高温作业专用服装设计

摘要

近年来，关于高温作业专用服热传递模型研究是一个热点问题。现有的模型根据热防护服是采用单层还是多层材料，分为单层和多层模型。但是在单层模型的基础上，许多学者研究了热防护服传递多层模型。Mell[1]提出了多层材料层与层之间的热传导模型。

本文通过对人体皮肤温度变化及外界环境温度变化之间的不同情形的温度进行分析并建立相应的数学模型，先后研究了热扩散率和热导率的转换、稳态下热量的传递、非稳态下热量的传递，建立了稳态下和非稳态下的一维热传导模型。使用有限差分格式对热传导方程进行求解，通过 Matlab 编程对模型进行了模拟，求出了任意位置温度随时间变化的分布，作出了温度随时空变化的三维图形，并与附件 1 数据比较，结果理想。

针对问题一建立了带有初边值问题的一维热传导模型，通过求解模型的数值解得到隔温层的温度分布，并与实验数据比较，温度分布比较吻合。求解过程中使用了 Fourier 热传导定律，将给定的已知各层厚度、比热、热传导率代入公式中，用 Matlab 求解，得出外界不同温度下人体皮肤外侧的温度分布。

针对问题二建立了单目标优化模型，使用遗传算法求出第Ⅱ层的最优厚度为 5.5mm。

针对问题三，考虑热传导在厚度、热传导率、比热三个变量下的关系式，结合 Fourier 热传导定律建立了双目标优化模型，通过变换化为单目标优化模型，使用类似问题二的方法在满足题意的条件下求得Ⅱ、Ⅳ层的最优厚度分别为 5.6mm、5.4mm。

关键词：热传导方程；数值模拟；有限差分；多层介质热传导

一、问题重述

消防及金属炼钢等行业中，工作人员常处于高温的环境，但是人体在此环境的耐受时间仅仅只有几秒钟，具有致命的危险。所以工作人员在工作时需要穿着专门服装以避免灼伤。当前专用服装通常由三层织物材料构成，分别记为 I、II、III 层。

条件：

1. Ⅰ层与外界环境接触；Ⅱ层、Ⅲ层中间层；Ⅲ层与皮肤之间的空隙为Ⅳ层。

2. 将体温控制在 37ºC 的假人放置在实验室的高温环境中，测量假人皮肤外侧的温度。

3. 降低研发成本、缩短研发周期。

利用数学模型来确定假人皮肤外侧的温度变化情况，并解决以下问题：

(1) 专用服装材料的某些参数值由附件 1 给出，对环境温度为 75ºC、II 层厚度为 6 mm、IV 层厚度为 5 mm、工作时间为 90 分钟的情形开展实验，测量得到假人皮肤外侧的温度（见附件 2）。建立数学模型，计算温度分布，并生成温度分布的 Excel 文件。

(2) 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。

(3) 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 47ºC，且超过 44ºC 的时间不超过 5 分钟。

二、问题分析

高温工作下的人们经常会遇到皮肤灼伤的情况，所以高温作业工作服的产生是有必要的，它既可以帮助高温工作的人们防止皮肤灼伤，而且也会使工作人员在穿戴上工作服后，使皮肤接触的工作环境处于一个相对适宜的温度。对于防护服我们主要用数学模型描述热防护服-空气层-皮肤系统内的热力学规律，并结合皮肤耐受温度，给出了各个温度等级及预测最高温度耐受时间参数的初步研究。同时，综合考虑皮肤层的热传递模型，给出问题的合理解决办法。为热防护服的功能性设计提供理论参考。

根据附件 1 提供的参数值得出：第Ⅰ、Ⅲ层材料的厚度不发生变化，其密度、比热、热传导也不发生变化。虽然第Ⅱ、Ⅳ层的厚度会发生变化，但是附表 1 中数据表明，密度、比热、热传导的参数值不会随厚度的变化而变化。这就方便了我们对热传导的参数进行应用。

1. 问题一分析：

在问题一中，温度是从外界通过防护服传导到皮肤表面的，所以我们引入了热传递的概念。热传递分为三种，分别为热传导、对流、辐射。其中对流是指液体和气体的传热方式，通过流动使热量均匀传播到整体的每个部分。但是对流需要物体厚度超过6.4mm 才会产生热对流。因此，在本题中不考虑对流这一因素。所以此题符合 Fourier 热传导定律($J=-\mathrm{kd}_T/d_x$)[2]，外界温度传递到人体皮肤任何一点的规律都相同，故我们对此建立了一维热传导模型对本题进行求解。

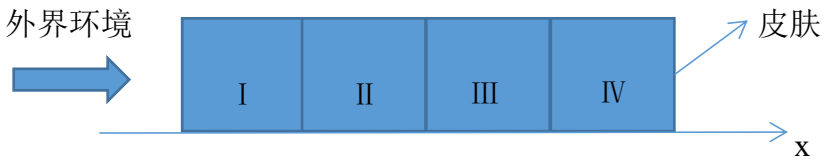


图 1 一维热传导图型

2. 问题二分析：

在问题二的叙述中，专用服装的Ⅰ、Ⅲ、Ⅳ层的厚度都是固定不变的，只需要确定Ⅱ层最优厚度。最优厚度又可称为饱和厚度，即在某一厚度为饱和厚度，其余超出的厚度对本问题的影响微乎其微。所以引入饱和厚度既可以降低研发成本、缩短研发周期也可以满足题意要求。因此我们在问题一的基础上我们对现有模型进行修改，转换为对Ⅱ层厚度的研究。

3. 问题三分析：

问题三要求专用服装的Ⅰ、Ⅲ层厚度保持不变，需要确定Ⅱ、Ⅳ层的最优厚度。所以此问题与问题二相似。因此我们对问题二模型进行修改，重新确定Ⅱ层的最优厚度的同时并计算出Ⅳ层的最优厚度。

三、问题假设

1. 热传递为垂直于皮肤方向进行，故可视为一维的；
2. 假设我们研究只研究热扩散不研究热对流；
3. 假设四种介质材料是同质的；
4. 相邻介质接触面积温度相同；

5. 第Ⅰ层材料的表面温度可以瞬间达到外界环境的温度；

6. 系统热传递仅考虑热辐射、热传导的传热，忽略水汽、汗液的影响，即不考虑湿传递；

7. 假设热防护织物没有熔融或者分解。

四、符号说明

符号	符号说明
$u(x,t)$	x位置t时刻的温度
x	到第Ⅰ层外表面的距离
a_i	第i层介质的热扩散系数
λ_i	第i层介质的热传导率
c_i	第i层介质的比热
ρ_i	第i层介质的密度
d_i	第i层介质的厚度
θ_i	稳态下介质第 i面的温度
ΔT	介质两面的恒定温差
Δt	通过介质的时间
S	热通过面积

五、模型建立与求解

（一）问题一模型：

1.基于稳态条件下建立模型（用于下述一般条件下建立模型的检验）：

由于温度不均匀，热量从温度高的地方向温度低的地方转移，称为热传导。热传导的起源是温度 T 不均匀，由高温部分向低温部分传递热量的过程，当温度的变化只是沿着一个方向进行时，导热速率的基本公式可写为 $\frac{\Delta Q}{\Delta t}=-\lambda\frac{\Delta T}{d}S$（此方程基于稳态条件下进行数学模型的建立）。

由空间角度考虑温度分布：

$$\frac{\Delta Q}{\Delta t}=-\lambda\frac{\Delta T}{d}S \tag{5-1}$$

由于处于稳态下，所以各介质中导热速率处处相等。

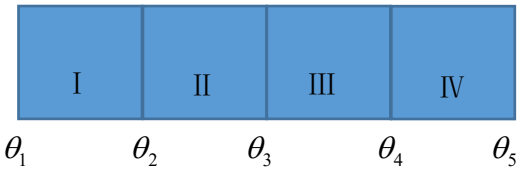


图 2 稳态下介质各面温度分布

由题可得：$Q_0=75^{\circ}C$；$d_{II}=6\times10^{-3}\text{m}$；$d_{IV}=5\times10^{-3}\text{m}$；；

由附件 1 得：$d_I=6\times10^{-4}\text{m}$；$d_{III}=3.6\times10^{-3}\text{m}$；

由附件 2 得：$Q_4=48^{\circ}C$；

因此可得：

$$\lambda_1\frac{75^{\circ}C-Q_2}{0.6mm}S=\lambda_2\frac{Q_2-Q_3}{6mm}S=\lambda_3\frac{Q_3-Q_4}{3.6mm}S=\lambda_4\frac{Q_4-48^{\circ}C}{5mm}S \tag{5-2}$$

由于所建立数学模型为一维模型，可设 S 为 1 单位面积。

所以

$$\begin{cases}\lambda_1\dfrac{75-Q_2}{6\times10^{-4}}=\lambda_2\dfrac{Q_2-Q_3}{6\times10^{-3}}\\ \lambda_2\dfrac{Q_2-Q_3}{6\times10^{-3}}=\lambda_3\dfrac{Q_3-Q_4}{3.6\times10^{-3}}\\ \lambda_3\dfrac{Q_3-Q_4}{3.6\times10^{-3}}=\lambda_4\dfrac{Q_4-48}{5\times10^{-3}}\end{cases} \tag{5-3}$$

将附件 1 中的$\lambda_1=0.082$；$\lambda_2=0.37$；$\lambda_3=0.045$；$\lambda_4=0.028$;代入（5-3）中化简得：

$$\begin{cases}82\times(75-\theta_2)=37\times(\theta_2-\theta_3)\\37\times6\times(\theta_2-\theta_3)=45\times(\theta_3-\theta_4)\\45\times5\times(\theta_3-\theta_4)=28\times3.6\times(\theta_4-48.08)\end{cases}\tag{5-4}$$

整理（5-4）得：

$$\begin{cases}119\theta_2-37\theta_3=6150\\222\theta_2-267\theta_3+45\theta_4=0\\225\theta_3-325.8\theta_4=-4846.464\end{cases}\tag{5-5}$$

由 Matlab 编程并根据附件 2 温度的精确度求得：

$$\begin{cases}\theta_2\approx74.30\\\theta_3\approx72.75\\\theta_4\approx65.12\end{cases}$$

2. 基于一般条件下建立模型：

（1）一维热传导方程的初边值问题的有限差分格式

考虑如下一维非齐次热传导方程初边值问题的有限差分方法，其中a为正常数（$a=\lambda/(\ell\cdot c)$且λ、ℓ、c为定值），$f(x,t),\varphi(x),\alpha(t),\beta(t)$为已知常数，

$$\frac{\partial u}{\partial t}=a\frac{\partial^2u}{\partial x^2}+f(x,t),\qquad c<x<d,\ \ 0<t\le T,\tag{5-8}$$

$$u(x,0)=\varphi(x),\qquad c\le x\le d,\tag{5-9}$$

$$u(c,t)=\alpha(t),\qquad u(d,t)=\beta(t),\qquad 0<t\le T\tag{5-10}$$

注：若无热源问题，则 $f(x,t)=0$

（2）差分格式的建立—— 向前 *Euler* 格式

将区间$[c,d]$作M等分，将$[0,T]$作N等分，并记 $h=(d-c)/M$，$\tau=T/N$，

$x_j=c+jh$，$0\le j\le M$，$t_k=k\tau$，$0\le k\le N$．分别称h和τ为空间步长和时间步长.

用两组平行直线：

$x=x_j$，$0\le j\le M$，

$t=t_k$，$0\le k\le N$

将Ω分割成矩形网格. 记$\Omega_h=\{x_j\mid 0\le j\le M\}$， $\Omega\tau=\{t_k\mid 0\le k\le N\}$，

$\Omega_{h\tau}=\Omega_h\times\Omega_\tau$.

称(x_j,t_k)为结点.

定义$\Omega_{h\tau}$上的网格函数 $\Omega=\{U_j^k\mid 0\le j\le M,0\le k\le N\}$，其中$U_j^k=u(x_j,t_k)$.

在结点(x_j,t_k)处有

$$\frac{\partial u(x_j,t_k)}{\partial t}=a\frac{\partial^2 u(x_j,t_k)}{\partial x^2}+f(x_j,t_k),\quad 1\le j\le M-1,\ 1\le k\le N-1. \tag{5-11}$$

将$u(x_j,t_{k+1})$以结点(x_j,t_k)为中心关于t运用泰勒级数展开，有

$$u(x_j,t_{k+1})=u(x_j,t_k)+u'(x_j,t_k)\tau+\frac{u''(x_j,t_k)\tau^2}{2!}+o(\tau^2).$$

整理有

$$\frac{u(x_j,t_{k+1})-u(x_j,t_k)}{\tau}=\frac{\partial u(x_j,t_k)}{\partial t}+\frac{\tau}{2}\frac{\partial^2 u(x_j,t_k)}{\partial t^2}+o(\tau). \tag{5-12}$$

再将(x_{j-1},t_k)，(x_{j+1},t_k)分别以结点(x_j,t_k)为中心关于x运用泰勒级数展开，有

$$u(x_{j-1},t_k)=u(x_j,t_k)+u'(x_j,t_k)(-h)+\frac{u''(x_j,t_k)(-h)^2}{2!}+\frac{u'''(x_j,t_k)(-h)^3}{3!}$$

$$+\frac{u^{(4)}(x_j,t_k)(-h)^4}{4!}+o(h^4),$$

$$u(x_{j+1},t_k)=u(x_j,t_k)+u'(x_j,t_k)h+\frac{u''(x_j,t_k)h^2}{2!}+\frac{u'''(x_j,t_k)h^3}{3!}$$

$$+\frac{u^{(4)}(x_j,t_k)h^4}{4!}+o(h^4).$$

由上述两式可得

$$\frac{u(x_{j-1},t_k)-2u(x_j,t_k)+u(x_{j+1},t_k)}{h^2}=\frac{\partial^2 u(x_j,t_k)}{\partial x^2}+\frac{h^2}{12}\frac{\partial^4(x_j,t_k)}{\partial x^4}+o(h^2). \tag{5-13}$$

由上可得：

$$\frac{u\left(x_j,t_{k+1}\right)\text{-u}\left(x_j,t_k\right)}{\tau}=a\frac{u\left(x_{j-1},t_k\right)-2u\left(x_j,t_k\right)+u\left(x_{j+1},t_k\right)}{h^2}+f\left(x_j,t_k\right)+R_j^k. \qquad (5\text{-}14)$$

其中 $R_j^k=-\frac{ah^2}{12}\frac{\partial^4u\left(x_j,t_k\right)}{\partial x^4}+\frac{\tau^2}{2}\frac{\partial^2\left(x_j,t_k\right)}{\partial t^2}+o(\tau+h^2)$ 为方程(5-1)的截断误差.

舍去截断误差，用 u_j^k 代替 $u\left(x_j,t_k\right)$，得到如下差分方程

$$\frac{u_j^{k+1}-u_j^k}{\tau}=a\frac{u_{j-1}^k-2u_j^k+u_{j+1}^k}{h^2}+f_j^k,\quad 1\le j\le M-1,\ \ 0\le k\le N-1. \qquad (5\text{-}15)$$

结合初边值条件，可得如下差分格式

$$\frac{u_j^{k+1}-u_j^k}{\tau}=a\frac{u_{j-1}^k-2u_j^k+u_{j+1}^k}{h^2}+f_j^k,\quad 1\le j\le M-1,\ \ 1\le k\le N-1, \qquad (5\text{-}16)$$

$$u_j^0=\varphi(x_j),\qquad 0\le j\le M, \qquad (5\text{-}17)$$

$$u_0^k=\alpha(t_j),\qquad u_M^k=\beta(t_k),\qquad 1\le k\le N. \qquad (5\text{-}18)$$

扩散系数公式：$\mathrm{a}=\lambda/(\ell\cdot c)$ （5-19）

根据附件 1 可得

$$\begin{cases}\rho_1=300\\ \rho_2=862\\ \rho_3=74.2\\ \rho_4=1.18\end{cases}\quad\begin{cases}\mathrm{c}_1=1377\\ \mathrm{c}_2=2100\\ \mathrm{c}_3=1726\\ \mathrm{c}_4=1005\end{cases}\quad\begin{cases}\lambda_1=0.082\\ \lambda_2=0.37\\ \lambda_3=0.045\\ \lambda_4=0.028\end{cases} \qquad (5\text{-}20)$$

将（2-2-10）代入（2-2-9）中得：

$$\begin{cases}\mathrm{a}_1=1.98\times10^{-7}\\ \mathrm{a}_2=2.04\times10^{-7}\\ \mathrm{a}_3=3.51\times10^{-7}\\ \mathrm{a}_4=2.6311\times10^{-7}\end{cases}$$

（3）差分格式的求解

向前 *Euler* 格式

记 $r=a\tau/h^2$ ，称 r 为步长比. 差分格式(5-6) ~ (5-8)中(5-6)可改写为

$$u_j^{k+1}=r\cdot u_{j-1}^k+(1-2r)u_j^k+r\cdot u_{j+1}^k+\tau\cdot f_j^k\ ,$$

$$1\le j\le M-1\ ,\ \ 0\le k\le N-1\ . \qquad (5\text{-}21)$$

将(2-3-1)写成如下形式

$$U^{k+1} = A \cdot U^k + \tau \cdot F^k + r \cdot F_1 \quad .$$

其中

$$U^{k+1} = \left[u_1^{k+1}, u_2^{k+1}, \cdots, u_{M-1}^{k+1} \right]^T, \quad U^k = \left[u_1^k, u_2^k, \cdots, u_{M-1}^k \right]^T,$$

$$F^k = \left[f_1^k, f_2^k, \cdots, f_{M-1}^k \right]^T, \quad F_1 = \left[u_0^k, 0, 0, \cdots, 0, u_M^k \right]^T_{(M-1)*1},$$

$$A = \begin{pmatrix} 1-2r & r & & & \\ r & 1-2r & r & & \\ & r & \ddots & \ddots & \\ & & \ddots & \ddots & r \\ & & & r & 1-2r \end{pmatrix}_{(M-1)*(M-1)} .$$

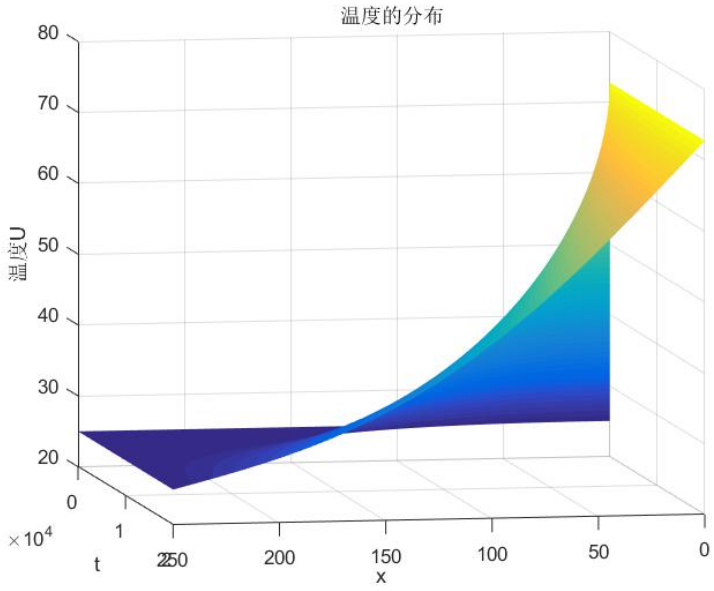
（向前 *Euler* 法）

根据以上模型建立编辑 matlab 代码绘制各层时间、厚度的温度分布图形。

最外层接触最外界，所以 θ_1 的值即为空气中的温度，即 $\theta_1 = 75^\circ C$ 。

温度经过Ⅰ、Ⅱ、Ⅲ、Ⅳ层后，根据热传导公式进行差分格式的建立，得到温度在各层的分布图形：

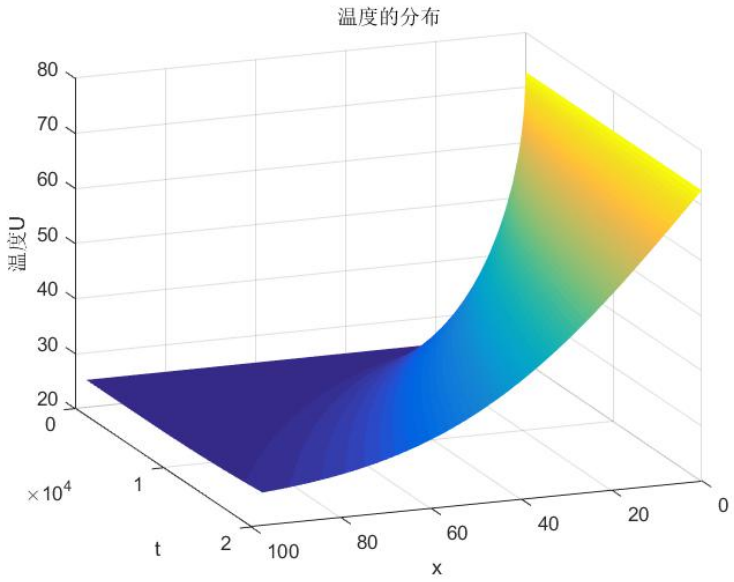
Ⅰ层温度时空分布图形：



$$\theta_2 = 74.30^\circ C$$

θ_2即表示外界温度75℃通过Ⅰ层后温度降低，直到温度降到74.30℃后保持稳定，温度仍在传递但是不再变化。

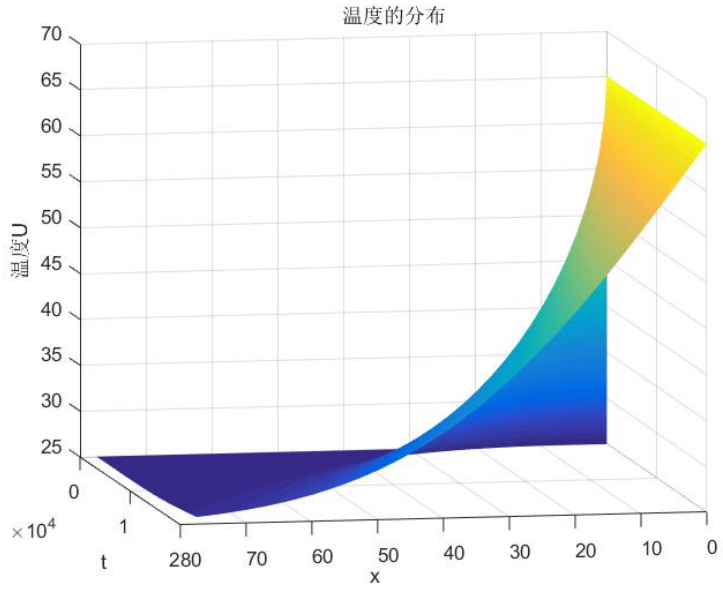
Ⅱ层温度时空分布图形：



$$\theta_3 = 72.75^{\circ}C$$

θ_3即表示通过Ⅰ层的温度在通过Ⅱ层的传递过程中温度降低，直到温度降到72.75℃后，虽然温度仍然在传递。但是温度保持稳定，不在变化。

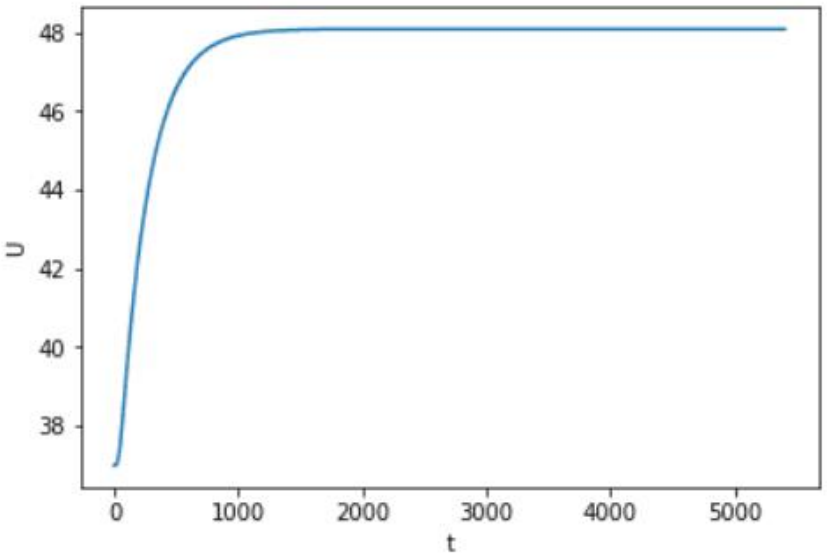
Ⅲ层温度时空分布图形：



$$\theta_4 = 65.12^{\circ}C$$

θ_4即表示传递过程中通过III层后温度降低，温度随还在传递，但也会趋于稳定在65.12℃

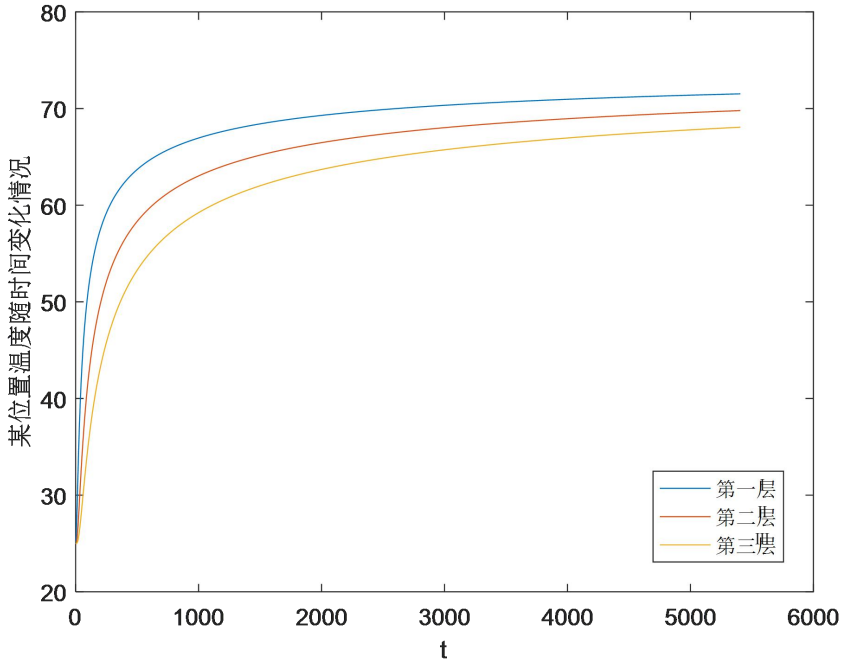
附件1实验数据皮肤外表面温度-时间分布图形：



$$\theta_5 = 48.08℃$$

因为第四层为空气层，所以比较特殊，故我们在matlab仿真时，运用线性制图，得出如上图图形，可以看出温度也会在48.08℃时趋于稳定。

最终各层温度分布结果图：



根据上图可看出，温度在各层的传递过程中，经过一段时间后都会趋于稳定，最终假人体皮肤保持 48.08℃。计算出温度分布。

温度分布数据见附件（problem1.xlsx）。

（二）问题二求解：

分析题意可得此题为约束优化问题，因此我们引入了遗传算法求解，由于所求为 mind_2。

所以，问题求解的标准形式可以表示为：

$$\text{目标函数：}\ \mathrm{mind}_2;$$

$$\text{约束条件：}\begin{cases}\mathrm{g_i}(\theta_2,\theta_3,\theta_4)\geq 0, i=1,2,3;\\ \theta_2,\theta_3,\theta_4\geq 0;\\ d_2\geq 0.\end{cases}$$

所以可设 S 为 1 单位面积。

由附件 1 可知

$$\begin{cases}\lambda_1=0.082\\ \lambda_2=0.37\\ \lambda_3=0.045\\ \lambda_4=0.028\end{cases}\qquad\begin{cases}\mathrm{d_1}=0.6\\ \mathrm{d_3}=.=3.6\end{cases}$$

代入傅里叶公式中后可解得

$$\begin{cases}\mathrm{d_4}=5.5\\ \theta_1=65\\ \theta_2=47\end{cases}$$

因此可得

$$\begin{cases}0.082\dfrac{65-\theta_2}{0.6\times10^{-3}}=0.37\dfrac{\theta_2-\theta_3}{\mathrm{d_2}\times10^{-3}}\\ 37\dfrac{\theta_2-\theta_3}{\mathrm{d_2}\times10^{-3}}=0.045\dfrac{\theta_3-\theta_4}{3.6\times10^{-3}}\\ 0.045\dfrac{\theta_3-\theta_4}{3.6\times10^{-3}}=0.028\dfrac{\theta_4-47}{5.5\times10^{-3}}\end{cases}$$

整理得：

$$\begin{cases}(222+82\mathrm{d}_2)\times\theta_2-222\theta_3=5330\mathrm{d}_2\\ 1332\theta_2-(1332+45\mathrm{d}_2)\theta_3+45\mathrm{d}_2\theta_4=0\\ 2475\theta_3-3483\theta_4+47376=0\end{cases}$$

我们所要求得到的为d_2的最优厚度，即要求取$\min(\mathrm{d}_2)$，由此通过 matlab 编程对方程进行最优解计算，求取$\min(\mathrm{d}_2)=5.5\mathrm{mm}$。

由结果逆推可知所得满足条件:55 分钟时假人皮肤超过 44℃，且在 5 分钟内不超过 47℃。

（三）问题三求解：

我们运用了遗传算法调节问题二中的模型参数，求出了第Ⅱ层与第Ⅳ层的最优厚度。遗传算法是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。 它是由美国的 J.Holland 教授 1975 年首先提出，其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。遗传算法在适应度函数选择不当的情况下有可能收敛于局部最优，而不能达到全局最优。

分析题意，问题三思想类似于问题二，仍为约束优化问题。只是问题二为单目标函数，问题三给双目标函数。因此我们针对问题二对问题三建立数学模型。

目标 $\begin{matrix}\max \mathrm{d}_4\\ \min d_2\end{matrix}$

所以，问题求解的标准形式可以化为：

目标函数：min（d_2 - d_4）

约束条件：

$$\begin{cases}\mathrm{g_i}(\theta_2,\theta_3,\theta_4)\geq 0, i=1,2,3;\\ \theta_2,\theta_3,\theta_4\geq 0;\\ d_2,d_4\geq 0.\end{cases}$$

所以可设 S 为 1 单位面积。

由题可知：

$$\begin{cases}\lambda_1=0.082\\ \lambda_2=0.37\\ \lambda_3=0.045\\ \lambda_4=0.028\end{cases}\qquad\begin{cases}\mathrm{d}_1=0.6\\ \mathrm{d}_3=.=3.6\end{cases}$$

因此可得：

$$\begin{cases}0.082\dfrac{80-\theta_2}{0.6\times10^{-3}}=0.37\dfrac{\theta_2-\theta_3}{\mathrm{d}_2\times10^{-3}}\\ 37\dfrac{\theta_2-\theta_3}{\mathrm{d}_2\times10^{-3}}=0.045\dfrac{\theta_3-\theta_4}{3.6\times10^{-3}}\\ 0.045\dfrac{\theta_3-\theta_4}{3.6\times10^{-3}}=0.028\dfrac{\theta_4-47}{\mathrm{d}_4\times10^{-3}}\end{cases}$$

整理得：

$$\begin{cases}(222+82\mathrm{d}_2)\times\theta_2-222\theta_3=6560\mathrm{d}_2\\ 133.2\theta_2-(133.2+45\mathrm{d}_2)\theta_3+45\mathrm{d}_2\theta_4=0\\ 45\mathrm{d}_4\theta_3-(100.8+45\mathrm{d}_4)\theta_4=-4737.6\end{cases}$$

所求得最优解 d2、d4 的值分别为 5.6㎜，6.4㎜，由结果逆推可知其满足条件 55 分时假人皮肤外侧温度超过 44℃，且在 5 分钟内不超过 47℃。

使用遗传算法进行热传导反问题参数识别的主要步骤为：

第 1 步：根据先验信息确定待识别参数的范围；

第 2 步：随机产生初始种群作为候选解；

第 3 步：计算每个个体的适应度；

第 4 步：进行重组操作；

第 5 步：进行变异操作；

第 7 步：执行收敛准则，如果不满足收敛准则，则转第 3 步；否则，适应度最高的个体的解就是反问题的解，停止反演计算过程。

六、模型评价

本文的工作可以归结为三部分：一是多层热防护热传递模型的建立及验证，二是根据温度变化，确定各个厚度下温度的传递规律，三是根据材料的密度、比热、热传导率并综合算法求出每个问题的最优解。

数学建模过程中有以下三个方向的创新点：

（1）模型建立上，我们将原先多维模型转化为简单的一维模型进行研究，简化了研究过程中曲面对数据的影响。

（2）在一维模型中，利用非齐次热传导方程初边值问题，分别求出稳态条件下与非稳态条件下的厚度，再利用蚁群算法计算两个数值，求出最优厚度，这样就满足了减少研究成本的要求。

（3）利用点面结合的办法，既研究了各层材料接触面之间的温度变化规律，也研究了每层材料内部的温度变化规律。

缺点：本模型所建立的热防护服-空气层-皮肤系统，在各个环节还有许多需要完善的地方，对于传递过程，本文还没考虑到湿传递，实际上这在高温条件下往往具有重要影响，不能给忽略。所以下一步的研究就需要考虑湿传递对热防服的影响。

参考文献：

[1] Mell W E，Lawson J R.A heat transfer model for firefighters’protective clothing[J].Fire Technology,2000,36（1）：39-68.俞昌铭.热传导及其数值分析[M].清华大学出版社，1981.

[2] 俞昌铭.热传导及其数值分析[M].清华大学出版社，1981.

[3] 肖本益，曲久辉，林佶侃. 常温厌氧生物处理技术[J]. 给水排水，2007,33(s1):174-178.

[4] 卢琳珍，徐定华，徐映红. 应用三层热防护服热传递改进模型的皮肤烧伤度预测[J]. 纺织学报，2018(1):111-118.

[5] 李勇贵. 舱壁热防护结构的热力学分析[D]. 哈尔滨工程大学，2013.

[6] 吕士钦.RBF 配点法在多层介质热传导反问题中的应用研究[D].太原理工大学，2013.

[7] 谢鸿政.多层介质内热传导方程初值问题解的逼近方法[J].哈尔滨工业大学学报，1985.02.10-15.

附录

（matlab 编写的求解问题一、二、三的代码）

```
%稳态求解
clear
clc
close all
result=solve('119*x-37*y=6150','222*x-267*y+45*z=0','225*y-325.8*z=-4846.46
4','x','y','z');
result.x
result.y
Result.z

%温度分布求解
clear
clc
close all
domain = 250; % 空间区域(0,domain)
time = 20000; % 时间区域(0,time)
a =0.082; % 扩散系数
c1 =75; % 左边值
c2 =25; % 右边值
g=25;
h = 1.5; % 空间步长
tau = 1; % 时间步长
lambda = tau / h^2; % 网格比
J = fix(domain / h); % 空间区域等分数
N = fix(time / tau); % 时间区域等分数
X = 0:h:domain;
T = 0:tau:time;
[X,T] = meshgrid(X,T); % 生成网格

U = zeros(N+1,J+1);
U(:,1) = c1; % 左边值
U(:,end) = c2; % 右边值

U(1,:) = g;
for n=1:N
    for j=2:J
        U(n+1,j) = (1-2*a*lambda)*U(n,j) + a*lambda*(U(n,j+1)+U(n,j-1));
```

```
    end
end

figure;
surf(X,T,U);
title('温度的分布');
xlabel('x')
ylabel('t')
zlabel('温度 U')
shading interp;

u90=U(:,2);
n=length(u90);
maxd=u90(1:3.7:n);



clear
clc
close all
domain = 250; % 空间区域(0,domain)
time = 20000; % 时间区域(0,time)
a =0.37; % 扩散系数
c1 =74.3; % 左边值
c2 = 25; % 右边值
g=25;
h = 1.5; % 空间步长
tau = 1; % 时间步长
lambda = tau / h^2; % 网格比
J = fix(domain / h); % 空间区域等分数
N = fix(time / tau); % 时间区域等分数
X = 0:h:domain;
T = 0:tau:time;
[X,T] = meshgrid(X,T); % 生成网格

U = zeros(N+1,J+1);
U(:,1) = c1; % 左边值
U(:,end) = c2; % 右边值

U(1,:) = g;
for n=1:N
    for j=2:J
        U(n+1,j) = (1-2*a*lambda)*U(n,j) + a*lambda*(U(n,j+1)+U(n,j-1));
    end
end
```

```
figure;
XX=X(:,1:60);
TT=T(:,1:60);
UU=U(:,1:60);
surf(XX,TT,UU);
title('温度的分布');
xlabel('x');
ylabel('t');
zlabel('温度 U');
shading interp;

u90=U(:,4);
n=length(u90);
maxd=u90(1:3.7:n);



clear
clc
close all
domain = 250; % 空间区域(0,domain)
time = 20000; % 时间区域(0,time)
a =0.045; % 扩散系数
c1 =72.75; % 左边值
c2 = 25; % 右边值
g=25;
h = 1.5; % 空间步长
tau = 1; % 时间步长
lambda = tau / h^2; % 网格比
J = fix(domain / h); % 空间区域等分数
N = fix(time / tau); % 时间区域等分数
X = 0:h:domain;
T = 0:tau:time;
[X,T] = meshgrid(X,T); % 生成网格

U = zeros(N+1,J+1);
U(:,1) = c1; % 左边值
U(:,end) = c2; % 右边值

U(1,:) = g;
for n=1:N
    for j=2:J
        U(n+1,j) = (1-2*a*lambda)*U(n,j) + a*lambda*(U(n,j+1)+U(n,j-1));
    end
```

```
end

figure;
XX=X(:,1:50);
TT=T(:,1:50);
UU=U(:,1:50);
surf(XX,TT,UU);
title('温度的分布');
xlabel('x');
ylabel('t');
zlabel('温度 U');
shading interp;

u90=U(:,6);
n=length(u90);
maxd=u90(1:3.7:n);

%最优厚度求解
function f=fun1(x)          %定义目标函数
%f=1332*(x(1)-x(2))/(45*(x(2)-x(3)));;
f=222*(x(1)-x(2))/(5330-82*x(1));


function [g,h]=fun2(x)      %非线性约束条件
g=[0
    0];
h=[109224*x(1)+9990*(x(2)-x(3))-7099560
    2475*x(2)-3483*x(3)+47376];


function [X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,HESSIAN] =
llw(FUN,X,A,B,Aeq,Beq,LB,UB,NONLCON,options,varargin)
%2018 年 9 月 16 日

defaultopt = struct( ...
    'Algorithm','interior-point', ...
    'AlwaysHonorConstraints','bounds', ...
    'DerivativeCheck','off', ...
    'Diagnostics','off', ...
    'DiffMaxChange',Inf, ...
    'DiffMinChange',0, ...
    'Display','final', ...
    'FinDiffRelStep', [], ...
    'FinDiffType','forward', ...
```

```
    'FunValCheck','off', ...
    'GradConstr','off', ...
    'GradObj','off', ...
    'HessFcn',[], ...
    'Hessian',[], ...
    'HessMult',[], ...
    'HessPattern','sparse(ones(numberOfVariables))', ...
    'InitBarrierParam',0.1, ...
    'InitTrustRegionRadius','sqrt(numberOfVariables)', ...
    'MaxFunEvals',[], ...
    'MaxIter',[], ...
    'MaxPCGIter','max(1,floor(numberOfVariables/2))', ...
    'MaxProjCGIter','2*(numberOfVariables-numberOfEqualities)', ...

'MaxSQPIter','10*max(numberOfVariables,numberOfInequalities+numberOfBounds)
', ...
    'ObjectiveLimit',-1e20, ...
    'OutputFcn',[], ...
    'PlotFcns',[], ...
    'PrecondBandWidth',0, ...
    'RelLineSrchBnd',[], ...
    'RelLineSrchBndDuration',1, ...
    'ScaleProblem','none', ...
    'SubproblemAlgorithm','ldl-factorization', ...
    'TolCon',1e-6, ...
    'TolConSQP',1e-6, ...
    'TolFun',1e-6, ...
    'TolPCG',0.1, ...
    'TolProjCG',1e-2, ...
    'TolProjCGAbs',1e-10, ...
    'TolX',[], ...
    'TypicalX','ones(numberOfVariables,1)', ...
    'UseParallel',false ...
    );

% If just 'defaults' passed in, return the default options in X
if nargin==1 && nargout <= 1 && strcmpi(FUN,'defaults')
   X = defaultopt;
   return
end

if nargin < 10
    options = [];
    if nargin < 9
```

```
        NONLCON = [];
        if nargin < 8
            UB = [];
            if nargin < 7
                LB = [];
                if nargin < 6
                    Beq = [];
                    if nargin < 5
                        Aeq = [];
                    end
                end
            end
        end
    end
end

problemInput = false;
if nargin == 1
    if isa(FUN,'struct')
        problemInput = true;
        [FUN,X,A,B,Aeq,Beq,LB,UB,NONLCON,options] =
separateOptimStruct(FUN);
    else % Single input and non-structure.
        error(message('optimlib:fmincon:InputArg'));
    end
end

% Prepare the options for the solver
[options, optionFeedback] = prepareOptionsForSolver(options, 'fmincon');

if nargin < 4 && ~problemInput
  error(message('optimlib:fmincon:AtLeastFourInputs'))
end

if isempty(NONLCON) && isempty(A) && isempty(Aeq) && isempty(UB) && isempty(LB)
   error(message('optimlib:fmincon:ConstrainedProblemsOnly'))
end

% Check for non-double inputs
msg = isoptimargdbl('FMINCON', {'X0','A','B','Aeq','Beq','LB','UB'}, ...
                                  X,   A,   B,   Aeq,   Beq,   LB,   UB);
if ~isempty(msg)
    error('optimlib:fmincon:NonDoubleInput',msg);
end
```

```
if nargout > 4
   computeLambda = true;
else
   computeLambda = false;
end

activeSet = 'medium-scale: SQP, Quasi-Newton, line-search';
sqp = 'sequential quadratic programming';
trustRegionReflective = 'trust-region-reflective';
interiorPoint = 'interior-point';

[sizes.xRows,sizes.xCols] = size(X);
XOUT = X(:);
sizes.nVar = length(XOUT);
% Check for empty X
if sizes.nVar == 0
   error(message('optimlib:fmincon:EmptyX'));
end

display = optimget(options,'Display',defaultopt,'fast');
flags.detailedExitMsg = ~isempty(strfind(display,'detailed'));
switch display
    case {'off','none'}
        verbosity = 0;
    case {'notify','notify-detailed'}
        verbosity = 1;
    case {'final','final-detailed'}
        verbosity = 2;
    case {'iter','iter-detailed'}
        verbosity = 3;
    case 'testing'
        verbosity = 4;
    otherwise
        verbosity = 2;
end

% Set linear constraint right hand sides to column vectors
% (in particular, if empty, they will be made the correct
% size, 0-by-1)
B = B(:);
Beq = Beq(:);

% Check for consistency of linear constraints, before evaluating
```

```
% (potentially expensive) user functions

% Set empty linear constraint matrices to the correct size, 0-by-n
if isempty(Aeq)
    Aeq = reshape(Aeq,0,sizes.nVar);
end
if isempty(A)
    A = reshape(A,0,sizes.nVar);
end

[lin_eq,Aeqcol] = size(Aeq);
[lin_ineq,Acol] = size(A);
% These sizes checks assume that empty matrices have already been made the correct
size
if Aeqcol ~= sizes.nVar
   error(message('optimlib:fmincon:WrongNumberOfColumnsInAeq', sizes.nVar))
end
if lin_eq ~= length(Beq)
    error(message('optimlib:fmincon:AeqAndBeqInconsistent'))
end
if Acol ~= sizes.nVar
   error(message('optimlib:fmincon:WrongNumberOfColumnsInA', sizes.nVar))
end
if lin_ineq ~= length(B)
    error(message('optimlib:fmincon:AeqAndBinInconsistent'))
end
% End of linear constraint consistency check

Algorithm = optimget(options,'Algorithm',defaultopt,'fast');

% Option needed for processing initial guess
AlwaysHonorConstraints =
optimget(options,'AlwaysHonorConstraints',defaultopt,'fast');

% Determine algorithm user chose via options. (We need this now
% to set OUTPUT.algorithm in case of early termination due to
% inconsistent bounds.)
if strcmpi(Algorithm,'active-set')
    OUTPUT.algorithm = activeSet;
elseif strcmpi(Algorithm,'sqp')
    OUTPUT.algorithm = sqp;
elseif strcmpi(Algorithm,'interior-point')
    OUTPUT.algorithm = interiorPoint;
elseif strcmpi(Algorithm,'trust-region-reflective')
```

```
    OUTPUT.algorithm = trustRegionReflective;
else
    error(message('optimlib:fmincon:InvalidAlgorithm'));
end

[XOUT,l,u,msg] = checkbounds(XOUT,LB,UB,sizes.nVar);
if ~isempty(msg)
   EXITFLAG = -2;
   [FVAL,LAMBDA,GRAD,HESSIAN] = deal([]);

   OUTPUT.iterations = 0;
   OUTPUT.funcCount = 0;
   OUTPUT.stepsize = [];
   if strcmpi(OUTPUT.algorithm,activeSet) || strcmpi(OUTPUT.algorithm,sqp)
       OUTPUT.lssteplength = [];
   else % trust-region-reflective, interior-point
       OUTPUT.cgiterations = [];
   end
   if strcmpi(OUTPUT.algorithm,interiorPoint) ||
strcmpi(OUTPUT.algorithm,activeSet) || ...
      strcmpi(OUTPUT.algorithm,sqp)
       OUTPUT.constrviolation = [];
   end
   OUTPUT.firstorderopt = [];
   OUTPUT.message = msg;

   X(:) = XOUT;
   if verbosity > 0
      disp(msg)
   end
   return
end

% Get logical list of finite lower and upper bounds
finDiffFlags.hasLBs = isfinite(l);
finDiffFlags.hasUBs = isfinite(u);

lFinite = l(finDiffFlags.hasLBs);
uFinite = u(finDiffFlags.hasUBs);

% Create structure of flags and initial values, initialize merit function
% type and the original shape of X.
flags.meritFunction = 0;
initVals.xOrigShape = X;
```

```
diagnostics =
strcmpi(optimget(options,'Diagnostics',defaultopt,'fast'),'on');
funValCheck =
strcmpi(optimget(options,'FunValCheck',defaultopt,'fast'),'on');
derivativeCheck =
strcmpi(optimget(options,'DerivativeCheck',defaultopt,'fast'),'on');

% Gather options needed for finitedifferences
% Write checked DiffMaxChange, DiffMinChage, FinDiffType, FinDiffRelStep,
% GradObj and GradConstr options back into struct for later use
options.DiffMinChange = optimget(options,'DiffMinChange',defaultopt,'fast');
options.DiffMaxChange = optimget(options,'DiffMaxChange',defaultopt,'fast');
if options.DiffMinChange >= options.DiffMaxChange
    error(message('optimlib:fmincon:DiffChangesInconsistent',
sprintf( '%0.5g', options.DiffMinChange ), sprintf( '%0.5g',
options.DiffMaxChange )))
end
% Read in and error check option TypicalX
[typicalx,ME] =
getNumericOrStringFieldValue('TypicalX','ones(numberOfVariables,1)', ...
    ones(sizes.nVar,1),'a numeric value',options,defaultopt);
if ~isempty(ME)
    throw(ME)
end
checkoptionsize('TypicalX', size(typicalx), sizes.nVar);
options.TypicalX = typicalx;
options.FinDiffType = optimget(options,'FinDiffType',defaultopt,'fast');
options = validateFinDiffRelStep(sizes.nVar,options,defaultopt);
options.GradObj = optimget(options,'GradObj',defaultopt,'fast');
options.GradConstr = optimget(options,'GradConstr',defaultopt,'fast');

flags.grad = strcmpi(options.GradObj,'on');

% Notice that defaultopt.Hessian = [], so the variable "hessian" can be empty
hessian = optimget(options,'Hessian',defaultopt,'fast');
% If calling trust-region-reflective with an unavailable Hessian option value,
% issue informative error message
if strcmpi(OUTPUT.algorithm,trustRegionReflective) && ...
        ~( isempty(hessian) || strcmpi(hessian,'on') ||
strcmpi(hessian,'user-supplied') || ...
           strcmpi(hessian,'off') || strcmpi(hessian,'fin-diff-grads')  )
    error(message('optimlib:fmincon:BadTRReflectHessianValue'))
end
```

```
if ~iscell(hessian) && ( strcmpi(hessian,'user-supplied') ||
strcmpi(hessian,'on') )
    flags.hess = true;
else
    flags.hess = false;
end

if isempty(NONLCON)
   flags.constr = false;
else
   flags.constr = true;
end

% Process objective function
if ~isempty(FUN)  % will detect empty string, empty matrix, empty cell array
   % constrflag in optimfcnchk set to false because we're checking the objective,
not constraint
   funfcn =
optimfcnchk(FUN,'fmincon',length(varargin),funValCheck,flags.grad,flags.hes
s,false,Algorithm);
else
   error(message('optimlib:fmincon:InvalidFUN'));
end

% Process constraint function
if flags.constr % NONLCON is non-empty
   flags.gradconst = strcmpi(options.GradConstr,'on');
   % hessflag in optimfcnchk set to false because hessian is never returned by
nonlinear constraint
   % function
   %
   % constrflag in optimfcnchk set to true because we're checking the constraints
   confcn =
optimfcnchk(NONLCON,'fmincon',length(varargin),funValCheck,flags.gradconst,
false,true);
else
   flags.gradconst = false;
   confcn = {'','','','','',''};
end

[rowAeq,colAeq] = size(Aeq);

if strcmpi(OUTPUT.algorithm,activeSet) || strcmpi(OUTPUT.algorithm,sqp)
```

```
    % See if linear constraints are sparse and if user passed in Hessian
    if issparse(Aeq) || issparse(A)
        warning(message('optimlib:fmincon:ConvertingToFull', Algorithm))
    end
    if flags.hess % conflicting options
        flags.hess = false;
        warning(message('optimlib:fmincon:HessianIgnoredForAlg',
Algorithm));
        if strcmpi(funfcn{1},'fungradhess')
            funfcn{1}='fungrad';
        elseif  strcmpi(funfcn{1},'fun_then_grad_then_hess')
            funfcn{1}='fun_then_grad';
        end
    end
elseif strcmpi(OUTPUT.algorithm,trustRegionReflective)
    % Look at constraint type and supplied derivatives, and determine if
    % trust-region-reflective can solve problem
    isBoundedNLP = isempty(NONLCON) && isempty(A) && isempty(Aeq); % problem
has only bounds and no other constraints
    isLinEqNLP = isempty(NONLCON) && isempty(A) && isempty(lFinite) ...
        && isempty(uFinite) && colAeq > rowAeq;
    if isBoundedNLP && flags.grad
        % if only l and u then call sfminbx
    elseif isLinEqNLP && flags.grad
        % if only Aeq beq and Aeq has more columns than rows, then call sfminle
    else
        if ~isBoundedNLP && ~isLinEqNLP
            error(message('optimlib:fmincon:ConstrTRR', ...
                addLink( 'Choosing the Algorithm', 'choose_algorithm' )))
        else
            % The user has a problem that satisfies the TRR constraint
            % restrictions but they haven't supplied gradients.
            error(message('optimlib:fmincon:GradOffTRR', ...
                addLink( 'Choosing the Algorithm', 'choose_algorithm' )))
        end
    end
end

lenvlb = length(l);
lenvub = length(u);

% Process initial point
shiftedX0 = false; % boolean that indicates if initial point was shifted
if strcmpi(OUTPUT.algorithm,activeSet)
```

```
    %
    % Ensure starting point lies within bounds
    %
    i=1:lenvlb;
    lindex = XOUT(i)<l(i);
    if any(lindex)
       XOUT(lindex)=l(lindex);
       shiftedX0 = true;
    end
    i=1:lenvub;
    uindex = XOUT(i)>u(i);
    if any(uindex)
       XOUT(uindex)=u(uindex);
       shiftedX0 = true;
    end
    X(:) = XOUT;
elseif strcmpi(OUTPUT.algorithm,trustRegionReflective)
    %
    % If components of initial x not within bounds, set those components
    % of initial point to a "box-centered" point
    %
    if isempty(Aeq)
        arg = (u >= 1e10); arg2 = (l <= -1e10);
        u(arg) = inf;
        l(arg2) = -inf;
        xinitOutOfBounds_idx = XOUT < l | XOUT > u;
        if any(xinitOutOfBounds_idx)
            shiftedX0 = true;
            XOUT = startx(u,l,XOUT,xinitOutOfBounds_idx);
            X(:) = XOUT;
        end
    else
       % Phase-1 for sfminle nearest feas. pt. to XOUT. Don't print a
       % message for this change in X0 for sfminle.
        XOUT = feasibl(Aeq,Beq,XOUT);
        X(:) = XOUT;
    end

elseif strcmpi(OUTPUT.algorithm,interiorPoint)
    % Variables: fixed, finite lower bounds, finite upper bounds
    xIndices = classifyBoundsOnVars(l,u,sizes.nVar,true);

    % If honor bounds mode, then check that initial point strictly satisfies
the
```

```
    % simple inequality bounds on the variables and exactly satisfies fixed
variable
    % bounds.
    if strcmpi(AlwaysHonorConstraints,'bounds') ||
strcmpi(AlwaysHonorConstraints,'bounds-ineqs')
        violatedFixedBnds_idx = XOUT(xIndices.fixed) ~= l(xIndices.fixed);
        violatedLowerBnds_idx = XOUT(xIndices.finiteLb) <=
l(xIndices.finiteLb);
        violatedUpperBnds_idx = XOUT(xIndices.finiteUb) >=
u(xIndices.finiteUb);
        if any(violatedLowerBnds_idx) || any(violatedUpperBnds_idx) ||
any(violatedFixedBnds_idx)
            XOUT = shiftInitPtToInterior(sizes.nVar,XOUT,l,u,Inf);
            X(:) = XOUT;
            shiftedX0 = true;
        end
    end
else % SQP
    % Classify variables: finite lower bounds, finite upper bounds
    xIndices = classifyBoundsOnVars(l,u,sizes.nVar,false);

    % SQP always honors the bounds. Check that initial point
    % strictly satisfies the bounds on the variables.
    violatedLowerBnds_idx = XOUT(xIndices.finiteLb) < l(xIndices.finiteLb);
    violatedUpperBnds_idx = XOUT(xIndices.finiteUb) > u(xIndices.finiteUb);
    if any(violatedLowerBnds_idx) || any(violatedUpperBnds_idx)
        finiteLbIdx = find(xIndices.finiteLb);
        finiteUbIdx = find(xIndices.finiteUb);
        XOUT(finiteLbIdx(violatedLowerBnds_idx)) =
l(finiteLbIdx(violatedLowerBnds_idx));
        XOUT(finiteUbIdx(violatedUpperBnds_idx)) =
u(finiteUbIdx(violatedUpperBnds_idx));
        X(:) = XOUT;
        shiftedX0 = true;
    end
end

% Display that x0 was shifted in order to honor bounds
if shiftedX0
    if verbosity >= 3
        if strcmpi(OUTPUT.algorithm,interiorPoint)

fprintf(getString(message('optimlib:fmincon:ShiftX0StrictInterior')));
            fprintf('\n');
```

```
        else
            fprintf(getString(message('optimlib:fmincon:ShiftX0ToBnds')));
            fprintf('\n');
        end
    end
end

% Evaluate function
initVals.g = zeros(sizes.nVar,1);
HESSIAN = [];

switch funfcn{1}
case 'fun'
   try
      initVals.f = feval(funfcn{3},X,varargin{:});
   catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:ObjectiveError', ...
            getString(message('optimlib:fmincon:ObjectiveError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
   end
case 'fungrad'
   try
      [initVals.f,initVals.g] = feval(funfcn{3},X,varargin{:});
   catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:ObjectiveError', ...
            getString(message('optimlib:fmincon:ObjectiveError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
   end
case 'fungradhess'
   try
      [initVals.f,initVals.g,HESSIAN] = feval(funfcn{3},X,varargin{:});
   catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:ObjectiveError', ...
            getString(message('optimlib:fmincon:ObjectiveError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
   end
case 'fun_then_grad'
   try
      initVals.f = feval(funfcn{3},X,varargin{:});
   catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:ObjectiveError', ...
```

```
            getString(message('optimlib:fmincon:ObjectiveError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
    try
       initVals.g = feval(funfcn{4},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:GradientError', ...
            getString(message('optimlib:fmincon:GradientError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
case 'fun_then_grad_then_hess'
    try
       initVals.f = feval(funfcn{3},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:ObjectiveError', ...
            getString(message('optimlib:fmincon:ObjectiveError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
    try
       initVals.g = feval(funfcn{4},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:GradientError', ...
            getString(message('optimlib:fmincon:GradientError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
    try
       HESSIAN = feval(funfcn{5},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:HessianError', ...
            getString(message('optimlib:fmincon:HessianError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
otherwise
    error(message('optimlib:fmincon:UndefinedCallType'));
end

% Check that the objective value is a scalar
if numel(initVals.f) ~= 1
    error(message('optimlib:fmincon:NonScalarObj'))
```

```
end

% Check that the objective gradient is the right size
initVals.g = initVals.g(:);
if numel(initVals.g) ~= sizes.nVar
   error('optimlib:fmincon:InvalidSizeOfGradient', ...

getString(message('optimlib:commonMsgs:InvalidSizeOfGradient',sizes.nVar)))
;
end

% Evaluate constraints
switch confcn{1}
case 'fun'
    try
        [ctmp,ceqtmp] = feval(confcn{3},X,varargin{:});
    catch userFcn_ME
        if strcmpi('MATLAB:maxlhs',userFcn_ME.identifier)
                error(message('optimlib:fmincon:InvalidHandleNonlcon'))
        else
            optim_ME = MException('optimlib:fmincon:NonlconError', ...
                getString(message('optimlib:fmincon:NonlconError')));
            userFcn_ME = addCause(userFcn_ME,optim_ME);
            rethrow(userFcn_ME)
        end
    end
    initVals.ncineq = ctmp(:);
    initVals.nceq = ceqtmp(:);
    initVals.gnc = zeros(sizes.nVar,length(initVals.ncineq));
    initVals.gnceq = zeros(sizes.nVar,length(initVals.nceq));
case 'fungrad'
   try
      [ctmp,ceqtmp,initVals.gnc,initVals.gnceq] =
feval(confcn{3},X,varargin{:});
   catch userFcn_ME
       optim_ME = MException('optimlib:fmincon:NonlconError', ...
           getString(message('optimlib:fmincon:NonlconError')));
       userFcn_ME = addCause(userFcn_ME,optim_ME);
       rethrow(userFcn_ME)
   end
   initVals.ncineq = ctmp(:);
   initVals.nceq = ceqtmp(:);
case 'fun_then_grad'
    try
```

```
        [ctmp,ceqtmp] = feval(confcn{3},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:NonlconError', ...
            getString(message('optimlib:fmincon:NonlconError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
    initVals.ncineq = ctmp(:);
    initVals.nceq = ceqtmp(:);
    try
        [initVals.gnc,initVals.gnceq] = feval(confcn{4},X,varargin{:});
    catch userFcn_ME
        optim_ME = MException('optimlib:fmincon:NonlconFunOrGradError', ...
            getString(message('optimlib:fmincon:NonlconFunOrGradError')));
        userFcn_ME = addCause(userFcn_ME,optim_ME);
        rethrow(userFcn_ME)
    end
case ''
   % No nonlinear constraints. Reshaping of empty quantities is done later
   % in this file, where both cases, (i) no nonlinear constraints and (ii)
   % nonlinear constraints that have one type missing (equalities or
   % inequalities), are handled in one place
   initVals.ncineq = [];
   initVals.nceq = [];
   initVals.gnc = [];
   initVals.gnceq = [];
otherwise
   error(message('optimlib:fmincon:UndefinedCallType'));
end

% Check for non-double data typed values returned by user functions
if ~isempty( isoptimargdbl('FMINCON',
{'f','g','H','c','ceq','gc','gceq'}, ...
   initVals.f, initVals.g, HESSIAN, initVals.ncineq, initVals.nceq,
initVals.gnc, initVals.gnceq) )

error('optimlib:fmincon:NonDoubleFunVal',getString(message('optimlib:common
Msgs:NonDoubleFunVal','FMINCON')));
end

sizes.mNonlinEq = length(initVals.nceq);
sizes.mNonlinIneq = length(initVals.ncineq);

% Make sure empty constraint and their derivatives have correct sizes (not
```

```
0-by-0):
if isempty(initVals.ncineq)
    initVals.ncineq = reshape(initVals.ncineq,0,1);
end
if isempty(initVals.nceq)
    initVals.nceq = reshape(initVals.nceq,0,1);
end
if isempty(initVals.gnc)
    initVals.gnc = reshape(initVals.gnc,sizes.nVar,0);
end
if isempty(initVals.gnceq)
    initVals.gnceq = reshape(initVals.gnceq,sizes.nVar,0);
end
[cgrow,cgcol] = size(initVals.gnc);
[ceqgrow,ceqgcol] = size(initVals.gnceq);

if cgrow ~= sizes.nVar || cgcol ~= sizes.mNonlinIneq
   error(message('optimlib:fmincon:WrongSizeGradNonlinIneq', sizes.nVar,
sizes.mNonlinIneq))
end
if ceqgrow ~= sizes.nVar || ceqgcol ~= sizes.mNonlinEq
   error(message('optimlib:fmincon:WrongSizeGradNonlinEq', sizes.nVar,
sizes.mNonlinEq))
end

if diagnostics
   % Do diagnostics on information so far

diagnose('fmincon',OUTPUT,flags.grad,flags.hess,flags.constr,flags.gradcons
t,...

XOUT,sizes.mNonlinEq,sizes.mNonlinIneq,lin_eq,lin_ineq,l,u,funfcn,confcn);
end

% Create default structure of flags for finitedifferences:
% This structure will (temporarily) ignore some of the features that are
% algorithm-specific (e.g. scaling and fault-tolerance) and can be turned
% on later for the main algorithm.
finDiffFlags.fwdFinDiff = strcmpi(options.FinDiffType,'forward');
finDiffFlags.scaleObjConstr = false; % No scaling for now
finDiffFlags.chkFunEval = false;     % No fault-tolerance yet
finDiffFlags.chkComplexObj = false;  % No need to check for complex values
finDiffFlags.isGrad = true;          % Scalar objective
```

```
% Check derivatives
if derivativeCheck && ...                % User wants to check derivatives...
   (flags.grad || ...                    % of either objective or ...
   flags.gradconst && sizes.mNonlinEq+sizes.mNonlinIneq > 0) % nonlinear
constraint function.
    validateFirstDerivatives(funfcn,confcn,X, ...
        l,u,options,finDiffFlags,sizes,varargin{:});
end

% call algorithm
if strcmpi(OUTPUT.algorithm,activeSet) % active-set
    defaultopt.MaxIter = 400; defaultopt.MaxFunEvals = '100*numberofvariables';
defaultopt.TolX = 1e-6;
    defaultopt.Hessian = 'off';
    problemInfo = []; % No problem related data
    [X,FVAL,LAMBDA,EXITFLAG,OUTPUT,GRAD,HESSIAN]=...

nlconst(funfcn,X,l,u,full(A),B,full(Aeq),Beq,confcn,options,defaultopt, ...

finDiffFlags,verbosity,flags,initVals,problemInfo,optionFeedback,varargin{:}
);
elseif strcmpi(OUTPUT.algorithm,trustRegionReflective) %
trust-region-reflective
   if (strcmpi(funfcn{1}, 'fun_then_grad_then_hess') || strcmpi(funfcn{1},
'fungradhess'))
      Hstr = [];
   elseif (strcmpi(funfcn{1}, 'fun_then_grad') || strcmpi(funfcn{1},
'fungrad'))
      n = length(XOUT);
      Hstr = optimget(options,'HessPattern',defaultopt,'fast');
      if ischar(Hstr)
         if strcmpi(Hstr,'sparse(ones(numberofvariables))')
            Hstr = sparse(ones(n));
         else
            error(message('optimlib:fmincon:InvalidHessPattern'))
         end
      end
      checkoptionsize('HessPattern', size(Hstr), n);
   end

   defaultopt.MaxIter = 400; defaultopt.MaxFunEvals = '100*numberofvariables';
defaultopt.TolX = 1e-6;
   defaultopt.Hessian = 'off';
   % Trust-region-reflective algorithm does not compute constraint
```

```
    % violation as it progresses. If the user requests the output structure,
    % we need to calculate the constraint violation at the returned
    % solution.
    if nargout > 3
        computeConstrViolForOutput = true;
    else
        computeConstrViolForOutput = false;
    end

    if isempty(Aeq)
       [X,FVAL,LAMBDA,EXITFLAG,OUTPUT,GRAD,HESSIAN] = ...

sfminbx(funfcn,X,l,u,verbosity,options,defaultopt,computeLambda,initVals.f,
initVals.g, ...

HESSIAN,Hstr,flags.detailedExitMsg,computeConstrViolForOutput,optionFeedbac
k,varargin{:});
    else
       [X,FVAL,LAMBDA,EXITFLAG,OUTPUT,GRAD,HESSIAN] = ...

sfminle(funfcn,X,sparse(Aeq),Beq,verbosity,options,defaultopt,computeLambda,
initVals.f, ...

initVals.g,HESSIAN,Hstr,flags.detailedExitMsg,computeConstrViolForOutput,op
tionFeedback,varargin{:});
    end
elseif strcmpi(OUTPUT.algorithm,interiorPoint)
    defaultopt.MaxIter = 1000; defaultopt.MaxFunEvals = 3000; defaultopt.TolX
= 1e-10;
    defaultopt.Hessian = 'bfgs';
    mEq = lin_eq + sizes.mNonlinEq + nnz(xIndices.fixed); % number of equalities
    % Interior-point-specific options. Default values for lbfgs memory is 10,
and
    % ldl pivot threshold is 0.01
    options =
getIpOptions(options,sizes.nVar,mEq,flags.constr,defaultopt,10,0.01);

    [X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,HESSIAN] =
barrier(funfcn,X,A,B,Aeq,Beq,l,u,confcn,options.HessFcn, ...

initVals.f,initVals.g,initVals.ncineq,initVals.nceq,initVals.gnc,initVals.g
nceq,HESSIAN, ...
        xIndices,options,optionFeedback,finDiffFlags,varargin{:});
else % sqp
```

```
    defaultopt.MaxIter = 400; defaultopt.MaxFunEvals =
'100*numberofvariables';
    defaultopt.TolX = 1e-6; defaultopt.Hessian = 'bfgs';
    % Validate options used by sqp
    options = getSQPOptions(options,defaultopt,sizes.nVar);
    optionFeedback.detailedExitMsg = flags.detailedExitMsg;
    % Call algorithm
    [X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,HESSIAN] =
sqpLineSearch(funfcn,X,full(A),full(B),full(Aeq),full(Beq), ...

full(l),full(u),confcn,initVals.f,full(initVals.g),full(initVals.ncineq),fu
ll(initVals.nceq), ...

full(initVals.gnc),full(initVals.gnceq),xIndices,options,finDiffFlags,verbo
sity,optionFeedback,varargin{:});
end



options = optimset('largescale','off');
[x]=llw('fun1',rand(3,1),[],[],[],[],zeros(3,1),[],'fun2',options)
%初始值是个随意的数字
y=5.5
```