，把它们转化等效位移 x，然后根据我们解出的温度随时间、等效位移的变化关系 $u(x,t)$ 计算每一个位置和时间对应的温度，并生成了 excel 文件。它的第一行代表了 500 个实际位移位置（以 mm 为单位）的值，第一列则是时间轴，每个位置和时间的对应一个温度。这是根据我们计算的温度随时间位移分布的三维图 (由于在时间超过 1600 秒后，温度变化已经不明显，所以省去这部分):



八、问题二

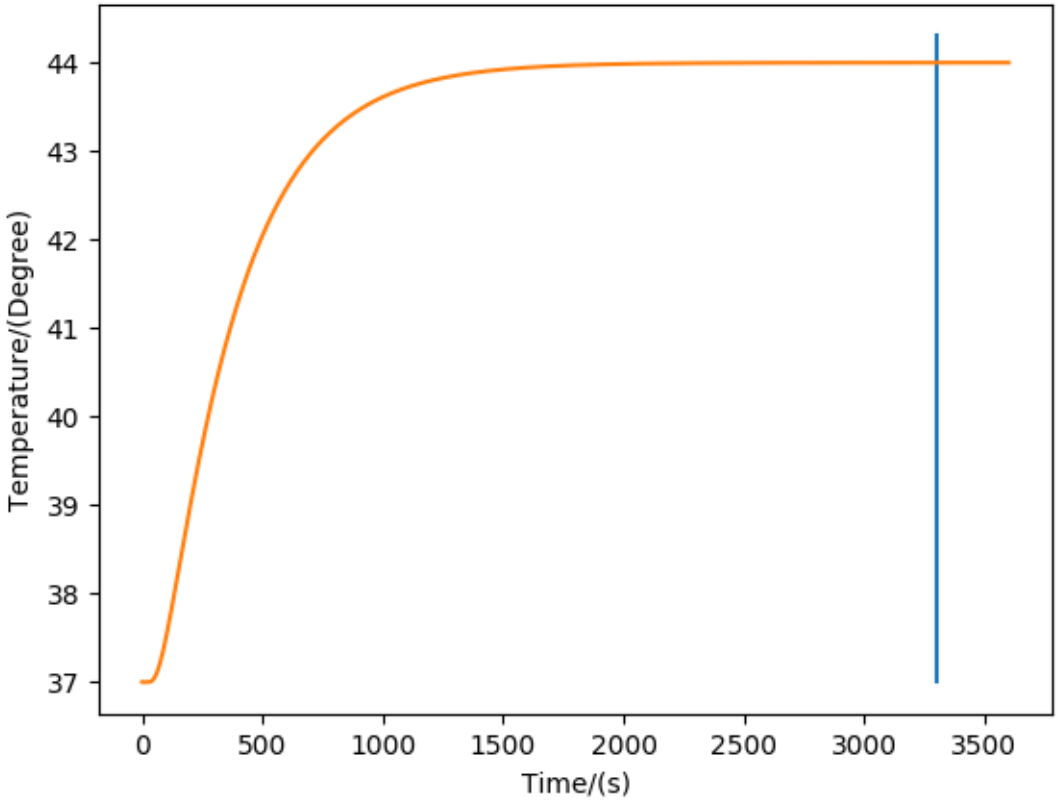
问题重述：当环境温度为 65ºC、IV 层的厚度为 5.5 mm 时，确定 II 层的最优厚度，确保工作 60 分钟时，假人皮肤外侧温度不超过 47ºC，且超过 44ºC 的时间不超过 5 分钟。

为了提高工作服的舒适度，应该让工作服尽量薄一些，而且同时还要满足隔热的要求，所以我们寻求这么一个最优解，既可以达到隔热要求，又能使衣物尽量薄。在第 4 层厚度已经确定的前提下，我们利用建立的模型，计算不同第 2 层厚度对应的温度时间变化关系。这里主要改变的是我们的模型中 a 和 α 变量，计算的仍然是皮肤表面的温度变化，$u(d_5, t)$，只要根据不同的第二层厚度,d_2 修改它们即可。计算公式如下：

$$a = \sum_{i=0}^{5} d_i \tag{17}$$

$$\alpha = 1 - d_0/a \tag{18}$$

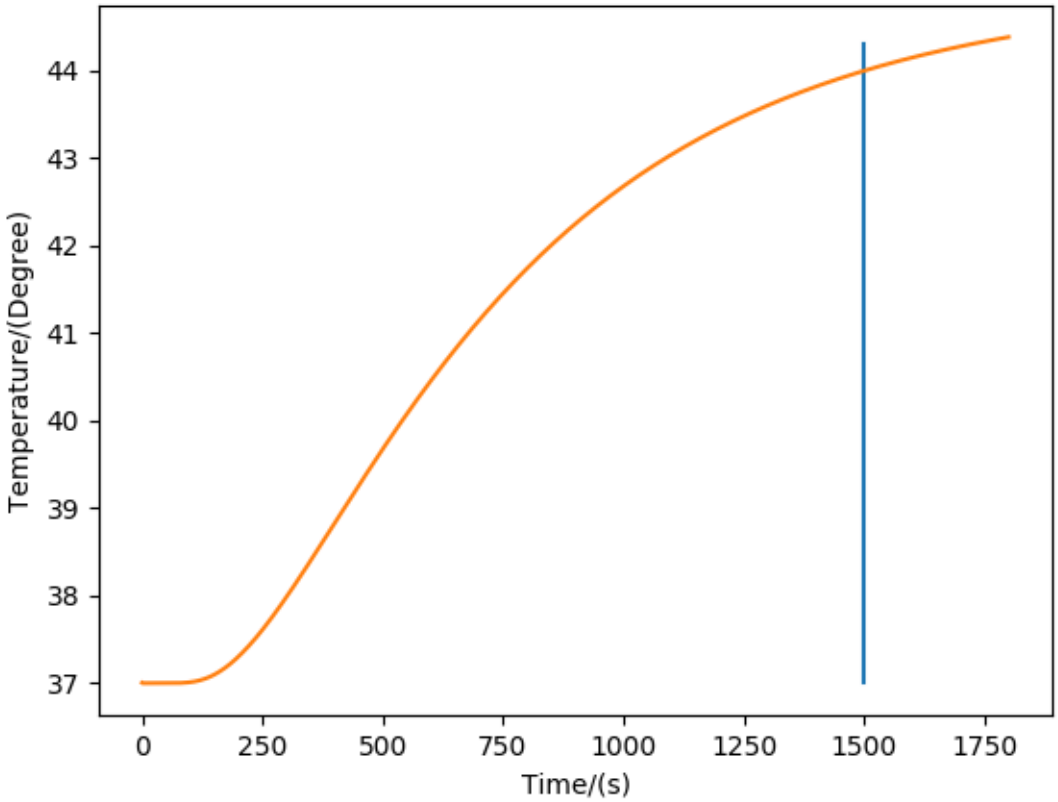
经过实验，我们找到了一个合适的 d_2 的值，$d_2 = 9.51529$mm，在这种情况下，皮肤的最高温度为 44.0001 度, 超过 44 度的时间是 296s。这是它的温度时间曲线（其中蓝色的线标志着温度超过 44 度）：



九、问题三

问题重述：当环境温度为 80 时，确定 II 层和 IV 层的最优厚度，确保工作 30 分钟时，假人皮肤外侧温度不超过 47°C，且超过 44°C 的时间不超过 5 分钟。

这个问题与问题二没有本质上的差别，我们依旧通过调整 d_2 的方式来解决。这里我们仍然保持 d_4 不变，为 5.5mm，因为这个厚度处于正常的范围，且没有什么用来衡量空气层对人体带来影响的指标。最终得到的解为，$d_2 = 18.366$mm。这种情况下，最高温度为 44.3855 度，超过 44 度的时间为 300 秒。以下使这种情况对应的时间温度变化曲线：


Temperature/(Degree)
44
43
42
41
40
39
38
37
0
250
500
750
1000
1250
1500
1750
Time/(s)

(19)

十、附录

10.1 编程环境

编程语言：python2.7

模块：torch numpy matplotlib xlrd xlsxwriter

编程思想：并行化计算设计思想，通过矩阵运算代替效率低下的 for 循环，加快运行速度

10.2 函数文件

```
# -*- coding: utf-8 -*-
import numpy as np
import torch
import matplotlib.pyplot as plt
import math
import xlsxwriter
import xlrd
from mpl_toolkits.mplot3d import axes3d

pi = math.pi
d5= 13.815158896
Alpha= 0.75
d0 = 11.84512721225
# 根据问题一计算出d_5和d_0后，记录在文件中，方便问题二和问题三的函数设计

def solution(x,a,T1,T2,alpha,t):                    # 计算 u(x,t)
    """
    用于计算u(x,t)的函数。
    输入：x --一个float数或可以转化成float数的变量，代表等效位移
          t --一个torch一维数组，代表测量温度的时间点
          T1 --一个数，代表u(0,t)处的温度
          T2 --一个数，代表u(a,t)处的温度
          alpha --一个0-1之间的数，计算公式：1-d_0/a
    输出：一个大小和t相同的一维torch数组，代表在等效位移为x，所有对应时间点预测的温度
    """
    temp = torch.arange(1,1001).float()      #创造一个1-1000的递增数列
    mask1 = 2*(T2-T1)/pi/temp            #计算求和数列的系数中的分数部分
```

```
    mask2 = (temp*pi*alpha).cos()       #计算求和数列的系数中的cos部分
    mask3 = torch.sin(temp*pi*x/a).float()  #计算求和项中的sin部分
    temp2 = t.view(-1,1).float()        #矩阵变形
    temp3 = -temp2 * (temp**2).view(1,-1) * pi**2/a**2
    mask4 = torch.exp(temp3)    #计算求和项的时间部分
    mask_1 = mask1 * mask2 *mask3
    mask_1 = mask_1.view(1,-1).repeat(len(t),1)
    ans = mask_1*mask4
    ans = ans.sum(-1)
    ans = ans
    ans += (T2-T1)*x/a +T1
    return ans

def plot(alpha):
    """
    用于画图、计算误差来比较模型预测值和实际值的函数
    输入:  alpha - = 1-d_0/a,这是模型的一个参数, 这个函数旨在通过误差比较
    寻找到一个合适的alpha值
    动作:  画出预测值和实际值的散点对比图
           显示预测值和实际值的误差的标准差
    """
    ratio = 0.3
   sheet=xlrd.open_workbook('CUMCM-2018-Problem-A-Chinese-Appendix.xlsx')
    sheet=sheet.sheets()[0]
    X = sheet.col_values(0)[2:int(ratio*(len(sheet.col_values(0))-2))+2]
    Y = sheet.col_values(1)[2:int(ratio*(len(sheet.col_values(0))-2))+2]
    X = torch.tensor(X,dtype =torch.float)
    Y = torch.tensor(Y,dtype =torch.float)
    a = 21.720222740795744/(alpha-(48.08-37)/(75-37))
    x = (48.08-37)/(75-37) * a
    print 'a=',a,'x=',x, 'alpha=',alpha
    Predict = solution(x,a,37,75,alpha,X)
    def Silmilarity(tensor1,tensor2):   #计算相关系数的函数
        mean1 = torch.mean(tensor1)
        mean2 = torch.mean(tensor2)
        ans1 = torch.sum((tensor1-mean1)*(tensor2-mean2))
```

```
        ans2 = (torch.sum((tensor1-mean1)**2)*torch.sum((tensor2-mean2)**2))**0.5
            return ans1.item()/ans2.item()
        def Mean_error(tensor1,tensor2):    #计算误差平方的平均的开根号函数
            return (torch.mean((tensor1-tensor2)**2)**0.5).item()
        print 'Mean Error =',Mean_error(Predict,Y)
        print 'Similarity =',Silmilarity(Predict,Y)

        plt.plot(X.numpy(),Predict.numpy(),'+',label='Predicted')
        # .numpy()是把torch张量转化为numpy张量的函数, 画图需要numpy张量
        plt.plot(X.numpy(),Y.numpy(),"x",label='Ground Turth')
        plt.xlabel('Time/(s)')
        plt.ylabel('Temperature/(Degree)')
        plt.legend()
        plt.show()

    def Distribution():
        """
        之前的实验中我们得到 a= 47.380508849 d_5= 13.815158896 alpha= 0.75
        第1、2、3、4层的等效位移的范围是 [13.815158896,35.53538163675]
        我们在这个范围内均匀设置500个坐标点用于计算温度时间变化曲线
        """
        import time
        t1 = time.time()
        alpha = 0.75
       sheet = xlrd.open_workbook('CUMCM-2018-Problem-A-Chinese-Appendix.xlsx').sheets()[1
        ts = sheet.col_values(0)[2:]
        # 读取测量温度的时间点数据
        thickness = 0.6+6+3.6+5
        ts = torch.tensor(ts)
        xs = torch.linspace(0,thickness,500)
        # 设置500个预测温度的位移点
        def transfer(x):
            x = x
         k1,k2,k3,k4 = 1.9849915274751876e-07,2.043973041652856e-07,3.513725392209836e-07
            c1,c2,c3,c4=1000*k1**0.5,1000*k2**0.5,1000*k3**0.5,1000*k4**0.5
            thick0 = 47.380508849*0.75-21.720222740795744
```

```
        thick1 = 5/c4 +thick0
        thick2 = thick1+3.6/c3
        thick3 = thick2+6/c2
        mask1 = ((0<=x) * (x<=5)).float()   # 计算分类矩阵
        mask2 = ((5<x) * (x<=8.6)).float()
        mask3 = ((8.6<x) * (x<=14.6)).float()
        mask4 = ((14.6<x) * (x<=15.2)).float()
        ans1 = mask1*(x/c4+thick0)
        ans2 = mask2*((x-5)/c3+thick1)
        ans3 = mask3*((x-8.6)/c2+thick2)
        ans4 = mask4*((x-14.6)/c1+thick3)
        return ans1+ans2+ans3+ans4
    # 设计把实际位移转化成等效位移的函数
    ans = torch.zeros((len(ts)+1,501))
    ans[0,1:] = xs # 第一行用位移点填满
    ans[1:,0] = ts # 第一列用时间点填满
    Xs = transfer(xs)
    for i in range(500):
        x = Xs[i].item()
        ans[1:,i+1] = solution(x,47.380508849,37,75,0.75,ts)
   # 如果用矩阵运算一步到位地计算温度分布，程序内存崩溃，所以还是采用一个for循环
    xlsx = xlsxwriter.Workbook('Problem1.xlsx')
    worksheet = xlsx.add_worksheet()
    for i in range(ans.size(0)):
        for j in range(ans.size(1)):
            worksheet.write(i,j,float(ans[i,j]))
    worksheet.write(0,0,'Time(s)')
    # 第1行第1列写上时间 (s)
    xlsx.close()
    t2 = time.time()
    print 'Finished in {} seconds'.format(t2-t1)


def Plot3d(rstride=100,cstride=1,ratio=0.4,stept = 30,stepx = 10):
    """
    用来读取生成的Problem1.xlsx文件并画出温度-时间-位移的三维图像的函数
```

```
输入:     rstride - 画图时，时间轴上，每隔rstride个数据点，在u(x,t)
                    曲面上画一条垂直于时间轴的线。默认为100，这时
                    候不画线，因为数据点小于100个。
          cstride - 类似于rstrice，但是是位移轴的参数，默认为1，会
                    在预测的u(x,t)曲面上画出密集的垂直于位移轴的图线
          ratio   - 对时间轴取样时，只去前面占ratio的部分，因为在后
                    期温度已经达到平衡，变化很小。默认为0.4，只显示
                    前40%的温度变化情况
          stept   - 对时间数据取样的间隔，最好为整数
          stepx   - 对位移数据取样的间隔，最好为整数

"""
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# 设置画布
sheet = xlrd.open_workbook('Problem1.xlsx').sheets()[0]
ts = np.array(sheet.col_values(0)[1:int(ratio*(sheet.nrows-1))+1])
xs = np.array(sheet.row_values(0)[1:])
# 读取excel文件中的数据并转化称numpy矩阵
lent = len(ts)
lenx = len(xs)
ts = ts[np.arange(0,lent,stept).astype(np.int)]
xs = xs[np.arange(0,lenx,stepx).astype(np.int)]
# 采样读取所有的位移、时间数据
ts = np.array(ts).reshape(-1,1).repeat(len(xs),1)
xs = np.array(xs).reshape(1,-1).repeat(len(ts),0)
# 把时间、位移矩阵拓展成2维
zs = np.zeros_like(ts)
for i in np.arange(0,lenx,stepx):
 zs[:,int(i/stepx)] = np.array(sheet.col_values(i+1))[np.arange(1,lent+1,stept).a
# 采样读取所有位移、时间对应的温度数据
ax.plot_wireframe(ts,xs,zs,rstride=rstride,cstride=cstride)
ax.set_xlabel('time(s)')
ax.set_ylabel('displace(mm)')
ax.set_zlabel('Temperature(degree)')
plt.show()
```

```
def Problem2(thickness,T2,time):
    """
    计算第二层厚度为thickness，最高温度为T2时，皮肤表面的温度时间变化关系。
    动作：  求出最高温度
            求出温度超过44度的时间
            画出皮肤表面温度时间变化关系图线
    """
    total_thickness = d0 + d5 + 1.3467032917824031 + \
        6.0732141078674315 + 5.5e-3/2.361075976051944e-05**0.5 + \
        thickness*1e-3/2.043973041652856e-07**0.5
    # total_thinckness 是总的等效厚度 d_0+...+d_5
    alpha = 1-d0/total_thickness
    t = torch.arange(time+1).float() # 设置时间断点
    ans = solution(d5,total_thickness,37,T2,alpha,t)  # 计算u(x,t)
    max_temp = max(ans)                                # 计算温度最大值
    print 'Max temperature is', max_temp
    i = 0
    if max_temp>44:
        while True:
            if ans[i] >44.0:                           # 寻找达到44度的位置
                break
            i += 1
        print 'temperature reaches 44 at {} s'.format(i)
        print 'temperature is higher than 44 for {} s'.format(time-i)
    plt.plot(i*np.ones(100),np.linspace(37,44.3,100))
    plt.plot(t.numpy(),ans.numpy())                    # 画图
    plt.xlabel('Time/(s)')
    plt.ylabel('Temperature/(Degree)')
    plt.show()
```

如果直接复制 pdf 可能导致代码无法运行，这里只起到解释代码的作用。可运行的代码可访问 github 链接：https://github.com/Ela-Boska/Savage.git中的 P2.py 文件
在 python 交互窗口中输入 import P2 后，就可以使用其中的函数