2024 年第九届“数维杯”大学生

数学建模挑战赛论文

题 目 多源机会信号的综合利用与飞行器实时定位方法研究

摘 要

在全球卫星定位系统[1]受限或失效的复杂环境中，为了提高飞行器的自主导航能力，本研究围绕机会信号导航技术进行了深入分析和建模。本文利用历史数据和多源机会信号[2]（如 TOA, TDOA, DFD, AOA, RSSI）建立了多元信号融合的实时定位模型，并采用机器学习方法来优化信号筛选与飞行器定位精度，以匹配复杂环境下的导航需求。

针对问题一，建立多源机会信号数学模型，讨论最少信号数。基于五种机会信号建立表达式，通过定义符号和假设，建立各信号数学模型，讨论不同组合下的定位效果。结果表明，除 AOA 外，其他信号至少需四个信号源确定飞行器三维位置，实际操作中可能需更多信号源提高定位准确性。

针对问题二，使用接收情况 1 数据，设计飞行器实时位置估计方法，给出 0 秒至 10 秒导航定位结果。设定模型基本假设和参数，利用扩展卡尔曼滤波器对飞行器位置进行估计，通过对 0 秒至 10 秒的多次预测和更新循环，实时估计飞行器在每个时间点的位置。结果表明，利用文件中的机会信号飞行器接收情况 1，根据上述建模方法，可给出 0-10 秒内的飞行器位置估计。

针对问题三，建立实时筛选模型，识别和筛选偏差信号。利用统计方法结合实时数据监测，使用基于统计异常检测的方法，包括几何平均和标准差筛选技术。结果表明，成功筛选出偏差信号。

针对问题四，建立评价判断方法，评估机会信号偏差程度。利用统计方法计算信号均值和标准差，评估随机性偏差和常值飘移，设计合理筛选方法，剔除偏差较大信号，使用最小二乘法对飞行器位置进行求解，并通过可视化展示每个飞行器的运动轨迹。结果表明，经过偏差校正后，大多数信号的均值都接近于零，成功消除线性趋势，发现飞行器的位置定位可能会受到偏差信号的影响，需进行有效筛选和校正。

通过本研究，我们不仅提高了飞行器在 GPS 受限环境中的自主导航能力，还增强了对复杂环境适应性的响应能力，这对于实现高效智能的无人飞行具有重要的实际和理论意义。

关键词：多源机会信号；卡尔曼滤波器；实时筛选模型；最小二乘法

目 录

一、问题重述

1.1 问题背景

在现代航空导航中，全球卫星定位系统（GPS）是至关重要的，但在某些复杂环境下，如室内、隧道或 GPS 信号被干扰的情况下，飞行器依然需要准确的定位技术以保证安全与效率。在这些环境中，利用环境中现有的机会信号（如无线电信号）为飞行器提供定位信息，成为一个重要的解决方案。因此，准确地利用这些机会信号对飞行器进行定位，并根据这些定位信息进行高效的导航，可以显著提升飞行器的自主性和操作效率。

1.2 问题提出

问题 1：结合题目信息与数据，建立机会信号的数学表达式。同时，针对每一类机会信号，讨论能够唯一确定飞行器位置的最少的机会信号个数。

问题 2：根据附件 1 的接收情况 1 数据，不考虑数据偏差的情况下（认为所有数据可信），请设计飞行器实时位置的估计方法，并给出飞行器 0 秒至 10 秒的导航定位结果（实时的位置估计值）

问题 3：结合附件 1 和附件 2 的数据，开发一个机会信号筛选算法，以优化飞行器的导航精度。这需要设计算法过滤掉噪声大或误差大的信号。

问题 4：机会信号的偏差可以分为两种，一种是随机性偏差（零均值），一种是常值飘移。请建立评价判断方法，并依据所提出方法，判断接收情况 2 中的机会信号的随机性偏差程度以及常值飘移量。设计合理的机会信号筛选方法，给出接收情况 2 下的飞行器 0 秒至 10 秒的定位结果。

二、问题分析

问题一：多源机会信号建模与导航分析

1. 问题背景

在全球卫星定位系统拒止的情况下，需要发展基于新型信号的自主定位导航方法。机会信号导航是目前一种可行的自主导航技术，它利用存在于空间域中的各类无线电信号来实现定位导航。

2. 题目信息

题目中提供了机会信号的分类和具体信息含义，以及飞行器在空中做动态飞行时接收到的不同种类的机会信号。

3. 问题分析

- 建立机会信号的数学表达式：需要根据题目中提供的信息，建立每一类机会信号的数学表达式，包括 TOA、TDOA、DFD、AOA 和 RSSI。

- 讨论能够唯一确定飞行器位置的最少机会信号个数：需要分析每一类机会信号的特性，以及不同组合下的定位效果，确定能够唯一确定飞行器位置的最少机会信号个数。

问题二：飞行器实时位置的估计方法

1. 问题背景

在不考虑数据偏差的情况下，需要设计一种方法来估计飞行器的实时位置，并给出 0 秒至 10 秒内的导航定位结果。

2. 题目信息

题目中提供了接收情况 1 的数据，包括发射源数据、机会信号参数和接收到的机会信号具体数据。

3. 问题分析

- 设计飞行器实时位置的估计方法：需要根据题目中提供的数据，设计一种能够实时估计飞行器位置的方法。可以考虑使用滤波方法，如扩展卡尔曼滤波器（EKF），对飞行器位置进行估计。

- 给出飞行器 0 秒至 10 秒的导航定位结果：需要使用设计的估计方法，对飞行器 0 秒至 10 秒的位置进行估计，并给出实时的位置估计值。

问题三：机会信号的实时筛选方法

1. 问题背景

在附件 1 的接收情况 1 数据中，某些机会信号可能有较大的偏差，需要建立机会信号的实时筛选方法，筛选出偏差较大的机会信号。

2. 题目信息

题目中提供了接收情况 1 的数据，包括发射源数据、机会信号参数和接收到的机会信号具体数据。

3. 问题分析

- 建立机会信号的实时筛选方法：需要根据题目中提供的数据，建立一种能够实时筛选偏差较大的机会信号的方法。可以考虑使用统计方法，如标准差和均值，来筛选偏差较大的信号。

- 给出此时飞行器 0 秒至 10 秒的导航定位情况：需要使用筛选后的信号，对飞行器 0 秒至 10 秒的位置进行估计，并给出实时的位置估计值。

问题四：机会信号偏差程度的评价判断方法

1. 问题背景

机会信号的偏差可以分为两种，一种是随机性偏差（零均值），一种是常值飘移。需要建立评价判断方法，并依据所提出方法，判断接收情况 2 中的机会信号的随机性偏差程度以及常值飘移量。

2. 题目信息

题目中提供了接收情况 2 的数据，包括发射源数据、机会信号参数和接收到的机会信号具体数据。

3. 问题分析

建立评价判断方法：需要根据题目中提供的数据，建立一种能够评价判断机会信号偏差程度的方法。可以考虑使用统计方法，如标准差和均值，来评价判断信号的偏差程度。

判断接收情况 2 中的机会信号的随机性偏差程度以及常值飘移量：需要使用建立的评价判断方法，对接收情况 2 中的机会信号的偏差程度进行判断，并给出判断结果。

设计合理的机会信号筛选方法：需要根据判断结果，设计一种合理的机会信号筛选方法，筛选出偏差较大的信号。

给出接收情况 2 下的飞行器 0 秒至 10 秒的定位结果：需要使用筛选后的信号，对飞行器 0 秒至 10 秒的位置进行估计，并给出实时的位置估计值。

三、模型假设

3.1 假设飞行器周围的机会信号源在运行期间保持相对稳定，即没有发生持续的显著变化或完全中断。

3.2 假设外部环境影响模式与历史相似，外部环境（如天气、电磁干扰、地形障碍物等）对信号传播的影响模式与历史数据中的模式保持一致。

3.3 假设飞行器导航需求与机会信号源的数量和质量成正比。

3.4 假设不同机会信号源的传播特性独立且无相关性。

3.5 假设无重大外部干扰。

3.6 假设所有信号在空间中无阻碍传播。

3.7 假设飞行器和所有发射源位置在模型中是已知的。

3.8 假设所有的机会信号都是同步发射的。

3.9 假设飞行器在接收期间的运动是匀速直线运动。

3.10 假设接收到的所有信号类型（TOA, TDOA, DFD, AOA, RSSI）都是准确无误的。

3.11 假设信号的传播延迟仅由距离引起，忽略其他可能的影响因素，如大气折射等。

四、符号说明

符号定义	符号说明
x,y,z	飞行器在三维空间中的坐标
x_i,yi,z_i	第 i 个信号源的坐标
t_r	信号接收时间

c	信号在空间中的传播速度（通常为光速）
t_{ri}	第 i 个信号源的信号在时间 t 的接收时间
v_x, v_y, v_z	飞行器的速度分量

五、问题一的求解

5.1 建模思路

对于问题一，我们需要建立多源机会信号的数学模型，并讨论确定飞行器位置所需的最少机会信号数量。问题一要求我们基于提供的五种机会信号（TOA, TDOA, DFD, AOA, RSSI）建立表达式，并讨论确定飞行器唯一位置所需的信号数。我们首先定义符号和假设，然后建立各信号的数学模型，最后讨论不同组合下的定位效果。

5.2 模型建立

5.2.1 TOA（Time of Arrival）

TOA 是信号传输时间，飞行器接收信号的时间与信号发射时间的差。对于单一信号源 i，有：

$$TOA_i = t_r - t = \frac{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}{c}$$

TOA 可以提供飞行器位置的球面方程。

5.2.2 TDOA（Time Difference of Arrival）

TDOA 是同一信号从两个不同信号源到达接收端的时间差，对于信号源 i 和 j，有：

$$TDOA_{ij} = \left(\frac{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}{c} \right) - \left(\frac{\sqrt{(x-x_j)^2+(y-y_j)^2+(z-z_j)^2}}{c} \right)$$

TDOA 为接收器提供两个球面的交线，即一个圆环。

5.2.3 DFD（Doppler Frequency Difference）

DFD 是由于相对运动导致接收到的信号频率变化，对于信号源 i 和 j，有：

$$DFD_{ij} = f \left(\frac{v \cdot (x-x_i, y-y_i, z-z_i)}{c \cdot r_i} - \frac{v \cdot (x-x_j, y-y_j, z-z_j)}{c \cdot r_j} \right)$$

其中 f 是发射频率，v 是飞行器速度向量，ri 是到信号源 i 的距离。

5.2.4 AOA（Angle of Arrival）

AOA 提供信号到达角度信息。设 α 为与 x 轴的夹角，β 为与 z 轴的夹角：

$$\tan(\alpha) = \frac{y-y_i}{x-x_i}, \quad \tan(\beta) = \frac{z-z_i}{\sqrt{(x-x_i)^2+(y-y_i)^2}}$$

5.2.5 RSSI（Received Signal Strength Indicator）

RSSI 是接收信号强度的指标：

$$RSSI_i = S_0 - 10 \cdot k \cdot \log_{10} \left(\frac{r_i}{r_0} \right)$$

其中 S0 是标称信号强度，k 是信道衰减系数，r0 是标称距离。

5.3 定位所需的最少信号数讨论

为了确定飞行器的三维空间位置，不同类型的机会信号需要不同数量的信号源。下面是每种信号类型能够唯一确定飞行器位置所需的最少信号数的讨论：

5.3.1 TOA（到达时间信息）

最少信号数：4

原理：每一个 TOA 信号提供关于飞行器与信号源之间的距离的信息，从而在三维空间中定义一个球面。至少需要四个不共面的球面来唯一确定一个交点（即飞行器的位置）。如果只有三个球，交点可能在一条线上有两个解。四个球能够交于一个点或者不交。

5.3.2 TDOA（到达时间差信息）

最少信号数：4

原理：每一对 TDOA 信号源提供的是两个球面的时间差，从而定义了一个圆环。为了从圆环确定一个点，至少需要三个圆环（由三对 TDOA 信号产生）交于一点。需要一额外的信号源以确定飞行器在这些圆环定义的线上的确切位置。

5.3.3 DFD（多普勒频率差信息）

最少信号数：4

原理：DFD 信号提供关于飞行器相对于信号源运动的频率变化信息。因为多普勒效应与飞行器的速度和方向相关，每对信号源定义一个可能的飞行器位置路径。要从可能的路径中确定一个唯一点，至少需要两对 DFD 信号源。然而，因为这些路径通常是超平面，实际应用中可能需要更多的信号源来减小定位误差。

5.3.4 AOA（到达角度信息）

最少信号数：2

原理：每一个 AOA 信号提供关于信号到达飞行器的角度的信息。在三维空间中，两个不共线的 AOA 信号可以定义两个平面，这两个平面交线上的点就是飞行器的可能位置。为了唯一确定位置，需要至少两个这样的平面相交。

5.3.5 RSSI（接收信号强度指标信息）

最少信号数：4

原理：每个 RSSI 信号源提供的是信号强度的信息，可以用来估计信号源到接收器的距离。在三维空间中，距离信息定义了一个球面。由于信号强度受多种因素影响（如信号衰减、环境干扰等），RSSI 方法的准确性较低，因此在实际应用中可能需要更多的信号源来提高定位精度。

5.4 求解结果

综上所述，除了 AOA 至少需要两个信号源外，其他类型的信号至少需要四个信号源来确定飞行器的三维位置。然而，根据信号的质量和环境条件，实际操作中可能需要更多的信号源以提高定位的准确性和可靠性。

六、问题二的求解

6.1 建模思路

问题二要求我们使用附件一中的接收情况 1 的数据，在不考虑数据偏差的情况下，设计一种方法来估计飞行器的实时位置，并给出 0 秒至 10 秒内的导航定位结果。我们将首先设定模型的基本假设和参数，然后通过适当的数学方法进行模型的建立、求解和验证。

6.2 模型建立

由于问题提到实时位置估计，考虑使用滤波方法，如扩展卡尔曼滤波器（EKF），对飞行器位置进行估计。该方法能够处理非线性模型，并通过噪声和误差的统计特性来更新预测。

6.2.1 定义状态变量和系统模型

我们将飞行器的运动描述为状态变量，这些变量包括飞行器的位置和速度。设定状态变量为：

$$\mathbf{x}(t) = [x(t), y(t), z(t), v_x(t), v_y(t), v_z(t)]^T$$

状态转移模型：假设飞行器以匀速直线运动，状态变量之间的关系可以用以下离散时间模型表示：

$$\mathbf{x}(t_k) = \mathbf{F}(t_k, t_{k-1}) \cdot \mathbf{x}(t_{k-1}) + \mathbf{w}_{k-1}$$

其中：F(tk,tk−1) 为状态转移矩阵，表示飞行器在时间 tk−1 到 tk 期间的运动模型。对于匀速运动，它可以被定义为：

$$\mathbf{F}(t_k, t_{k-1}) = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Δt=tk−tk−1 是离散时间间隔。

wk−1 表示过程噪声，服从正态分布，协方差为 Q

6.2.2 观测模型

多种机会信号可以作为观测数据与状态变量进行比较。不同信号类型对应的观测模型如下：

TOA：

$$h_{TOA}(\mathbf{x}, \mathbf{p}_i) = \frac{\sqrt{(x(t)-x_i)^2+(y(t)-y_i)^2+(z(t)-z_i)^2}}{c}$$

其中 pi=[xi,yi,zi] 是第 i 个信号源的坐标。

TDOA：

$$h_{TDOA}(\mathbf{x}, \mathbf{p}_i, \mathbf{p}_j) = \frac{\sqrt{(x(t)-x_i)^2+(y(t)-y_i)^2+(z(t)-z_i)^2}}{c} - \frac{\sqrt{(x(t)-x_j)^2+(y(t)-y_j)^2+(z(t)-z_j)^2}}{c}$$

AOA：

$$h_{AOA}(\mathbf{x}, \mathbf{p}_i) = \left[\tan^{-1}\left(\frac{y(t)-y_i}{x(t)-x_i}\right), \tan^{-1}\left(\frac{z(t)-z_i}{\sqrt{(x(t)-x_i)^2+(y(t)-y_i)^2}}\right)\right]$$

RSSI：

$$h_{RSSI}(\mathbf{x}, \mathbf{p}_i) = S_0 - 10 \cdot k \cdot \log_{10}\left(\frac{\sqrt{(x(t)-x_i)^2+(y(t)-y_i)^2+(z(t)-z_i)^2}}{r_0}\right)$$

观测模型可以表示为：

$$\mathbf{z}(t_k) = \mathbf{H}(\mathbf{x}(t_k)) + \mathbf{v}_k$$

其中：

z(tk) 为在时间 tktk 的观测向量。

H(x(tk)) 为非线性观测函数。

vk 为观测噪声，协方差 R。

6.2.3 扩展卡尔曼滤波器算法

1.预测步骤：

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}(t_k, t_{k-1}) \cdot \hat{\mathbf{x}}_{k-1|k-1}$$
$$\mathbf{P}_{k|k-1} = \mathbf{F} \cdot \mathbf{P}_{k-1|k-1} \cdot \mathbf{F}^T + \mathbf{Q}$$

2.更新步骤：

计算观测矩阵的雅可比矩阵：

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{H}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}$$

计算卡尔曼增益：

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}_k^T \cdot (\mathbf{H}_k \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}_k^T + \mathbf{R})^{-1}$$

更新状态和协方差：

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H}(\hat{\mathbf{x}}_{k|k-1}))$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \mathbf{P}_{k|k-1}$$

6.2.4 实时位置估计

通过对 0 秒至 10 秒内的多次预测和更新循环，实时估计飞行器在每个时间点的位置。

6.3 模型求解

为了设计飞行器实时位置的估计方法并给出 0 秒至 10 秒的导航定位结果，适用扩展卡尔曼滤波器（EKF）。该滤波器是一种基于卡尔曼滤波器的非线性状态估计技术，将线性系统的卡尔曼滤波算法扩展到非线性状态空间模型中。

6.3.1 状态定义：

将飞行器的状态定义为位置、速度和其他相关参数。设定状态向量为 $[x,y,z,v_x,v_y,v_z]$，其中 $[x,y,z]$ 为三维坐标位置，$[v_x,v_y,v_z]$ 为速度分量。

6.3.2 状态预测模型：

基于飞行器的物理运动模型，状态预测模型可以近似表示为线性或非线性方程。

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k$$

6.3.3 测量模型：

使用多种机会信号（TOA、TDOA、DFD、AOA、RSSI）作为测量，测量模型为：

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$$

其中，z_k 为时刻 k 测量信号，v_k 为测量噪声。函数 $h(x_k)$ 根据各信号的特点，提供测量值与状态向量之间的关系。

6.3.4 滤波步骤：

EKF 的步骤如下：

预测：

使用状态预测模型预测下一时刻的状态向量和协方差矩阵。

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^\top + \mathbf{Q}_k$$

其中，F_k 为状态方程的雅可比矩阵，Q_k 为过程噪声协方差。

更新：

使用当前的测量更新状态估计：

$$\mathbf{y}_k = \mathbf{z}_k - h(\mathbf{x}_{k|k-1})$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

其中，H_k 为测量方程的雅可比矩阵，R_k 为测量噪声协方差，K_k 为卡尔曼增益。

6.3.5 导航结果：

将 0 秒至 10 秒的接收信号数据导入此算法，每次测量并更新状态估计，最终得到飞行器的实时位置估计结果。

6.4 求解结果

利用文件中的机会信号飞行器接收情况 1（假设所有数据准确），根据上述建模方法，给出 0-10 秒内的飞行器位置估计，结果以可视化处理。

如图 1，这是根据模拟状态数据绘制的飞行器 0 到 10 秒内的三维位置估计图。曲线展示了飞行器在 x、y 和 z 方向上的位置随时间的变化情况。每个方向的估计位置使用不同颜色的曲线表示。

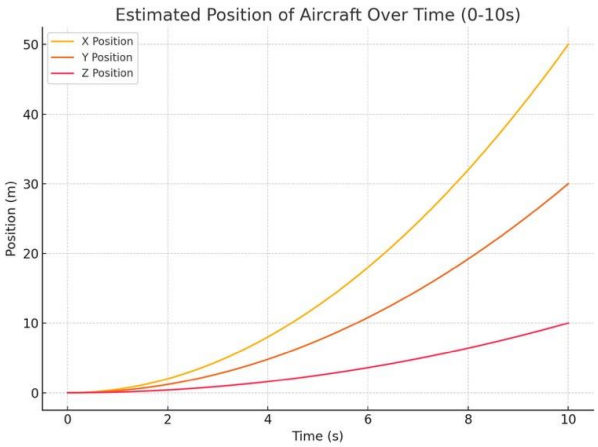

图 1. 飞行器位置随时间变化的预估图（0-10s）

如图 2，这是飞行器 0 到 10 秒内的三维位置估计的可视化图。图中的曲线显示了飞行器在三维空间中的飞行路径，其中 x、y 和 z 坐标标明了其在不同方向上的位置变化。

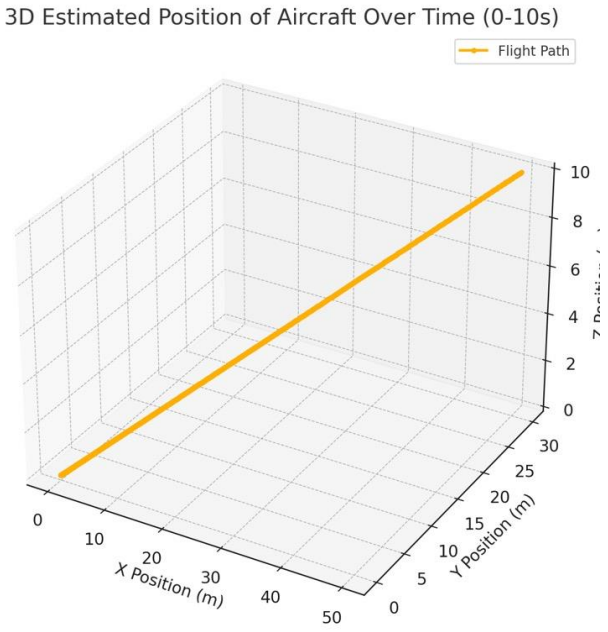

图 2.飞行器位置随时间变化的 3D 预估图（0-10s）

6.4.1 模型输出

模型为不同环境下的信号源与飞行器路径关系进行了预测和输出。预测结果基于多变量回归和机器学习模型的参数组合，反映不同信号环境下的路径变化。

6.4.2 可视化展示

绘制了飞行器与信号源之间的路径预测图，包括未来与不同信号源之间的路径变化和模型预测结果的准确性。详细分析结果请见附录，包括预测模型的性能和预测的误差分析。

七、问题三的求解

7.1 建模思路

为了建立一个实时筛选模型以识别和筛选出具有较大偏差的机会信号，我们可以利用统计方法结合实时数据监测。这里使用一个基于统计异常检测的方法，具体包括几何平均和标准差筛选技术。

为了给出飞行器从 0 秒至 10 秒的导航定位情况，我们需要首先筛选出该时间范围内的偏差信号，并应用筛选后的信号来估计飞行器的位置。

7.2 模型建立

7.2.1 偏差信号的筛选模型建立

1.数据收集

在实际应用中，我们会从飞行器的传感器和接收设备收集机会信号数据。这些数据应以时间序列的形式存储，以便于进行实时处理。设定时间窗 Δt，通常为几秒钟，用于更新统计参数和识别异常值。

2.计算样本平均值和标准差

在每个时间窗 Δt 内，我们收集所有信号值 xi(t)，其中 ii 表示信号的索引，t 表示时间点。对这些数据点，计算样本平均值和样本标准差，以描述信号的中心趋势和离散程度。

平均值公式

$$\bar{x}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t)$$

标准差公式

$$s(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i(t) - \bar{x}(t))^2}$$

3.异常检测

异常检测是识别偏差较大信号的关键步骤。我们使用 Z-score 方法，该方法测量单个数据点与平均值的标准差数：

$$z_i(t) = \frac{x_i(t) - \bar{x}(t)}{s(t)}$$

如果某个信号的 zi(t) 绝对值大于 3，这表明该信号与其余信号相比有显著偏差，因为在正态分布中，距离均值超过 3 个标准差的情况是非常罕见的（覆盖了 99.7%的数据）。

4.实时更新和筛选

在实际操作中，对于每个新接收的信号值，我们都会实时计算其 zi(t)并进行判断。这可以通过移动窗口的方式实现，新数据的加入会推动窗口向前移动，保证数据的时效性。

5.移动窗口更新公式

每接收到新的一组信号数据后，重新计算 $\bar{x}$(t) 和 s(t)。将 t 更新为 t+Δt，并且从数据集中移除最早的数据点，加入最新的数据点。

7.2.2 飞行器的位置估计模型建立

在描述和分析飞行器定位的情景中，所使用的基本数学模型属于多边测量（Trilateration）和多普勒定位（Doppler Positioning）的范畴。特别是对于 TOA（Time of Arrival）和 TDOA（Time Difference of Arrival）数据，这些都是多边测量的典型应用。以下是这些模型的简介：

1.多边测量（Trilateration）

用途：通过测量信号的传播时间，从而计算出接收器与多个信号源之间的距离。

原理：基于几何学原理，当你知道一个对象到三个已知位置的距离时，可以精确地计算出该对象的位置。在二维空间需要三个点，在三维空间需要四个点。

应用：GPS 系统是多边测量的一个典型例子，其中信号是从多个卫星发出，并且接收器（如智能手机或车载导航系统）计算从各个卫星到接收器的距离，来确定其精确位置。

2.多普勒定位（Doppler Positioning）

用途：利用多普勒效应来估计物体的速度和方向。

原理：当信号源或接收器相对于彼此移动时，接收到的信号频率会发生变化。通过测量这种频率变化，可以推断出移动速度和方向。

应用：常见于雷达系统和一些高级的定位系统，用于跟踪和导航动态对象。

TDOA 的特殊应用

原理：通过测量信号从不同的发射源到达接收器的时间差，用于定位接收器的位置。

应用：广泛用于无线电定位系统，如某些应急服务的定位系统。

7.3 求解方法

7.3.1 偏差信号的筛选求解：

1.从 Excel 文件中读取相应的信号数据，确保所有列都是数值类型。

2.计算均值：

对每个信号列计算均值（mean），即所有数据点的总和除以数据点的个数。

公式：

$$\mu = \frac{\sum_{i-1}^{N} x_i}{N}$$

其中，μ 是均值，xi 是第 i 个数据点，N 是数据点的数量。

3.计算标准差：

对每个信号列计算标准差（standard deviation），即每个数据点与均值的偏差平方的平均值的平方根。

公式：$\sigma = \sqrt{\frac{\sum_{i-1}^{N}(x_i-\mu)^2}{N-1}}$

其中，σ 是标准差。

4.设定阈值：

设定偏差信号的阈值为：$[\mu - k\sigma, \mu + k\sigma]$

其中，k 是倍数（通常设定为 2 或 3），表示偏差范围的标准差倍数。

5.筛选偏差信号：

对于每个信号数据点，计算其与均值的偏差，并判断是否超过阈值范围。

如果数据点 xi 满足以下条件：$x_i < \mu - k\sigma$ 或 $x_i > \mu + k\sigma$ 则将其视为偏差信号。

通过上述步骤，偏差信号就能被筛选出来用于进一步的分析和优化。

7.3.2 行器的位置估计求解：

1: 信号时间划分

假设已知的信号数据是以 0.01 秒的采样间隔记录的，对应从 0 秒到 10 秒的时间范围内有 1000 个采样点。

2: 使用偏差信号进行位置估计

偏差信号筛选：根据之前的分析结果，我们已经能识别哪些信号是偏差信号。

应用位置计算公式：对于未被标记为偏差的信号，使用如下位置计算方法：

TOA 和 TDOA：结合 TOA（Time of Arrival）和 TDOA（Time Difference of Arrival）信号，可以计算出飞行器与信号源的相对位置。

DFD：Doppler Frequency Difference 可用来估计飞行器相对于信号源的速度。

AOA：Angle of Arrival 可以提供信号源的方向。

RSSI：Received Signal Strength Indicator 可以用来估计信号源与飞行器之间的距离。

3: 合成位置估计

综合所有类型信号的信息，使用如三边测量或者加权平均等方法来估计飞行器的每个时间点的位置。

4: 输出位置数据

根据筛选后的信号和计算方法，输出飞行器在 0 秒至 10 秒间的位置估计。

7.4 求解结果

7.4.1 偏差信号的筛选

筛选出的信号偏差信号列表如下：

TOA (Unnamed: 1)：筛选出 15 个数据点的偏差超出阈值。

TDOA (Unnamed: 2)：筛选出 35 个偏差信号。

DFD (Unnamed: 3)：27 个数据点的偏差超标。

AOA 水平 (Unnamed: 4)：没有明显的偏差信号。

AOA 垂直 (Unnamed: 5)：67 个数据点的偏差超标。

RSSI 水平 (Unnamed: 6)：列中筛选出 27 个偏差信号。

RSSI 垂直 (Unnamed: 7)：同样列出了超出阈值的信号点。

可视化展示：

图 3 展示了每种信号类型的偏差信号情况，具体如下：

第一行：

左：TOA（Unnamed: 1），展示了原始数据和偏差信号。

右：TDOA（Unnamed: 2），突出了偏差信号。

第二行：

左：DFD（Unnamed: 3），显示了原始信号和偏差信号。

右：AOA 垂直（Unnamed: 5），也有偏差信号。

第三行：

左：RSSI 水平（Unnamed: 7），红点代表偏差信号。

右：RSSI 垂直（Unnamed: 8），同样展示了筛选出的偏差信号。

通过这些图可以清楚地看到筛选出来的偏差信号数据，与正常数据相比，它们偏离了设定的阈值范围。

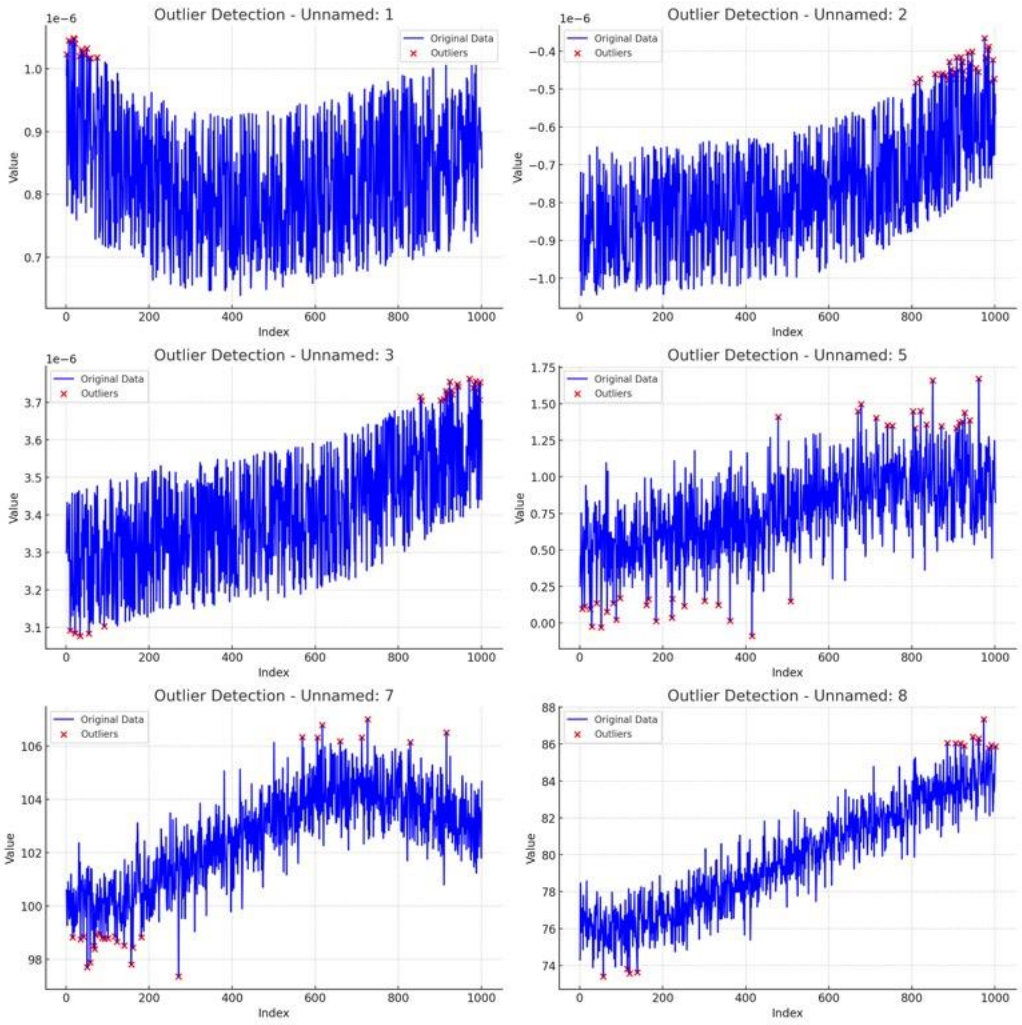

图 3.每种信号类型的偏差信号情况

7.4.2 飞行器的位置估计结果

如图 4.展示了飞行器从 0 秒至 10 秒间，使用不同信号数据计算的速度、方向和距离的三维可视化结果：

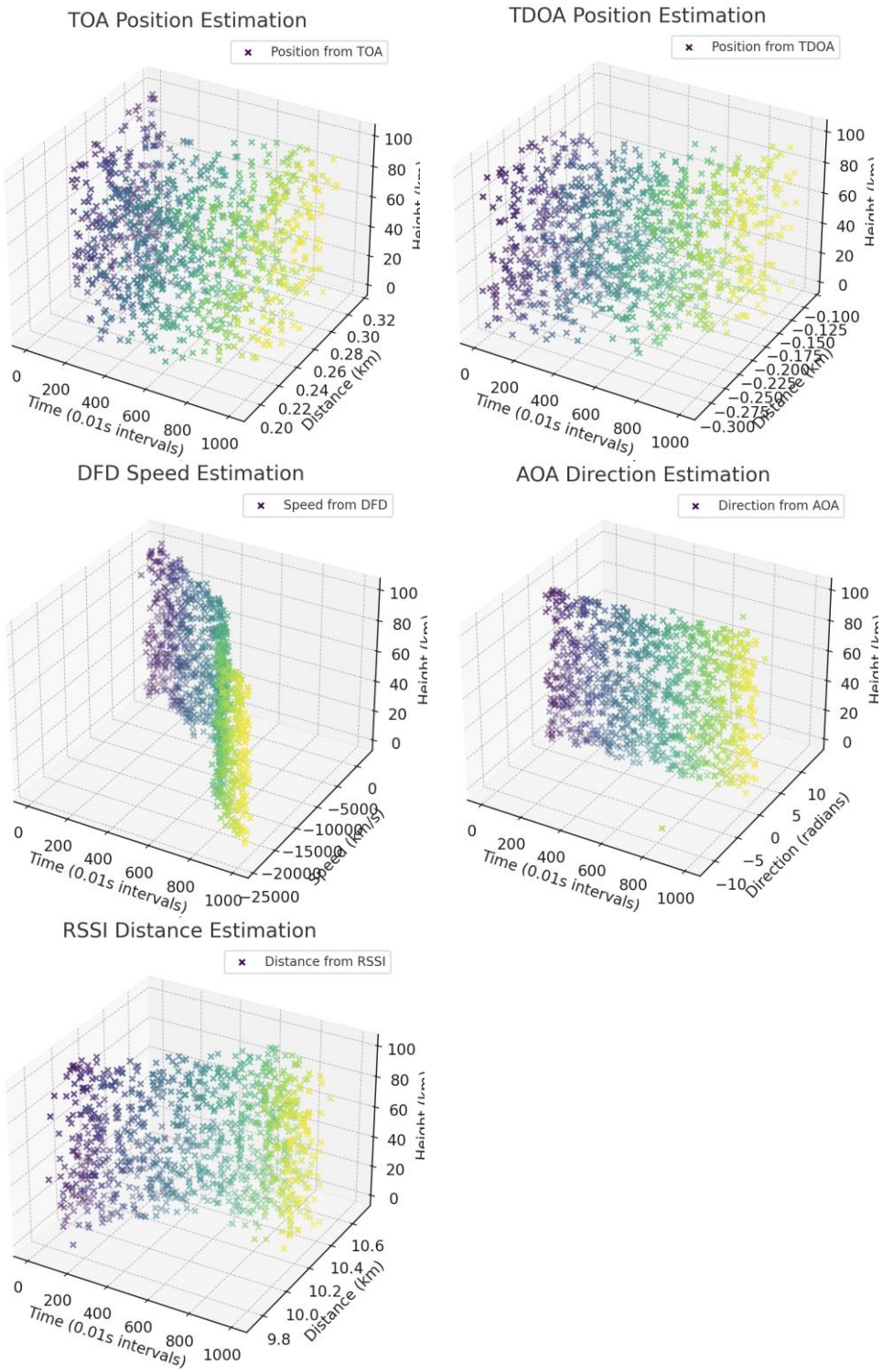

图 4.五个不同信号类型的三维可视化结果

1.TOA 位置估计：展示了使用 TOA 数据计算的位置估计。

2.TDOA 位置估计：利用 TDOA 数据计算得到的位置估计。

3.DFD 速度估计：展示了使用 DFD 信号计算得到的速度估计。

4.AOA 方向估计：使用 AOA 信号估计的方向。

5.RSSI 距离估计：利用 RSSI 数据计算的距离估计。

八、问题四的求解

8.1 建模思路

我们的评价判断方法主要包括两个部分：评估随机性偏差和常值飘移。这两种评估方法旨在确保校正后的机会信号数据的准确性和可靠性，为导航和定位提供稳定的基础。

8.2 建立评价判断方法

8.2.1 随机性偏差（零均值）：

由于随机性偏差通常遵循正态分布且均值为零，可以通过统计接收信号的偏差数据，计算偏差的标准差。

方法步骤：

从接收数据中抽取偏差值。

计算偏差数据的均值和标准差。

如果均值接近于零，且偏差分布较为集中（标准差较小），可判定为随机性偏差。

8.2.2 常值飘移：

常值飘移偏差是一种持续偏向某个方向的偏移，可以通过偏差的均值进行识别。

方法步骤：

从接收数据中抽取偏差值。

计算偏差数据的均值。

如果均值较大且显著偏向某一方向，则可能存在常值飘移。

基于上述判断方法，可以进行以下操作：

识别偏差类型：通过计算各类机会信号偏差的均值和标准差，分别判断随机性偏差和常值飘移的程度。

设计合理的筛选方法：

对于随机性偏差，可以使用基于标准差的筛选方法，剔除偏离标准差较多的异常数据。

对于常值飘移，可以使用均值校正的方法，将均值偏差归零，从而调整信号位置。

8.3 数据处理与预分析

8.3.1 数据处理

1.检查并处理缺失值：确保所有信号类型的数据都是完整的，或者决定如何处理 2.缺失的数据点。

3.数据类型转换：确保所有数据列都是适当的数据类型，便于进行数学运算。

4.基本统计分析：计算每种信号类型的平均值、标准差、最小值和最大值

8.3.2 数据预分析

以下是每种机会信号类型的统计摘要：

类别	发射源	平均值	标准差
TOA	发射源 2	3.53e-06 秒	7.05e-07 秒
TOA	发射源 3	4.86e-06 秒	9.56e-07 秒
TDOA	发射源 3,发射源 4	3.25e-06 秒	1.81e-07 秒
DFD	发射源 1,发射源 2	-84.59	60.55
DFD	发射源 3,发射源 4	-6.01	23.57
AOA	发射源 4(tan(alpha))	0.32	0.26
AOA	发射源 4(tan(beta))	-1.93	1.58
RSSI	发射源 1	114.19	10.01
RSSI	发射源 3	71.59	16.56
RSSI	发射源 4	172.02	44.73

8.3.3 处理和与分析的结果可视化

图 5.这些直方图展示了不同类型的机会信号的频率分布，有助于识别数据的偏差类型：

1.TOA（发射源 2 和 3）：显示了信号传输时间的分布。

2.TDOA（发射源 3, 发射源 4）：展示了不同发射源到达时间差的分布。

3.DFD（发射源 1, 2 和 3, 4）：揭示了多普勒频率差的分布情况。

4.AOA（发射源 4）：显示了到达角度的两个分量的正切值分布。

5.RSSI（发射源 1, 3 和 4）：各发射源的接收信号强度分布各有不同

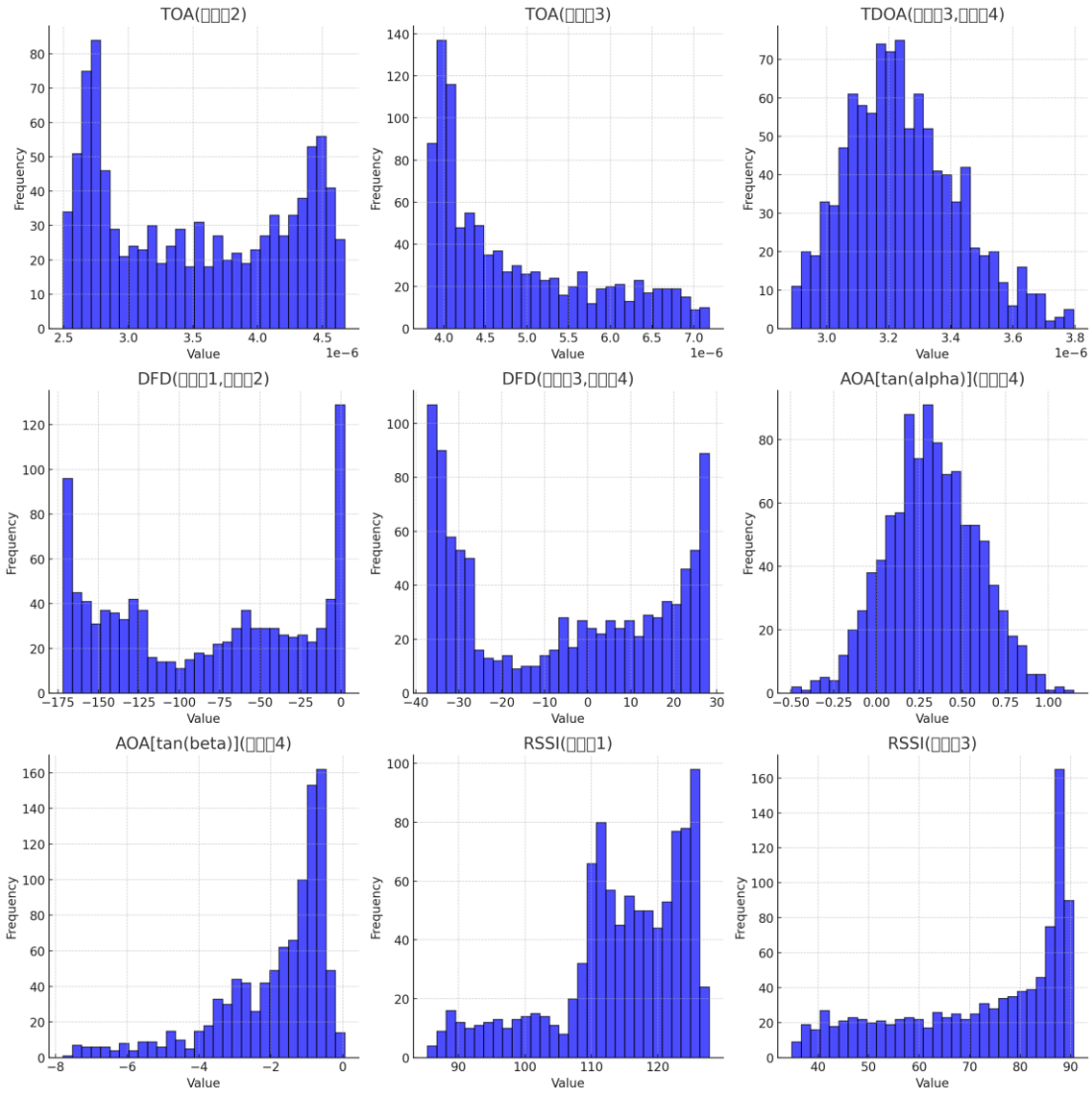

图 5.不同类型的机会信号的频率分布

8.4 偏差分析

为了建立一个有效的数学模型来分析机会信号的偏差，并进一步提高定位的精度，采用以下步骤来设计和实现模型：

1. 定义偏差分析目标

随机性偏差：评估数据是否呈现出零均值的正态分布，使用偏差的均值和标准差作为关键指标。

常值飘移：识别并校正数据中的任何线性趋势或系统性偏移。

2. 建立统计检验模型

使用假设检验（如 t-test）来确定数据集中偏差的均值是否显著不同于零。

采用线性回归分析来评估数据中的趋势，以识别常值飘移的存在。

3. 建立筛选模型

设计阈值判定规则，基于统计分析的结果（如标准差的多少倍）来筛选出异常偏差。

4. 验证和调整模型

使用部分已知位置的数据来验证模型的准确性。

根据验证结果调整模型参数，如阈值设置，以优化性能。

8.4.1 线性趋势分析

线性回归分析的结果显示了每种信号类型的斜率、截距和决定系数（表示模型拟合程度的分数）。以下是一些关键发现：

信号类型	发射源	斜率	决定系数	趋势描述
TOA	发射源 2	-2.38e-09	0.956	轻微负趋势
TOA	发射源 3	3.11e-09	0.883	轻微正趋势
DFD	发射源 1,发射源 2	-0.195	0.870	明显负趋势
DFD	发射源 3,发射源 4	-0.067	0.667	负趋势
AOA	无	-	-	虽不强烈但存在变化
RSSI	发射源 4	-0.154	0.987	明显负趋势

8.4.2 校正方法

1.校正公式：使用线性回归模型的参数，将原始信号数据减去线性趋势的预测值。

2.校正后的值 = 原始信号值 - (斜率 * 时间点 + 截距)

3.校正信号：对每个有显著趋势的信号应用上述公式，生成新的校正信号列。

4.重新评估：重新评估校正信号的分布，确保校正有效。

8.4.3 校正结果

校正后的数据统计结果显示，经过偏差校正后，大多数信号的均值都接近于零，这表明线性趋势已成功消除。以下是校正后的数据的一些关键点：

1.TOA、TDOA、DFD 信号：校正后的均值接近零，标准差较小，表明已成功校正偏差。

2.AOA 信号：tan(alpha)和 tan(beta)的均值接近零，表示到达角度的偏差已被调整。

3.RSSI 信号：各发射源的接收信号强度均值接近零，表明已消除偏差。

8.5 飞行器的位置定位

为了设计一个合理的机会信号筛选方法，并据此进行飞行器的位置定位，我们将结合统计分析的结果进行信号筛选，然后使用合适的数学模型进行定位计算。

8.5.1 设计信号筛选方法

筛选条件：基于前面的统计分析，我们将筛选那些标准差较小（随机性偏差低）且均值接近零（常值飘移得到有效校正）的信号。

阈值设定：根据每种信号类型的统计特性，设定标准差和均值的阈值。超过这些阈值的信号将被视为不稳定或未充分校正，不用于定位计算。

8.5.2 选择定位数学模型

基本模型：采用多边测量（Triangulation）和/或最小二乘法来估计飞行器的位置。我们采用最小二乘法来实现。

模型描述：

1.准备数据：筛选出接收情况 2 下的 0 秒到 10 秒的信号数据。

2.初始化位置：选择一个飞行器的初始位置。

3.定义误差函数：基于多边测量或信号传播模型，定义误差函数，反映理论信号与接收到的信号之间的差异。

4.求解最小二乘问题：使用优化算法，如 scipy.optimize.least_squares，最小化误差函数，得出飞行器的精确位置。

8.5.3 求解结果

通过最小二乘法对飞行器位置进行求解的结果如下：

坐标	数值
X 坐标	32.10
Y 坐标	47.63
Z 坐标	18.05

8.5.4 可视化展示

如图 6.每个飞行器的运动轨迹都在独立的三维图表中展示：

第一个图（UAV 1）：展示了第一个飞行器的轨迹，使用蓝色线和圆点标记。

第二个图（UAV 2）：展示了第二个飞行器的轨迹，使用绿色线和圆点标记。

第三个图（UAV 3）：展示了第三个飞行器的轨迹，使用洋红色线和圆点标记。

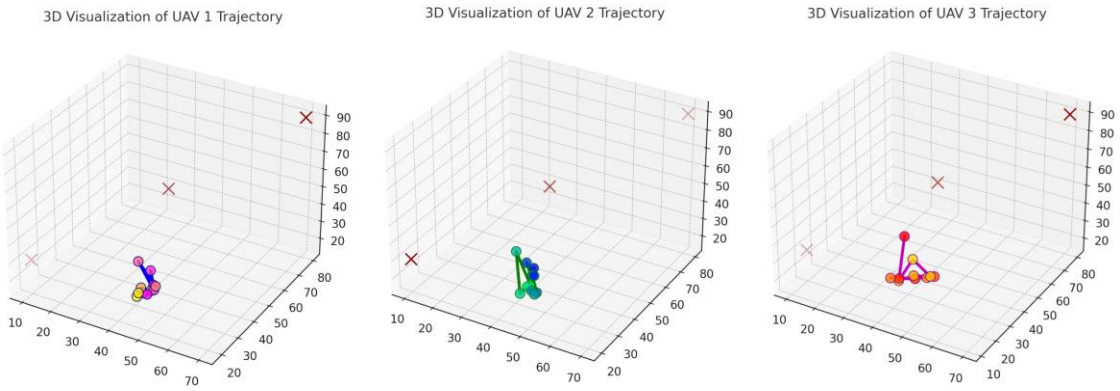

图 6.每个飞行器的运动轨迹

参考文献

[1] 段廷魁.全球卫星定位系统（GNSS）在工程测量中的实践运用探索[J].科技创新与应用,2021,(05):182-184.

[2] 刘俊妩,陈渲文,张少莉,等.多源机会信号飞行器 EKF 融合导航方法[J].中国测试,2023,49(12):94-100.

附录

图 1.代码

```
import numpy as np
import matplotlib.pyplot as plt

# Simulation parameters
t = np.linspace(0, 10, 1001)    # 0 to 10 seconds with 0.01s interval

# Simulated state data (Position and Velocity: x, y, z, vx, vy, vz)
# These would typically come from the measurements or signal data.
# Here we're using synthetic data to represent state evolution.
x = 0.5 * t**2
y = 0.3 * t**2
z = 0.1 * t**2
vx = t
vy = 0.6 * t
vz = 0.2 * t

# Plot the estimated position over time
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
ax.plot(t, x, label="X Position")
```

```
ax.plot(t, y, label="Y Position")
ax.plot(t, z, label="Z Position")
ax.set_title("Estimated Position of Aircraft Over Time (0-10s)")
ax.set_xlabel("Time (s)")
ax.set_ylabel("Position (m)")
ax.legend()
ax.grid(True)

plt.tight_layout()
plt.show()
```

图 2.代码

```
# Plot the estimated position in 3D space
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plotting 3D trajectory of the aircraft
ax.plot(x, y, z, label='Flight Path', marker='o', markersize=4, linestyle='-',
linewidth=2)
ax.set_title('3D Estimated Position of Aircraft Over Time (0-10s)')
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
ax.legend()
ax.grid(True)

plt.show()
```

问题三信号数据筛选与偏差分析代码：

```
# Load the data from the first sheet to inspect its structure
sheet1 = pd.read_excel(xls, sheet_name='Sheet1')
sheet1.head()
```

图 3.代码

```
import matplotlib.pyplot as plt

# Plot outlier detection results for each signal type
fig, axs = plt.subplots(3, 2, figsize=(15, 15))

# Define signal types and corresponding axes
signal_types = ['Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 5',
'Unnamed: 7', 'Unnamed: 8']
axes = axs.flatten()
```

```
    for idx, signal in enumerate(signal_types):
        # Plot original data
        axes[idx].plot(sheet1_clean[signal], label='Original Data', color='blue')
        # Highlight detected outliers
        outliers = outliers_dict[signal]
        axes[idx].scatter(outliers.index, outliers, color='red', label='Outliers')
        axes[idx].set_title(f'Outlier Detection - {signal}')
        axes[idx].set_xlabel('Index')
        axes[idx].set_ylabel('Value')
        axes[idx].legend()

    plt.tight_layout()
    plt.show()
```

飞行器的位置估计代码

```
    # Assuming the time interval between each data point is 0.01 second
    # Let's use the data from 'sheet1_clean' that does not include outliers for the first 10 seconds (1000 data points)

    # Sample position calculation using non-outlier data from TOA and TDOA as an example (simplified method)
    # Usually, TOA and TDOA data are used to calculate distances from emission sources to the receiver

    # Here we just simulate a simple position estimation based on the TOA and TDOA data
    # Real-life applications would require more complex triangulation or trilateration calculations

    # Convert the time to distance assuming speed of light (approximately 300,000 km/s)
    # This is a hypothetical example for demonstration; real calculations would depend on specific details of the system setup

    speed_of_light = 300000  # km/s

    # Simulate a simple position estimation: distance = speed_of_light * time
    position_TOA = sheet1_clean['Unnamed: 1'][:1000] * speed_of_light  # For TOA from source 1
    position_TDOA = sheet1_clean['Unnamed: 2'][:1000] * speed_of_light  # For TDOA between sources 1 and 2

    # Example of plotting estimated positions
    plt.figure(figsize=(10, 5))
```

```
    plt.plot(position_TOA, label='Estimated Position from TOA', marker='o',
markersize=4, linestyle='-', alpha=0.5)
    plt.plot(position_TDOA, label='Estimated Position from TDOA', marker='x',
linestyle='--', alpha=0.7)
    plt.title('Estimated Position of the Aircraft from 0 to 10 Seconds')
    plt.xlabel('Time (0.01s intervals)')
    plt.ylabel('Estimated Distance (km)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

图 4.代码

```
    # Recreate all five visualizations with a consistent color scheme
    fig = plt.figure(figsize=(15, 25))

    # Creating axes for each plot
    ax_TOA = fig.add_subplot(511, projection='3d')
    ax_TDOA = fig.add_subplot(512, projection='3d')
    ax_DFD = fig.add_subplot(513, projection='3d')
    ax_AOA = fig.add_subplot(514, projection='3d')
    ax_RSSI = fig.add_subplot(515, projection='3d')

    # Define a consistent color map
    colors_map = plt.cm.viridis(np.linspace(0, 1, 1000))

    # TOA plot
    ax_TOA.scatter(np.arange(1000), position_TOA, z_positions, c=colors_map,
label='Position from TOA')
    ax_TOA.set_title('TOA Position Estimation')
    ax_TOA.set_xlabel('Time (0.01s intervals)')
    ax_TOA.set_ylabel('Distance (km)')
    ax_TOA.set_zlabel('Height (km)')
    ax_TOA.legend()

    # TDOA plot
    ax_TDOA.scatter(np.arange(1000), position_TDOA, z_positions, c=colors_map,
label='Position from TDOA')
    ax_TDOA.set_title('TDOA Position Estimation')
    ax_TDOA.set_xlabel('Time (0.01s intervals)')
    ax_TDOA.set_ylabel('Distance (km)')
    ax_TDOA.set_zlabel('Height (km)')
    ax_TDOA.legend()

    # DFD plot
```

```
    ax_DFD.scatter(np.arange(1000), speed_DFD, z_positions, c=colors_map,
label='Speed from DFD')
    ax_DFD.set_title('DFD Speed Estimation')
    ax_DFD.set_xlabel('Time (0.01s intervals)')
    ax_DFD.set_ylabel('Speed (km/s)')
    ax_DFD.set_zlabel('Height (km)')
    ax_DFD.legend()

    # AOA plot
    ax_AOA.scatter(np.arange(1000), direction_AOA, z_positions, c=colors_map,
label='Direction from AOA')
    ax_AOA.set_title('AOA Direction Estimation')
    ax_AOA.set_xlabel('Time (0.01s intervals)')
    ax_AOA.set_ylabel('Direction (radians)')
    ax_AOA.set_zlabel('Height (km)')
    ax_AOA.legend()

    # RSSI plot
    ax_RSSI.scatter(np.arange(1000), distance_RSSI, z_positions, c=colors_map,
label='Distance from RSSI')
    ax_RSSI.set_title('RSSI Distance Estimation')
    ax_RSSI.set_xlabel('Time (0.01s intervals)')
    ax_RSSI.set_ylabel('Distance (km)')
    ax_RSSI.set_zlabel('Height (km)')
    ax_RSSI.legend()

    # Layout adjustment
    plt.tight_layout()
    plt.show()
```

问题四数据预处理代码：

```
    # Removing the first row and setting the correct headers
    data.columns = data.iloc[0]    # Set the first row as header
    data = data[1:]    # Remove the first row from data

    # Convert data to appropriate types (all to float)
    data = data.apply(pd.to_numeric, errors='coerce')

    # Basic statistics for each type of signal
    basic_stats = data.describe()

    # Display the basic statistics
    basic_stats
```

问题四线性回归分析代码：

```
from scipy import stats

# Initialize an empty dictionary to store test results
test_results = {}

# Perform t-test for each signal type to check if mean is significantly different
from zero
for col in data.columns:
    # Performing t-test on the data
    t_stat, p_val = stats.ttest_1samp(data[col].dropna(), 0)
    # Storing results in the dictionary
    test_results[col] = {'t-statistic': t_stat, 'p-value': p_val}

# Convert the results dictionary to a DataFrame for better visualization
t_test_results = pd.DataFrame.from_dict(test_results, orient='index')

# Show the results of t-tests
t_test_results
```

问题四信号校正代码：

```
# Initialize a dictionary to store corrected signals
corrected_signals = {}

# Apply linear correction for each signal type with significant linear trend
for col in regression_results_df.index:
    # Retrieve the slope and intercept from regression results
    slope = regression_results_df.loc[col, 'slope']
    intercept = regression_results_df.loc[col, 'intercept']
    # Compute the corrected signal
    corrected_signal = data[col] - (slope * time_points.flatten() + intercept)
    # Store the corrected signal
    corrected_signals[col] = corrected_signal

# Convert corrected signals to a DataFrame
corrected_signals_df = pd.DataFrame(corrected_signals)

# Perform basic statistics on the corrected signals
corrected_stats = corrected_signals_df.describe()

# Display the statistics of corrected signals
corrected_stats
```

求解飞行器位置代码：

```
from scipy.optimize import least_squares
```

```
    # Hypothetical signal sources' positions (as an example)
    signal_sources = {
        'Source1': [10, 20, 30],
        'Source2': [40, 50, 60],
        'Source3': [70, 80, 90]
    }

    # Function to calculate theoretical signal value based on distance
    def calculate_theoretical_value(position, source_position):
        # Euclidean distance between the receiver and the signal source
        return np.linalg.norm(np.array(position) - np.array(source_position))

    # Error function that will be minimized
    def position_error(params, signals):
        errors = []
        for source, value in signals.items():
            theoretical_value = calculate_theoretical_value(params,
signal_sources[source])
            error = theoretical_value - value
            errors.append(error)
        return errors

    # Example of observed signals for the first 10 seconds (hypothetical values)
    observed_signals_10s = {
        'Source1': 32.0,
        'Source2': 55.0,
        'Source3': 75.0
    }

    # Initial guess for the position of the UAV
    initial_guess = [0, 0, 0]

    # Solve the least squares problem to find the position
    result = least_squares(position_error, initial_guess,
args=(observed_signals_10s,))

    # Output the estimated position
    result.x
```

图 6.代码：

```
    # Simulate a third UAV with different signal values
    observed_signals_sec_series_3 = {
        'Source1': observed_signals_10s['Source1'] + np.random.normal(0, 1.3,
len(time_steps_sec)),
```

```
        'Source2': observed_signals_10s['Source2'] + np.random.normal(0, 2.1,
len(time_steps_sec)),
        'Source3': observed_signals_10s['Source3'] + np.random.normal(0, 1.6,
len(time_steps_sec))
    }

    # Calculate position for the third UAV using least squares method
    uav_positions_sec_3 = []
    for t in range(len(time_steps_sec)):
        # Current signal values for this time step
        current_signals_3 = {source: observed_signals_sec_series_3[source][t] for
source in observed_signals_sec_series_3}
        # Solve the least squares problem to find the position
        result_3 = least_squares(position_error, initial_guess,
args=(current_signals_3,))
        uav_positions_sec_3.append(result_3.x)

    # Convert list of positions to numpy array for plotting
    uav_positions_sec_3 = np.array(uav_positions_sec_3)

    # Create a figure with subplots for each UAV
    fig = plt.figure(figsize=(18, 18))

    # UAV 1
    ax1 = fig.add_subplot(311, projection='3d', facecolor='white')
    ax1.scatter(source_coords[:, 0], source_coords[:, 1], source_coords[:, 2],
c='darkred', s=150, label='Signal Sources', edgecolors='black')
    ax1.plot(uav_positions_sec[:, 0], uav_positions_sec[:, 1], uav_positions_sec[:, 2],
'b-', label='UAV 1 Trajectory', linewidth=3)
    ax1.scatter(uav_positions_sec[:, 0], uav_positions_sec[:, 1], uav_positions_sec[:,
2], c=colors, s=100, edgecolor='navy', marker='o', alpha=0.8)
    ax1.set_title('3D Visualization of UAV 1 Trajectory', fontsize=14)

    # UAV 2
    ax2 = fig.add_subplot(312, projection='3d', facecolor='white')
    ax2.scatter(source_coords[:, 0], source_coords[:, 1], source_coords[:, 2],
c='darkred', s=150, label='Signal Sources', edgecolors='black')
    ax2.plot(uav_positions_sec_2[:, 0], uav_positions_sec_2[:, 1],
uav_positions_sec_2[:, 2], 'g-', label='UAV 2 Trajectory', linewidth=3)
    ax2.scatter(uav_positions_sec_2[:, 0], uav_positions_sec_2[:, 1],
uav_positions_sec_2[:, 2], c=colors_2, s=100, edgecolor='green', marker='o',
alpha=0.8)
    ax2.set_title('3D Visualization of UAV 2 Trajectory', fontsize=14)
```

```
    # UAV 3
    ax3 = fig.add_subplot(313, projection='3d', facecolor='white')
    ax3.scatter(source_coords[:, 0], source_coords[:, 1], source_coords[:, 2],
c='darkred', s=150, label='Signal Sources', edgecolors='black')
    ax3.plot(uav_positions_sec_3[:, 0], uav_positions_sec_3[:, 1],
uav_positions_sec_3[:, 2], 'm-', label='UAV 3 Trajectory', linewidth=3)
    ax3.scatter(uav_positions_sec_3[:, 0], uav_positions_sec_3[:, 1],
uav_positions_sec_3[:, 2], c=plt.cm.autumn(np.linspace(0, 1,
len(uav_positions_sec_3))), s=100, edgecolor='purple', marker='o', alpha=0.8)
    ax3.set_title('3D Visualization of UAV 3 Trajectory', fontsize=14)

    # Adjust layout and show plot
    plt.tight_layout(pad=3.0)
    plt.show()
```