火箭残骸音爆定源问题的求解与优化

摘要

在火箭发射任务中，多级火箭及其助推器在完成各自任务后会分离并坠落，过程中产生音爆。为实现残骸的快速回收和准确定位，需要在潜在坠落区域布置多台震动波监测设备。本文提出了一种通过监测设备接收到的音爆信号抵达时间，反推残骸在空中的位置和时间的数学模型。模型考虑了多残骸坠落产生的多组音爆信号，并提出了如何在存在记录误差的情况下实现高精度定位的方法。研究结果对提升火箭残骸回收效率、减少环境影响具有重要意义。

问题一 我们构建了单残骸音爆定源模型。假设出残骸发生音爆的位置和时间，共含四个未知数，则可以利用空间坐标距离公式以及残骸音爆位置和某一台设备之间的距离等于震动波速度×（音爆接收时间–音爆发生时间）列出方程，四台设备即可列出四个方程，从而求解出四个未知数，即 **最少只需四台设备** 即可确定残骸发射音爆的位置和时间。对于题目所给具体算例，将方程求解转化为最小二乘问题，通过牛顿法求解出结果，再用模拟退火算法求解以进一步提高结果的可靠性，最终确定残骸发生音爆时的位置和时间分别为 **经度 110.728°，纬度 27.025°，高程 15215.429m，时间为-45.523s（相对于观测系统时钟 0 时）**

问题二 针对多残骸音爆定源，最核心的问题便是对设备接收到的时间数据进行分类，确定每台设备中的哪一条数据属于同一个残骸。然而若某一组时间数据不属于同一个残骸，那么对于问题一的优化目标函数值便不能充分收敛至0。如果遍历所有的时间数据组合，得到一组求解出的目标函数值，只需选取其最小值便能很好地确定某残骸所对应的时间数据，并套用问题一模型解出其坐标和时间。

问题三 其为问题二的一个具体算例，只需将题中所给数据进行预处理后代入问题二的模型中进行求解即可得到以下结果

残骸	经度(°)	纬度(°)	高程(m)	时间(s)
1	110.500	27.310	12513.956	12.000
2	110.300	27.650	11477.885	14.000
3	110.700	27.650	13468.163	15.000
4	110.500	27.950	11528.802	13.002

问题四 在问题二的原有模型上对音爆接收时间项叠加一个 0.5s 的随机误差项即可完成对模型的修正，再将问题三中的数据代入修正后的模型中用同样的方法求解即可得到修正后的结果，最后将修正前后的结果进行对比分析误差。

关键字：单、多残骸音爆定源　牛顿法　模拟退火算法　音爆接收时间分类　误差修正

一、 问题重述

1.1 问题背景

在火箭发射任务中，多级火箭的各级火箭或助推器完成任务后会分离并坠落地面，产生跨音速音爆。为了实现残骸的快速回收和准确定位，需要在残骸落区布置多台震动波监测设备，通过监测设备记录的音爆信号时间，确定残骸音爆发生时的空中位置和时间，并预测其落地点。多残骸同时坠落的复杂性以及设备记录误差带来的挑战，使得残骸的精准定位成为一个重要且复杂的问题。

1.2 问题提出

问题一 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？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）如表 2 所示：

表 **2**

设备	经度(°)	纬度(°)	高程(m)	音爆抵达时间(s)
A	110.241	27.204	824	100.767
B	110.780	27.456	727	112.220
C	110.712	27.785	742	188.020
D	110.251	27.825	850	258.985
E	110.524	27.617	786	118.443
F	110.467	27.921	678	266.871
G	110.047	27.121	575	163.024

问题二 火箭残骸除了一级残骸，还有两个或者四个助推器。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？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？

问题三 假设各台监测设备布置的坐标和 4 个音爆抵达时间分别如表 3 所示：设备经度(°)纬度(°)高程(m)音爆抵达时间利用问题 2 所建立的数学模型，从上表中选取合适的数据，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）。

表 **3**

设备	经度(°)	纬度(°)	高程(m)	音爆抵达时间(s)			
A	110.241	27.204	824	100.767	164.229	214.850	270.065
B	110.783	27.456	727	92.453	112.220	169.362	196.583
C	110.762	27.785	742	75.560	110.696	156.936	188.020
D	110.251	28.025	850	94.653	141.409	196.517	258.985
E	110.524	27.617	786	78.600	86.216	118.443	126.669
F	110.467	28.081	678	62.274	166.270	175.482	266.871
G	110.047	27.521	575	103.738	163.024	206.789	210.306

问题四 假设设备记录时间存在 0.5 s 的随机误差，请修正问题 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。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 km），并自行根据问题 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：

题目要求我们建立数学模型，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？根据物理公式：距离等于时间乘以速度，假设出残骸发生音爆的位置和时间，共含四个未知数，则残骸发生音爆的位置和设备之间的距离等于震动波速度*(音爆接收时间-音爆发生时间)，利用一台设备的提供的数据即可列出一个方程，求解出四个未知数所需的最小方程数即为最小设备数。对于题目所给具体算例，通过选取合适的数据再将其代入模型中求解并进行验证即可。

2.2 问题二分析:

针对多残骸音爆定源，最核心的问题便是对设备接收到的时间数据进行分类，确定每台设备中的哪一条数据属于同一个残骸。然而若某一组时间数据不属于同一个残骸，那么对于问题一的优化目标函数值便不能充分收敛至0。如果遍历所有的时间数据组合，得到一组求解出的目标函数值，只需选取其最小值便能很好地确定某残骸所对应的时间数据，并套用问题一模型解出其坐标和时间。

2.3 问题三分析:

问题三是问题二的一个具体算例，只需将题中所给数据进行预处理后代入问题二的模型中进行求解即可得到结果。

2.4 问题四分析:

题目要求我们考虑0.5 s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并通过对问题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。只需在问题二的原有模型上对音爆接收时间项叠加一个0.5s的随机误差项即可完成对模型的修正，再将问题三中的数据代入修正后的模型中用同样的方法求解即可得到修正后的结果，最后将修正前后的结果进行对比分析误差。

三、 模型假设

1. 计算两点间距离时可忽略地面曲率
2. 震动波的传播速度为 340 m/s
3. 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

四、 符号说明

表 **4**

符号	意义	单位
d_i	第 i 台设备与音爆的距离	km
x_0	音爆位置横座标	km
y_0	音爆位置纵坐标	km
z_0	音爆位置高坐标	km
x_i	第 i 台设备位置的横坐标	km
y_i	第 i 台设备位置的纵坐标	km
z_i	第 i 台设备位置的高坐标	km
t_0	音爆发生时间	s
t_i	第 i 台设备捕捉到音爆抵达时间	s
v	震动波速度	km/s
T_0	多残骸时间数据矩阵	s
E	随机误差矩阵	s
e	随机误差	s

五、 模型建立与求解

5.1 问题一模型建立与求解

5.1.1 建立坐标系

每台监测设备与音爆位置均由纬度、经度和高程来描述，为了更直观地显示出各台监测设备的几何位置以便选取合适的数据，我们以经度 110°，纬度 27°，高程 0m 为坐标原点，并分别以经度、纬度、高程方向为坐标轴建立三维坐标系。

各设备在三维空间坐标系中所对应的坐标及音爆抵达时间:

表 **5**　坐标表

设备	横坐标x(km)	纵坐标y(km)	高坐标z(km)	音爆抵达时间(s)
A	23.4503	22.6977	0.8240	100.767
B	75.8971	50.7359	0.7270	112.220
C	69.2804	87.3415	0.7420	188.020
D	24.4233	91.7920	0.8500	258.985
E	50.9873	68.6493	0.7860	118.443
F	45.4410	102.4732	0.6780	266.871
G	4.5733	13.4628	0.5750	163.024

下图为各设备位置的三维示意图

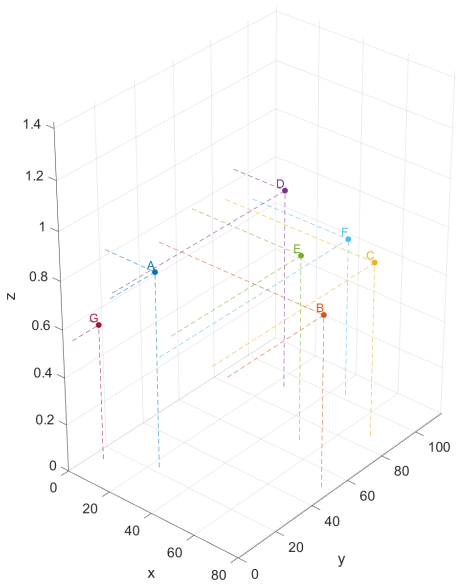

图 **1**

在具体选择数据点时，设备的几何分布应具有多样性和对称性，有良好的几何约束，因此根据图 **1** 和表 **4** 选择 A、B、C、D、F、G 六台设备的数据。

5.1.2 单残骸音爆定源模型建立

将监测设备 A、B、C、D、F、G 六台设备分别编号为 1、2、3、4、5、6 号设备，由空间坐标计算此单残骸音爆位置与第i台设备之间的距离

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \tag{1}$$

而由音爆传播时间得到音爆与设备之间的距离

$$d_i{}' = v(t_i - t_0) \tag{2}$$

注意到对$\forall(x, y, z, t)$有

$$\lim_{(x,y,z,t)\to(x_0,y_0,z_0,t_0)} |d_i - d_i{}'| = 0 \tag{3}$$

令$f_i = |d_i - d_i{}'|$，则原问题转化为以下优化问题

$$\min \sum_{i=1}^{6} f_i^2 \tag{4}$$

5.1.3 单残骸音爆定源模型求解

由 **5.1.2** 知，可令

$$S(x, y, z, t) = \sum_{i=1}^{6} f_i^2 \tag{5}$$

则有优化问题

$$\min S(x, y, z, t) \tag{6}$$

对于优化求解问题通常可以利用牛顿法、模拟退火算法等进行求解。

5.1.3.1 牛顿法

牛顿法又称为牛顿-拉弗森方法，是一种在实数域和复数域上近似求解方程的方法，其利用函数曲线的切线来逐步逼近逼近方程的根。牛顿法是一种迭代求解方法，在实际使用时，要选择合适的初始猜测值以确保算法的收敛性和处理可能的数值问题。

STEP 1 通过分析题意知该无约束优化问题为

$$\min_{\boldsymbol{x}\in R^4} S(x, y, z, t), \boldsymbol{x} = (x_0, y_0, n_0, t_0) \tag{7}$$

其中$\boldsymbol{x}^* = ((x_0)^*, (y_0)^*, (n_0)^*, (t_0)^*)$ 为该目标函数的极小点。

STEP 2 设第k次迭代值为$\boldsymbol{x}^k$，则可在$\boldsymbol{x}^k$附近进行二阶泰勒展开

$$s(\boldsymbol{x}) = s\left(\boldsymbol{x}^k\right) + \left(g_{k(\boldsymbol{x}-\boldsymbol{x}^k)}\right)^T + \frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{x}^k\right)^T H\left(\boldsymbol{x}^k\right)\left(\boldsymbol{x}-\boldsymbol{x}^k\right) \tag{8}$$

其中，$g_k = g\left(\boldsymbol{x}^k\right) = \nabla s\left(\boldsymbol{x}^k\right)$是$s(\boldsymbol{x})$的梯度向量在点$\boldsymbol{x}^k$的值，$H\left(\boldsymbol{x}^k\right)$是$s(\boldsymbol{x})$的海塞矩阵

$$H(\boldsymbol{x}) = \left[\frac{\partial^2 s}{(\partial x_i)(\partial x_j)}\right]_{n*n} \tag{9}$$

在点$\boldsymbol{x}^k$处的值。函数$s(\boldsymbol{x})$有极值的必要条件是在极值点处一阶导数为 0，即梯度向量为 0。特别的当$H(\boldsymbol{x})$是正定矩阵时，函数$s(\boldsymbol{x})$的极值为极小值。

5.1.3.2 模拟退火算法

模拟退火算法是基于蒙特卡洛迭代求解策略的一种随机优化算法，其出发点是基于物理中固体物质的退火过程和一般组合优化问题之间的相似性，从某一较高初温出发，伴随温度参数的不断下降，结合概率突跳特性在求解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。

STEP 1 考虑优化函数f，给定初始温度T_0，终止温度$T_m > 0$，温度下降率α，此时T_0应尽量大，T_m应尽量小，α应尽量接近1^-，使得温度下降速度尽量慢，能有足够多的轮数进行以下步骤。

STEP 2 使当前温度$T = T_0$，采取随机方式在f的定义域中选取初始解x_0，并使最优解$x = x_0$，接下来采取随机方式在x的邻域中选取新解x'，若$f(x') < f(x)$时，则$x = x'$，否则有$e^{-\frac{|f(x')-f(x)|}{T}}$的概率使$x = x'$。

STEP 3 当前温度进行下降，即$T = \alpha T$，若$T < T_m$，则终止计算，否则继续选取新解，进入 **STEP 2** 计算。

设置$T_0 = 1$，$\alpha = 1 - 10^{-5}$，$T_m = 10^{-30}$，经编程求解后可以得到最优解$(70.850, 2.814, 15.215, -45.523)$，作示意图有

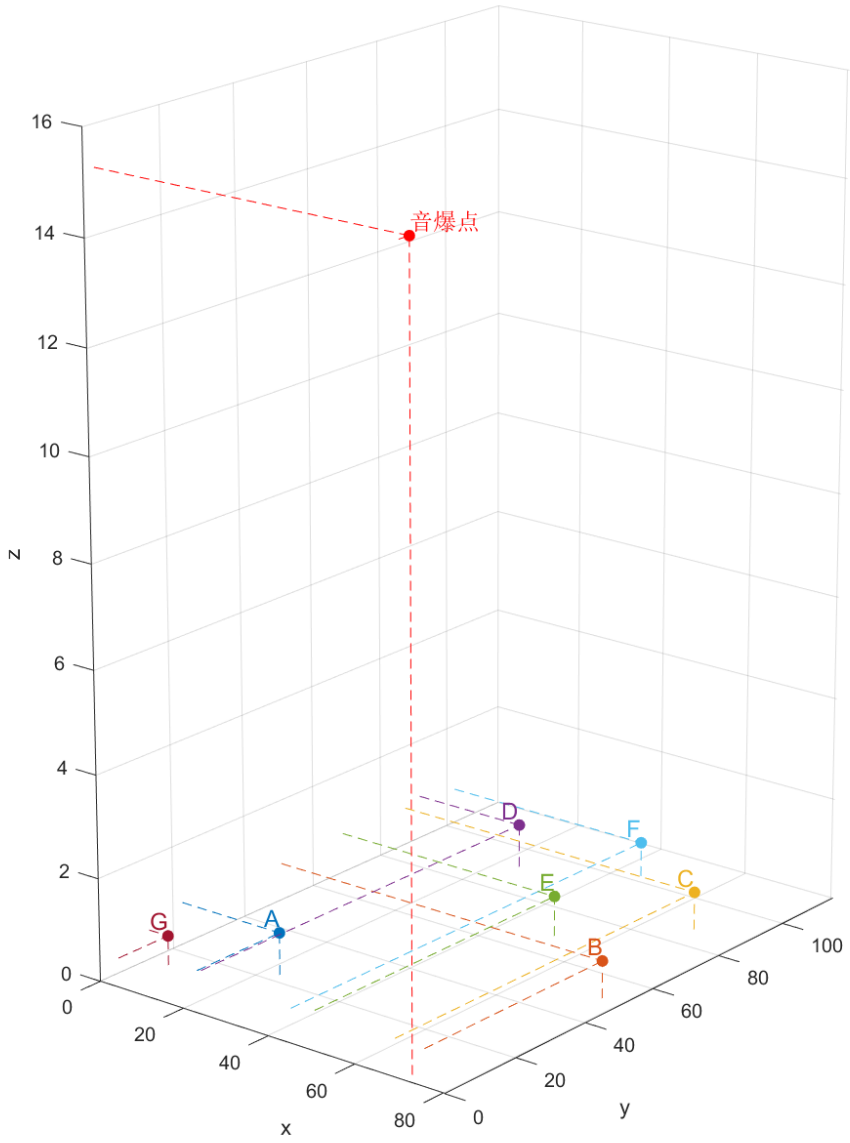

图 **2**

经坐标转化为经纬度坐标后经坐标转化为经纬度坐标后即有：

该残骸音爆位置经度 110.728°，纬度 27.025°，高程 15215.429m，时间为-45.523s（相对于观测系统时钟 0 时）

5.2 问题二模型建立与分析

5.2.1 多残骸音爆定位模型建立

设第j个残骸发生音爆时坐标为(x_j, y_j, z_j)，时间为t_j。考虑有n台设备分别接收到 4 组音爆数据。设第i台设备坐标为(x_i, y_i, z_i)，4 组音爆接收时间分别为t_{i1}，t_{i2}，t_{i3}，t_{i4}，有音爆接收时间矩阵

$$T_0 = \begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ \vdots & \vdots & \vdots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & t_{n4} \end{pmatrix} \tag{10}$$

对每个残骸所产生的音爆而言，会分别被n台设备接收到一次数据，有集合$T_j = \left\{t_{1k_1}, t_{2k_2}, \cdots, t_{nk_n}\right\}$，$k_i \in \{1,2,3,4\}$，$i \in \{1,2,\cdots,n\}$表示第$j$个残骸在每台设备所产生的时间数据。

对T_0中每一行的四个元素而言，对应该设备依次接收到 4 个残骸所产生的音爆，故对于T_0中某一列中元素不会属于同一个T_j，则此模型目的之一便是求出每个T_j所含的元素，且$\bigcap_{j=1}^{4} T_j = \varnothing$，此处假设$t_{11} \in T_1$，$t_{12} \in T_2$，$t_{13} \in T_3$，$t_{14} \in T_4$。

根据 **5.1** 模型，对于优化问题

$$\begin{gathered} f_i = \left| \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} - v\left(t_{ik_i} - t_j\right) \right| \\ \min \sum_{i=1}^{n} f_i^2 \end{gathered} \tag{11}$$

若在T_0中的一组音爆接收时间$\left(t_{1k_1}, t_{2k_2}, \cdots, t_{nk_n}\right)$，$k_1 \in \{1,2,3,4\}$，$i \in \{1,2,\cdots,n\}$不属于同一个$T_j$，则$\varepsilon = \min \sum_{i=1}^{n} f_i^2$将无法充分收敛至 0。

综上所述，本模型可转化为如下时间复杂度为$O(4^n)$的优化问题：

对于第j个残骸，有$T_j = \left\{t_{1j}, t_{2k_2}, t_{2k_3}, \cdots, t_{nk_n}\right\}$，$k_i \in \{1,2,3,4\}$，$i \in \{2,3,\cdots,n\}$，即对于所有可能的$T_j$元素组合有优化问题

$$\min \varepsilon\left(t_{1j}, t_{2k_2}, t_{2k_3}, \cdots, t_{nk_n}\right) = \min \sum_{i=1}^{n} f_i^2 \tag{12}$$

其中

$$\begin{gathered} f_i = \left| \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} - v\left(t_{ik_i} - t_j\right) \right| \\ k_i \in \{j, k_2, k_3, \cdots, k_n\} \end{gathered} \tag{13}$$

5.2.2 多残骸音爆定位模型分析与求解

易知此优化问题的决策变量$\boldsymbol{x}=\left(t_{1j},t_{2k_2},t_{2k_3},\cdots,t_{nk_n}\right)$为在已知范围内的离散量，不妨采取遍历的方式选取$\boldsymbol{x}$并采用牛顿法计算优化问题

$$\varepsilon=\min\sum_{i=1}^{n}f_i^2 \tag{14}$$

得到ε的最小值以及所对应的$\boldsymbol{x}$和上述优化问题的解，随即得到各残骸发生音爆时的位置和时间。

通过以上分析可知，每台设备可提供一组有关 4 个残骸的音爆接收时间，要将音爆接收时间合理分类并找到 4 个残骸发生音爆时的位置和时间，只需在给定一组来自不同设备的音爆接收时间的情况下可求解出一个相应的解，即只需 4 台设备所提供的数据便可求解出 4 个残骸发生音爆时的位置和时间。

5.3 问题三的具体算例求解

5.3.1 建立坐标系

同问题一，先将题中各设备的位置转化为三维坐标，得如下数据表：

表 **6**

设备	横坐标x(km)	纵坐标y(km)	高坐标z(km)
A	23.4503	22.6977	0.8240
B	76.1890	50.7359	0.7270
C	74.1456	87.3415	0.7420
D	24.4233	114.0446	0.8500
E	50.9873	68.6493	0.7860
F	45.4410	120.2753	0.6780
F	4.5733	57.9680	0.5750

七台设备位置的三维示意图为：

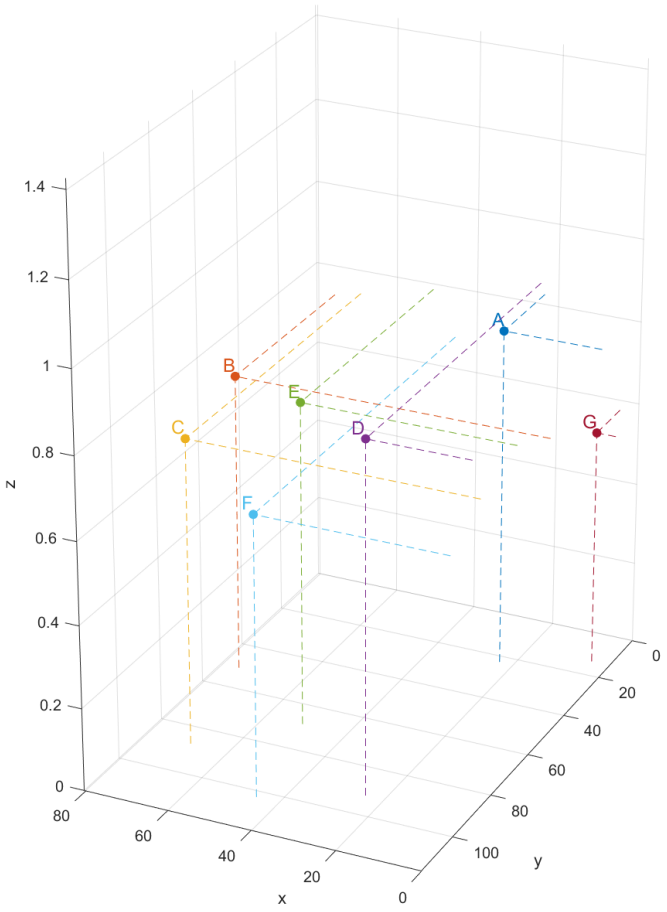

图 **3**

5.3.2 多残骸音爆定位模型具体算例求解

为提高结果的鲁棒性和准确性，使用题中所给的全部七台设备的数据代入问题二中的模型进行求解。

$$T_0 = \begin{pmatrix} 100.767 & 164.229 & 214.850 & 270.065 \\ 92.453 & 112.220 & 169.362 & 196.583 \\ 75.560 & 110.696 & 156.936 & 188.020 \\ 94.653 & 141.409 & 196.517 & 258.985 \\ 78.600 & 86.216 & 118.443 & 126.669 \\ 67.274 & 166.270 & 175.482 & 266.871 \\ 103.738 & 163.024 & 206.789 & 210.306 \end{pmatrix} \tag{15}$$

5.3.3 求解结果及分析

经编程求解可得 4 个残骸相对于各台设备的音爆接收时间如下表所示

表 **7**

残骸	各设备接收时间(s)						
	A	B	C	D	E	F	G
1	110.767	112.220	188.020	258.985	118.443	266.871	163.024
2	164.229	169.362	156.936	141.409	86.216	166.270	103.738
3	214.850	92.453	75.560	196.517	78.600	175.482	210.306
4	270.065	196.583	110.696	94.653	126.669	67.274	206.789

4 个残骸发生音爆时的位置和时间如下表所示：

表 **8**

残骸	横坐标x(km)	纵坐标y(km)	高坐标z(km)	时间(s)
1	48.652	34.491	12.514	12.000
2	29.191	72.321	11.477	14.000
3	68.112	72.321	13.468	15.000
4	48.652	105.700	11.529	13.002

经坐标转换为经纬度坐标后有

表 **9**

残骸	经度(°)	纬度(°)	高程(m)	时间(s)
1	110.500	27.310	12513.956	12.000
2	110.300	27.650	11477.885	14.000
3	110.700	27.650	13468.163	15.000
4	110.500	27.950	11528.802	13.002

该结果中，每两个残骸发生音爆的时间的最大差距约为 3s，满足 4 个残骸产生音爆的时间互相差别不超过 5s 的要求。

依据表 **8** 数据可作残骸 1、2、3、4 的音爆点的位置示意图如下

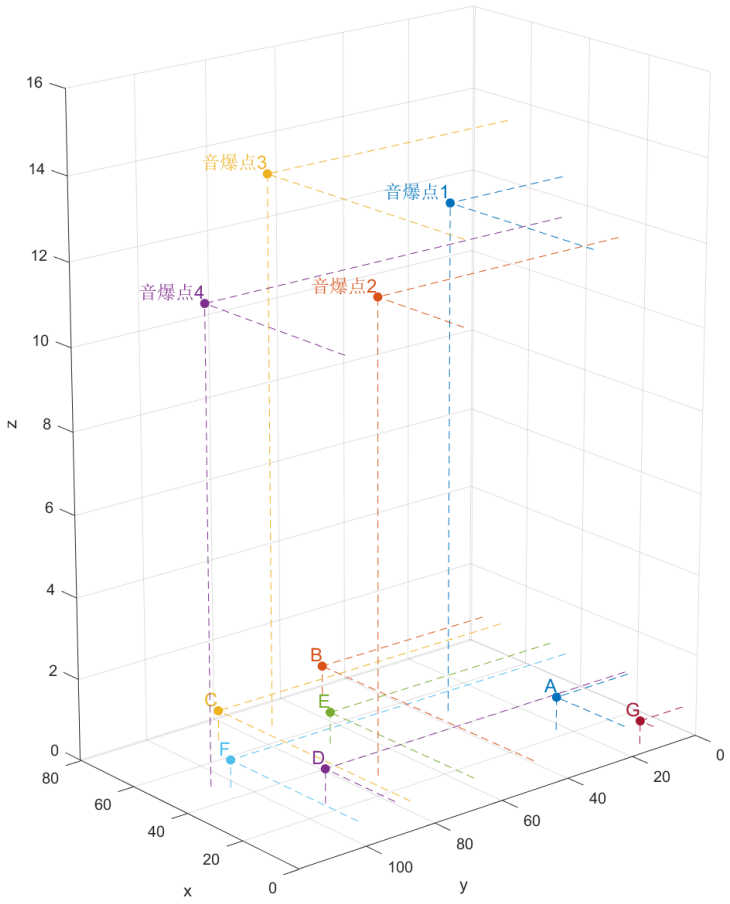

图 **4**

5.4 多残骸音爆定源问题误差修正

5.4.1 问题二模型的修正

在问题二原模型中T_0上追加随机误差$e \in [-0.5, 0.5]$，考虑以下修正模型

$$\min \varepsilon\left(t_{1j}, t_{2k_2}, t_{2k_3}, \cdots, t_{nk_n}\right) = \min \sum_{i=1}^{n} f_i^2 \tag{16}$$

其中

$$f_i = \left| \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} - v\left[\left(t_{ik_i} + e_i\right) - t_j\right] \right| \tag{17}$$

5.4.2 问题三修正模型算例

5.4.2.1 生成 0.5s 的随机误差矩阵

$$E=\begin{pmatrix}0.099 & -0.005 & -0.132 & 0.232\\ -0.228 & -0.480 & 0.461 & -0.233\\ 0.088 & 0.295 & -0.340 & -0.004\\ -0.314 & -0.461 & -0.173 & 0.252\\ -0.275 & 0.2134 & 0.137 & -0.208\\ -0.145 & 0.216 & -0.043 & -0.368\\ -0.279 & 0.435 & -0.2195 & 0.088\end{pmatrix} \tag{18}$$

5.4.2.2 叠加随机误差得到问题 3 中新的音爆时间矩阵

$$T_0=\begin{pmatrix}100.866 & 164.224 & 214.718 & 270.297\\ 92.225 & 111.740 & 169.823 & 196.350\\ 75.648 & 110.991 & 156.596 & 188.016\\ 94.339 & 140.948 & 196.344 & 259.237\\ 78.325 & 86.429 & 118.580 & 126.461\\ 67.129 & 166.486 & 175.439 & 266.503\\ 103.459 & 163.459 & 206.570 & 210.394\end{pmatrix} \tag{19}$$

5.4.2.3 求解结果及分析

对于修正模型算例，以同样的方法通过编程求解得到 4 个残骸相对于各台设备的音爆接收时间如下表所示四个残骸发生音爆时的位置和时间如下表：

表 **10**

残骸	各设备接收时间(s)						
	A	B	C	D	E	F	G
1	110.866	111.74	188.016	259.237	118.580	266.503	163.459
2	164.224	169.823	156.596	140.948	86.429	166.486	103.459
3	214.718	92.225	75.648	196.344	78.325	175.439	210.394
4	270.297	196.350	110.991	94.339	126.461	67.129	206.570

四个残骸发生音爆时的位置和时间如下表：

表 **11**

残骸	横坐标x(km)	纵坐标y(km)	高坐标z(km)	时间(s)
1	48.846	34.335	12.961	12.331
2	29.095	72.379	11.424	13.939
3	68.089	72.293	13.390	15.011
4	48.561	105.641	11.218	13.280

经过简单计算可得，各残骸在修正误差前后的位置距离和时间差为

表 **12**

残骸	1	2	3	4
修正前后位置距离/km	0.512	0.125	0.867	0.329
音爆时间差(修正前-修正后)	0.669	0.061	−0.011	−0.279

可知位置误差小于 1km，即第二问所建立模型实际上是相对准确的。

六、 模型的评价与改进

1. 牛顿法收敛速度快，可快速得到最优解，但易陷入局部最优解，使用牛顿法后再采用模拟退火算法进行全局搜索，可有效提高结果的鲁棒性和准确性。
2. 问题二三中使用遍历求解ε最小值，在此问题中可快速得到较为准确的结果，但不适合用来求解大规模问题，如果所求问题规模较大，可以考虑使用匈牙利算法进行求解。
3. 在多残骸情况下，还可引入分类算法（如聚类分析）对不同残骸的音爆信号进行分类识别，提高多残骸同时坠落情况下的定位精度和效率
4. 结合其他传感器数据（如 GPS、雷达），通过数据融合技术，提升模型对复杂坠落环境的适应能力和定位准确性。

七、 参考文献

[1] 王强, 李伟, 龚建泽, 丁思炜, 和 雷鹏,《基于火箭残骸实时定位信息的落点计算模型》, 计算机测量与控制, 2021.

[2] 廖彦杰, 薛松柏, 龚小维, 和 叶琪玮, 《火箭整流罩残骸定位跟踪系统》. 2021 年.

[3] 王洪, 刘昌忠, 汪学刚, 和 吴宏刚, 《一种多点定位的目标位置精确解算方法》, 航空学报, 卷 32, 期 7, 页 6–7, 2011.

八、 附录

8.1 成员分工

- **林起赋 2200100067** 主要负责问题分析并建模
- **刘洪源 2200100055** 主要负责编程及模型数学化，论文模板编写
- **庄梦佳 2200100156** 主要负责论文写作

8.2 Matlab 实现模拟退火算法

```
function best_solution = SA(fun, init_solution)

    best_solution = init_solution; % 最优解

    init_temperature = 1; % 初始温度
    temperature = init_temperature;
    last_temperature = 10^(-30); % 终止温度
    rate = 0.99999; % 降温速度

    while temperature > last_temperature
        next_solution = zeros(1, length(best_solution));
        for i = 1 : 1 : length(best_solution)
            next_solution(i) = best_solution(i) + (rand-0.5)/100;
        end
        if fun(next_solution) < fun(best_solution)
            best_solution = next_solution;
        else
            if rand < exp(-(fun(next_solution) - fun(best_solution)) / temperature)
                best_solution = next_solution;
            end
        end
        temperature = temperature * rate;
    end
end
```

8.3 Matlab 经纬坐标转换函数

```
function [x_t, y_t, z_t] = coord2xyz(long, lat, height)
    base_long = 110;
    base_lat = 27;
    x_t = (long - base_long) * 97.304;
    y_t = (lat - base_lat) * 111.263;
    z_t = height / 1000;
end

function [long, lat, height] = xyz2coord(x, y, z)
    base_long = 110;
    base_lat = 27;
    long = x / 97.304 + base_long;
    lat = y / 111.263 + base_lat;
    height = z * 1000;
end
```

8.4 Matlab 问题一算法程序

```
clear;clc;format long g;

coords_ori = [
    110.241 27.204 824;
    110.780 27.456 727;
    110.712 27.785 742;
    110.251 27.825 850;
    110.524 27.617 786;
    110.467 27.921 678;
    110.047 27.121 575
];

[x_t, y_t, z_t] = coord2xyz(coords_ori(:, 1), coords_ori(:, 2), coords_ori(:, 3));

times = [100.767 112.220 188.020 258.985 118.443 266.871 163.024];

for i = 1 : 1 : 7
    eval(sprintf('f%d = @(x) sqrt((x(1) - x_t(%d))^2 + (x(2) - y_t(%d))^2 + (x(3) -
z_t(%d))^2) - (times(%d)-x(4))*0.34;', i, i, i, i, i));
end

sum = @(x) f1(x)^2+f2(x)^2+f3(x)^2+f4(x)^2+f6(x)^2+f7(x)^2;

x_n = fminunc(sum, [0 0 0 0]); % 使用牛顿法
x_s = SA(sum, [0 0 0 0]); % 使用模拟退火算法
disp(x_n);
disp(x_s);
```

8.5 Matlab 问题三算法程序

```
clc;clear;format long g;

function [c, ceq] = nonlinConstraints(x)
    c = -x(3);
    ceq = [];
end

coords = [
    110.241 27.204 824;
    110.783 27.456 727;
    110.762 27.785 742;
    110.251 28.025 850;
    110.524 27.617 786;
    110.467 28.081 678;
    110.047 27.521 575
];
[x_t, y_t, z_t] = coord2xyz(coords(:, 1), coords(:, 2), coords(:, 3));
times = [
    100.767 164.229 214.850 270.065;
    92.453  112.220 169.362 196.583;
    75.560  110.696 156.936 188.020;
    94.653  141.409 196.517 258.985;
    78.600  86.216  118.443 126.669;
    67.274  166.270 175.482 266.871;
    103.738 163.024 206.789 210.306
    ];
final = [];
for sonic = 1 : 4
  len = length(times(1, :));
  results = [];
  results_index = [];
  for i = 1 : len
    for j = 1 : len
      for k = 1 : len
        for l = 1 : len
          for m = 1 : len
            for n = 1 : len
              time = [times(1, 1) times(2, i) times(3, j) times(4, k) times(5, l) times(6,
m) times(7, n)];
              for t = 1 : 1 : 7
                  eval(sprintf('f%d = @(x) sqrt((x(1) - x_t(%d))^2 + (x(2) - y_t(%d))^2 +
(x(3) - z_t(%d))^2) - (time(%d)-x(4))*0.34;', t, t, t, t, t));
              end
              sum = @(x) f1(x)^2+f2(x)^2+f3(x)^2+f4(x)^2+f5(x)^2+f6(x)^2+f7(x)^2;
              opt = optimoptions('fmincon', 'Display','none');
            [x, fval] = fmincon(sum, [0 0 0 0], [], [], [], [], [], @nonlinConstraints,
opt);
              results = [results; times(1, 1) times(2, i) times(3, j) times(4, k) times(5,
l) times(6, m) times(7, n) fval x];
              results_index = [results_index; 1 i j k l m n fval];
            end
          end
        end
      end
    end
  end
```

```
  [~, min_index0] = min(results(:, 8));
  [~, min_index1] = min(results_index(:, 8));
  result = results(min_index0, :);
  result_index = results_index(min_index1, :);
  for i = 1 : 7
    eval(sprintf('times_r%d = times(%d, :);', i, i));
    eval(sprintf('times_r%d(result_index(%d))=[];', i, i));
  end
  times = [times_r1;times_r2;times_r3;times_r4;times_r5;times_r6;times_r7];
  final = [final; result];
end
```

8.6 Matlab 问题四算法代码

```
clc;clear;format long g;
function [c, ceq] = nonlinConstraints(x)
    c = -x(3);
    ceq = [];
end
coords = [
    110.241 27.204 824;
    110.783 27.456 727;
    110.762 27.785 742;
    110.251 28.025 850;
    110.524 27.617 786;
    110.467 28.081 678;
    110.047 27.521 575
];
[x_t, y_t, z_t] = coord2xyz(coords(:, 1), coords(:, 2), coords(:, 3));
times = [
    100.767 164.229 214.850 270.065;
    92.453  112.220 169.362 196.583;
    75.560  110.696 156.936 188.020;
    94.653  141.409 196.517 258.985;
    78.600  86.216  118.443 126.669;
    67.274  166.270 175.482 266.871;
    103.738 163.024 206.789 210.306
    ];
rand_t = rand(7, 4) - 0.5; % 随机误差
times = times + rand_t; % 叠加误差
final = [];
for sonic = 1 : 4
  len = length(times(1, :));
  results = [];
  results_index = [];
  for i = 1 : len
    for j = 1 : len
      for k = 1 : len
        for l = 1 : len
          for m = 1 : len
            for n = 1 : len
              time = [times(1, 1) times(2, i) times(3, j) times(4, k) times(5, l) times(6,
m) times(7, n)];
              for t = 1 : 1 : 7
                  eval(sprintf('f%d = @(x) sqrt((x(1) - x_t(%d))^2 + (x(2) - y_t(%d))^2 +
(x(3) - z_t(%d))^2) - (time(%d)-x(4))*0.34;', t, t, t, t, t));
              end
              sum = @(x) f1(x)^2+f2(x)^2+f3(x)^2+f4(x)^2+f5(x)^2+f6(x)^2+f7(x)^2;
              opt = optimoptions('fmincon', 'Display','none');
            [x, fval] = fmincon(sum, [0 0 0 0], [], [], [], [], [], @nonlinConstraints,
opt);
              results = [results; times(1, 1) times(2, i) times(3, j) times(4, k) times(5,
l) times(6, m) times(7, n) fval x];
              results_index = [results_index; 1 i j k l m n fval];
            end
          end
        end
      end
    end
  end
```

```
  [~, min_index0] = min(results(:, 8));
  [~, min_index1] = min(results_index(:, 8));
  result = results(min_index0, :);
  result_index = results_index(min_index1, :);
  for i = 1 : 7
    eval(sprintf('times_r%d = times(%d, :);', i, i));
    eval(sprintf('times_r%d(result_index(%d))=[];', i, i));
  end
  times = [times_r1;times_r2;times_r3;times_r4;times_r5;times_r6;times_r7];
  final = [final; result];
end
```