基于粒子群算法的火箭残骸精确定位算法

摘要

大部分火箭为多级火箭，火箭助推器完成既定任务后就会脱落。因此精确回收火箭残骸显得尤为重要，本文聚焦于火箭残骸的精确定位的一系列问题展开分析。

问题一，类比平面内通过三个圆确定一个点的思路，得到空间内不在同一平面的四个球可以确定一个点，然后列出空间球方程，发现至少需要**四个方程**才可以精确确定火箭残骸的空间位置和脱离时刻的数据。然后通过**粒子群算法**，综合 7 个监测点数据，得出最符合条件的结果，得出残骸**经度为 110.67°**，**纬度为 27.07°**，**高程为 1103.70m**，音爆发生时间相对于开始计时时刻为**-16.77s**。

问题二，在多个残骸发生音爆时，监测设备会接受到多组时间数据，但是由于残骸发生音爆位置与时间均不同，抵达顺序会有所变化，因此采取对抵达顺序进行全排列的，求解误差最小即为正确的抵达顺序。构建**非线性规划模型**，再利用**粒子群算法**求解，与问题一同理，**至少需要四个监测设备**才能定位。

问题三，运用题给数据，在问题二建立的模型的基础上，求解四个残骸的音爆位置与时间，发现第一个残骸结果已在第一问给出，简化求解残骸数量，确定**目标函数与约束条件**，同样运用**粒子群算法**求解，结果详见附录 A。

问题四，要求在监测设备有无法消除的 0.5s 的误差的情况下，给出修正模型，做到能在 1km 误差范围内精确定位。首先用随机数在问题三数据基础上修改，给出示例数据。接着将示例数据代入运用**蒙特卡洛算法**改进后的问题二模型，得到各抵达顺序组合的误差表，对误差进行**层次聚类**，以**曼哈顿距离**为聚类依据，筛选出 8 个更具有优势的可能抵达顺序组合。接着将范围缩小为**8 个组合**，运用加减随机数的修正模型进行修正计算，得出编号为 81 的组合具有更好的趋向性，在**10000 次**修正中，指向 81 号组合有**3867 次**。最后，固定组合，再次对示例数据进行修正，得到四个残骸结果如附录 A。**最大位置误差为 831.83m**，**平均时间误差较原数据为 1.001s**，认为修正模型效果良好。

关键字： 粒子群算法； 非线性规划； 层次聚类法； 曼哈顿距离； 蒙特卡洛算法

目录

一、 问题重述

1.1 问题背景

大部分火箭为多级火箭，火箭助推器完成既定任务后就会脱落。此时，准确判断火箭残骸推落时间，脱离的位置，对火箭回收利用有着极其重要的作用。为了快速回收火箭，在火箭理论落点位置附近，设置震动波检测装置，通过不同火箭残骸从空中传来的跨音速音爆，然后根据音爆抵达的时间，定位空中残骸发生音爆时的位置，再采用弹道外推实现残骸落地点的快速精准定位。

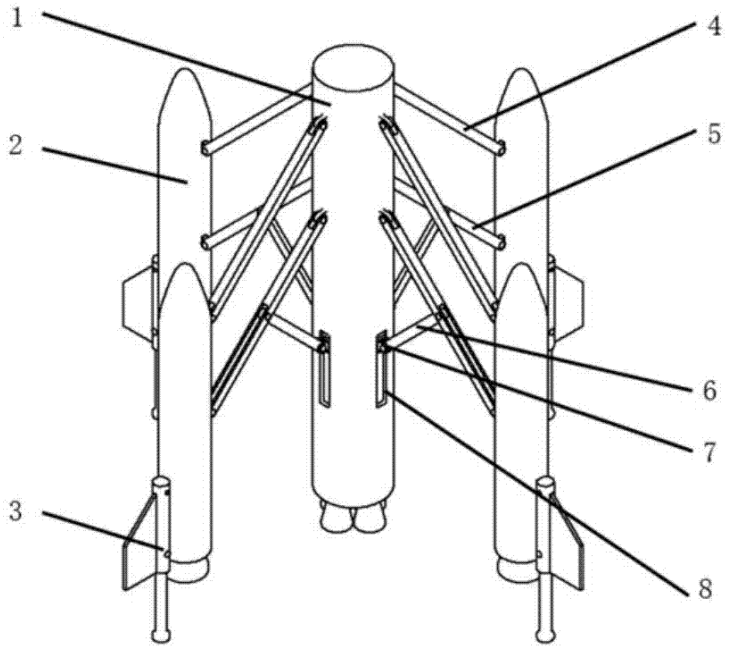

图 **1**　火箭助推器分离

1.2 问题要求

本文围绕通过监测设备的监测数据，对火箭残骸位置进行定位的一系列问题的解决来展开。

1.2.1 问题一

(1) 建立数学模型分析如果要精准确定空中单个残骸发生音爆时的位置坐标（经度、纬度、高程）和时间，至少需要布置几台监测设备?

(2) 假设某火箭一级残骸分离后，在落点附近布置了 7 台监测设备，各台设备三维坐标（经度、纬度、高程）、音爆抵达时间（相对于观测系统时钟 0 时），选取合适的数据，计算残骸发生音爆时的位置和时间。

1.2.2 问题二

(1) 在多个残骸发生音爆时，监测设备在监测范围内可能会采集到几组音爆数据。假设空中有 4 个残骸，每个设备按照时间先后顺序收到 4 组震动波。建立数学模型，分析如何确定监测设备接收到的震动波是来自哪一个残骸？

(2) 如果要确定 4 个残骸在空中发生音爆时的位置和时间，至少需要布置多少台监测设备？

1.2.3 问题三

根据题给的四个音爆抵达时间的数据，利用问题 2 所建立的数学模型，选取合适的数据，确定 4 个残骸在空中发生音爆时的位置和时间（4 个残骸产生音爆的时间可能不同，但互相差别不超过 5 s）。

1.2.4 问题四

(1) 假设设备记录时间存在 0.5s 的随机误差，修正问题 2 所建立的模型以较精确地确定 4 个残骸在空中发生音爆时的位置和时间。

(2) 通过对问题 3 表中数据叠加随机误差，给出修正模型的算例，并分析结果误差。如果时间误差无法降低，提供一种解决方案实现残骸空中的精准定位（误差 <1km）。

(3) 根据问题 3 所计算得到的定位结果模拟所需的监测设备位置和音爆抵达时间数据，验证相关模型。

二、 问题分析

2.1 问题一分析

2.1.1 问题分析

要解决该问题，首先要了解该监测设备的定位原理，如图 2，以监测站点为中心，画一个以传播距离为半径的圆，如果在一个平面内，三个圆就能准确得定位一个点

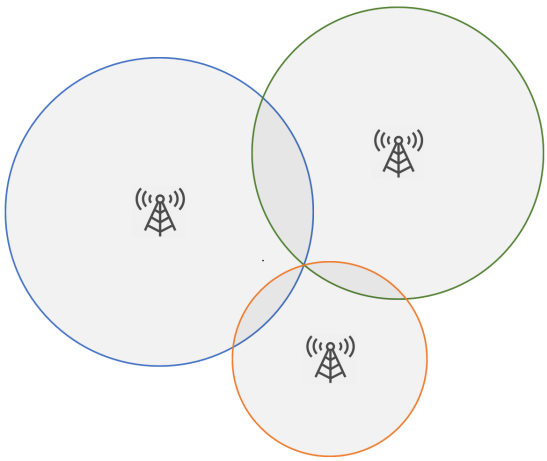

图 **2**　监测设备定位原理

而对于三维空间内的此类问题，不难发现，可以直接用球方程求解：

$$\sqrt{(x-a_i)^2+(y-b_i)^2+(z-c_i)^2}=r \tag{1}$$

接着确定未知数的个数，按照题给要求，我们需要精确确定火箭残骸的经度，纬度，高程，发生音爆的时间，即 4 个未知数，不难看出，需要四个方程连立求解。因此要得到关于火箭残骸发生音爆的位置和时间，至少要布置 4 个监测站点，且这 4 个监测站点不能有相同的经度、纬度或高程。

2.1.2 求解思路

首先，经过问题分析，需要求解的是一组四元二次方程组，求解该类方程的计算会较为复杂，因此将该问题转化为二次规划问题，目标函数确定为方程的误差平方和，然后利用启发式算法求最优解。

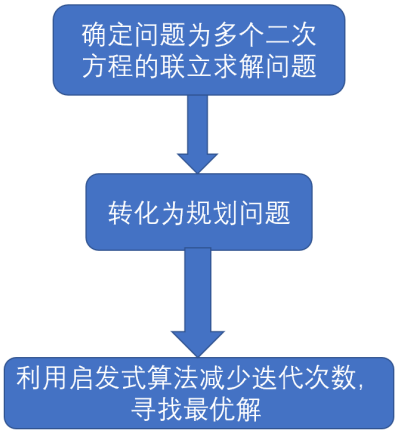

图 **3**　问题一流程图

2.2 问题二分析

2.2.1 问题分析

相对于监测设备而言，接收到四个残骸的时间数据会因为残骸发生音爆的位置与监测设备的位置偏差不同而导致先脱落的残骸可能因为离监测设备较远而声音传播到监测设备所需的时间很久，而后脱落的残骸可能因为离监测设备较近而声音传播到监测设备所需的时间要更短，当二者传播的时间差值大于二者发生音爆的时间差值，就会出现监测设备先检测到后脱落的残骸发出的震动波而后检测到先脱落残骸发出的震动波，所以该问题主要是要对监测设备所测得的时间值与残骸进行一对一匹配。

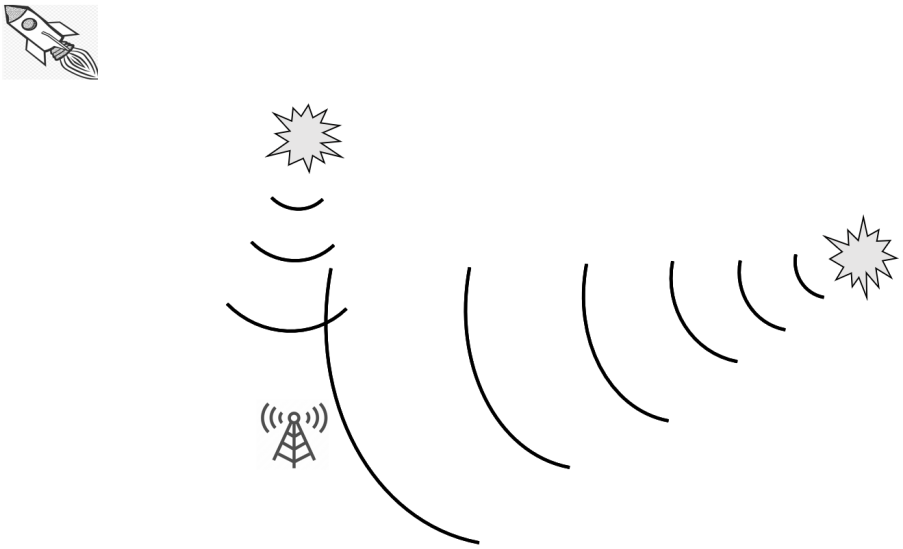

图 **4**　声波传播示意图

而由问题一知，单个残骸需要四个站点的数据，而各个残骸之间在时间经过匹配之后数据相互独立，所以也同样需要四个站点的数据。

2.2.2 求解思路

经过问题分析，我们首先需要获得不同的时间与残骸的匹配情况。可以通过对残骸标号的方式进行全排列，获得时间序列对应残骸序号的所有可能情况，对四个站点都有全排列的可能情况，将其再进行组合就可以获得四个站点的对应序号矩阵。

而后在确定残骸在每个站点的对应时间后，延续第一问的思路构建目标函数为四个残骸的误差平方和总和，然后利用启发式算法求出目标函数最小值，再比对不同组合的目标函数最小值，误差平方和总和最小的对应序号矩阵就是残骸音爆震动波传至监测设备的相对顺序。

2.3 问题三分析

2.3.1 问题分析

根据第二问所建立的模型，对第三问数据带入进行计算。首先，观察数据，发现第三问所给数据中有第一问已经给出的重复数据，为降低计算复杂度和简化模型，认为四个残骸中有一个残骸已知，即第一问计算出来的残骸坐标。接着利用第二问模型确立相应的目标函数，写出对应的约束条件。并利用粒子群算法求解。

2.3.2 求解思路

(1) 对题给数据进行筛选，得出剩下三个残骸的音爆抵达时间表。
(2) 对残骸可能出现的顺序进行全排列，给出对应的序号矩阵。
(3) 按照目标函数和对应的约束条件，运用粒子群算法得出所有符合条件的解，按照适应度和实际情况筛选剩余的解。

2.4 问题四分析

2.4.1 问题分析

问题四要求在第三问问给出的模型的基础上，优化结果要求结果能应对监测设备可能存在的 0.5s 的误差。通过分析第二问建立的模型，可以发现，寻找残骸相利用对所有可能出现的抵达顺序进行枚举，然后一一计算它们的适应度，选取适应度最小的组合作为正确组合。但是，0.5s 的随机误差不仅仅放大了定位精确距离的误差，而且可能在残骸的相对抵达顺序方面给出错误解。由于修正算法可能会进一步放大这种误差，所以这种错误对精确定位带来的影响是巨大的，下面针对这个方面来优化算法。

2.4.2 求解思路

求解思路主要聚焦于缩小可能的抵达顺序范围，防止在模型修正计算时选取到错误组合，影响定位精度。

(1) 数据准备
对题给数据进行加减 0.5s 以内的随机误差。
(2) 计算每种组合的误差值
按 (1) 得到的数据进行问题三相同的处理，得到每种可能的抵达顺序组合的数据。
(3) 聚类选取误差最小的几个组合
对 (2) 中数据进行分析，聚类，找到几个能稳定给出较低误差的组合，将这几个组合作为新的抵达顺序组合的集合。
(4) 在误差小的组合范围内修正选取趋向性最高的组合

运用修正算法，具体为：对 (1) 中数据随机还原 0.5s 以内的随机误差，进行多次重复计算，并利用蒙特卡洛寻找初始解的方法降低时间复杂度，在抵达组合中找到趋向性最高的一个组合，这个组合即为目标组合。

(5) 再次修正得出结果

接着以确定的目标组合，按照随机还原 0.5s 以内随机误差的方法进行重复计算，误差最低的结果即为修正后的结果。

三、 模型假设

(1) 音爆的传播的平均速度始终为 340m/s，不受介质变化的影响。

(2) 计算两点间距离时可忽略地面曲率，纬度间每度距离值近似为 111.263 km，经度间每度距离值近似为 97.304 km。

(3) 假设问题一与问题三给出的数据没有测量误差。

四、 符号说明

符号	说明
x	火箭残骸经度计算值
y	火箭残骸纬度计算值
z	火箭残骸高程计算值
a_i	监测点经度计算值
b_i	监测点纬度计算值
c_i	监测点高程计算值
t	火箭残骸音爆实际发生时刻
t_i	音爆抵达时间
c_1	个体认知加速常数
c_2	社会经验加速常数
$pbest_{i,d}$	粒子 i 在维度 d 上的个体最优位置
$x_{i,d}(t)$	粒子 i 在时间 t 在维度 d 上的当前位置
$gbest_d$	全局最优在维度 d 上的位置

五、 模型的建立与求解

5.1 问题一

5.1.1 数据预处理

针对第一问数据，为方便计算，取 A 监测设备的近卫笃为原点坐标，忽视地面曲率后，取纬度间每度距离值近似为 111.263 km，经度间每度距离值近似为 97.304 km。数据转化后如表 1：

表 **1**　检测设备采集的数据 **(**转换后**)**

X(m)	Y(m)	高程 (m)	音爆抵达时间 (s)
0	0	824	100.77
52446.85	28038.27	727	112.22
45830.18	64643.80	742	188.02
973.04	69094.32	850	258.98
27537.03	45951.62	786	118.44
21990.70	79775.57	678	266.87

5.1.2 模型的建立

根据问题分析，不难发现我们需要至少 4 个监测设备才可以确定残骸的位置，对所取的 B,D,F,G 四个监测设备做初步可视化，发现可以找到一个近似的残骸位置。

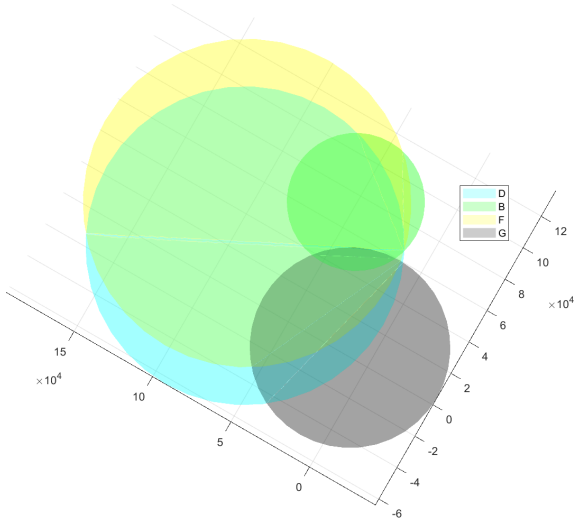

图 **5**　**B,D,F,G** 站点的模拟图

首先，确定目标函数，由于起爆位置和时间未知，设音爆残骸位置为 x,y,z, 爆炸时刻为 t。构建如公式（2）的目标函数。

$$min \sum_{i=1} |\sqrt{(x-a_i)^2+(y-b_i)^2+(z-c_i)^2}-(t-t_i)\times 340| \tag{2}$$

接着，运用启发式算法对该目标函数求解，下面运用粒子群算法求解:

$$v_{i,d}(t+1) = w \cdot v_{i,d}(t) + c_1 \cdot r_1 \cdot (pbest_{i,d} - x_{i,d}(t)) + c_2 \cdot r_2 \cdot (gbest_d - x_{i,d}(t)) \tag{3}$$

这个公式是粒子群算法中用于更新粒子速度的核心公式。它包含三个部分:

1. 惯性项 (Inertia): $w \cdot v_{i,d}(t)$
   这部分表示粒子当前的速度对下一时刻速度的影响。w 是惯性权重，控制前一速度的影响程度。
2. 个体认知项 (Cognitive): $c_1 \cdot r_1 \cdot (pbest_{i,d} - x_{i,d}(t))$
   这部分表示粒子对自己历史最优位置的认知。
   c_1 是个体认知加速常数，控制这部分的影响。
   r_1 是一个随机数，增加搜索的随机性。
   $pbest_{i,d}$ 是粒子 i 在维度 d 上的个体最优位置。
   $x_{i,d}(t)$ 是粒子 i 在时间 t 在维度 d 上的当前位置。
3. 社会经验项 (Social): $c_2 \cdot r_2 \cdot (gbest_d - x_{i,d}(t))$
   这部分表示粒子对群体最优位置的认知。
   c_2 是社会经验加速常数，控制这部分的影响。
   r_2 是另一个随机数。
   $gbest_d$ 是全局最优在维度 d 上的位置。
   通过这三部分的综合作用, 粒子的速度会不断更新, 从而引导粒子向全局最优位置移动。这就是粒子群算法的核心思想。

粒子位置的更新过程,其中 $x_{i,d}(t)$ 是粒子 i 在维度 d 上在时间 t 的当前位置,$v_{i,d}(t+1)$ 是粒子 i 在维度 d 上在时间 $t+1$ 的速度，$x_{i,d}(t+1)$ 是粒子 i 在维度 d 上在时间 $t+1$ 的更新后的位置。

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \tag{4}$$

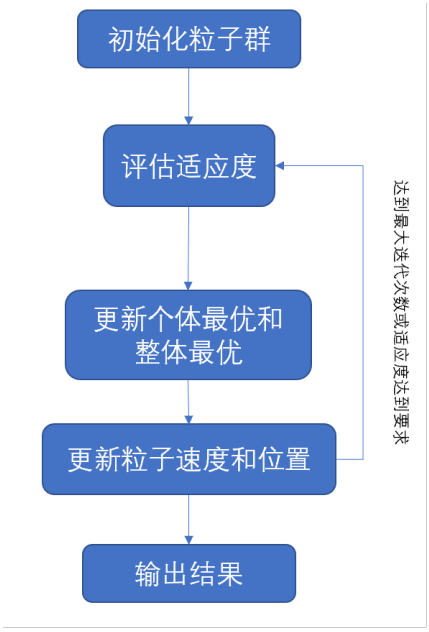

图 **6**　粒子群算法流程图

5.1.3 模型的求解

根据算法思想计算最终结果如表 2：

表 **2**　第一问计算所得最终结果

经度	纬度	高程 (m)	音爆发生时间 (s)	累计误差
110.6728286°	27.07314712°	1103.69901	-16.7707758	30999.25

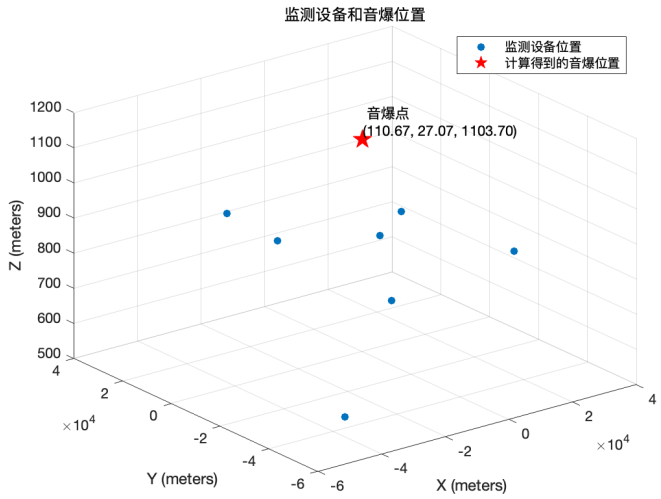

图 **7**　第一问残骸位置

表中累计误差为所能找到的最小的目标函数值，但是仍与我们期望有不小的的距离。将计算结果带入各个监测设备的方程得到各个基监测设备的误差值，发现误差和的主要来源来自与 C，D，E 三个监测设备，因此认为这三个监测设备在此次监测任务中受到了较大的影响，不能很好地反映火箭残骸的音爆位置，在此选择性剔除。

5.2 问题二

5.2.1 数据预备

根据问题分析，我们首先要得到四个监测设备对应残骸的时间序列矩阵。标记全排列各个排列的序号如表所示

表 **3**　排列对应序号 **(**部分**)**

序号	1	2	3	4	5	6	7	8
	1234	1243	1324	1342	1423	1432	2134	2143
序号	9	10	11	12	13	14	15	16
	2314	2341	2413	2431	3124	3142	3214	3241
序号	17	18	19	20	21	22	23	24
	3412	3421	4123	4132	4213	4231	4312	4321

每个监测设备选取 24 个序号中的一个组合成时间序列矩阵，用四重循环实现。

5.2.2 模型的建立

确定目标函数，由于四个音爆残骸的位置和时间都未知，故设四个音爆残骸位置为 x_j,y_j,z_j, 爆炸时刻为 t_j，第 i 个监测设备接收到第 j 个残骸的音爆时间为 t_{ij}，构建如公式（5）的目标函数。

$$min\sum_{j=1}^{4}\sum_{i=1}^{4}\sqrt{(x_j-a_i)^2+(y_j-b_i)^2+(z_j-c_i)^2}-(t_{ij}-t_j)\times 340 \tag{5}$$

接着参照问题一，利用粒子群算法得到适应值最小的序号矩阵，获得残骸音爆震动波传至监测设备的相对顺序即可。

5.3 问题三

5.3.1 数据的预处理

首先筛选出三个残骸的可能的时间数据，如表 4。

表 **4**　三个残骸的可能的时间数据

设备	音爆抵达时间 (s)		
A	164.229	214.85	270.065
B	92.453	169.362	196.583
C	75.56	110.696	156.936
D	94.653	141.409	196.517
E	78.6	86.216	126.669
F	67.274	166.27	175.482

接着根据问题一得出的结论，认为 A，B，F，G 四个监测装置的数据更具有可靠性，对这四组音爆抵达时间进行全排列处理，得到 216 个 3x4 的矩阵，其中每个矩阵代表一种可能的音爆抵达顺序。

5.3.2 模型的建立与求解

对简化后的问题利用粒子群算法进行求解，具体算法详见附录 C，按照适应度进行筛选，最符合的结果如下表：

表 **5**　三个残骸的抵达时间和位置

	x(m)	y(m)	高程 (m)	抵达时间 (s)
残骸一	34672.7249	84715.9729	23224.4855	-17.6949759
残骸二	50041.0653	50999.0696	28000.9286	-12.7055895
残骸三	-6386.82961	50634.5237	36808.5605	-17.7055895

残骸的音爆相对抵达数据如下，即残骸一的音爆在所求解的三个残骸中第一个抵达 A 监测点，第三个抵达 B 监测点，第三个抵达 F 监测点，第一个抵达 G 监测点，以此类推，得到了残骸音爆抵达的顺序和位置。

表 **6**　三个残骸的音爆在四个监测点的相对抵达顺序

	A	B	F	G
残骸一	1	3	3	1
残骸二	2	1	2	3
残骸三	3	2	1	2

最终结果如表 7 所示，接着加入第一问中的残骸音爆数据，画出四个残骸的音爆位置与监测点的相对空间图。

表 **7**　四个残骸的音爆位置

	经度 (°)	纬度 (°)	高程	时间
残骸一	110.5973	27.9654	23224.49	-17.695
残骸二	110.7553	27.66237	28000.93	-12.7056
残骸三	110.1754	27.65909	36808.56	-17.7056
残骸四	110.6728	27.07315	1103.699	-16.7708

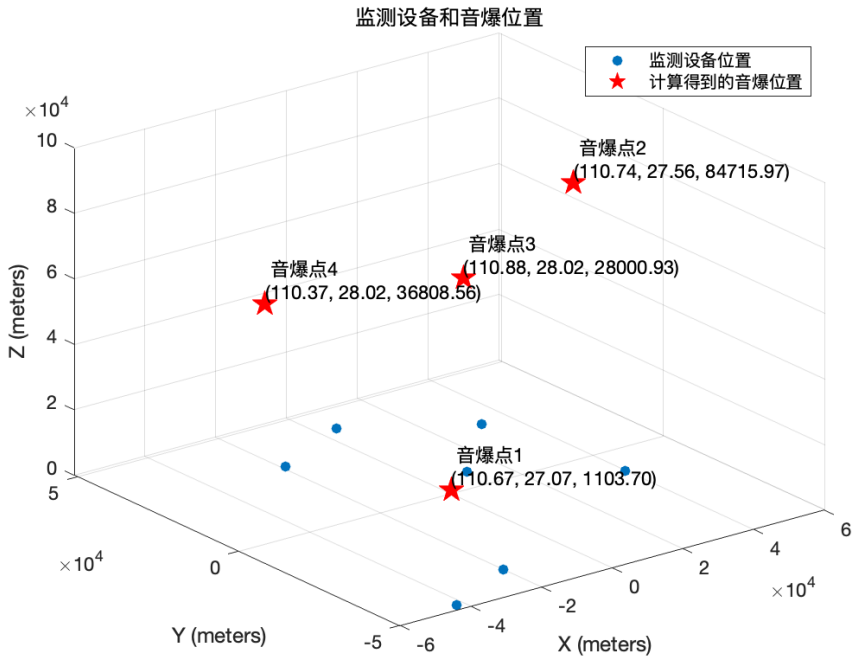

图 **8**　四个残骸的音爆位置图

5.4 问题四

5.4.1 数据的预处理与模型建立

首先对第三问中的时间数据加小于等于 0.5s 的随机误差。如表 8：

表 **8**　对数据加随机误差

监测点	经度	纬度	高程	残骸音爆抵达时间（s）			
A	110.241	27.204	824	100.767	164.229	214.85	270.065
B	110.783	27.456	727	92.453	112.22	169.362	196.583
F	110.467	28.081	678	67.274	166.27	175.482	266.871
G	110.047	27.521	575	103.738	163.024	206.789	210.306

依照表中数据使用粒子群算法进行第一次计算，计算结果，首先画出相对抵达顺序组合编号——适应度的折线图，发现编号为 68,72,77,81,95,99,104,108 的排列方式的误差值远小于总体，再运用层次聚类的方法来验证该结论。

这里使用曼哈顿距离作为聚类依据，曼哈顿距离的计算公式为：

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i| \tag{6}$$

这里由于只对适应度单列数据进行聚类分析，因此公式简化为 (7)，其中 c_i 表示第 i 个中心点。结果如图11所示，同样说明了编号为 68,72,77,81,95,99,104,108 的组合更加符合要求。

$$d(x,c_i)=|x-c_i| \tag{7}$$

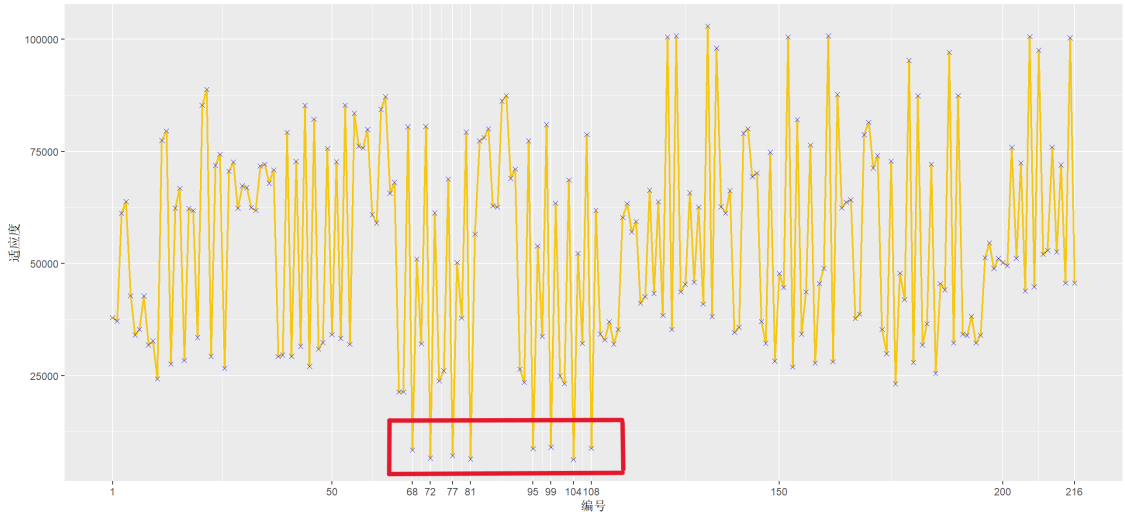

图 **9**　计算适应度结果折线图

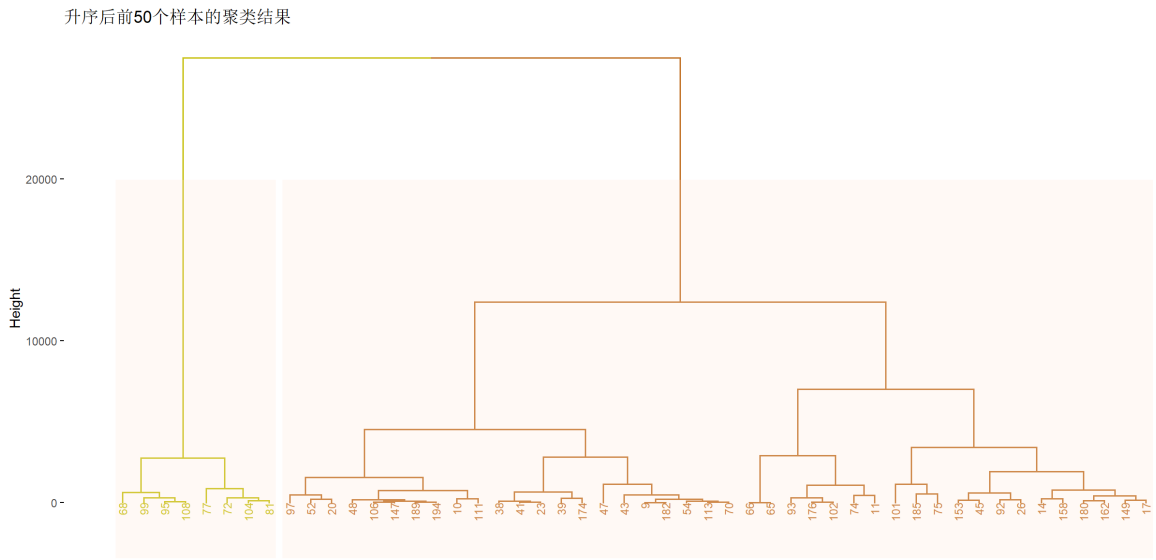

图 **10**　升序排列后前 **50** 个数据的树状聚类图

5.4.2 修正模型的求解

在上文提到的更优的组合中，对原数据进行 0.5s 以内的随机修正，再辅之以蒙特卡洛算法求初始解的方式，来降低时间复杂度，具体代码详见附录 C。求解结果显示，编号为 81 号的组合在多次计算中都显示出了较低的适应度，结果显示对 81 号抵达顺序的趋向性较其他组合高，在 10000 次的随机修正中，81 号组合显示更低适应度的次数达到了 3867 次。故认为 81 号组合为目标组合。

接着确定组合后，再次进行 0.5s 以内的随机修正，得出最终结果如下表，其中 x，y 为相对于 A 监测点的距离 (m).

表 **9**　修正算法得出结果

	x	y	高程	抵达时间
残骸一	34553.7041	84410.5488	22459.9577	-16.2085486
残骸二	49868.0507	51036.95	27532.522	-11.7707758
残骸三	-6242.73407	50665.1467	36382.2927	-16.7707758
残骸四	41490.6482	-14441.084	1055.69901	-16.1207328

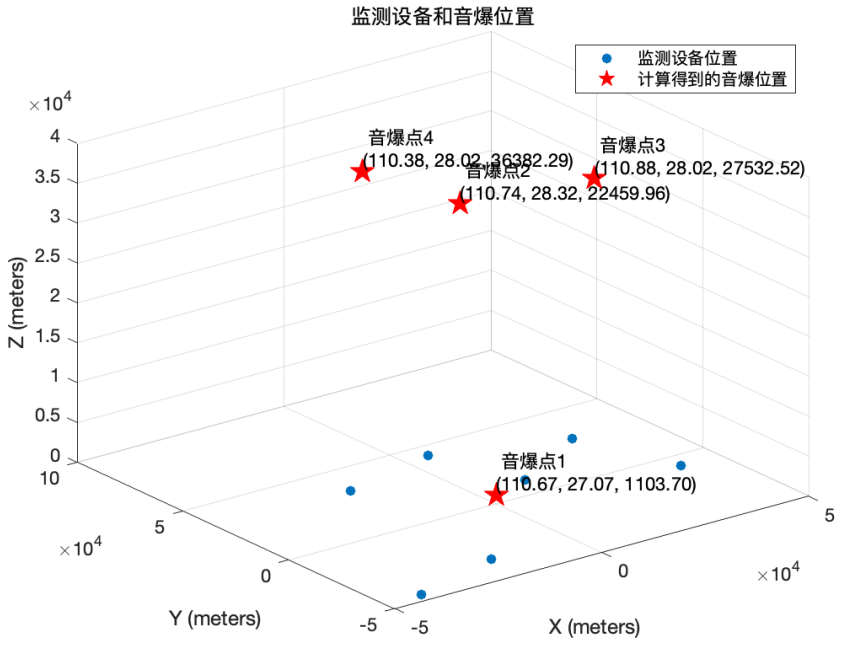

图 **11**　修正后相对位置图

5.4.3 修正模型误差分析

假定第三问计算结果为火箭残骸实际精确音爆发生位置，综合计算模型各残骸平均时间误差为 1.001513675s，平均位置误差为 581.6912729m

表 **10**　修正前后数据对比

原数据	x	y	高程	抵达时间
残骸一	34672.7249	84715.9729	23224.4855	-17.6949759
残骸二	50041.0653	50999.0696	28000.9286	-12.7055895
残骸三	-6386.82961	50634.5237	36808.5605	-17.7055895
残骸四	42018.6482	-14559.084	1103.69901	-16.7707758
修正后	x	y	高程	抵达时间
残骸一	34553.7041	84410.5488	22459.9577	-16.2085486
残骸二	49868.0507	51036.95	27532.522	-11.7707758
残骸三	-6242.73407	50665.1467	36382.2927	-16.7707758
残骸四	41490.6482	-14441.084	1055.69901	-16.1207328
抵达时间之差	-1.4864273	-0.9348137	-0.9348137	-0.650043
相对位置之差	831.8368762	500.7731217	451.0050222	543.1500713

可以发现该算法将位置误差控制在了 1km 以内，认为满足题目要求。

六、 模型的评价

6.1 模型的优点

(1) 粒子群算法

简单易实现：粒子群算法易于理解和实现，不需要太多高级数学知识。

全局搜索能力：PSO 能够在大范围内搜索解空间，并且具有一定的全局搜索能力。

容易并行化：粒子之间的独立性使得 PSO 易于并行化，可以有效地利用多核计算资源

(2) 层次聚类法

无需预先指定聚类数量：层次聚类不需要预先指定聚类的数量，可以根据聚类结果的树状结构灵活地选择合适的聚类数量。

可视化展示：层次聚类的结果可以通过树状图（树状图）直观展示不同层次的聚类结构。

对异常值不敏感：层次聚类对异常值的影响较小，因为它是基于数据点之间的相似性进行聚类的。

(3) 蒙特卡洛算法

这里运用蒙特卡洛算法来求初始解，限制了粒子群算法易陷入局部最优的缺点。并且，蒙特卡洛算法的计算过程通常可以并行化，可以利用并行计算资源提高计算效率。

6.2 模型的缺点

(1) 粒子群算法过早收敛：粒子群算法容易陷入局部最优解，特别是在高维空间和复杂的优化问题中。

参数敏感：PSO 的性能很大程度上取决于参数的选择，不同问题需要调整不同的参数。

缺乏理论支持：PSO 缺乏深入的理论基础，对于其工作原理的解释并不十分清晰。

(2) 随机数修正模型计算复杂度大，可能会有重复求解的情况，无法穷举所有情况。求解结果随机性大，可能无法稳定的求出统一的结果。

(3) 问题四模型问题四模型涉及多层重复计算步骤，时间复杂度极大，因此采取了多种方式来简化和优化模型。最终结果的呈现只是对原题数据的一种随机误差计算，未能考虑到所有情况下，改模型的普适性。可能存在极端情况，普适性难以论证。

参考文献

[1] 杨维, 李歧强. 粒子群优化算法综述 [J]. 中国工程科学,2004,6(5):87-94. DOI:10.3969/j.issn.1009-1742.2004.05.018.

[2] 李爱国, 覃征, 鲍复民, 等. 粒子群优化算法 [J]. 计算机工程与应用,2002,38(21):1-3,17. DOI:10.3321/j.issn:1002-8331.2002.21.001.

[3] 吕振肃, 侯志荣. 自适应变异的粒子群优化算法 [J]. 电子学报,2004,32(3):416-420. DOI:10.3321/j.issn:0372-2112.2004.03.016.

[4] 王靖, 张金锁. 综合评价中确定权重向量的几种方法比较 [J]. 河北工业大学学报,2001,30(2):52-57. DOI:10.3969/j.issn.1007-2373.2001.02.012.

[5] 庄锁法. 基于层次分析法的综合评价模型 [J]. 合肥工业大学学报（自然科学版）,2000,23(4):582-585,590. DOI:10.3969/j.issn.1003-5060.2000.04.029.

[6] GUOQIANG MAO, BARIS FIDAN, BRIAN D.O. ANDERSON. Wireless sensor network localization techniques[J]. Computer networks,2007,51(10):2529-2553.

[7] 邓平, 李莉, 范平志. 一种 TDOA/AOA 混合定位算法及其性能分析 [J]. 电波科学学报,2002,17(6):633-636. DOI:10.3969/j.issn.1005-0388.2002.06.018.

[8] 韩霜, 罗海勇, 陈颖, 等. 基于 TDOA 的超声波室内定位系统的设计与实现 [J]. 传感技术学报,2010,23(3):347-353. DOI:10.3969/j.issn.1004-1699.2010.03.012.

附录 A　问题三四答案

表 **12**　问题三结果

	经度 (°)	纬度 (°)	高程	时间
残骸一	110.5973	27.9654	23224.49	-17.695
残骸二	110.7553	27.66237	28000.93	-12.7056
残骸三	110.1754	27.65909	36808.56	-17.7056
残骸四	110.6728	27.07315	1103.699	-16.7708

表 **13**　问题四结果

	经度 (°)	纬度 (°)	高程	时间
残骸一	110.5961108	27.96265785	23224.4855	-17.6949759
残骸二	110.7534974	27.6627055	28000.9286	-12.7055895
残骸三	110.176843	27.65936384	36808.5605	-17.7055895
残骸四	110.6674023	27.07420767	1103.69901	-16.7707758

附录 B　第一问粒子群算法

```
clc;clear;close all;

%原始数据
data=xlsread("firstdata.xlsx");
%以一个点为原点
%经度
x0=(data(:,1)-data(1,1))*97304;
%纬度
y0=(data(:,2)-data(1,2))*111263;
%海拔
z0=data(:,3);
%收取时间
t0=data(:,4);
```

```
% 适应度函数表达式(求这个函数的最大值)
%x=[x,y,z,t]
f= @(x) abs(sqrt((x(1)-x0(1))^2+(x(2)-y0(1))^2+(x(3)-z0(1))^2)-340*(t0(1)-x(4)))...
+abs(sqrt((x(1)-x0(3))^2+(x(2)-y0(3))^2+(x(3)-z0(3))^2)-340*(t0(3)-x(4)))...
+abs(sqrt((x(1)-x0(5))^2+(x(2)-y0(5))^2+(x(3)-z0(5))^2)-340*(t0(5)-x(4)))...
+abs(sqrt((x(1)-x0(4))^2+(x(2)-y0(4))^2+(x(3)-z0(4))^2)-340*(t0(4)-x(4)))...
+abs(sqrt((x(1)-x0(2))^2+(x(2)-y0(2))^2+(x(3)-z0(2))^2)-340*(t0(2)-x(4)))...
+abs(sqrt((x(1)-x0(6))^2+(x(2)-y0(6))^2+(x(3)-z0(6))^2)-340*(t0(6)-x(4)))...
+abs(sqrt((x(1)-x0(7))^2+(x(2)-y0(7))^2+(x(3)-z0(7))^2)-340*(t0(7)-x(4)));

%figure(1);
%fplot(f, [0 20], 'b-');           % 画出初始图像

d = 4;                          % 空间维数
N = 1500;                         % 初始种群个数
x_limit = [-100000, 130000;
-100000, 130000;
0, 100000;
-100,100];            % 设置位置限制
v_limit = [-100, 100;-100, 100;-10, 10;-1, 1];      % 设置速度限制
for i=1:4
x(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1); %初始每个粒子的位置
end

for i=1:4
v(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
end% 初始每个粒子的速度

pbest = x;                      % 初始化每个个体的历史最佳位置
gbest = inf(1, d);          % 初始化种群的历史最佳位置

fp_best = inf(N, 1);        % 初始化每个个体的历史最佳适应度 为 正无穷
fg_best = inf;               % 初始化种群历史最佳适应度 为 +无穷

iter = 2000;                    % 最大迭代次数
w = 0.9;                        % 惯性权重
c1 = 0.7;                       % 自我学习因子
c2 = 0.2;                       % 群体学习因子

%hold on;
%plot(x, f(x), 'ro');
%title('初始状态图');


record = zeros(iter, 1);    % 记录器(用于记录 fg_best 的变化过程)
figure(2);
i = 1;
```

```
cnt=0;
while i <= iter
for k=1:N
fx(k)=f(x(k,:));
end              % 计算个体当前适应度
for j = 1:N
if fp_best(j) > fx(j)
fp_best(j) = fx(j); % 更新个体历史最佳适应度
pbest(j,:) = x(j,:); % 更新个体历史最佳位置
end
end
% if fix(min(fp_best)*1000)==fix(fg_best*1000)
%     cnt=cnt+1;
% end
% if cnt==100
%     cnt=0;
%     for k=1:4
% x(1:N,k) = x_limit(k,1) + (x_limit(k,2) - x_limit(k,1)) * rand(N,1); %初始每个粒子的位置
%     end
%     v = rand(N, d); % 初始每个粒子的速度
%     i=i+1;
%     disp(['2第',num2str(i),'次最佳适应度: ',num2str(min(fx))]);
%     continue;
% end

if fg_best > min(fp_best)
[fg_best,ind_min] = min(fp_best); % 更新群体历史最佳适应度
gbest = pbest(ind_min,:);          % 更新群体历史最佳位置，其中 ind_max 是最大值所在的下标
% 注: 将上面的两个式子换成下面两个是不可以的
% [gbest, ind_max] = max(pbest); % 更新群体历史最佳位置，其中 ind_max 是最大值所在的下标
% fg_best = fp_best(ind_max);    % 更新群体历史最佳适应度
end
for k=1:N
for p=1:4
v(k,p) = v(k,p) * w + c1 * rand() * (pbest(k,p) - x(k,p)) + c2 * rand() * (gbest(p) - x(k,p));
    % 速度更新
end
end
% 注:  repmat(A,r1,r2):可将矩阵 扩充 为每个单位为A的r1*r2的矩阵

% 边界速度处理
for k=1:N
for p=1:4
v(v(N,p) > v_limit(p,2)) = v_limit(p,2);
v(v(N,p) < v_limit(p,1)) = v_limit(p,1);
end
end
```

```
x = x + v;% 位置更新
%x = x + v*(iter-i)/iter;% 位置更新,模拟退火

% 边界位置处理
for k=1:N
for p=1:4
x(x(N,p) > x_limit(p,2)) = x_limit(p,2);
x(x(N,p) < x_limit(p,1)) = x_limit(p,1);
end
end

record(i) = fg_best;%最大值记录

% 画动态展示图
% zuo_x = 0 : 0.01 : 20;
% plot(zuo_x, f(zuo_x), 'b-', x, f(x), 'ro');
% title('状态位置变化')
% pause(0.1)
i = i + 1;
%    if mod(i,10) == 0 % 显示进度
%        i
%    end
disp(['第',num2str(i),'次最佳适应度: ',num2str(fg_best)]);
end
% figure(3);
% plot(record);
% title('收敛过程');
% zuo_x = 0 : 0.01 : 20;

% figure(4);
% plot(zuo_x, f(zuo_x), 'b-', x, f(x), 'ro');
% title('最终状态图')

disp(['最佳适应度: ',num2str(fg_best)]);
disp(['最佳粒子的位置x: ',num2str(gbest)]);
```

附录 C　第三问粒子群算法

```
%原始数据
data3=Untitled;
%以一个点为原点
%经度
x0=data3(:,1);
%纬度
```

```
y0=data3(:,2);
%海拔
z0=data3(:,3);
%收取时间
tdata=raidi;
cntt=1;



iter = 5000;                  % 最大迭代次数
w = 0.9;                      % 惯性权重
c1 = 0.7;                     % 自我学习因子
c2 = 0.2;                     % 群体学习因子
d = 12;                       % 空间维数
N = 150;                      % 初始种群个数
x_limit = [-100000, 130000;
-100000, 130000;
-100, 10000;
-21.7707758,-11.7707758];          % 设置位置限制
v_limit = [-100, 100;-100, 100;-10, 10;-1, 1];      % 设置速度限制
for cl=1:216
cel=newcel{cl};
t1=[tdata(1,cel(1,1)),tdata(2,cel(1,2)),tdata(6,cel(1,3)),tdata(7,cel(1,4))];
t2=[tdata(1,cel(2,1)),tdata(2,cel(2,2)),tdata(6,cel(2,3)),tdata(7,cel(2,4))];
t3=[tdata(1,cel(3,1)),tdata(2,cel(3,2)),tdata(6,cel(3,3)),tdata(7,cel(3,4))];

% 适应度函数表达式(求这个函数的最大值)
%x=[x,y,z,t]
f= @(x21,x22,x23,x24)
    abs(sqrt((x21(1)-x0(1))^2+(x21(2)-y0(1))^2+(x21(3)-z0(1))^2)-340*(t1(1)-x21(4)))...
+abs(sqrt((x21(1)-x0(2))^2+(x21(2)-y0(2))^2+(x21(3)-z0(2))^2)-340*(t1(2)-x21(4)))...
+abs(sqrt((x21(1)-x0(6))^2+(x21(2)-y0(6))^2+(x21(3)-z0(6))^2)-340*(t1(3)-x21(4)))...
+abs(sqrt((x21(1)-x0(7))^2+(x21(2)-y0(7))^2+(x21(3)-z0(7))^2)-340*(t1(4)-x21(4)))...
...
+abs(sqrt((x22(1)-x0(1))^2+(x22(2)-y0(1))^2+(x22(3)-z0(1))^2)-340*(t2(1)-x22(4)))...
+abs(sqrt((x22(1)-x0(2))^2+(x22(2)-y0(2))^2+(x22(3)-z0(2))^2)-340*(t2(2)-x22(4)))...
+abs(sqrt((x22(1)-x0(6))^2+(x22(2)-y0(6))^2+(x22(3)-z0(6))^2)-340*(t2(3)-x22(4)))...
+abs(sqrt((x22(1)-x0(7))^2+(x22(2)-y0(7))^2+(x22(3)-z0(7))^2)-340*(t2(4)-x22(4)))...
...
+abs(sqrt((x23(1)-x0(1))^2+(x23(2)-y0(1))^2+(x23(3)-z0(1))^2)-340*(t3(1)-x23(4)))...
+abs(sqrt((x23(1)-x0(2))^2+(x23(2)-y0(2))^2+(x23(3)-z0(2))^2)-340*(t3(2)-x23(4)))...
+abs(sqrt((x23(1)-x0(6))^2+(x23(2)-y0(6))^2+(x23(3)-z0(6))^2)-340*(t3(3)-x23(4)))...
+abs(sqrt((x23(1)-x0(7))^2+(x23(2)-y0(7))^2+(x23(3)-z0(7))^2)-340*(t3(4)-x23(4)));



for i=1:4
x21(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1); %初始每个粒子的位置
```

```
x22(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1); %初始每个粒子的位置
x23(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1); %初始每个粒子的位置
end

for i=1:4
v1(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
v2(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
v3(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
end% 初始每个粒子的速度

pbest = [x21,x22,x23];                  % 初始化每个个体的历史最佳位置
gbest = zeros(1, d);          % 初始化种群的历史最佳位置

fp_best = inf(N, 1);        % 初始化每组个体的历史最佳适应度 为 正无穷
fg_best = inf;               % 初始化种群历史最佳适应度 为 +无穷


record = zeros(iter, 1);    % 记录器(用于记录 fg_best 的变化过程)
% figure(2);
i = 1;
cnt=0;
while i <= iter
for k=1:N
fx(k)=f(x21(k,:),x22(k,:),x23(k,:));
end               % 计算个体当前组适应度
for j = 1:N
if fp_best(j) > fx(j)
fp_best(j) = fx(j); % 更新个体历史最佳适应度
pbest(j,:) = [x21(j,:),x22(j,:),x23(j,:)]; % 更新个体历史最佳位置
end
end

if fg_best > min(fp_best)
[fg_best,ind_min] = min(fp_best); % 更新群体历史最佳适应度
gbest = pbest(ind_min,:);          % 更新群体历史最佳位置，其中 ind_max 是最大值所在的下标
end
for k=1:N
for p=1:4
v1(k,p) = v1(k,p) * w + c1 * rand() * (pbest(k,p) - x21(k,p)) + c2 * rand() * (gbest(p) -
    x21(k,p)); % 速度更新
v2(k,p) = v2(k,p) * w + c1 * rand() * (pbest(k,p+4) - x22(k,p)) + c2 * rand() * (gbest(p+4) -
    x22(k,p));
v3(k,p) = v3(k,p) * w + c1 * rand() * (pbest(k,p+8) - x23(k,p)) + c2 * rand() * (gbest(p+8) -
    x23(k,p));
end
end
```

```
% 边界速度处理
for k=1:N
for p=1:4
if v1(k,p) > v_limit(p,2)
v1(k,p)= v_limit(p,2);
end
if v1(k,p) < v_limit(p,1)
v1(k,p)= v_limit(p,1);
end

if v2(k,p) > v_limit(p,2)
v2(k,p)= v_limit(p,2);
end
if v2(k,p) < v_limit(p,1)
v2(k,p)= v_limit(p,1);
end

if v3(k,p) > v_limit(p,2)
v3(k,p)= v_limit(p,2);
end
if v3(k,p) < v_limit(p,1)
v3(k,p)= v_limit(p,1);
end

% v2(v2(N,p) > v_limit(p,2)) = v_limit(p,2);
% v2(v2(N,p) < v_limit(p,1)) = v_limit(p,1);
%
% v3(v3(N,p) > v_limit(p,2)) = v_limit(p,2);
% v3(v3(N,p) < v_limit(p,1)) = v_limit(p,1);
end
end

x21 = x21 + v1;% 位置更新
x22 = x22 + v2;% 位置更新
x23 = x23 + v3;% 位置更新

% 边界位置处理
% 边界位置处理
for k=1:N
for p=1:3
if x21(k,p) > x_limit(p,2)
x21(k,p)= x_limit(p,2);
end
if x21(k,p) < x_limit(p,1)
x21(k,p)=x_limit(p,1);
end
if x23(k,p) > x_limit(p,2)
```

```
x23(k,p)= x_limit(p,2);
end
if x23(k,p) < x_limit(p,1)
x23(k,p)=x_limit(p,1);
end
if x22(k,p) > x_limit(p,2)
x22(k,p)= x_limit(p,2);
end
if x22(k,p) < x_limit(p,1)
x22(k,p)=x_limit(p,1);
end
end
end
for k=1:N
for p=4:4
if x21(k,p) > x_limit(p,2)
x21(k,p)= x_limit(p,2);
end
if x21(k,p) < x_limit(p,1)
x21(k,p)=x_limit(p,1);
end

if x22(k,p) < max(x_limit(p,1),x21(k,p)-5)
x22(k,p)= max(x_limit(p,1),x21(k,p)-5);
end
if x22(k,p) > min(x_limit(p,2),x21(k,p)+5)
x22(k,p)=min(x_limit(p,2),x21(k,p)+5);
end

if x23(k,p) < max(max(x_limit(p,1),x21(k,p)-5),x22(k,p)-5)
x23(k,p)= max(max(x_limit(p,1),x21(k,p)-5),x22(k,p)-5);
end
if x23(k,p) > min(min(x_limit(p,2),x21(k,p)+5),x22(k,p)+5)
x23(k,p)=min(min(x_limit(p,2),x21(k,p)+5),x22(k,p)+5);
end
% x22(x22(N,p) > x_limit(p,2)) = x_limit(p,2);
% x22(x22(N,p) < x_limit(p,1)) = x_limit(p,1);
%
% x23(x23(N,p) > x_limit(p,2)) = x_limit(p,2);
% x23(x23(N,p) < x_limit(p,1)) = x_limit(p,1);
end
end

record(i) = fg_best;%最大值记录

i = i + 1;
%disp(['第',num2str(i),'次最佳适应度: ',num2str(fg_best)]);
```

```
end
ans(cntt,:)=[fg_best,gbest];
disp(['第',num2str(cntt),'次最佳适应度: ',num2str(fg_best)]);
cntt=cntt+1;
end

disp(['最佳适应度: ',num2str(fg_best)]);
disp(['最佳粒子的位置x: ',num2str(gbest)]);
gbest=gbest';
[lastbest,lastmin] = min(ans(:,1));
```

附录 D　第四问粒子群算法结合蒙特卡洛

```
clc;clear;close all;

%原始数据
data3=xlsread("thirddata.xlsx");
%以一个点为原点
%经度
x0=(data3(:,1)-data3(1,1))*97304;
%纬度
y0=(data3(:,2)-data3(1,2))*111263;
%海拔
z0=data3(:,3);
%收取时间
tdata=data3(:,4:7);
da=load("q2time");
tdata=da.raidi;
cntt=1;
c=load("newcell.mat");

numbe=[68,72,77,81,95,99,104,108];

iter = 2000;                  % 最大迭代次数
w = 0.9;                      % 惯性权重
c1 = 0.7;                     % 自我学习因子
c2 = 0.2;                     % 群体学习因子
d = 12;                       % 空间维数
N = 5000;                     % 初始种群个数
x_limit = [-100000, 130000;
-100000, 130000;
0, 100000;
-21.7707758,-11.7707758];          % 设置位置限制
v_limit = [-350, 350;-350, 350;-250, 250;-1, 1]; % 设置速度限制
```

```
r1=rand(1,4)-0.5;
r2=rand(1,4)-0.5;
r3=rand(1,4)-0.5;
% for cl=1:216
for numbee=4:4
cel=c.newcel{numbe(numbee)};
t1=[tdata(1,cel(1,1)),tdata(2,cel(1,2)),tdata(6,cel(1,3)),tdata(7,cel(1,4))];
t2=[tdata(1,cel(2,1)),tdata(2,cel(2,2)),tdata(6,cel(2,3)),tdata(7,cel(2,4))];
t3=[tdata(1,cel(3,1)),tdata(2,cel(3,2)),tdata(6,cel(3,3)),tdata(7,cel(3,4))];

t1=t1+r1;
t2=t2+r2;
t3=t3+r3;


% 适应度函数表达式(求这个函数的最大值)
%x=[x,y,z,t]
f= @(x21,x22,x23)
    abs(sqrt((x21(1)-x0(1))^2+(x21(2)-y0(1))^2+(x21(3)-z0(1))^2)-340*(t1(1)-x21(4)))...
+abs(sqrt((x21(1)-x0(2))^2+(x21(2)-y0(2))^2+(x21(3)-z0(2))^2)-340*(t1(2)-x21(4)))...
+abs(sqrt((x21(1)-x0(6))^2+(x21(2)-y0(6))^2+(x21(3)-z0(6))^2)-340*(t1(3)-x21(4)))...
+abs(sqrt((x21(1)-x0(7))^2+(x21(2)-y0(7))^2+(x21(3)-z0(7))^2)-340*(t1(4)-x21(4)))...
...
+abs(sqrt((x22(1)-x0(1))^2+(x22(2)-y0(1))^2+(x22(3)-z0(1))^2)-340*(t2(1)-x22(4)))...
+abs(sqrt((x22(1)-x0(2))^2+(x22(2)-y0(2))^2+(x22(3)-z0(2))^2)-340*(t2(2)-x22(4)))...
+abs(sqrt((x22(1)-x0(6))^2+(x22(2)-y0(6))^2+(x22(3)-z0(6))^2)-340*(t2(3)-x22(4)))...
+abs(sqrt((x22(1)-x0(7))^2+(x22(2)-y0(7))^2+(x22(3)-z0(7))^2)-340*(t2(4)-x22(4)))...
...
+abs(sqrt((x23(1)-x0(1))^2+(x23(2)-y0(1))^2+(x23(3)-z0(1))^2)-340*(t3(1)-x23(4)))...
+abs(sqrt((x23(1)-x0(2))^2+(x23(2)-y0(2))^2+(x23(3)-z0(2))^2)-340*(t3(2)-x23(4)))...
+abs(sqrt((x23(1)-x0(6))^2+(x23(2)-y0(6))^2+(x23(3)-z0(6))^2)-340*(t3(3)-x23(4)))...
+abs(sqrt((x23(1)-x0(7))^2+(x23(2)-y0(7))^2+(x23(3)-z0(7))^2)-340*(t3(4)-x23(4)));


% for i=1:4
%     x21(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1);
    %初始每个粒子的位置
%     x22(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1);
    %初始每个粒子的位置
%     x23(1:N,i) = x_limit(i,1) + (x_limit(i,2) - x_limit(i,1)) * rand(N,1);
    %初始每个粒子的位置
% end用蒙特卡洛做优化
x21=zeros(N,4);
x22=zeros(N,4);
x23=zeros(N,4);
```

```
for i = 1:N
% 初始化最优解和最优评估值
best_solution1 = zeros(1,4);
best_solution2=zeros(1,4);
best_solution3=zeros(1,4);
best_eval = Inf;
mc_iterations = 10000; %蒙特卡洛
for iter = 1:mc_iterations
solution1=zeros(1,4);
solution2=zeros(1,4);
solution3=zeros(1,4);
for j=1:4
solution1(1,j) = x_limit(j,1) + (x_limit(j,2) - x_limit(j,1)) * rand(1,1); %初始每个粒子的位置
solution2(1,j) = x_limit(j,1) + (x_limit(j,2) - x_limit(j,1)) * rand(1,1); %初始每个粒子的位置
solution3(1,j) = x_limit(j,1) + (x_limit(j,2) - x_limit(j,1)) * rand(1,1); %初始每个粒子的位置
end
% 评估当前解
current_eval = f(solution1,solution2,solution3);
% 如果当前解更优，则更新最优解
if current_eval < best_eval
best_solution1 = solution1;
best_solution2 = solution2;
best_solution3 = solution3;
best_eval = current_eval;
end
end
% 将找到的最优解放入种群中
x21(i, :) = best_solution1;
x22(i, :) = best_solution2;
x23(i, :) = best_solution3;
end

for i=1:4
v1(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
v2(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
v3(1:N,i) = v_limit(i,1) + (v_limit(i,2) - v_limit(i,1)) * rand(N,1);
end% 初始每个粒子的速度

pbest = [x21,x22,x23];                  % 初始化每个个体的历史最佳位置
gbest = zeros(1, d);          % 初始化种群的历史最佳位置

fp_best = inf(N, 1);        % 初始化每组个体的历史最佳适应度 为 正无穷
fg_best = inf;               % 初始化种群历史最佳适应度 为 +无穷


record = zeros(iter, 1);    % 记录器(用于记录 fg_best 的变化过程)
% figure(2);
```

```
i = 1;
cnt=0;
iter=5000;
while i <= iter
for k=1:N
fx(k)=f(x21(k,:),x22(k,:),x23(k,:));
end              % 计算个体当前组适应度
for j = 1:N
if fp_best(j) > fx(j)
fp_best(j) = fx(j); % 更新个体历史最佳适应度
pbest(j,:) = [x21(j,:),x22(j,:),x23(j,:)]; % 更新个体历史最佳位置
end
end

if fg_best > min(fp_best)
[fg_best,ind_min] = min(fp_best); % 更新群体历史最佳适应度
gbest = pbest(ind_min,:);          % 更新群体历史最佳位置，其中 ind_max 是最大值所在的下标
end
for k=1:N
for p=1:4
v1(k,p) = v1(k,p) * w + c1 * rand() * (pbest(k,p) - x21(k,p)) + c2 * rand() * (gbest(p) -
    x21(k,p)); % 速度更新
v2(k,p) = v2(k,p) * w + c1 * rand() * (pbest(k,p+4) - x22(k,p)) + c2 * rand() * (gbest(p+4)
    - x22(k,p));
v3(k,p) = v3(k,p) * w + c1 * rand() * (pbest(k,p+8) - x23(k,p)) + c2 * rand() * (gbest(p+8)
    - x23(k,p));
end
end

% 边界速度处理
for k=1:N
for p=1:4
if v1(k,p) > v_limit(p,2)
v1(k,p)= v_limit(p,2);
end
if v1(k,p) < v_limit(p,1)
v1(k,p)= v_limit(p,1);
end

if v2(k,p) > v_limit(p,2)
v2(k,p)= v_limit(p,2);
end
if v2(k,p) < v_limit(p,1)
v2(k,p)= v_limit(p,1);
end

if v3(k,p) > v_limit(p,2)
```

```
v3(k,p)= v_limit(p,2);
end
if v3(k,p) < v_limit(p,1)
v3(k,p)= v_limit(p,1);
end

% v2(v2(N,p) > v_limit(p,2)) = v_limit(p,2);
% v2(v2(N,p) < v_limit(p,1)) = v_limit(p,1);
%
% v3(v3(N,p) > v_limit(p,2)) = v_limit(p,2);
% v3(v3(N,p) < v_limit(p,1)) = v_limit(p,1);
end
end

x21 = x21 + v1;% 位置更新
x22 = x22 + v2;% 位置更新
x23 = x23 + v3;% 位置更新

% 边界位置处理
% 边界位置处理
for k=1:N
for p=1:3
if x21(k,p) > x_limit(p,2)
x21(k,p)= x_limit(p,2);
end
if x21(k,p) < x_limit(p,1)
x21(k,p)=x_limit(p,1);
end
if x23(k,p) > x_limit(p,2)
x23(k,p)= x_limit(p,2);
end
if x23(k,p) < x_limit(p,1)
x23(k,p)=x_limit(p,1);
end
if x22(k,p) > x_limit(p,2)
x22(k,p)= x_limit(p,2);
end
if x22(k,p) < x_limit(p,1)
x22(k,p)=x_limit(p,1);
end
end
end
for k=1:N
for p=4:4
if x21(k,p) > x_limit(p,2)
x21(k,p)= x_limit(p,2);
end
```

```
if x21(k,p) < x_limit(p,1)
x21(k,p)=x_limit(p,1);
end

if x22(k,p) < max(x_limit(p,1),x21(k,p)-5)
x22(k,p)= max(x_limit(p,1),x21(k,p)-5);
end
if x22(k,p) > min(x_limit(p,2),x21(k,p)+5)
x22(k,p)=min(x_limit(p,2),x21(k,p)+5);
end

if x23(k,p) < max(max(x_limit(p,1),x21(k,p)-5),x22(k,p)-5)
x23(k,p)= max(max(x_limit(p,1),x21(k,p)-5),x22(k,p)-5);
end
if x23(k,p) > min(min(x_limit(p,2),x21(k,p)+5),x22(k,p)+5)
x23(k,p)=min(min(x_limit(p,2),x21(k,p)+5),x22(k,p)+5);
end
% x22(x22(N,p) > x_limit(p,2)) = x_limit(p,2);
% x22(x22(N,p) < x_limit(p,1)) = x_limit(p,1);
%
% x23(x23(N,p) > x_limit(p,2)) = x_limit(p,2);
% x23(x23(N,p) < x_limit(p,1)) = x_limit(p,1);
end
end

record(i) = fg_best;%最大值记录

i = i + 1;
disp(['第',num2str(i),'次最佳适应度: ',num2str(fg_best)]);
end
ans(cntt,:)=[fg_best,gbest];
disp(['第',num2str(cntt),'次最佳适应度: ',num2str(fg_best)]);
cntt=cntt+1;
end

disp(['最佳适应度: ',num2str(fg_best)]);
disp(['最佳粒子的位置x: ',num2str(gbest)]);
gbest=gbest';
[lastbest,lastmin] = min(ans(:,1));
```