室内定位问题

摘要

本文根据题目中所给的 4 个已知的信标节点定位参数与其他节点的位置参数，求解关于未知节点位置、信标节点毁坏的影响及其调整方案与节点之间相互定位的问题。

针对求解未知节点的位置，本文采取分析 RSSI 节点定位问题常用的对数常态模型，即未知节点所接收到的 RSSI 信号与未知节点到信标节点的距离的对数成正比，以此作为基础，通过最小二乘拟合求得模型中所必需的常数 A 和路径损耗因子 n。再通过四边定位算法，将 RSSI 信号转换成测量距离后，通过再次最小二乘拟合求得未知节点的预估位置。鉴于四边定位算法仍有一定的误差，再通过 Taylor 级数算法，在初始点进行泰勒展开，并通过迭代，获得未知节点误差数据，与预估位置结合起来就可以获得未知节点的优化预估位置。结果为 $(3.1180, 3.0637)$; $(9.2740, 1.2393)$; $(4.3094, 2.9400)$; $(1.8137, 3.5926)$; $(9.0509, 1.1196)$; $(9.8367, 0.6013)$; $(4.3870, 1.4846)$; $(1.1659, 1.5496)$; $(2.5605, 2.0592)$; $(4.1029, 2.5278)$。

针对缺少某一信标节点，本文先假设信标节点 C 被毁坏，通过计算测量点与任一信标节点的距离误差函数，再结合多元随机变量的知识，求得在 3 个信标节点的情况下平面上一点出现的概率 $p(x, y)$，通过对其计算得到期望点，计算距离误差。再与 4 个信标节点的结果，相互比较得到影响，并同时验证了表 1 的数据。简单的定性结论是：4 个信标节点的定位效果比缺少任何一个信标节点的情况都要准确，同时缺少信标节点 C 的情况下对于表 1 的数据影响最大。

针对信标节点毁坏后的信标节点位置调整方案，本文通过数学定理的证明得到一个简单易懂的结论：当测量节点与三个信标节点的连线成 120° 时，测量误差最小，定位精度最高。

针对节点之间相互定位的问题，本文采取 Markov 链的思维，先通过问题一的模型求得初始的优化预估位置，通过足够多次的扰动，可以得到密度最高的区域，选取概率最高的 1 ~ 4 个点 (看图选定)，取平均后减去中心的坐标就是相对与中心的偏移量，将偏移量加上优化预估位置数据就能获得未知节点的位置数据。结果为 $(1.79127801, 4.29584893)$; $(6.16185836, 3.93273367)$; $(4.71724098, 1.70506389)$; $(7.48347061, 2.72824931)$; $(6.52969316, 1.17499307)$。并在确定任意信标节点毁坏的影响时，采用同样的算法，得到结果，再计算距离误差，得到结论：4 个信标节点的定位效果仍然比缺少任何一个信标节点的情况都要准确，缺少信标节点 B 对于表 2 的数据影响最小，缺少信标节点 A 影响最大。

关键词：对数常态模型　四边定位算法　Taylor 级数算法　高斯误差　概率分布　距离误差　Markov 链　偏移

一、 问题重述

1.1 问题背景

通过采用无线通讯、基站定位等技术方式，可以实现在室内环境中的位置定位，从而更好地追踪人员或物体的行迹。本问题所使用较常用的 RSSI 测距方法来讨论一系列人或物体室内位置的定位问题。

1.2 问题提出

在某一矩形房间内放置有 4 个已知位置的信标节点，并提供了 10 个已知位置的测量点的各部分参数，从而回答以下问题：

(1) 通过 10 个已知位置节点的数据来估计另外 10 个未知位置节点的位置数据；
(2) 讨论 4 个信标节点任一毁坏对定位精度产生的影响，验证之；并调整未毁坏的另外 3 个信标节点的位置使毁坏的信标节点影响尽可能地小；
(3) 确定 5 个既可以接收 4 个信标节点，又可以接收其他 4 个未知位置节点 RSSI 信息的未知位置节点的位置，并讨论 4 个信标节点任一毁坏对定位精度产生的影响。

二、 问题分析

2.1 问题一的分析

题目的数据是基于 RSSI 测距技术得到的，其大致原理是通过测量发射信号强度的衰减程度来测得物体与发射点之间的距离。对于信号传播，常见的有哈他模型、自由空间传播模型、对数常态 (对数正态) 模型[3]。综合考虑测量空间是一个有遮挡物、噪声、空气等干扰因素的空间，以上三种模型中最合适的便是对数常态模型。同时考虑到手工计算的困难，我们借助 Matlab 软件来求解所需数值。

同时考虑到测量精度和误差会使 RSSI 测量值产生不稳定的波动，我们采取最小二乘法拟合来获得对数常态模型中需得到的变量值。之后通过四边定位算法来最小二乘拟合出未知节点的所在位置。并通过与 Taylor 级数算法的结合，更加精确的获取未知节点的位置坐标信息。

2.2 问题二的分析

为了得到某一信标节点被毁坏的影响，我们需要先知道已知节点在问题一的模型下所得到的优化位置到 O 点 (也可取其余的未毁坏的点，具有等价性) 的误差函数，在得到误差

函数之后，我们可以计算得到平面上一点 (x, y) 在这点出现的联合概率密度函数 $p(x, y)$，绘制概率云图，得到某一片区域的质心，该质心即为剩余 3 个信标节点所测得的测量点位置。

再与用 4 个信标节点得到的测量点位置进行比较，获得某一信标节点被毁坏的影响，同时也完成了表 1 数据代入的验证，同时也可以获得定量与定性的结论。调整方案可以利用数学结论进行完成，证明只要保证测量节点与三个信标节点的连线成 120°，测量误差就是最小的即可。

2.3 问题三的分析

借助 Markov 链的模型，我们可以将通过问题一求得的 5 个未知节点的优化预估位置作为初始状态，通过足够多的微小扰动，求得密度最大的区域，选取概率最高的 n 个点取平均后减去中心点的位置，获得偏移量，把优化预估位置加上偏移量，就能得到 5 个未知节点的测量位置数据。

再将相应的信标节点的信息抹去，计算某一信标节点被毁坏时，在 Markov 链模型下，未知节点最有可能出现的位置信息，并通过分析得到结论。

三、 模型假设

为了更好地建立并求解模型，不妨对模型作出以下合理的假设：

(1) 4 个信标节点所处环境、工作能力等设备性能完全一致；
(2) 所给数据的误差在较小范围内，对结果的影响较小；
(3) 调整信标节点时不会对其设备性能产生影响；
(4) 不同设备之间的信号的相互干扰在对数常态模型的接受范围内；
(5) 接收节点和发射节点之间完全无遮挡；
(6) 信标节点 O, A, B, C 发射功率、天线收发信号的能力稳定；
(7) 系统损耗因子相同；
(8) 工作信号波长相同；
(9) 电缆和缆头损耗忽略不计。

四、 主要符号说明

符号	符号意义	符号	符号意义
i	第 i 个未知节点	j	第 j 个已知节点
RSSI_d	发射点距 d 时接收点 接收的信号强度	A_P	距发射节点 P 为 1 m 时 接收节点接收到的信号强度
n_P	点 P 的路径损耗因子	$d_{P_{i(j)}}$	第 $i(j)$ 个接收点距 发射点 P 的距离
$\mathrm{RSSI}_{P_{i(j)}}$	第 $i(j)$ 个接收点接收到 发射点 P 的信号强度	(x_P, y_P)	信标节点 P 的坐标位置
$Z_i(u_i, v_i)$	第 i 个未知节点的 估计坐标位置	dd_{P_i}	在 Z_i 处接收到的 信标节点 P 的信号强度
$(\Delta x_i, \Delta y_i)$	第 i 个未知节点的 RSSI 位置减去估计位置	$f_{P_i}(x, y)$	第 i 个未知节点的估计位置 到信标节点 P 的距离
$(optx_j, opty_j)$	第 j 个已知节点的优化位置	xz_{P_j}	第 j 个已知节点的优化位置 到信标节点 P 的距离
wc_{P_j}	第 j 个已知节点的优化位置 到信标节点 P 的距离 减去给定距离的绝对值	(x_j, y_j)	第 j 个已知节点的给定位置
$r_{P_{(j)}}$	平面任意一个点到 信标节点 P 的距离	$p(x, y)$	平面上一点出现的概率
avg_P	缺少信标节点 P 时 的平均距离误差	avg	信标节点全存在时的平均距离误差
f_P	信标节点 P 在某一点的 概率密度函数	R_{Pj}	平面任意一个点到 信标节点 P 的 RSSI 距离

五、 模型的建立和求解

5.1 数据预处理

距离发射节点 d 处天线的接收功率由弗里斯传输公式[1] 给出：

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{5.1}$$

其中，$P_r(d)$ 是接收功率，P_t 是发射功率，G_t 是发射天线增益，G_r 是接收天线增益，λ 是信号波长，d 是接收节点和发射节点之间的距离，L 是系统损耗因子 (与传播无关)。天线增益是衡量天线朝某特定方向收发信号的能力。

基于之前的模型假设第 (5) ~ (9) 条，我们认定所给数据是稳定可靠的。

5.2 问题一模型的建立和求解

5.2.1 对数常态模型

对数常态模型[2] 考虑了环境中较多因素，如遮挡物、噪声、信号干扰等，其模型公式为

$$PL(d) = PL(d_0) - 10n\lg\left(\frac{d}{d_0}\right) + X_\sigma \tag{5.2}$$

其中，$PL(d)$ 为发射点距 d 时接收点接收的信号强度；$PL(d_0)$ 为发射点距 d_0 时接收点接收的信号强度；n 为路径损耗因子 (与环境有关)；d 为某接收点距发射点的距离；d_0 为参考距离；X_σ 为遮蔽因子，并服从 $N(0,\sigma^2)$；信号强度的单位均为 dBm，距离单位为 m。

在实际应用中，为便于计算，可以忽略遮蔽因子，同时取 $d_0 = 1$，则对数常态模型公式可以简写成：

$$\mathrm{RSSI}_d = A - 10n\lg(d) \tag{5.3}$$

在计算 RSSI_d 之前，我们还需要计算确定常数 A 和路径损耗因子 n。我们可以通过最小二乘法来计算题目中表 1 所给的数据，下面以 RSSI_O 为例，简单介绍计算过程：

(1) 先通过每个点的 X 坐标和 Y 坐标计算出每个点到该信标节点 (目前是 O 点) 距离 d_{O_i}；

$$d_{O_i} = \sqrt{(x_i - x_O)^2 + (y_i - y_O)^2} \tag{5.4}$$

(2) 对于每一个 (x_i, y_i) 都有

$$\mathrm{RSSI}_{O_i} = A - 10n\lg(d_{O_i}) \tag{5.5}$$

将其抽象成矩阵的形式，则有

$$\begin{pmatrix} \mathrm{RSSI}_{O_1} \\ \mathrm{RSSI}_{O_2} \\ \vdots \\ \mathrm{RSSI}_{O_{10}} \end{pmatrix} = \begin{pmatrix} 1 & -10n\lg(d_{O_1}) \\ 1 & -10n\lg(d_{O_2}) \\ \vdots & \vdots \\ 1 & -10n\lg(d_{O_{10}}) \end{pmatrix} \begin{pmatrix} A_O \\ n_O \end{pmatrix} \tag{5.6}$$

(3) 将式 (5.6) 等号左边的矩阵记为 R_O，等号右边分别记为 C_O, X_O，则通过最小二乘法可以解得

$$\hat{X}_O = (C_O^{\mathrm{T}} C_O)^{-1} C_O^{\mathrm{T}} R_O \tag{5.7}$$

得到的结果 $\hat{X}_O = \begin{pmatrix} A_O \\ n_O \end{pmatrix}$。

(4) 通过类似的步骤可以得到对于另外信标节点 A, B, C 的 $A_A, A_B, A_C, n_A, n_B, n_C$。

5.2.2 四边定位算法计算未知节点的位置

在得到以上需要的数据之后，我们可以通过四边定位算法[2] 来计算另外 10 个未知节点的位置。此算法的大致原理为：已知 4 个不共线的点 A, B, C, D 的位置坐标和一个未知位置坐标的点 P，且点 P 到这 4 个点的距离分别为 d_1, d_2, d_3, d_4，我们就可以以这 4 个点为圆心，d_i 为半径作圆，在理想状态下，这 4 个圆会相交于一点，该点就是点 P，如下图 (1)，但是在实际情况中，由于测量误差的影响，这 4 个圆会相交出一块区域，而实际的点 P 的位置应该就在这片区域内，如下图 (2)。

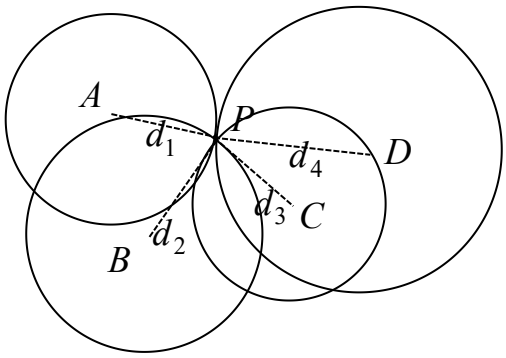

图 1: 理想状态下的 4 个圆

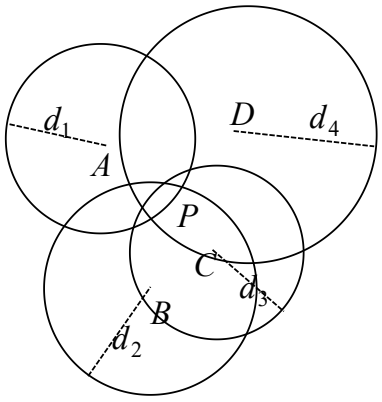

图 2: 实际情况中的 4 个圆

那么对于本问题来说，我们已知的是信标节点 O, A, B, C 的位置坐标，每个点对应的常数 A 和路径损耗因子 n，我们再以 10 个未知点中的第 1 个点 $Z_1(u_1, v_1)$ 为例，简单介绍计算过程：

(1) 因为在题目所给的表 2 中已经详细给出了未知点在 4 个信标节点处的位置，我们将其转变为方程组：

$$\begin{cases} (u_1 - x_O)^2 + (v_1 - y_O)^2 = dd_{O_1}^2 \\ (u_1 - x_A)^2 + (v_1 - y_A)^2 = dd_{A_1}^2 \\ (u_1 - x_B)^2 + (v_1 - y_B)^2 = dd_{B_1}^2 \\ (u_1 - x_C)^2 + (v_1 - y_C)^2 = dd_{C_1}^2 \end{cases} \tag{5.8}$$

其中，dd_{P_i} 表示的是在 Z_i 处接收到的信标节点 P 的信号强度 (称为 RSSI 位置)[4]，即

$$dd_{P_i} = 10^{\frac{A_P - \mathrm{RSSI}_{P_i}}{10 n_P}} \tag{5.9}$$

(2) 将式 (5.8) 展开，可得

$$\begin{cases} u_1^2+v_1^2-2x_Ou_1-2y_Ov_1=dd_{O_1}^2-x_O^2-y_O^2 \\ u_1^2+v_1^2-2x_Au_1-2y_Av_1=dd_{A_1}^2-x_A^2-y_A^2 \\ u_1^2+v_1^2-2x_Bu_1-2y_Bv_1=dd_{B_1}^2-x_B^2-y_B^2 \\ u_1^2+v_1^2-2x_Cu_1-2y_Cv_1=dd_{C_1}^2-x_C^2-y_C^2 \end{cases} \tag{5.10}$$

同时减去第 4 行的等式，可得

$$\begin{cases} 2(x_C-x_O)u_1+2(y_C-y_O)v_1=dd_{O_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_O^2-y_O^2 \\ 2(x_C-x_A)u_1+2(y_C-y_A)v_1=dd_{A_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_A^2-y_A^2 \\ 2(x_C-x_B)u_1+2(y_C-y_B)v_1=dd_{B_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_B^2-y_B^2 \end{cases} \tag{5.11}$$

(3) 将式 (5.11) 写成矩阵形式：

$$\begin{pmatrix} 2(x_C-x_O) & 2(y_C-y_O) \\ 2(x_C-x_A) & 2(y_C-y_A) \\ 2(x_C-x_B) & 2(y_C-y_B) \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} dd_{O_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_O^2-y_O^2 \\ dd_{A_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_A^2-y_A^2 \\ dd_{B_1}^2-dd_{C_1}^2+x_C^2+y_C^2-x_B^2-y_B^2 \end{pmatrix} \tag{5.12}$$

(4) 将式 (5.12) 的等号左边分别记为 C, X，等号右边记为 D，则通过最小二乘法可解得

$$\hat{X}=(C^{\mathrm{T}}C)^{-1}C^{\mathrm{T}}D \tag{5.13}$$

于此，我们便解得了 $Z_1(u_1, v_1)$ 的坐标。

(5) 通过类似的步骤，我们可以解得其他 9 个未知点的位置。

解得的 10 个未知点的预估位置坐标如表 (1)：

表 1: 10 个未知点的预估位置坐标数据

节点标号	1	2	3	4	5
X 坐标	3.1182	9.2740	4.3106	1.8138	9.0450
Y 坐标	3.0645	1.2392	2.9434	3.5929	1.1309
节点标号	6	7	8	9	10
X 坐标	9.8470	4.3856	1.1658	2.5606	4.1029
Y 坐标	0.5842	1.4940	1.5497	2.0591	2.5279

为更直观，我们可以利用 Matlab 将这些点绘制在图片中：

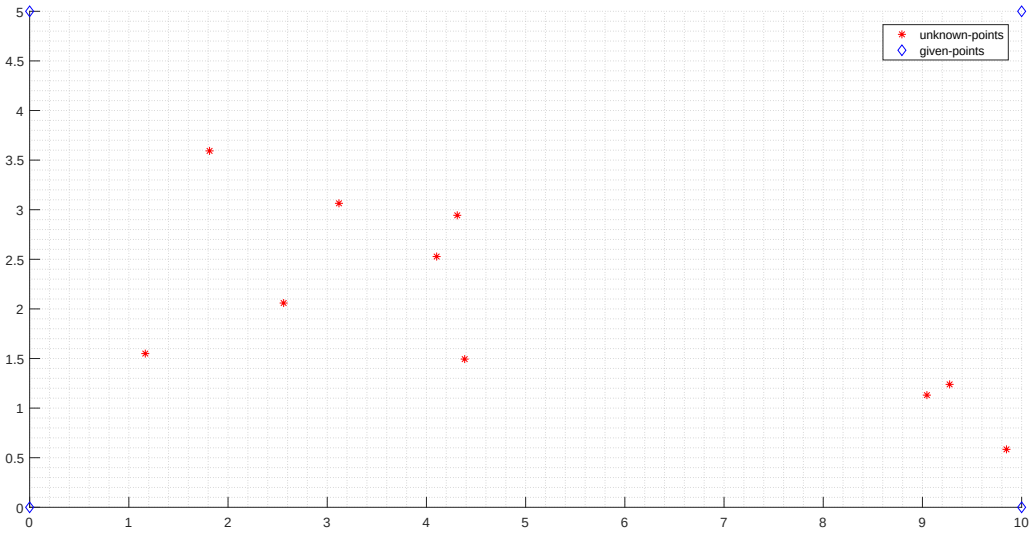

图 3: 10 个未知点的预估位置图

5.2.3 Taylor 级数算法优化未知节点的位置

我们之前所提到的四边定位算法是一种特殊的最大似然估计法，我们可以通过求出的估计位置作为一个初始迭代点进行迭代，而迭代的算法是定位精度较高的 Taylor 级数算法[3]。利用求出的估计位置作为初始迭代点，能够更快更有效的使 Taylor 级数算法收敛，减少迭代的次数，提高定位的精度。

以某一个未知节点为例，大致思路如下：

(1) 第 i 个未知节点的估计位置根据前文可计算得 (u_i, v_i)，(U_{P_i}, V_{P_i}) 为用 RSSI 公式求出的第 i 个未知节点到信标节点 P 的距离，则会存在一个位置偏差

$$(\Delta x_i, \Delta y_i) = (U_{P_i}, V_{P_i}) - (u_i, v_i) \tag{5.14}$$

(2) 令 $f_{P_i}(x, y)$ 表示第 i 个未知节点的估计位置到信标节点 P 的距离，即

$$f_{P_i}(u_i, v_i) = \sqrt{(u_i - x_P)^2 + (v_i - y_P)^2} \tag{5.15}$$

(3) 将式 (5.15) 在初始迭代点处进行泰勒展开，可得

$$\begin{aligned} f_{P_i}(U_i, V_i) &= f_{P_i}(u_i + \Delta x_i, v_i + \Delta y_i) \\ &\approx f_{P_i}(u_i, v_i) + \left.\frac{\partial f_{P_i}}{\partial x}\right|_{u_i} \Delta x_i + \left.\frac{\partial f_{P_i}}{\partial y}\right|_{v_i} \Delta y_i \\ &= f_{P_i}(u_i, v_i) + \frac{x_P - u_i}{\sqrt{(u_i - x_P)^2 + (v_i - y_P)^2}} \Delta x_i + \frac{y_P - v_i}{\sqrt{(u_i - x_P)^2 + (v_i - y_P)^2}} \Delta y_i \end{aligned} \tag{5.16}$$

其中，为保持整体的线性性，式 (5.16) 省略了一阶偏导数后面的项，同时进行化简，我们可得到

$$f_{P_i}(U_i, V_i) - f_{P_i}(u_i, v_i) = \frac{x_P - u_i}{\sqrt{(u_i - x_P)^2 + (v_i - y_P)^2}}\Delta x_i + \frac{y_P - v_i}{\sqrt{(u_i - x_P)^2 + (v_i - y_P)^2}}\Delta y_i \quad (5.17)$$

(4) 将式 (5.17) 写成矩阵形式，即

$$\begin{pmatrix} \frac{x_O - u_1}{f_{O_1}(u_1, v_1)} & \frac{y_O - v_1}{f_{O_1}(u_1, v_1)} \\ \frac{x_A - u_1}{f_{A_1}(u_1, v_1)} & \frac{y_A - v_1}{f_{A_1}(u_1, v_1)} \\ \frac{x_B - u_1}{f_{B_1}(u_1, v_1)} & \frac{y_B - v_1}{f_{B_1}(u_1, v_1)} \\ \frac{x_C - u_1}{f_{C_1}(u_1, v_1)} & \frac{y_C - v_1}{f_{C_1}(u_1, v_1)} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta y_1 \end{pmatrix} = \begin{pmatrix} f_{O_1}(u_1, v_1) - dd_{O_1}^2 \\ f_{A_1}(u_1, v_1) - dd_{A_1}^2 \\ f_{B_1}(u_1, v_1) - dd_{B_1}^2 \\ f_{C_1}(u_1, v_1) - dd_{C_1}^2 \end{pmatrix} \quad (5.18)$$

(5) 将式 (5.18) 等号左边记为 G, X，等号右边记为 h，则根据最小二乘原则可得

$$\hat{X} = (G^{\mathrm{T}}G)^{-1}G^{\mathrm{T}}h \quad (5.19)$$

(6) 为改进迭代的精度我们要增加一个判断迭代终止的条件[5]，条件为

$$|\Delta x_i + \Delta y_i| < 0.01 \quad (5.20)$$

如果式 (5.20) 不满足，则将 (u_i, v_i) 更改为 $(u_i + \Delta x_i, v_i + \Delta y_i)$，再回到步骤 (5)，重新计算，直到满足式 (5.20)。

(7) 同理可求得另外 9 个点的 $(\Delta x_i, \Delta y_i)(i = 1, 2, \cdots, 10)$，再加上原来的预估位置 (u_i, v_i)，即可求得通过 Taylor 级数算法优化的新的预估位置 $(u_i', v_i') = (u_i, v_i) + (\Delta x_i, \Delta y_i)$。

5.2.4 10 个未知节点的优化预估位置

解得的 10 个未知点的优化预估位置坐标如表 (2)：

表 2: 10 个未知点的优化预估位置坐标数据

节点标号	1	2	3	4	5
X 坐标	3.1180	9.2740	4.3094	1.8137	9.0509
Y 坐标	3.0637	1.2393	2.9400	3.5926	1.1196
节点标号	6	7	8	9	10
X 坐标	9.8367	4.3870	1.1659	2.5605	4.1029
Y 坐标	0.6013	1.4846	1.5496	2.0592	2.5278

可视化图如下，图中红色的点为经过 Taylor 级数算法优化的位置，蓝色的点为仅用四边定位算法得到的位置结果：

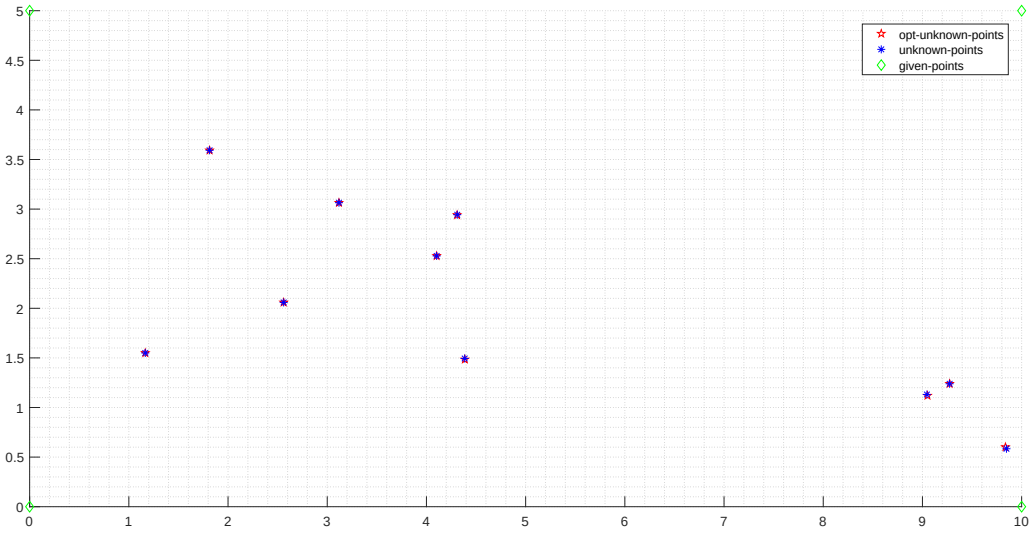

图 4: 10 个未知点的优化预估位置图

以上，问题一所求的 10 个未知节点的位置数据已呈现。

5.3 问题二模型的建立和求解

5.3.1 四边定位泰勒优化算法的高斯误差函数

在确定任意一个信标节点的毁坏对定位精度的影响前，我们先计算通过问题一的模型得到的误差函数，并且这个误差函数针对的是 10 个已知节点，即利用 10 个已知节点所接收到的 $\mathrm{RSSI}_O, \mathrm{RSSI}_A, \mathrm{RSSI}_B, \mathrm{RSSI}_C$ 信号，通过 RSSI 公式、四边定位算法和 Taylor 级数算法算得的优化位置与已知的某一个信标节点之间的误差函数[7]。

我们使用常用的高斯函数 (也称高斯误差函数、正态分布概率密度函数) 求解高斯误差，高斯函数的形式如下：

$$f(x) = \frac{1}{\sqrt{2\pi}c} \exp\left[-\frac{(x-b)^2}{2c^2}\right] \tag{5.21}$$

其中，$\frac{1}{\sqrt{2\pi}c}$ 是得到的曲线高度，b 是曲线的对称轴，c 是曲线的半宽度信息。

利用高斯函数求误差函数的原理及步骤如下，同时为方便获取误差信息，我们计算的是 10 个已知节点的优化位置到 O 点的距离误差 (选取其他的点具有等价性)：

(1) 目前对 10 个已知节点有 10 个优化位置数据，为 $(optx_j, opty_j)(j = 1, 2, \cdots, 10)$，并计算这个优化位置到信标节点 O 的距离：

$$xz_{Oj} = \sqrt{(optx_j - x_O)^2 + (opty_j - y_O)^2} \tag{5.22}$$

(2) 我们假设的高斯函数为如下形式：

$$wc_{Oj} = \frac{1}{\sqrt{2\pi}\sigma_O} \exp\left[-\frac{(xz_{O_j} - \mu_O)^2}{2\sigma_O^2}\right] \tag{5.23}$$

(3) 根据高斯误差函数的性质，可得

$$\begin{cases} \mu_O = \dfrac{\sum\limits_{j=1}^{10} xz_{Oj}}{10} \\ \sigma_O^2 = \dfrac{\sum\limits_{j=1}^{10} (xz_{Oj} - \mu_O)^2}{10} \end{cases} \tag{5.24}$$

(4) 根据 Matlab 代码，我们可以求得

$$\begin{cases} \mu_O = 0.0167 \\ \sigma_O^2 = 3.9877 \times 10^{-5} \end{cases} \tag{5.25}$$

(5) 根据以上过程，可以绘制出所需的误差函数图像如图 (5) 所示，发现图中对称轴 $x = \mu_O$ 附近的值特别大，远离对称轴的部分值很小，这也符合高斯函数的 3σ 准则，在均值 (即为图上的对称轴) 附近的 3σ 范围内，可能出现的概率值大约占了全部的 99.7%。

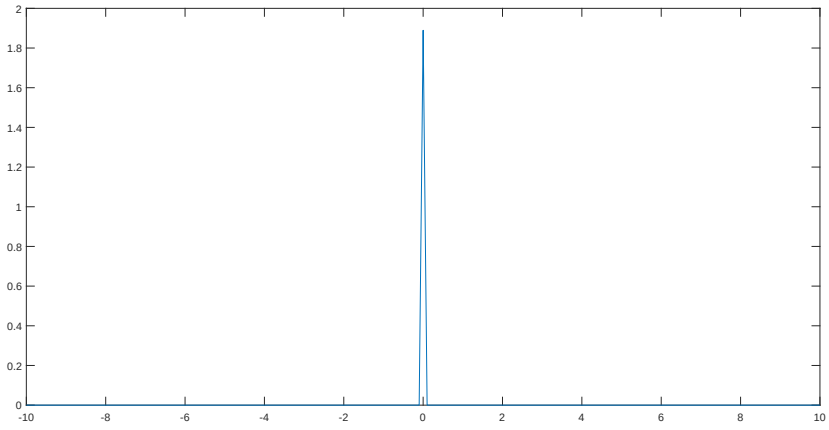

图 5: 四边定位与 Taylor 级数算法所得优化预估位置的高斯误差函数

5.3.2 计算联合概率密度函数并绘制概率云图

为了获知 4 个信标节点任一被毁坏的影响，我们需要知道对平面上任意点 (x,y) 在真实点是 (x_0,y_0) 时的条件联合概率密度函数 $p(x,y)$。为方便叙述，不妨先假设信标节点 C 被毁坏，且真实点为 $(3,4)$，则 $p(x,y)=\phi(r_O,r_A,r_B)$，计算步骤如下：

(1) 已知坐标点 $O(0,0),A(0,5),B(10,0),C(10,5)$，则由第 5.2.2小节的内容可以求得：

$$C=\begin{pmatrix}-20 & 0\\ -20 & 10\end{pmatrix},D=\begin{pmatrix}-100+r_B^2-r_O^2\\ -75+r_B^2-r_A^2\end{pmatrix} \tag{5.26}$$

(2) 由式 (5.26) 可解得

$$\begin{cases}x=\dfrac{1}{20}(100-r_B^2+r_O^2)\\ y=\dfrac{1}{10}(25+r_A^2-r_B^2)\end{cases} \tag{5.27}$$

同时可解得，

$$\begin{cases}r_A=\sqrt{25+r_O^2-10y}\\ r_B=\sqrt{100+r_O^2-20x}\end{cases} \tag{5.28}$$

(3) 根据多元随机变量和多重积分变量替换的知识，可得

$$p(x,y)=\int_{-\infty}^{+\infty}f_O(r_O)f_A(g_A)f_B(g_B)\left\|\begin{matrix}\dfrac{\partial r_O}{\partial r_O} & \dfrac{\partial r_O}{\partial X} & \dfrac{\partial r_O}{\partial Y}\\ \dfrac{\partial g_A}{\partial r_O} & \dfrac{\partial g_A}{\partial X} & \dfrac{\partial g_A}{\partial Y}\\ \dfrac{\partial g_B}{\partial r_O} & \dfrac{\partial g_B}{\partial X} & \dfrac{\partial g_B}{\partial Y}\end{matrix}\right\|\mathrm{d}r_O \tag{5.29}$$

其中，f_O,f_A,f_B 分别为 r_O,r_A,r_B 的概率密度函数，表达式为：

$$\begin{cases}f_O=\dfrac{1}{\sqrt{2\pi}\sigma}\exp\left[-\dfrac{(r_O^2-5)}{2\sigma^2}\right]\\ f_A=\dfrac{1}{\sqrt{2\pi}\sigma}\exp\left[-\dfrac{(r_A^2-\sqrt{10})}{2\sigma^2}\right]\\ f_B=\dfrac{1}{\sqrt{2\pi}\sigma}\exp\left[-\dfrac{(r_B^2-\sqrt{65})}{2\sigma^2}\right]\end{cases} \tag{5.30}$$

(4) 通过求得函数，我们可以绘制出在 $(3,4)$ 的 3σ 领域，即 $[3-3\sigma,3+3\sigma]\times[4-3\sigma,4+3\sigma]$ 的矩形区域内的概率云图。

利用 Matlab，我们可以得到信标节点 C 被毁坏时，若真实点为 $(3,4)$ 时，在 $(3,4)$ 的 3σ 领域内的概率云图：

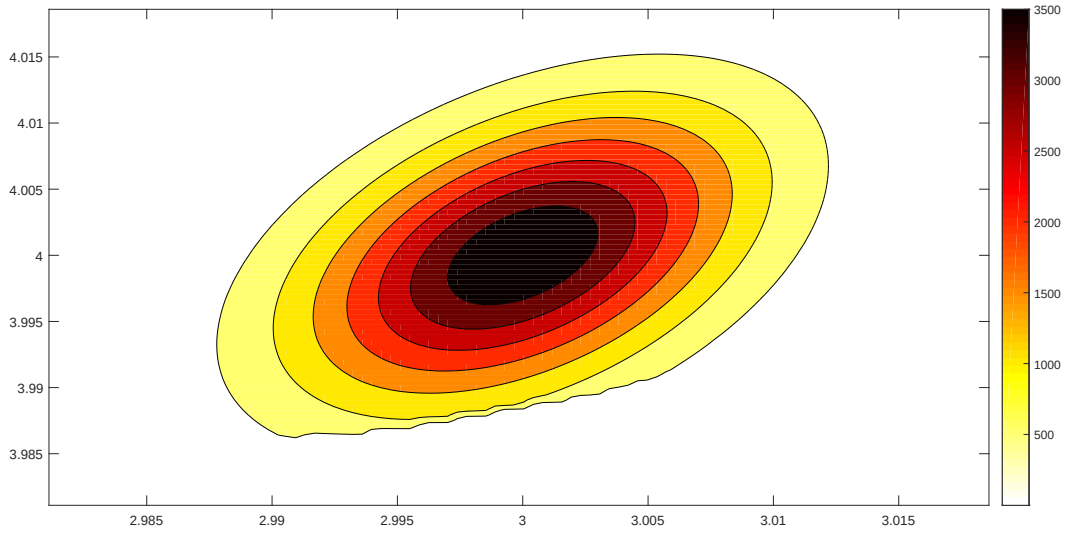

图 6: 信标节点 C 被毁坏 $(3,4)$ 的 3σ 领域内的概率云图

特别说明：显示的图片的颜色块中可能会出现一些白色的细线，这属于 LaTeX 插入 eps 文件时的自带问题，非 Matlab 的绘图问题，阅读时忽略这些细线即可。

5.3.3 计算概率质心

根据微积分中的知识，我们知道平面薄板的质心分布为

$$\bar{x} = \frac{\iint_D x\rho(x,y)\,\mathrm{d}\sigma}{\iint_D \rho(x,y)\,\mathrm{d}\sigma}, \bar{y} = \frac{\iint_D y\rho(x,y)\,\mathrm{d}\sigma}{\iint_D \rho(x,y)\,\mathrm{d}\sigma} \tag{5.31}$$

其中，D 是平面薄板在坐标轴上的区域，$\rho(x,y)$ 是其密度分布。

而在本问题中，由于 Matlab 求 $p(x,y)$ 的局限性，我们只能求出一定量点的概率，即数值解，而无法求出具体的解析式，但当点的数量足够多的时候，我们就可以近似认为

$$\bar{x} \approx \frac{\sum_{i=1}^{n} xp(x,y)}{\sum_{i=1}^{n} p(x,y)}, \bar{y} \approx \frac{\sum_{i=1}^{n} yp(x,y)}{\sum_{i=1}^{n} p(x,y)} \tag{5.32}$$

于是，我们就可以得到在缺失信标节点 C 的情况下，假如真实点是 $(3,4)$ 的条件下，其他 3 个信标节点所测量到的期望点是

$$\bar{x} = 3.000089678303547, \bar{y} = 4.000089678303530 \tag{5.33}$$

这个计算得到的质心点就是缺少信标节点 C 时会产生的误差影响，为方便我们可以用到真实点的距离来衡量影响的大小。

5.3.4 利用题目中表 1 的数据验证影响

验证影响的思路很简单，只需将求 $p(x,y)$ 时使用的点 $(3,4)$ 改成表 1 中所给的 10 个已知节点的位置数据即可，通过同样的步骤，我们可以算得：(由于 Matlab 计算时的局限性与积分下限的可变性，所得数据均为一组一组输入的结果，附录中只提供了上一小节中例子的代码，以下代码省略)

(1) 信标节点 C 被毁坏时，得到的表 1 中 10 个已知节点的质心位置：

表 3: 缺 C 时 10 个已知节点的质心位置

节点标号	1	2	3	4	5
X 坐标	6.43674544	3.80650985	8.119409084	5.328270805	3.508008894
Y 坐标	1.03674544	1.52650985	2.349409084	1.148270805	4.218008894
节点标号	6	7	8	9	10
X 坐标	9.389974781	8.75997637	5.503209666	6.220005029	5.857730799
Y 坐标	0.969974781	1.12997637	0.853209666	1.140005029	2.167730799

所求得的质心位置与表 1 所提供的坐标数据间的距离为

表 4: 缺 C 时 10 个已知节点的质心位置与提供位置的距离

节点标号	1	2	3	4	5
距离	0.004602642	0.023348454	0.000835681	0.002445452	0.00281585
节点标号	6	7	8	9	10
距离	3.56655×10^{-5}	3.34178×10^{-5}	0.004539153	7.11224×10^{-6}	0.017351271

(2) 信标节点 B 被毁坏的情况：

表 5: 缺 B 时 10 个已知节点的质心位置

节点标号	1	2	3	4	5
X 坐标	6.439902073	3.790023834	8.11884251	5.329995557	3.491392449
Y 坐标	1.039902073	1.510023834	2.34884251	1.149995557	4.201392449
节点标号	6	7	8	9	10
X 坐标	9.390023104	8.76002239	5.487268901	6.22007254	5.864834796
Y 坐标	0.970023104	1.13002239	0.837268901	1.14007254	2.174834796

所求得的质心位置与表 1 所提供的坐标数据间的距离为

表 6: 缺 B 时 10 个已知节点的质心位置与提供位置的距离

节点标号	1	2	3	4	5
距离	0.00013849	3.37059×10^{-5}	0.001636938	6.28347×10^{-6}	0.026315051
节点标号	6	7	8	9	10
距离	3.26744×10^{-5}	3.16637×10^{-5}	0.018004493	0.000102587	0.007304702

(3) 信标节点 A 被毁坏的情况：

表 7: 缺 A 时 10 个已知节点的质心位置

节点标号	1	2	3	4	5
X 坐标	6.439998751	3.802942785	8.118976215	5.327742302	3.50998541
Y 坐标	1.039998751	1.522942785	2.348976215	1.147742302	4.21998541
节点标号	6	7	8	9	10
X 坐标	9.381521683	8.754120563	5.500279856	6.221057896	5.866745102
Y 坐标	0.961521683	1.124120563	0.850279856	1.141057896	2.176745102

所求得的质心位置与表 1 所提供的坐标数据间的距离为

表 8: 缺 A 时 10 个已知节点的质心位置与提供位置的距离

节点标号	1	2	3	4	5
距离	1.76571×10^{-6}	0.018303862	0.001447851	0.003192868	2.0633×10^{-5}
节点标号	6	7	8	9	10
距离	0.011990151	0.008314779	0.000395777	0.001496091	0.00460312

(4) 信标节点 O 被毁坏：

表 9: 缺 O 时 10 个已知节点的质心位置

节点标号	1	2	3	4	5
X 坐标	6.429993569	3.786984574	8.119865534	5.329988935	3.501110263
Y 坐标	1.029993569	1.506984574	2.349865534	1.149988935	4.211110263
节点标号	6	7	8	9	10
X 坐标	9.389991417	8.75999397	5.498853407	6.208739066	5.86706835
Y 坐标	0.969991417	1.12999397	0.848853407	1.128739066	2.17706835

所求得的质心位置与表 1 所提供的坐标数据间的距离为

表 10: 缺 O 时 10 个已知节点的质心位置与提供位置的距离

节点标号	1	2	3	4	5
距离	0.014151231	0.004264457	0.000190164	1.56486×10^{-5}	0.012571987
节点标号	6	7	8	9	10
距离	1.21377×10^{-5}	8.52812×10^{-6}	0.001621527	0.015925365	0.00414598

(5) 使用这样的方法，也可以计算 4 个信标节点都没有毁坏的时候的数据：

表 11: 4 个信标节点时 10 个已知节点的质心位置

节点标号	1	2	3	4	5
X 坐标	6.440267169	3.789998452	8.119981994	5.331268545	3.507652315
Y 坐标	1.040267169	1.509998452	2.349981994	1.151268545	4.217652315
节点标号	6	7	8	9	10
X 坐标	9.391479416	8.759993565	5.506309638	6.22284757	5.868752146
Y 坐标	0.971479416	1.129993565	0.856309638	1.14284757	2.178752146

所求得的质心位置与表 1 所提供的坐标数据间的距离为

表 12: 4 个信标节点时 10 个已知节点的质心位置与提供位置的距离

节点标号	1	2	3	4	5
距离	0.000377834	2.18897×10^{-6}	2.54647×10^{-5}	0.001793994	0.003320129
节点标号	6	7	8	9	10
距离	0.00209221	9.10011×10^{-6}	0.008923175	0.004027072	0.001764732

(6) 从上述表 (3)~ 表 (12) 中，可以得到每种情况的平均误差

$$\begin{aligned} avg_C &= 0.00560147, avg_B = 0.005360659, \\ avg_A &= 0.00497669, avg_O = 0.005290702, \\ avg &= 0.00223359 \end{aligned} \tag{5.34}$$

(7) 上方都是定量的分析，同样地通过式 (5.34)，我们可以知道定性地知道：

(a) 相较于 4 个信标节点，无论是缺少哪一个信标节点，其距离误差都会大于 4 个信标节点的距离误差；

(b) 相较于缺少某一个信标节点，信标节点 C 被毁坏对表 1 数据的距离误差影响最大。

5.3.5 调整信标节点位置的方案

根据数学原理知识的推导，我们可以得到：当任意三个信标节点与待测点的连线夹角都成 120° 时，测量的精度最高。这个结论的推导过程如下[8]：

(1) 在没有任何误差的情况下，待测点 P 的真实位置 (x, y) 与 3 个信标节点 $P_i(x_i, y_i)$ 满足方程组

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2 \\ (x - x_3)^2 + (y - y_3)^2 = d_3^2 \end{cases} \tag{5.35}$$

其中，d_i 是待测点的真实位置到信标节点 (x_i, y_i, z_i) 的距离。

(2) 但是，因为测量误差、环境因素的影响，测得的 d_i 会在一个领域 $U(d_i, \varepsilon_i)$ 内。为简化分析，我们不妨假设这些 ε_i 是相等的，均等于 ε。于是，这三个圆就会相交于一个小区域 $\mathcal{S}$，并假设以下符号：

$$\begin{cases} \mathcal{S}_i = \{(x, y) | (d_i - \varepsilon)^2 \le (x - x_i)^2 + (y - y_i)^2 \le (d_i + \varepsilon)^2\} \\ \mathcal{S} = \left\{ (x, y) \left| x \in \bigcap_{i=1}^{3} \mathcal{S}_i, y \in \bigcap_{i=1}^{3} \mathcal{S}_i \right. \right\} \\ \mathcal{T} = \{(x, y) | x^2 + y^2 = \varepsilon^2, \varepsilon > 0\} \end{cases} \tag{5.36}$$

(3) 连接点 P 和 P_i 后形成的直线会与 $\mathcal{T}$ 这个圆形区域交于两个点 $Q_{ij}(j = 1, 2)$，并过点 Q_{ij} 作 $\mathcal{T}$ 区域的切线，最终这 6 条切线会交于 6 个点，形成一个任意形状的六边形 $\tilde{\mathcal{S}}$，如图 (7) 所示。虽然严格来说，$\mathcal{S}$ 的区域的边界应该是曲线，但由于在合适的测量条件下，ε 很小，就可以大致认为 $\mathcal{S}$ 区域的边界是直线，且整块区域就约等于 $\tilde{\mathcal{S}}$。

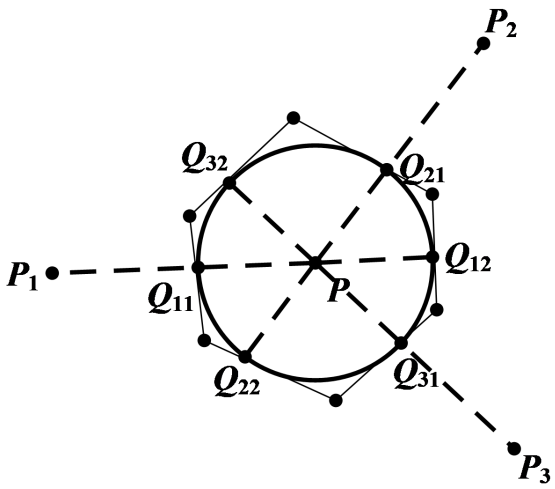

图 7: 定位误差所构成的六边形

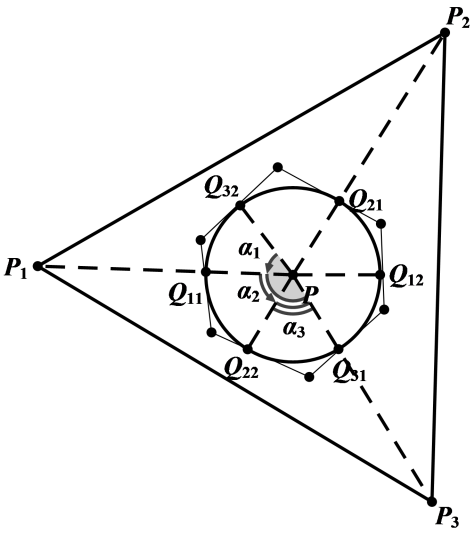

图 8: 最小化六边形面积

(4) 因此，最小化定位误差，即 $\min \varepsilon$ 的问题，可以转变成最小化 $\tilde{S}$ 的区域面积，即 $\min S(\tilde{S})$ 的问题。

(5) 如图 (8)，令 $\angle Q_{32}PQ_{11} = \alpha_1, \angle Q_{11}PQ_{22} = \alpha_2, \angle Q_{22}PQ_{31} = \alpha_3$，则

$$S(\tilde{S}) = 2\varepsilon^2 \left(\tan\frac{\alpha_1}{2} + \tan\frac{\alpha_2}{2} + \tan\frac{\alpha_3}{2}\right) \tag{5.37}$$

其中，$\alpha_1 + \alpha_2 + \alpha_2 = \pi$。

(6) 根据基本不等式 $a + b + c \geq 3\sqrt[3]{abc}$，当且仅当 $a = b = c$ 时取到等号，由此可得

$$\begin{aligned} S(\tilde{S}) &= 2\varepsilon^2 \left(\tan\frac{\alpha_1}{2} + \tan\frac{\alpha_2}{2} + \tan\frac{\alpha_3}{2}\right) \\ &\geq 6\varepsilon^2 \sqrt[3]{\tan\frac{\alpha_1}{2} \cdot \tan\frac{\alpha_2}{2} \cdot \tan\frac{\alpha_3}{2}} \end{aligned} \tag{5.38}$$

当且仅当 $\dfrac{\alpha_1}{2} = \dfrac{\alpha_2}{2} = \dfrac{\alpha_3}{2}$，即 $\alpha_1 = \alpha_2 = \alpha_2 = \dfrac{\pi}{3}$ 时，取到等号。

(7) 当 $\alpha_1 = \alpha_2 = \alpha_2 = \dfrac{\pi}{3}$ 时，三个信标节点与待测点的连线夹角都成 120°，此时定位的误差最小。

因此，只需在测量时保证测量节点与三个信标节点的连线成 120° 即可。

5.4 问题三模型的建立和求解

5.4.1 确定 5 个未知节点的位置

(1) 我们采取 Markov 链模型的想法，先通过问题一的模型求得 5 个位置节点的优化预估位置：

表 13: 5 个未知节点的优化预估位置

节点标号	1	2	3	4	5
X 坐标	1.79127658	6.16187959	4.71723798	7.48346161	6.52971436
Y 坐标	4.29585533	3.93274474	1.70506089	2.72825536	1.17447707

(2) 然后计算 f_P 的概率密度函数在 R_{Pj} 处的概率值 (R_{Pj} 表示第 j 个未知节点到节点 P 的 RSSI 距离，其中 RSSI 公式中的 A 和 n 均取第一问求得值的平均值，即 $A = -58.9650, n = 1.2324$)，则在某状态下的某个点 (x, y) 的状态信息可以表示为[9][10]

$$Q_0 = \prod_{P=1}^{C} \prod_{j\neq i, j=1}^{5} f_P(R_{Pj}) \tag{5.39}$$

其中，$P = 1, 2, 3, 4, O, A, B, C$，数字表示 5 个未知节点。并记

$$E_0 = -\ln Q_0 \tag{5.40}$$

(3) 再对这个状态进行足够多次的微小扰动，扰动的结果就是让某个点 (x, y) 要么不移动，要么让 x, y 均减小一个较小值，要么让 x, y 均增大一个较小值。经过扰动之后，此时在某状态下的某个点 (x, y) 的新的状态信息可以表示为

$$Q_1 = \prod_{P=1}^{C} \prod_{j\neq i, j=1}^{5} f_P(R_{Pj}) \tag{5.41}$$

并记

$$E_1 = -\ln Q_1 \tag{5.42}$$

(4) 是否从老的状态改变成新的状态的判断依据如下：

$$\begin{cases} \text{以概率为 1 进行移动}, & E_0 \geq E_1 \\ \text{以概率为} \exp[-(E_1 - E_0)] \text{进行移动}, & E_0 < E_1 \end{cases} \tag{5.43}$$

(5) 经过足够多次的扰动之后，在平面上密度最高的区域选取最高的 n 个数据取平均或最高的峰值的坐标数据 (如果峰值间差距较小，选用取 n 个点求平均；如果峰值间差距较大，则选择取最高值所在的位置数据)，减去中心坐标就是相对于中心的偏移量 $\Delta\varepsilon$，加上 5 个未知节点的优化预估位置就是在 Markov 链模型下的测量位置。

根据 Matlab 绘图，我们可以得到在未缺失任何一个信标节点的情况下，5 个未知节点的位置数据：

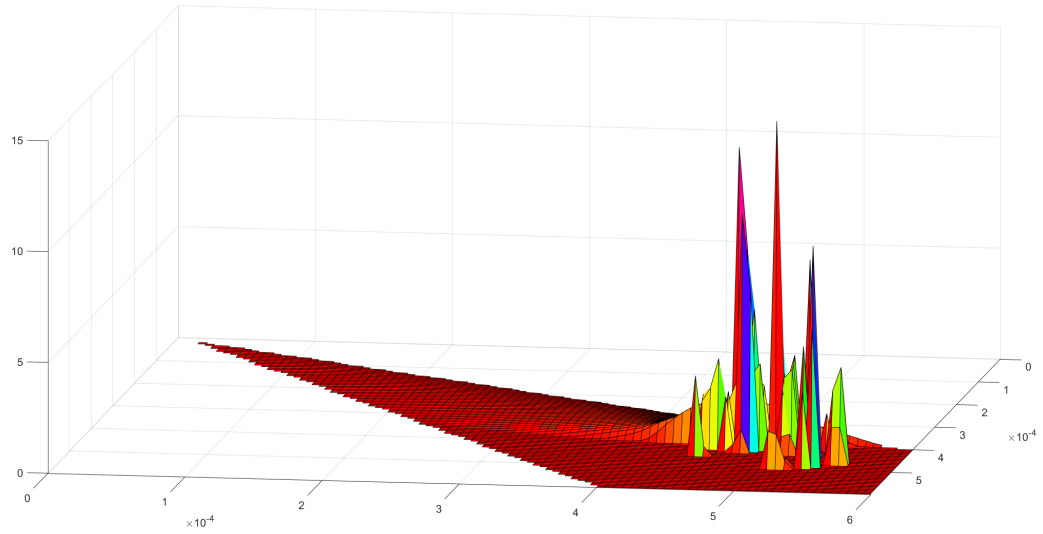

图 9: 未缺失信标节点时，未知节点 1 偏移的区域

因为图 (9) 中的峰值差距较小，选择最高的三个峰值数值，并取平均，可得到三个坐标与取平均后的数据：

$$\begin{cases} x_1 = 4.877 \times 10^{-4}, x_2 = 5.083 \times 10^{-4}, x_3 = 5.083 \times 10^{-4} \\ y_1 = 2.438 \times 10^{-4}, y_2 = 2.644 \times 10^{-4}, y_3 = 2.541 \times 10^{-4} \\ \bar{x} = 5.014 \times 10^{-4}, \bar{y} = 2.541 \times 10^{-4} \end{cases} \tag{5.44}$$

减去中心位置 $(5 \times 10^{-4}, 2.5 \times 10^{-4})$，可得

$$\Delta\varepsilon = (5.014 \times 10^{-4}, 2.541 \times 10^{-4}) - (5 \times 10^{-4}, 2.5 \times 10^{-4}) = (0.014 \times 10^{-4}, 0.041 \times 10^{-4}) \tag{5.45}$$

则在未缺失任何一个信标节点的情况下，第 1 个未知节点的坐标数据为

$$(x_1, y_1) = \left(0.014 \times 10^{-4}, 0.041 \times 2 \times 10^{-4}\right) + (1.79127658, 4.29585533) = (1.79127801, 4.29584893) \tag{5.46}$$

从而可以求得所有未知节点的测量数据：

表 14: 全部信标节点下 5 个未知节点的测量位置

节点标号	1	2	3	4	5
X 坐标	1.79127801	6.16185836	4.71724098	7.48347061	6.52969316
Y 坐标	4.29584893	3.93273367	1.70506389	2.72824931	1.17499307

可以绘制图：

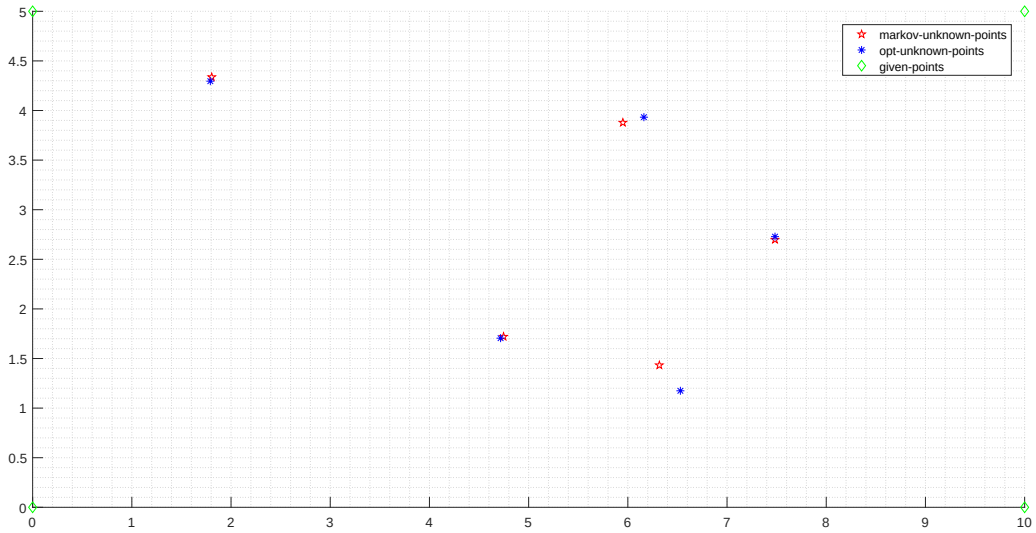

图 10: 未缺失信标节点时，所有未知节点的测量位置

5.4.2 4 个信标节点任一毁坏对定位精度产生的影响

通过上一小节的模型，我们可以求得在缺少某个信标节点时，未知节点的位置数据：

(1) 缺少信标节点 C 时，5 个未知节点的测量位置数据：

表 15: 缺少信标节点 C 时 5 个未知节点的测量位置

节点标号	1	2	3	4	5
X 坐标	1.7912796	6.16190989	4.71726838	7.48346639	6.52971133
Y 坐标	4.29591045	3.93274774	1.70510429	2.72823106	1.1744478

与优化预估位置之间的距离为：

表 16: 缺少信标节点 C 时 5 个未知节点与全部节点的测量位置的距离误差

节点标号	1	2	3	4	5	平均距离误差
距离 ($\times 10^{-5}$ m)	6.154	5.342	4.882	1.873	54.56	14.56

(2) 缺少信标节点 B 的情况：

表 17: 缺少信标节点 B 时 5 个未知节点的测量位置

节点标号	1	2	3	4	5
X 坐标	1.79127658	6.16180079	4.71725918	7.48342221	6.52972041
Y 坐标	4.29583113	3.93274174	1.70502149	2.72830076	1.17448927

与优化预估位置之间的距离为：

表 18: 缺少信标节点 B 时 5 个未知节点与全部节点的测量位置的距离误差

节点标号	1	2	3	4	5	平均距离误差
距离 ($\times 10^{-5}$ m)	1.786	5.813	4.614	7.064	50.45	13.95

(3) 缺少信标节点 A 的情况：

表 19: 缺少信标节点 A 时 5 个未知节点的测量位置

节点标号	1	2	3	4	5
X 坐标	1.79132213	6.16191594	4.71718648	7.48349496	6.52974616
Y 坐标	4.29587953	3.93276594	1.70500339	2.72817966	1.17452402

与优化预估位置之间的距离为：

表 20: 缺少信标节点 A 时 5 个未知节点与全部节点的测量位置的距离误差

节点标号	1	2	3	4	5	平均距离误差
距离 ($\times 10^{-5}$ m)	5.369	6.601	8.143	7.378	47.20	14.94

(4) 缺少信标节点 O 的情况：

表 21: 缺少信标节点 O 时 5 个未知节点的测量位置

节点标号	1	2	3	4	5
X 坐标	1.79124328	6.16183816	4.71725466	7.48352324	6.52971436
Y 坐标	4.29584013	3.93270941	1.70501544	2.72831903	1.17445277

与优化预估位置之间的距离为：

表 22: 缺少信标节点 O 时 5 个未知节点与全部节点的测量位置的距离误差

节点标号	1	2	3	4	5	平均距离误差
距离 ($\times 10^{-5}$ m)	3.583	3.157	5.034	8.735	54.07	14.92

(5) 根据表 (14)~ 表 (22)，我们可以得到定量的数据结论，同时也可以获得定性的结论：

(a) 无论是缺少哪一个信标节点，对测量精度的影响都小于全部信标节点同时进行定位得到的结果；

(b) 在缺少信标节点的时候，缺少信标节点 B 对测量精度的影响最小，缺少信标节点 A 对测量精度的影响最大。

(c) 第 5 个未知节点在 4 个距离误差表中的误差均最大，我们有理由怀疑这个点在表 3 所给的数据中可能出现误差，或者该点处于某个或某些节点信号的盲区中。

六、 模型的评价与推广

6.1 模型的评价

6.1.1 模型的优点

本文问题一选用的对数常态模型、四边定位算法与 Taylor 级数算法相结合方法，首先是简单易懂、不需要很强大的理论知识支撑，并且结合 Matlab 的计算，得到计算结果所需的时间很少，运行速度较快，编程难度也较低。同时通过后续对已知节点的回代检验，能看出所得到的位置数据十分接近真实数据，精度较高、误差较小，模型十分稳定。

问题二从高斯误差函数入手，再计算概率密度函数，绘制概率云图，可视化效果佳，并且脱离原数据，可以计算平面上的任意一点的偏差。调整方案也是从数学原理出发，不需要高深的数学知识，只需要有简单的高中不等式与初中平面几何的知识便能看懂。

问题三借助 Markov 链模型，去思考微小扰动的结果，效果精度较高，同样可视化效果佳。并且人的直观感觉，在有未知节点的相互定位的前提下，问题三缺少信标节点的影响 (距离误差) 应该小于问题二的影响，本文的计算结果也符合我们的预计，增加未知节点间相互定位效果比没有相互定位要好。

6.1.2 模型的缺点

本文问题一的模型虽然具有良好的优点，但是同样也具有一定的缺点，如对于某几个已知节点计算得到的优化预估位置与真实位置相差较远；根据数据查找室内环境的常数 A 和路径损耗因子 n 发现本文中得到的 A, n 与真实值偏差较大[6]，需要建立考虑更多干扰或实际因素的模型来减少误差。

问题二中计算概率密度时对于编程的要求较高，并且经常会出现一个代码运行时间过久的状况，此点需要编程人员进行优化。并且调整方案仅仅作了数学上的证明，当未知节点较少时，可以通过不停调整剩余 3 个信标节点的位置来得到精度较高的位置数据，但是当未知节点数量较多时，一次又一次的调整信标节点的位置很费时间，如何寻找到一个通用普遍的调整方法是本文没有去思考的。

问题三使用 Markov 链模型，入手难度较高，需要一定的关于随机过程的知识，如果有更加简单易懂、高效可行的模型来进行会更加容易让人接受。在对 Markov 链模型进行微小扰动的时候，使用 rand() 函数产生随机数来进行扰动，这使得每次程序运行的结果都有一定的不同，虽然数据对结果的影响很小，但为方便获得统一的结果，代码上还需要进行优化。

6.2 模型的推广

本文使用最多的一个想法就是最小二乘拟合，这个方法在大量数学建模问题中都会涉及。同时问题一种的四边定位算法可以再拓展为多边定位算法，用于更多信标节点的定位问题。Taylor 级数算法在测量未知节点时可以使用，也可以用于已知节点测量的优化。

问题二中的调整方案是偏理论性的，既然可以求得 3 个信标节点时理论上的最佳调整方案，那同样地，也可以尝试去求得更多信标节点时的调整规则。

三个问题的算法都可以进行推广用于计算测量点的位置信息，如果能结合更加精准的优化方案，将是一系列不错的定位模型。

七、 参考文献

[1] 赵帅豪, 王坚. 一种基于低功耗蓝牙的室内定位技术 [J]. 北京测绘,2020,34(2):238-242.

[2] 陶为戈, 朱昳华, 贾子彦. 基于 RSSI 混合滤波和最小二乘参数估计的测距算法 [J]. 传感技术学报,2012,25(12):1748-1753.

[3] 肖婷婷. 基于 RSSI 测距的室内定位算法研究及改进 [D]. 江西: 江西师范大学,2015.

[4] 杜亚江, 高立兵. 基于最小二乘法的 RSSI 测距环境参数修正方案 [J]. 计算机系统应用,2012,21(2):221-224.

[5] 陈金星, 任进. 基于 RSSI 的改进泰勒级数室内定位算法 [J]. 北方工业大学学报,2016,28(3):25-29.

[6] 陈红阳. 基于测距技术的无线传感器网络定位技术研究 [D]. 四川: 西南交通大学,2006.

[7] CSDN. 高斯函数的详细分析 [EB/OL].https://blog.csdn.net/jorg_zhao/article/details/52687448,2016-09-28.

[8] 朱剑, 赵海, 徐久强, 等. 三角形定位算法的误差分析 [J]. 东北大学学报 (自然科学版),2009,30(5):648-651.

[9] 陈石平, 马利滨, 徐伟强. 基于马尔科夫链的北斗行驶记录仪数据存储算法研究 [J]. 全球定位系统,2015(3):42-45.

[10] 蔡文学, 邱珠成, 黄晓宇, 等. 基于 WiFi 指纹的室内轨迹定位模型 [J]. 计算机工程,2015(6):76-82.

八、 附录

附录一：四边定位算法的 Matlab 代码

文件名：sibian.m

```
clear; clc;
% 信标节点及表一已知节点数据录入
O = [0, 0];
A = [0, 5];
B = [10, 0];
C = [10, 5];
X = [6.44, 3.79, 8.12, 5.33, 3.51, 9.39, 8.76, 5.50, 6.22, 5.87];
Y = [1.04, 1.51, 2.35, 1.15, 4.22, 0.97, 1.13, 0.85, 1.14, 2.18];
RO = [-69.0, -66.5, -70.4, -68.0, -68.1, -71.0, -70.6, -68.2, -68.8, -68.8];
RA = [-69.8, -67.7, -70.4, -69.0, -65.8, -71.4, -71.1, -69.3, -69.6, -69.0];
RB = [-66.0, -68.9, -64.9, -67.4, -69.9, -59.7, -61.7, -67.1, -66.3, -67.2];
RC = [-67.9, -69.5, -65.3, -68.6, -69.0, -66.5, -66.5, -68.7, -68.0, -67.6];
% 表二未知节点数据录入
RRO = [-66.8, -70.9, -67.8, -66.4, -70.8, -71.2, -67.2, -62.5, -65.4, -67.4];
RRA = [-66.0, -71.3, -67.3, -63.5, -71.2, -71.7, -68.2, -65.8, -66.2, -67.3];
RRB = [-69.8, -61.2, -68.9, -70.7, -61.0, -56.4, -68.4, -70.7, -69.9, -68.9];
RRC = [-69.5, -66.1, -68.6, -70.3, -66.4, -66.9, -69.1, -71.0, -70.1, -68.9];
% 拟合RSSI公式中的A和路径损耗因子n时用到的系数矩阵
MO = []; MA = []; MB = []; MC = [];

for i = 1:10
    % 表一已知节点到各信标节点的距离
    dO(i) = sqrt((X(i) - O(1))^2 + (Y(i) - O(2))^2);
    dA(i) = sqrt((X(i) - A(1))^2 + (Y(i) - A(2))^2);
```

```
    dB(i) = sqrt((X(i) - B(1))^2 + (Y(i) - B(2))^2);
    dC(i) = sqrt((X(i) - C(1))^2 + (Y(i) - C(2))^2);
    % 建立系数矩阵
    MO(end + 1, :) = [1, -10 * log10(dO(i))];
    MA(end + 1, :) = [1, -10 * log10(dA(i))];
    MB(end + 1, :) = [1, -10 * log10(dB(i))];
    MC(end + 1, :) = [1, -10 * log10(dC(i))];
end

% 利用lsqr函数最小二乘法求解线性方程组得到参数A和n
ZO = lsqr(MO, (RO)');
ZA = lsqr(MA, (RA)');
ZB = lsqr(MB, (RB)');
ZC = lsqr(MC, (RC)');

for i = 1:10
    % 用RSSI公式求出各未知节点到各信标节点的距离
    ddO(i) = 10^((ZO(1) - RRO(i)) / (10 * ZO(2)));
    ddA(i) = 10^((ZA(1) - RRA(i)) / (10 * ZA(2)));
    ddB(i) = 10^((ZB(1) - RRB(i)) / (10 * ZB(2)));
    ddC(i) = 10^((ZC(1) - RRC(i)) / (10 * ZC(2)));
end

R = [O; A; B; C];
% R和T是合并矩阵，接下来循环用到其列向量和矩阵中元素
T = [ddO; ddA; ddB; ddC];
% 求解未知节点坐标的系数矩阵
U = [];
% 求解未知节点坐标的列向量
V = [];
% 用来储存未知节点的坐标信息,每一列是一个坐标
Q = [];

for i = 1:10

    for j = 1:3
        U(end + 1, :) = [R(4, 1) - R(j, 1); R(4, 2) - R(j, 2)];
        V(end + 1, :) = [((T(j, i))^2 - (T(4, i))^2 + (R(4, 1))^2 + (R(4, 2))^2 - (
            R(j, 1))^2 - (R(j, 2))^2) / 2];
    end

    Q(:, end + 1) = lsqr(U, V);
```

```
    U = []; V = [];
end

% 作图
QX = Q(1, :);
QY = Q(2, :);
scatter(QX, QY, '*r')
grid minor
hold on
scatter([0, 0, 10, 10], [0, 5, 0, 5], 'bd')
legend('unknown-points', 'given-points')
axis([0, 10, 0, 5])
```

附录二：Taylor 级数算法的 Matlab 代码

文件名：taile.m

```
clear; clc;
% 信标节点及表一已知节点数据录入
O = [0, 0];
A = [0, 5];
B = [10, 0];
C = [10, 5];
X = [6.44, 3.79, 8.12, 5.33, 3.51, 9.39, 8.76, 5.50, 6.22, 5.87];
Y = [1.04, 1.51, 2.35, 1.15, 4.22, 0.97, 1.13, 0.85, 1.14, 2.18];
RO = [-69.0, -66.5, -70.4, -68.0, -68.1, -71.0, -70.6, -68.2, -68.8, -68.8];
RA = [-69.8, -67.7, -70.4, -69.0, -65.8, -71.4, -71.1, -69.3, -69.6, -69.0];
RB = [-66.0, -68.9, -64.9, -67.4, -69.9, -59.7, -61.7, -67.1, -66.3, -67.2];
RC = [-67.9, -69.5, -65.3, -68.6, -69.0, -66.5, -66.5, -68.7, -68.0, -67.6];
% 表二未知节点数据录入
RRO = [-66.8, -70.9, -67.8, -66.4, -70.8, -71.2, -67.2, -62.5, -65.4, -67.4];
RRA = [-66.0, -71.3, -67.3, -63.5, -71.2, -71.7, -68.2, -65.8, -66.2, -67.3];
RRB = [-69.8, -61.2, -68.9, -70.7, -61.0, -56.4, -68.4, -70.7, -69.9, -68.9];
RRC = [-69.5, -66.1, -68.6, -70.3, -66.4, -66.9, -69.1, -71.0, -70.1, -68.9];
% 拟合RSSI公式中的A和路径损耗因子n时用到的系数矩阵
MO = []; MA = []; MB = []; MC = [];

for i = 1:10
    % 表一已知节点到各信标节点的距离
    dO(i) = sqrt((X(i) - O(1))^2 + (Y(i) - O(2))^2);
    dA(i) = sqrt((X(i) - A(1))^2 + (Y(i) - A(2))^2);
    dB(i) = sqrt((X(i) - B(1))^2 + (Y(i) - B(2))^2);
    dC(i) = sqrt((X(i) - C(1))^2 + (Y(i) - C(2))^2);
    % 建立系数矩阵
```

```
    MO(end + 1, :) = [1, -10 * log10(dO(i))];
    MA(end + 1, :) = [1, -10 * log10(dA(i))];
    MB(end + 1, :) = [1, -10 * log10(dB(i))];
    MC(end + 1, :) = [1, -10 * log10(dC(i))];
end

% 利用lsqr函数最小二乘法求解线性方程组得到参数A和n
ZO = lsqr(MO, (RO)');
ZA = lsqr(MA, (RA)');
ZB = lsqr(MB, (RB)');
ZC = lsqr(MC, (RC)');

% 用RSSI公式求出各未知节点到各信标节点的距离
for i = 1:10
    ddO(i) = 10^((ZO(1) - RRO(i)) / (10 * ZO(2)));
    ddA(i) = 10^((ZA(1) - RRA(i)) / (10 * ZA(2)));
    ddB(i) = 10^((ZB(1) - RRB(i)) / (10 * ZB(2)));
    ddC(i) = 10^((ZC(1) - RRC(i)) / (10 * ZC(2)));
end

R = [O; A; B; C];
% R和T是合并矩阵，接下来循环用到其列向量和矩阵中元素
T = [ddO; ddA; ddB; ddC];
% 求解未知节点坐标的系数矩阵
U = [];
% 求解未知节点坐标的列向量
V = [];
% 用来储存未知节点的坐标信息,每一列是一个坐标
Q = [];

for i = 1:10

    for j = 1:3
        U(end + 1, :) = [R(4, 1) - R(j, 1); R(4, 2) - R(j, 2)];
        V(end + 1, :) = [((T(j, i))^2 - (T(4, i))^2 + (R(4, 1))^2 + (R(4, 2))^2 - (
            R(j, 1))^2 - (R(j, 2))^2) / 2];
    end

    Q(:, end + 1) = lsqr(U, V);
    U = []; V = [];
end

```

```
xt = Q(1, :);
yt = Q(2, :);

% 估计位置到各信标节点的距离
for i = 1:10
    d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
    d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
    d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
    d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
end

delta_rho = [];
delta_rho_true = [];
i = 1;

while true

    K = [(O(1) - xt(i)) / (d_O(i)), (O(2) - yt(i)) / (d_O(i)); (A(1) - xt(i)) / (
        d_A(i)), (A(2) - yt(i)) / (d_A(i)); (B(1) - xt(i)) / (d_B(i)), (B(2) - yt(i
        )) / (d_B(i)); (C(1) - xt(i)) / (d_C(i)), (C(2) - yt(i)) / (d_C(i))];
    D = [d_O(i) - ddO(i); d_A(i) - ddA(i); d_B(i) - ddB(i); d_C(i) - ddC(i)];
    delta_rho(:, i) = lsqr(K, D);

    if abs(delta_rho(1, i) + delta_rho(2, i)) >= 0.01
        xt(i) = xt(i) + delta_rho(1, i);
        yt(i) = yt(i) + delta_rho(2, i);
        d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
        d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
        d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
        d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
    else
        delta_rho_true(:, i) = delta_rho(:, i);
        i = i + 1;
    end

    if i == 11
        break
    end

end

loca = Q + delta_rho_true;
```

```
109
110 QX = Q(1, :);
111 QY = Q(2, :);
112 locaX = loca(1, :);
113 locaY = loca(2, :);
114 scatter(locaX, locaY, 'pr')
115 hold on
116 scatter(QX, QY, '*b')
117 hold on
118 scatter([0, 0, 10, 10], [0, 5, 0, 5], 'gd')
119 grid minor
120 legend('opt-unknown-points', 'unknown-points', 'given-points')
121 axis([0, 10, 0, 5])
```

附录三：计算缺失 C 点的概率云图以及质心的 Matlab 代码

文件名：probability_C_example.m

```
1  clear; clc;
2  % 信标节点及表一已知节点数据录入
3  O = [0, 0]; A = [0, 5]; B = [10, 0]; C = [10, 5];
4  X = [6.44, 3.79, 8.12, 5.33, 3.51, 9.39, 8.76, 5.50, 6.22, 5.87];
5  Y = [1.04, 1.51, 2.35, 1.15, 4.22, 0.97, 1.13, 0.85, 1.14, 2.18];
6  RO = [-69.0, -66.5, -70.4, -68.0, -68.1, -71.0, -70.6, -68.2, -68.8, -68.8];
7  RA = [-69.8, -67.7, -70.4, -69.0, -65.8, -71.4, -71.1, -69.3, -69.6, -69.0];
8  RB = [-66.0, -68.9, -64.9, -67.4, -69.9, -59.7, -61.7, -67.1, -66.3, -67.2];
9  RC = [-67.9, -69.5, -65.3, -68.6, -69.0, -66.5, -66.5, -68.7, -68.0, -67.6];
10 % 拟合RSSI公式中的A和路径损耗因子n时用到的系数矩阵
11 MO = []; MA = []; MB = []; MC = [];
12
13 for i = 1:10
14     % 表一已知节点到各信标节点的距离,这是真实值的距离
15     dO(i) = sqrt((X(i) - O(1))^2 + (Y(i) - O(2))^2);
16     dA(i) = sqrt((X(i) - A(1))^2 + (Y(i) - A(2))^2);
17     dB(i) = sqrt((X(i) - B(1))^2 + (Y(i) - B(2))^2);
18     dC(i) = sqrt((X(i) - C(1))^2 + (Y(i) - C(2))^2);
19     % 建立系数矩阵
20     MO(end + 1, :) = [1, -10 * log10(dO(i))];
21     MA(end + 1, :) = [1, -10 * log10(dA(i))];
22     MB(end + 1, :) = [1, -10 * log10(dB(i))];
23     MC(end + 1, :) = [1, -10 * log10(dC(i))];
24 end
25
26 % 利用lsqr函数最小二乘法求解线性方程组得到参数A和n
```

```
ZO = lsqr(MO, (RO)');
ZA = lsqr(MA, (RA)');
ZB = lsqr(MB, (RB)');
ZC = lsqr(MC, (RC)');

for i = 1:10
    % 用RSSI公式计算已知节点到各信标节点的距离，这是测量值的距离
    yzdO(i) = 10^((ZO(1) - RO(i)) / (10 * ZO(2)));
    yzdA(i) = 10^((ZA(1) - RA(i)) / (10 * ZA(2)));
    yzdB(i) = 10^((ZB(1) - RB(i)) / (10 * ZB(2)));
    yzdC(i) = 10^((ZC(1) - RC(i)) / (10 * ZC(2)));
end

R = [O; A; B; C];
% R和T是合并矩阵，接下来循环用到其列向量和矩阵中元素
TT = [yzdO; yzdA; yzdB; yzdC];
% 验证已知节点坐标的系数矩阵
UU = [];
% 验证已知节点坐标的列向量
VV = [];
% 用来储存验证已知节点的坐标信息,每一列是一个坐标
QQ = [];

for i = 1:10

    for j = 1:3
        UU(end + 1, :) = [R(4, 1) - R(j, 1); R(4, 2) - R(j, 2)];
        VV(end + 1, :) = [((TT(j, i))^2 - (TT(4, i))^2 + (R(4, 1))^2 + (R(4, 2))^2
            - (R(j, 1))^2 - (R(j, 2))^2) / 2];
    end

    QQ(:, end + 1) = lsqr(UU, VV);
    UU = []; VV = [];
end

xt = QQ(1, :);
yt = QQ(2, :);

% 估计位置到各信标节点的距离
for i = 1:10
    d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
    d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
```

```
    d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
    d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
end

delta_rho = [];

delta_rho = [];
delta_rho_true = [];
i = 1;

while true

    K = [(O(1) - xt(i)) / (d_O(i)), (O(2) - yt(i)) / (d_O(i)); (A(1) - xt(i)) / (
        d_A(i)), (A(2) - yt(i)) / (d_A(i)); (B(1) - xt(i)) / (d_B(i)), (B(2) - yt(i
        )) / (d_B(i)); (C(1) - xt(i)) / (d_C(i)), (C(2) - yt(i)) / (d_C(i))];
    D = [d_O(i) - yzdO(i); d_A(i) - yzdA(i); d_B(i) - yzdB(i); d_C(i) - yzdC(i)];
    delta_rho(:, i) = lsqr(K, D);

    if abs(delta_rho(1, i) + delta_rho(2, i)) >= 0.01
        xt(i) = xt(i) + delta_rho(1, i);
        yt(i) = yt(i) + delta_rho(2, i);
        d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
        d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
        d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
        d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
    else
        delta_rho_true(:, i) = delta_rho(:, i);
        i = i + 1;
    end

    if i == 11
        break
    end

end

% 用泰勒展开修正过后的修正估计位置坐标
loca = QQ + delta_rho;

for i = 1:10
    xz_dO(i) = sqrt((loca(1, i) - O(1))^2 + (loca(2, i) - O(2))^2);
    xz_dA(i) = sqrt((loca(1, i) - A(1))^2 + (loca(2, i) - A(2))^2);
```

```
    xz_dB(i) = sqrt((loca(1, i) - B(1))^2 + (loca(2, i) - B(2))^2);
    xz_dC(i) = sqrt((loca(1, i) - C(1))^2 + (loca(2, i) - C(2))^2);
    wc_dO(i) = abs(xz_dO(i) - dO(i));
    wc_dA(i) = abs(xz_dA(i) - dA(i));
    wc_dB(i) = abs(xz_dB(i) - dB(i));
    wc_dC(i) = abs(xz_dC(i) - dC(i));
end

% 高斯误差均值
gauss_mean = sum(wc_dO) / 10;
gauss_sum = 0;

for i = 1:10
    gauss_sum = gauss_sum + (wc_dO(i) - gauss_mean)^2;
end

% 高斯误差系数sigma
gauss_xigma2 = gauss_sum / 10;
% 高斯误差系数a = 1 / sqrt(2 * pi * xigma2)
gauss_a = 1 / sqrt(2 * pi * gauss_xigma2);

% 高斯误差函数
x = [-10:0.1:10];
y = gauss_a * exp(-(x - gauss_mean).^2 / (2 * gauss_xigma2));
% plot(x, y)

% 真实点为(3,4)，考虑C点损坏
% 其余情况下需改变g2，g3，jf_j，f以及integral的积分下限，以下略去
clear x y
syms x y r1
g2 = sqrt(r1^2 - 10 * y + 25);
g3 = sqrt(r1^2 - 20 * x + 100);
jf_j = abs(det(jacobian([r1, g2, g3], [r1, x, y])));
jf_z = [];
k = 1;

for i = 3 - 0.0189:0.0005:3 + 0.0189

    for j = 4 - 0.0189:0.0005:4 + 0.0189
        jf_g2 = subs(g2, y, j);
        jf_g3 = subs(g3, x, i);
        jf_jj = subs(jf_j, {x, y}, {i, j});
```

```
        f = ((1 / sqrt(2 * pi * gauss_xigma2)))^3 * exp(-(r1 - 5)^2 / (2 *
            gauss_xigma2)) * exp(-(jf_g2 - sqrt(10))^2 / (2 * gauss_xigma2)) * exp
            (-(jf_g3 - sqrt(65))^2 / (2 * gauss_xigma2)) * jf_jj;
        jf_z(k) = integral(matlabFunction(f), sqrt(10 * j - 25), inf);
        k = k + 1;
    end

end

jf = [];

for i = 3 - 0.0189:0.0005:3 + 0.0189

    for j = 4 - 0.0189:0.0005:4 + 0.0189
        jf(end + 1, :) = [i, j];
    end

end

[X, Y, Z] = griddata(jf(:, 1), jf(:, 2), (jf_z)', linspace(min(jf(:, 1)), max(jf
    (:, 1)))', linspace(min(jf(:, 2)), max(jf(:, 2))), 'natural');
contourf(X, Y, Z)
colorbar;
colormap(flipud(hot));

% 计算质心
xoab_ba_fm = 0;

for i = 1:5776
    xoab_ba_fm = xoab_ba_fm + jf_z(i);
end

xoab_ba_fz = 0; k = 1;

for i = 3 - 0.0189:0.0005:3 + 0.0189

    for j = 1:76
        xoab_ba_fz = xoab_ba_fz + i * jf_z(k);
        k = k + 1;
    end

end
```

```
189
190 xoab_ba = xoab_ba_fz / xoab_ba_fm;
191
192 yoab_ba_fm = 0;
193
194 for i = 1:5776
195     yoab_ba_fm = yoab_ba_fm + jf_z(i);
196 end
197
198 yoab_ba_fz = 0; k = 1;
199
200 for i = 4 - 0.0189:0.0005:4 + 0.0189
201
202     for j = 1:76
203         yoab_ba_fz = yoab_ba_fz + i * jf_z(k);
204         k = k + 1;
205     end
206
207 end
208
209 y_ba = yoab_ba_fz / yoab_ba_fm;
```

附录四：利用问题一的模型计算 5 个未知节点的优化预估位置的 Matlab 代码

文件名：taile_T3.m

```
1  clear; clc;
2  % 信标节点及表一已知节点数据录入
3  O = [0, 0];
4  A = [0, 5];
5  B = [10, 0];
6  C = [10, 5];
7  X = [6.44, 3.79, 8.12, 5.33, 3.51, 9.39, 8.76, 5.50, 6.22, 5.87];
8  Y = [1.04, 1.51, 2.35, 1.15, 4.22, 0.97, 1.13, 0.85, 1.14, 2.18];
9  RO = [-69.0, -66.5, -70.4, -68.0, -68.1, -71.0, -70.6, -68.2, -68.8, -68.8];
10 RA = [-69.8, -67.7, -70.4, -69.0, -65.8, -71.4, -71.1, -69.3, -69.6, -69.0];
11 RB = [-66.0, -68.9, -64.9, -67.4, -69.9, -59.7, -61.7, -67.1, -66.3, -67.2];
12 RC = [-67.9, -69.5, -65.3, -68.6, -69.0, -66.5, -66.5, -68.7, -68.0, -67.6];
13 % 表三未知节点数据录入
14 T3RO = [-67.2, -69.1, -67.0, -69.6, -68.6];
15 T3RA = [-59.1, -68.4, -67.9, -70.0, -69.5];
16 T3RB = [-70.8, -67.6, -67.7, -66.1, -64.4];
17 T3RC = [-70.2, -64.8, -68.4, -64.1, -67.0];
18 % 拟合RSSI公式中的A和路径损耗因子n时用到的系数矩阵
```

```
MO = []; MA = []; MB = []; MC = [];

for i = 1:10
    % 表一已知节点到各信标节点的距离
    dO(i) = sqrt((X(i) - O(1))^2 + (Y(i) - O(2))^2);
    dA(i) = sqrt((X(i) - A(1))^2 + (Y(i) - A(2))^2);
    dB(i) = sqrt((X(i) - B(1))^2 + (Y(i) - B(2))^2);
    dC(i) = sqrt((X(i) - C(1))^2 + (Y(i) - C(2))^2);
    % 建立系数矩阵
    MO(end + 1, :) = [1, -10 * log10(dO(i))];
    MA(end + 1, :) = [1, -10 * log10(dA(i))];
    MB(end + 1, :) = [1, -10 * log10(dB(i))];
    MC(end + 1, :) = [1, -10 * log10(dC(i))];
end

% 利用lsqr函数最小二乘法求解线性方程组得到参数A和n
ZO = lsqr(MO, (RO)');
ZA = lsqr(MA, (RA)');
ZB = lsqr(MB, (RB)');
ZC = lsqr(MC, (RC)');

% 用RSSI公式求出各未知节点到各信标节点的距离
for i = 1:5
    ddO(i) = 10^((ZO(1) - T3RO(i)) / (10 * ZO(2)));
    ddA(i) = 10^((ZA(1) - T3RA(i)) / (10 * ZA(2)));
    ddB(i) = 10^((ZB(1) - T3RB(i)) / (10 * ZB(2)));
    ddC(i) = 10^((ZC(1) - T3RC(i)) / (10 * ZC(2)));
end

R = [O; A; B; C];
% R和T是合并矩阵，接下来循环用到其列向量和矩阵中元素
T = [ddO; ddA; ddB; ddC];
% 求解未知节点坐标的系数矩阵
U = [];
% 求解未知节点坐标的列向量
V = [];
% 用来储存未知节点的坐标信息,每一列是一个坐标
Q = [];

for i = 1:5

    for j = 1:3
```

```
        U(end + 1, :) = [R(4, 1) - R(j, 1); R(4, 2) - R(j, 2)];
        V(end + 1, :) = [((T(j, i))^2 - (T(4, i))^2 + (R(4, 1))^2 + (R(4, 2))^2 - (
            R(j, 1))^2 - (R(j, 2))^2) / 2];
    end

    Q(:, end + 1) = lsqr(U, V);
    U = []; V = [];
end

xt = Q(1, :);
yt = Q(2, :);

% 估计位置到各信标节点的距离
for i = 1:5
    d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
    d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
    d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
    d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
end

delta_rho = [];
delta_rho_true = [];
i = 1;

while true

    K = [(O(1) - xt(i)) / (d_O(i)), (O(2) - yt(i)) / (d_O(i)); (A(1) - xt(i)) / (
        d_A(i)), (A(2) - yt(i)) / (d_A(i)); (B(1) - xt(i)) / (d_B(i)), (B(2) - yt(i
        )) / (d_B(i)); (C(1) - xt(i)) / (d_C(i)), (C(2) - yt(i)) / (d_C(i))];
    D = [d_O(i) - ddO(i); d_A(i) - ddA(i); d_B(i) - ddB(i); d_C(i) - ddC(i)];
    delta_rho(:, i) = lsqr(K, D);

    if abs(delta_rho(1, i) + delta_rho(2, i)) >= 0.01
        xt(i) = xt(i) + delta_rho(1, i);
        yt(i) = yt(i) + delta_rho(2, i);
        d_O(i) = sqrt((O(1) - xt(i))^2 + (O(2) - yt(i))^2);
        d_A(i) = sqrt((A(1) - xt(i))^2 + (A(2) - yt(i))^2);
        d_B(i) = sqrt((B(1) - xt(i))^2 + (B(2) - yt(i))^2);
        d_C(i) = sqrt((C(1) - xt(i))^2 + (C(2) - yt(i))^2);
    else
        delta_rho_true(:, i) = delta_rho(:, i);
        i = i + 1;
```

```
    end

    if i == 6
        break
    end

end

loca = Q + delta_rho_true;
```

附录五：用于 Markov 模型的高斯函数的 Matlab 代码
文件名：function_Gauss.m

```
function Gauss = function_Gauss(x, mu, sigma)
    % mu是均值，sigma^2是方差，输出是概率密度函数在x点的值
    Gauss = (1 / sqrt(2 * pi * sigma^2) * exp(-(x - mu)^2 / (2 * sigma^2)));
end
```

附录六：用于 Markov 模型的计算 RSSI 距离的 Matlab 代码
文件名：RSSI_distance.m

```
function distance = RSSI_distance(X)
    % A = -58.9650, n = 1.2324
    distance = 10^((-58.9650 - X) / (10 * 1.2324));
end
```

附录七：用于 Markov 模型的所有信标节点的概率密度的 Matlab 代码
文件名：function_P_all.m

```
function P = function_P(X)
    % 计算概率密度P
    % X是十维特征向量
    sigma = 0.0063;
    P1_2 = function_Gauss(RSSI_distance(-67.4), sqrt((X(1) - X(2))^2 + (X(6) - X
        (7))^2), sigma);
    P1_3 = function_Gauss(RSSI_distance(-67.3), sqrt((X(1) - X(3))^2 + (X(6) - X
        (8))^2), sigma);
    P1_4 = function_Gauss(RSSI_distance(-69.2), sqrt((X(1) - X(4))^2 + (X(6) - X
        (9))^2), sigma);
    P1_5 = function_Gauss(RSSI_distance(-68.9), sqrt((X(1) - X(5))^2 + (X(6) - X
        (10))^2), sigma);
    P1_6 = function_Gauss(RSSI_distance(-67.2), sqrt((X(1) - 0)^2 + (X(6) - 0)^2),
         sigma);
    P1_7 = function_Gauss(RSSI_distance(-59.1), sqrt((X(1) - 0)^2 + (X(6) - 5)^2),
         sigma);
```

```
    P1_8 = function_Gauss(RSSI_distance(-70.8), sqrt((X(1) - 10)^2 + (X(6) - 0)^2)
        , sigma);
    P1_9 = function_Gauss(RSSI_distance(-70.2), sqrt((X(1) - 10)^2 + (X(6) - 5)^2)
        , sigma);

    P2_1 = function_Gauss(RSSI_distance(-68.0), sqrt((X(2) - X(1))^2 + (X(7) - X
        (6))^2), sigma);
    P2_3 = function_Gauss(RSSI_distance(-64.4), sqrt((X(2) - X(3))^2 + (X(7) - X
        (8))^2), sigma);
    P2_4 = function_Gauss(RSSI_distance(-60.7), sqrt((X(2) - X(4))^2 + (X(7) - X
        (9))^2), sigma);
    P2_5 = function_Gauss(RSSI_distance(-65.6), sqrt((X(2) - X(5))^2 + (X(7) - X
        (10))^2), sigma);
    P2_6 = function_Gauss(RSSI_distance(-69.1), sqrt((X(2) - 0)^2 + (X(7) - 0)^2),
         sigma);
    P2_7 = function_Gauss(RSSI_distance(-68.4), sqrt((X(2) - 0)^2 + (X(7) - 5)^2),
         sigma);
    P2_8 = function_Gauss(RSSI_distance(-67.6), sqrt((X(2) - 10)^2 + (X(7) - 0)^2)
        , sigma);
    P2_9 = function_Gauss(RSSI_distance(-64.8), sqrt((X(2) - 10)^2 + (X(7) - 5)^2)
        , sigma);

    P3_1 = function_Gauss(RSSI_distance(-66.7), sqrt((X(3) - X(1))^2 + (X(8) - X
        (6))^2), sigma);
    P3_2 = function_Gauss(RSSI_distance(-64.7), sqrt((X(3) - X(2))^2 + (X(8) - X
        (7))^2), sigma);
    P3_4 = function_Gauss(RSSI_distance(-64.8), sqrt((X(3) - X(4))^2 + (X(8) - X
        (9))^2), sigma);
    P3_5 = function_Gauss(RSSI_distance(-62.7), sqrt((X(3) - X(5))^2 + (X(8) - X
        (10))^2), sigma);
    P3_6 = function_Gauss(RSSI_distance(-67.0), sqrt((X(3) - 0)^2 + (X(8) - 0)^2),
         sigma);
    P3_7 = function_Gauss(RSSI_distance(-67.9), sqrt((X(3) - 0)^2 + (X(8) - 5)^2),
         sigma);
    P3_8 = function_Gauss(RSSI_distance(-67.7), sqrt((X(3) - 10)^2 + (X(8) - 0)^2)
        , sigma);
    P3_9 = function_Gauss(RSSI_distance(-68.4), sqrt((X(3) - 10)^2 + (X(8) - 5)^2)
        , sigma);

    P4_1 = function_Gauss(RSSI_distance(-68.8), sqrt((X(4) - X(1))^2 + (X(9) - X
        (6))^2), sigma);
    P4_2 = function_Gauss(RSSI_distance(-61.0), sqrt((X(4) - X(2))^2 + (X(9) - X
```

```
        (7))^2), sigma);
    P4_3 = function_Gauss(RSSI_distance(-64.7), sqrt((X(4) - X(3))^2 + (X(9) - X
        (9))^2), sigma);
    P4_5 = function_Gauss(RSSI_distance(-63.9), sqrt((X(4) - X(5))^2 + (X(9) - X
        (10))^2), sigma);
    P4_6 = function_Gauss(RSSI_distance(-69.6), sqrt((X(4) - 0)^2 + (X(9) - 0)^2),
         sigma);
    P4_7 = function_Gauss(RSSI_distance(-70.0), sqrt((X(4) - 0)^2 + (X(9) - 5)^2),
         sigma);
    P4_8 = function_Gauss(RSSI_distance(-66.1), sqrt((X(4) - 10)^2 + (X(9) - 0)^2)
        , sigma);
    P4_9 = function_Gauss(RSSI_distance(-64.1), sqrt((X(4) - 10)^2 + (X(9) - 5)^2)
        , sigma);

    P5_1 = function_Gauss(RSSI_distance(-69.2), sqrt((X(5) - X(1))^2 + (X(10) - X
        (6))^2), sigma);
    P5_2 = function_Gauss(RSSI_distance(-65.3), sqrt((X(5) - X(2))^2 + (X(10) - X
        (7))^2), sigma);
    P5_3 = function_Gauss(RSSI_distance(-62.6), sqrt((X(5) - X(4))^2 + (X(10) - X
        (9))^2), sigma);
    P5_4 = function_Gauss(RSSI_distance(-63.3), sqrt((X(5) - X(5))^2 + (X(10) - X
        (10))^2), sigma);
    P5_6 = function_Gauss(RSSI_distance(-68.6), sqrt((X(5) - 0)^2 + (X(10) - 0)^2)
        , sigma);
    P5_7 = function_Gauss(RSSI_distance(-69.5), sqrt((X(5) - 0)^2 + (X(10) - 5)^2)
        , sigma);
    P5_8 = function_Gauss(RSSI_distance(-64.4), sqrt((X(5) - 10)^2 + (X(10) - 0)
        ^2), sigma);
    P5_9 = function_Gauss(RSSI_distance(-67.0), sqrt((X(5) - 10)^2 + (X(10) - 5)
        ^2), sigma);

    P = P1_2 * P1_3 * P1_4 * P1_5 * P1_6 * P1_7 * P1_8 * P1_9 * P2_1 * P2_3 * P2_4
         * P2_5 * P2_6 * P2_7 * P2_8 * P2_9 * P3_1 * P3_2 * P3_4 * P3_5 * P3_6 *
        P3_7 * P3_8 * P3_9 * P4_1 * P4_2 * P4_3 * P4_5 * P4_6 * P4_7 * P4_8 * P4_9
        * P5_1 * P5_2 * P5_3 * P5_4 * P5_6 * P5_7 * P5_8 * P5_9;
end
```

附录八：用于 Markov 模型的扰动函数的 Matlab 代码

文件名：disturb.m

```
function new = disturb(X)
    % -58.9650 1.2324
    for n = 1:10
```

```
        xx = rand();

        if ((xx >= 0 && xx <= 1/3))
            X(n) = X(n) - 0.0000001;
        elseif ((xx > 1/3) && (xx <= 2/3))
            X(n) = X(n);
        elseif ((xx > 2/3) && (xx <= 1))
            X(n) = X(n) + 0.0000001;
        end

    end

    new = X;
end
```

附录九：计算问题三偏移量的 Matlab 代码

文件名：main_all.m

```
clear; clc;
% 初值
X0 = [1.7913 6.1619 4.7172 7.4835 6.5297 4.2959 3.9327 1.7051 2.7283 1.1745];
X = X0;
% 出现次数
N_1 = zeros(10000, 10000);
N_2 = zeros(10000, 10000);
N_3 = zeros(10000, 10000);
N_4 = zeros(10000, 10000);
N_5 = zeros(10000, 10000);

for i = 1:1000000
    % 扰动得到新的X
    new_X = disturb(X);
    E_before = -log(function_P(X));
    E_after = -log(function_P(new_X));
    % X更新
    if (E_after <= E_before)
        X = new_X;
    end

    if (E_after > E_before)
        xx = rand();
        % X一定概率更新
        if (xx < exp(-(E_after - E_before)))
```

```
            X = new_X;
        end

    end

    % 加5000，使得平均值在5000处
    X_1 = round((X(1) - X0(1)) / 0.0000001 + 5000);
    X_2 = round((X(2) - X0(2)) / 0.0000001 + 5000);
    X_3 = round((X(3) - X0(3)) / 0.0000001 + 5000);
    X_4 = round((X(4) - X0(4)) / 0.0000001 + 5000);
    X_5 = round((X(5) - X0(5)) / 0.0000001 + 5000);
    Y_1 = round((X(6) - X0(6)) / 0.0000001 + 5000);
    Y_2 = round((X(7) - X0(7)) / 0.0000001 + 5000);
    Y_3 = round((X(8) - X0(8)) / 0.0000001 + 5000);
    Y_4 = round((X(9) - X0(9)) / 0.0000001 + 5000);
    Y_5 = round((X(10) - X0(10)) / 0.0000001 + 5000);
    N_1(X_1, Y_1) = N_1(X_1, Y_1) + 1;
    N_2(X_2, Y_2) = N_2(X_2, Y_2) + 1;
    N_3(X_3, Y_3) = N_3(X_3, Y_3) + 1;
    N_4(X_4, Y_4) = N_4(X_4, Y_4) + 1;
    N_5(X_5, Y_5) = N_5(X_5, Y_5) + 1;
end

N = zeros(4004002, 3);
k = 1;

for i = 4000:6000

    for j = 4000:6000
        N(k, :) = [i * 0.001, j * 0.0005, N_1(i, j)];
        k = k + 1;
    end

end

% 这里以画N_1的图为基准，其他的图只需略加修改即可
[X, Y, Z] = griddata(N(:, 1), N(:, 2), N(:, 3), linspace(min(N(:, 1)), max(N(:, 1)
    ))', linspace(min(N(:, 2)), max(N(:, 2))), 'natural');
surf(X, Y, Z);
colormap(hsv)
```

附录九：绘制问题三全部信标节点时未知节点位置的 Matlab 代码

文件名：huatuT3all.m

```
clear; clc;
x0 = [1.7913, 6.1619, 4.7172, 7.4835, 6.5297];
y0 = [4.2959, 3.9327, 1.7051, 2.7283, 1.1745];
x1 = [1.8053, 5.9499, 4.7472, 7.4835, 6.3177];
y1 = [4.3369, 3.8777, 1.7201, 2.6980, 1.4325];
O = [0, 0];
A = [0, 5];
B = [10, 0];
C = [10, 5];

scatter(x1, y1, 'pr')
hold on
scatter(x0, y0, 'b*')
hold on
scatter([0, 0, 10, 10], [0, 5, 0, 5], 'bd')
grid minor
legend('markov-unknown-points', 'opt-unknown-points', 'given-points')
axis([0, 10, 0, 5])
```

附录十：问题三计算信标节点 C 缺失时概率的 Matlab 代码

文件名：function_P_cNone.m

```
function P = function_P_cNone(X)
    % 计算概率密度P
    % X是十维特征向量
    sigma = 0.0063;
    P1_2 = function_Gauss(RSSI_distance(-67.4), sqrt((X(1) - X(2))^2 + (X(6) - X
        (7))^2), sigma);
    P1_3 = function_Gauss(RSSI_distance(-67.3), sqrt((X(1) - X(3))^2 + (X(6) - X
        (8))^2), sigma);
    P1_4 = function_Gauss(RSSI_distance(-69.2), sqrt((X(1) - X(4))^2 + (X(6) - X
        (9))^2), sigma);
    P1_5 = function_Gauss(RSSI_distance(-68.9), sqrt((X(1) - X(5))^2 + (X(6) - X
        (10))^2), sigma);
    P1_6 = function_Gauss(RSSI_distance(-67.2), sqrt((X(1) - 0)^2 + (X(6) - 0)^2),
         sigma);
    P1_7 = function_Gauss(RSSI_distance(-59.1), sqrt((X(1) - 0)^2 + (X(6) - 5)^2),
         sigma);
    P1_8 = function_Gauss(RSSI_distance(-70.8), sqrt((X(1) - 10)^2 + (X(6) - 0)^2)
        , sigma);

```

```
P2_1 = function_Gauss(RSSI_distance(-68.0), sqrt((X(2) - X(1))^2 + (X(7) - X(6))^2), sigma);
P2_3 = function_Gauss(RSSI_distance(-64.4), sqrt((X(2) - X(3))^2 + (X(7) - X(8))^2), sigma);
P2_4 = function_Gauss(RSSI_distance(-60.7), sqrt((X(2) - X(4))^2 + (X(7) - X(9))^2), sigma);
P2_5 = function_Gauss(RSSI_distance(-65.6), sqrt((X(2) - X(5))^2 + (X(7) - X(10))^2), sigma);
P2_6 = function_Gauss(RSSI_distance(-69.1), sqrt((X(2) - 0)^2 + (X(7) - 0)^2), sigma);
P2_7 = function_Gauss(RSSI_distance(-68.4), sqrt((X(2) - 0)^2 + (X(7) - 5)^2), sigma);
P2_8 = function_Gauss(RSSI_distance(-67.6), sqrt((X(2) - 10)^2 + (X(7) - 0)^2), sigma);

P3_1 = function_Gauss(RSSI_distance(-66.7), sqrt((X(3) - X(1))^2 + (X(8) - X(6))^2), sigma);
P3_2 = function_Gauss(RSSI_distance(-64.7), sqrt((X(3) - X(2))^2 + (X(8) - X(7))^2), sigma);
P3_4 = function_Gauss(RSSI_distance(-64.8), sqrt((X(3) - X(4))^2 + (X(8) - X(9))^2), sigma);
P3_5 = function_Gauss(RSSI_distance(-62.7), sqrt((X(3) - X(5))^2 + (X(8) - X(10))^2), sigma);
P3_6 = function_Gauss(RSSI_distance(-67.0), sqrt((X(3) - 0)^2 + (X(8) - 0)^2), sigma);
P3_7 = function_Gauss(RSSI_distance(-67.9), sqrt((X(3) - 0)^2 + (X(8) - 5)^2), sigma);
P3_8 = function_Gauss(RSSI_distance(-67.7), sqrt((X(3) - 10)^2 + (X(8) - 0)^2), sigma);

P4_1 = function_Gauss(RSSI_distance(-68.8), sqrt((X(4) - X(1))^2 + (X(9) - X(6))^2), sigma);
P4_2 = function_Gauss(RSSI_distance(-61.0), sqrt((X(4) - X(2))^2 + (X(9) - X(7))^2), sigma);
P4_3 = function_Gauss(RSSI_distance(-64.7), sqrt((X(4) - X(3))^2 + (X(9) - X(9))^2), sigma);
P4_5 = function_Gauss(RSSI_distance(-63.9), sqrt((X(4) - X(5))^2 + (X(9) - X(10))^2), sigma);
P4_6 = function_Gauss(RSSI_distance(-69.6), sqrt((X(4) - 0)^2 + (X(9) - 0)^2), sigma);
P4_7 = function_Gauss(RSSI_distance(-70.0), sqrt((X(4) - 0)^2 + (X(9) - 5)^2), sigma);
```

```
    P4_8 = function_Gauss(RSSI_distance(-66.1), sqrt((X(4) - 10)^2 + (X(9) - 0)^2)
        , sigma);

    P5_1 = function_Gauss(RSSI_distance(-69.2), sqrt((X(5) - X(1))^2 + (X(10) - X
        (6))^2), sigma);
    P5_2 = function_Gauss(RSSI_distance(-65.3), sqrt((X(5) - X(2))^2 + (X(10) - X
        (7))^2), sigma);
    P5_3 = function_Gauss(RSSI_distance(-62.6), sqrt((X(5) - X(4))^2 + (X(10) - X
        (9))^2), sigma);
    P5_4 = function_Gauss(RSSI_distance(-63.3), sqrt((X(5) - X(5))^2 + (X(10) - X
        (10))^2), sigma);
    P5_6 = function_Gauss(RSSI_distance(-68.6), sqrt((X(5) - 0)^2 + (X(10) - 0)^2)
        , sigma);
    P5_7 = function_Gauss(RSSI_distance(-69.5), sqrt((X(5) - 0)^2 + (X(10) - 5)^2)
        , sigma);
    P5_8 = function_Gauss(RSSI_distance(-64.4), sqrt((X(5) - 10)^2 + (X(10) - 0)
        ^2), sigma);

    P = P1_2 * P1_3 * P1_4 * P1_5 * P1_6 * P1_7 * P1_8 * P2_1 * P2_3 * P2_4 * P2_5
         * P2_6 * P2_7 * P2_8 * P3_1 * P3_2 * P3_4 * P3_5 * P3_6 * P3_7 * P3_8 *
        P4_1 * P4_2 * P4_3 * P4_5 * P4_6 * P4_7 * P4_8 * P5_1 * P5_2 * P5_3 * P5_4
        * P5_6 * P5_7 * P5_8;
end
```

附录十一：问题三计算信标节点 C 缺失时未知节点坐标的 Matlab 代码

文件名：main_cNone.m

```
clc; clear;
X0 = [1.7913 6.1619 4.7172 7.4835 6.5297 4.2959 3.9327 1.7051 2.7283 1.1745]; %初
    值
X = X0;
N_1 = zeros(10000, 10000); %出现次数
N_2 = zeros(10000, 10000); %出现次数
N_3 = zeros(10000, 10000); %出现次数
N_4 = zeros(10000, 10000); %出现次数
N_5 = zeros(10000, 10000); %出现次数

for i = 1:1000000
    new_X = disturb(X); %扰动得到新的X
    E_before = -log(function_P_cNone(X));
    E_after = -log(function_P_cNone(new_X));

    if (E_after <= E_before) %X更新
```

```
        X = new_X;
    end

    if (E_after > E_before)
        xx = rand();

        if (xx < exp(-(E_after - E_before))) %X一定概率更新
            X = new_X;
        end

    end

    X_1 = round((X(1) - X0(1)) / 0.0000001 + 5000); %加5000，使得平均值在5000处
    X_2 = round((X(2) - X0(2)) / 0.0000001 + 5000);
    X_3 = round((X(3) - X0(3)) / 0.0000001 + 5000);
    X_4 = round((X(4) - X0(4)) / 0.0000001 + 5000);
    X_5 = round((X(5) - X0(5)) / 0.0000001 + 5000);
    Y_1 = round((X(6) - X0(6)) / 0.0000001 + 5000);
    Y_2 = round((X(7) - X0(7)) / 0.0000001 + 5000);
    Y_3 = round((X(8) - X0(8)) / 0.0000001 + 5000);
    Y_4 = round((X(9) - X0(9)) / 0.0000001 + 5000);
    Y_5 = round((X(10) - X0(10)) / 0.0000001 + 5000);
    N_1(X_1, Y_1) = N_1(X_1, Y_1) + 1;
    N_2(X_2, Y_2) = N_2(X_2, Y_2) + 1;
    N_3(X_3, Y_3) = N_3(X_3, Y_3) + 1;
    N_4(X_4, Y_4) = N_4(X_4, Y_4) + 1;
    N_5(X_4, Y_4) = N_5(X_4, Y_4) + 1;
end

%五个未知节点的图
N = zeros(4004002, 3);
k = 1;

for i = 4000:6000

    for j = 4000:6000
        N(k, :) = [i * 0.0000001, j * 0.0000001, N_1(i, j)];
        k = k + 1;
    end

end

```

```
[X, Y, Z] = griddata(N(:, 1), N(:, 2), N(:, 3), linspace(min(N(:, 1)), max(N(:, 1)))', linspace(min(N(:, 2)), max(N(:, 2))), 'natural');
surf(X, Y, Z); colormap(hsv)
```

附录十二：问题三计算信标节点 B 缺失时概率的 Matlab 代码

文件名：function_P_bNone.m

```
function P = function_P_bNone(X)
% 计算概率密度P
% X是十维特征向量
sigma = 0.0063;
P1_2 = function_Gauss(RSSI_distance(-67.4), sqrt((X(1) - X(2))^2 + (X(6) - X(7))^2), sigma);
P1_3 = function_Gauss(RSSI_distance(-67.3), sqrt((X(1) - X(3))^2 + (X(6) - X(8))^2), sigma);
P1_4 = function_Gauss(RSSI_distance(-69.2), sqrt((X(1) - X(4))^2 + (X(6) - X(9))^2), sigma);
P1_5 = function_Gauss(RSSI_distance(-68.9), sqrt((X(1) - X(5))^2 + (X(6) - X(10))^2), sigma);
P1_6 = function_Gauss(RSSI_distance(-67.2), sqrt((X(1) - 0)^2 + (X(6) - 0)^2), sigma);
P1_7 = function_Gauss(RSSI_distance(-59.1), sqrt((X(1) - 0)^2 + (X(6) - 5)^2), sigma);

P1_9 = function_Gauss(RSSI_distance(-70.2), sqrt((X(1) - 10)^2 + (X(6) - 5)^2), sigma);

P2_1 = function_Gauss(RSSI_distance(-68.0), sqrt((X(2) - X(1))^2 + (X(7) - X(6))^2), sigma);
P2_3 = function_Gauss(RSSI_distance(-64.4), sqrt((X(2) - X(3))^2 + (X(7) - X(8))^2), sigma);
P2_4 = function_Gauss(RSSI_distance(-60.7), sqrt((X(2) - X(4))^2 + (X(7) - X(9))^2), sigma);
P2_5 = function_Gauss(RSSI_distance(-65.6), sqrt((X(2) - X(5))^2 + (X(7) - X(10))^2), sigma);
P2_6 = function_Gauss(RSSI_distance(-69.1), sqrt((X(2) - 0)^2 + (X(7) - 0)^2), sigma);
P2_7 = function_Gauss(RSSI_distance(-68.4), sqrt((X(2) - 0)^2 + (X(7) - 5)^2), sigma);

P2_9 = function_Gauss(RSSI_distance(-64.8), sqrt((X(2) - 10)^2 + (X(7) - 5)^2), sigma);

```

```
P3_1 = function_Gauss(RSSI_distance(-66.7), sqrt((X(3) - X(1))^2 + (X(8) - X(6))^2), sigma);
P3_2 = function_Gauss(RSSI_distance(-64.7), sqrt((X(3) - X(2))^2 + (X(8) - X(7))^2), sigma);
P3_4 = function_Gauss(RSSI_distance(-64.8), sqrt((X(3) - X(4))^2 + (X(8) - X(9))^2), sigma);
P3_5 = function_Gauss(RSSI_distance(-62.7), sqrt((X(3) - X(5))^2 + (X(8) - X(10))^2), sigma);
P3_6 = function_Gauss(RSSI_distance(-67.0), sqrt((X(3) - 0)^2 + (X(8) - 0)^2), sigma);
P3_7 = function_Gauss(RSSI_distance(-67.9), sqrt((X(3) - 0)^2 + (X(8) - 5)^2), sigma);

P3_9 = function_Gauss(RSSI_distance(-68.4), sqrt((X(3) - 10)^2 + (X(8) - 5)^2), sigma);

P4_1 = function_Gauss(RSSI_distance(-68.8), sqrt((X(4) - X(1))^2 + (X(9) - X(6))^2), sigma);
P4_2 = function_Gauss(RSSI_distance(-61.0), sqrt((X(4) - X(2))^2 + (X(9) - X(7))^2), sigma);
P4_3 = function_Gauss(RSSI_distance(-64.7), sqrt((X(4) - X(3))^2 + (X(9) - X(9))^2), sigma);
P4_5 = function_Gauss(RSSI_distance(-63.9), sqrt((X(4) - X(5))^2 + (X(9) - X(10))^2), sigma);
P4_6 = function_Gauss(RSSI_distance(-69.6), sqrt((X(4) - 0)^2 + (X(9) - 0)^2), sigma);
P4_7 = function_Gauss(RSSI_distance(-70.0), sqrt((X(4) - 0)^2 + (X(9) - 5)^2), sigma);

P4_9 = function_Gauss(RSSI_distance(-64.1), sqrt((X(4) - 10)^2 + (X(9) - 5)^2), sigma);

P5_1 = function_Gauss(RSSI_distance(-69.2), sqrt((X(5) - X(1))^2 + (X(10) - X(6))^2), sigma);
P5_2 = function_Gauss(RSSI_distance(-65.3), sqrt((X(5) - X(2))^2 + (X(10) - X(7))^2), sigma);
P5_3 = function_Gauss(RSSI_distance(-62.6), sqrt((X(5) - X(4))^2 + (X(10) - X(9))^2), sigma);
P5_4 = function_Gauss(RSSI_distance(-63.3), sqrt((X(5) - X(5))^2 + (X(10) - X(10))^2), sigma);
P5_6 = function_Gauss(RSSI_distance(-68.6), sqrt((X(5) - 0)^2 + (X(10) - 0)^2), sigma);
```

```
    P5_7 = function_Gauss(RSSI_distance(-69.5), sqrt((X(5) - 0)^2 + (X(10) - 5)^2)
        , sigma);

    P5_9 = function_Gauss(RSSI_distance(-67.0), sqrt((X(5) - 10)^2 + (X(10) - 5)
        ^2), sigma);

    P = P1_2 * P1_3 * P1_4 * P1_5 * P1_6 * P1_7 * P1_9 * P2_1 * P2_3 * P2_4 * P2_5
         * P2_6 * P2_7 * P2_9 * P3_1 * P3_2 * P3_4 * P3_5 * P3_6 * P3_7 * P3_9 *
        P4_1 * P4_2 * P4_3 * P4_5 * P4_6 * P4_7 * P4_9 * P5_1 * P5_2 * P5_3 * P5_4
        * P5_6 * P5_7 * P5_9;
end
```

附录十三：问题三计算信标节点 B 缺失时未知节点坐标的 Matlab 代码

文件名：main_bNone.m

```
clc; clear;
X0 = [1.7913 6.1619 4.7172 7.4835 6.5297 4.2959 3.9327 1.7051 2.7283 1.1745]; %初
    值
X = X0;
N_1 = zeros(10000, 10000); %出现次数
N_2 = zeros(10000, 10000); %出现次数
N_3 = zeros(10000, 10000); %出现次数
N_4 = zeros(10000, 10000); %出现次数
N_5 = zeros(10000, 10000); %出现次数

for i = 1:1000000
    new_X = disturb(X); %扰动得到新的X
    E_before = -log(function_P_bNone(X));
    E_after = -log(function_P_bNone(new_X));

    if (E_after <= E_before) %X更新
        X = new_X;
    end

    if (E_after > E_before)
        xx = rand();

        if (xx < exp(-(E_after - E_before))) %X一定概率更新
            X = new_X;
        end

    end

```

```
    X_1 = round((X(1) - X0(1)) / 0.0000001 + 5000); %加5000，使得平均值在5000处
    X_2 = round((X(2) - X0(2)) / 0.0000001 + 5000);
    X_3 = round((X(3) - X0(3)) / 0.0000001 + 5000);
    X_4 = round((X(4) - X0(4)) / 0.0000001 + 5000);
    X_5 = round((X(5) - X0(5)) / 0.0000001 + 5000);
    Y_1 = round((X(6) - X0(6)) / 0.0000001 + 5000);
    Y_2 = round((X(7) - X0(7)) / 0.0000001 + 5000);
    Y_3 = round((X(8) - X0(8)) / 0.0000001 + 5000);
    Y_4 = round((X(9) - X0(9)) / 0.0000001 + 5000);
    Y_5 = round((X(10) - X0(10)) / 0.0000001 + 5000);
    N_1(X_1, Y_1) = N_1(X_1, Y_1) + 1;
    N_2(X_2, Y_2) = N_2(X_2, Y_2) + 1;
    N_3(X_3, Y_3) = N_3(X_3, Y_3) + 1;
    N_4(X_4, Y_4) = N_4(X_4, Y_4) + 1;
    N_5(X_4, Y_4) = N_5(X_4, Y_4) + 1;
end

%逐个画五个未知节点的图，只需将N_1改为N_2~N_5即可

N = zeros(4004002, 3);
k = 1;

for i = 4000:6000

    for j = 4000:6000
        N(k, :) = [i * 0.0000001, j * 0.0000001, N_1(i, j)];
        k = k + 1;
    end

end

[X, Y, Z] = griddata(N(:, 1), N(:, 2), N(:, 3), linspace(min(N(:, 1)), max(N(:, 1)
    ))', linspace(min(N(:, 2)), max(N(:, 2))), 'natural');
surf(X, Y, Z); colormap(hsv)
```

附录十四：问题三计算信标节点 A 缺失时概率的 Matlab 代码

文件名：function_P_aNone.m

```
    function P = function_P_aNone(X)
    % 计算概率密度P
    % X是十维特征向量
    sigma = 0.0063;
    P1_2 = function_Gauss(RSSI_distance(-67.4), sqrt((X(1) - X(2))^2 + (X(6) - X
```

```
    (7))^2), sigma);
P1_3 = function_Gauss(RSSI_distance(-67.3), sqrt((X(1) - X(3))^2 + (X(6) - X
    (8))^2), sigma);
P1_4 = function_Gauss(RSSI_distance(-69.2), sqrt((X(1) - X(4))^2 + (X(6) - X
    (9))^2), sigma);
P1_5 = function_Gauss(RSSI_distance(-68.9), sqrt((X(1) - X(5))^2 + (X(6) - X
    (10))^2), sigma);
P1_6 = function_Gauss(RSSI_distance(-67.2), sqrt((X(1) - 0)^2 + (X(6) - 0)^2),
     sigma);

P1_8 = function_Gauss(RSSI_distance(-70.8), sqrt((X(1) - 10)^2 + (X(6) - 0)^2)
    , sigma);
P1_9 = function_Gauss(RSSI_distance(-70.2), sqrt((X(1) - 10)^2 + (X(6) - 5)^2)
    , sigma);

P2_1 = function_Gauss(RSSI_distance(-68.0), sqrt((X(2) - X(1))^2 + (X(7) - X
    (6))^2), sigma);
P2_3 = function_Gauss(RSSI_distance(-64.4), sqrt((X(2) - X(3))^2 + (X(7) - X
    (8))^2), sigma);
P2_4 = function_Gauss(RSSI_distance(-60.7), sqrt((X(2) - X(4))^2 + (X(7) - X
    (9))^2), sigma);
P2_5 = function_Gauss(RSSI_distance(-65.6), sqrt((X(2) - X(5))^2 + (X(7) - X
    (10))^2), sigma);
P2_6 = function_Gauss(RSSI_distance(-69.1), sqrt((X(2) - 0)^2 + (X(7) - 0)^2),
     sigma);

P2_8 = function_Gauss(RSSI_distance(-67.6), sqrt((X(2) - 10)^2 + (X(7) - 0)^2)
    , sigma);
P2_9 = function_Gauss(RSSI_distance(-64.8), sqrt((X(2) - 10)^2 + (X(7) - 5)^2)
    , sigma);

P3_1 = function_Gauss(RSSI_distance(-66.7), sqrt((X(3) - X(1))^2 + (X(8) - X
    (6))^2), sigma);
P3_2 = function_Gauss(RSSI_distance(-64.7), sqrt((X(3) - X(2))^2 + (X(8) - X
    (7))^2), sigma);
P3_4 = function_Gauss(RSSI_distance(-64.8), sqrt((X(3) - X(4))^2 + (X(8) - X
    (9))^2), sigma);
P3_5 = function_Gauss(RSSI_distance(-62.7), sqrt((X(3) - X(5))^2 + (X(8) - X
    (10))^2), sigma);
P3_6 = function_Gauss(RSSI_distance(-67.0), sqrt((X(3) - 0)^2 + (X(8) - 0)^2),
     sigma);

```

```
    P3_8 = function_Gauss(RSSI_distance(-67.7), sqrt((X(3) - 10)^2 + (X(8) - 0)^2)
        , sigma);
    P3_9 = function_Gauss(RSSI_distance(-68.4), sqrt((X(3) - 10)^2 + (X(8) - 5)^2)
        , sigma);

    P4_1 = function_Gauss(RSSI_distance(-68.8), sqrt((X(4) - X(1))^2 + (X(9) - X
        (6))^2), sigma);
    P4_2 = function_Gauss(RSSI_distance(-61.0), sqrt((X(4) - X(2))^2 + (X(9) - X
        (7))^2), sigma);
    P4_3 = function_Gauss(RSSI_distance(-64.7), sqrt((X(4) - X(3))^2 + (X(9) - X
        (9))^2), sigma);
    P4_5 = function_Gauss(RSSI_distance(-63.9), sqrt((X(4) - X(5))^2 + (X(9) - X
        (10))^2), sigma);
    P4_6 = function_Gauss(RSSI_distance(-69.6), sqrt((X(4) - 0)^2 + (X(9) - 0)^2),
         sigma);

    P4_8 = function_Gauss(RSSI_distance(-66.1), sqrt((X(4) - 10)^2 + (X(9) - 0)^2)
        , sigma);
    P4_9 = function_Gauss(RSSI_distance(-64.1), sqrt((X(4) - 10)^2 + (X(9) - 5)^2)
        , sigma);

    P5_1 = function_Gauss(RSSI_distance(-69.2), sqrt((X(5) - X(1))^2 + (X(10) - X
        (6))^2), sigma);
    P5_2 = function_Gauss(RSSI_distance(-65.3), sqrt((X(5) - X(2))^2 + (X(10) - X
        (7))^2), sigma);
    P5_3 = function_Gauss(RSSI_distance(-62.6), sqrt((X(5) - X(4))^2 + (X(10) - X
        (9))^2), sigma);
    P5_4 = function_Gauss(RSSI_distance(-63.3), sqrt((X(5) - X(5))^2 + (X(10) - X
        (10))^2), sigma);
    P5_6 = function_Gauss(RSSI_distance(-68.6), sqrt((X(5) - 0)^2 + (X(10) - 0)^2)
        , sigma);

    P5_8 = function_Gauss(RSSI_distance(-64.4), sqrt((X(5) - 10)^2 + (X(10) - 0)
        ^2), sigma);
    P5_9 = function_Gauss(RSSI_distance(-67.0), sqrt((X(5) - 10)^2 + (X(10) - 5)
        ^2), sigma);

    P = P1_2 * P1_3 * P1_4 * P1_5 * P1_6 * P1_8 * P1_9 * P2_1 * P2_3 * P2_4 * P2_5
         * P2_6 * P2_8 * P2_9 * P3_1 * P3_2 * P3_4 * P3_5 * P3_6 * P3_8 * P3_9 *
        P4_1 * P4_2 * P4_3 * P4_5 * P4_6 * P4_8 * P4_9 * P5_1 * P5_2 * P5_3 * P5_4
        * P5_6 * P5_8 * P5_9;
end
```

附录十五：问题三计算信标节点 A 缺失时未知节点坐标的 Matlab 代码

文件名：main_aNone.m

```
clc; clear;
X0 = [1.7913 6.1619 4.7172 7.4835 6.5297 4.2959 3.9327 1.7051 2.7283 1.1745]; %初值
X = X0;
N_1 = zeros(10000, 10000); %出现次数
N_2 = zeros(10000, 10000); %出现次数
N_3 = zeros(10000, 10000); %出现次数
N_4 = zeros(10000, 10000); %出现次数
N_5 = zeros(10000, 10000); %出现次数

for i = 1:1000000
    new_X = disturb(X); %扰动得到新的X
    E_before = -log(function_P_aNone(X));
    E_after = -log(function_P_aNone(new_X));

    if (E_after <= E_before) %X更新
        X = new_X;
    end

    if (E_after > E_before)
        xx = rand();

        if (xx < exp(-(E_after - E_before))) %X一定概率更新
            X = new_X;
        end

    end

    X_1 = round((X(1) - X0(1)) / 0.0000001 + 5000); %加5000，使得平均值在5000处
    X_2 = round((X(2) - X0(2)) / 0.0000001 + 5000);
    X_3 = round((X(3) - X0(3)) / 0.0000001 + 5000);
    X_4 = round((X(4) - X0(4)) / 0.0000001 + 5000);
    X_5 = round((X(5) - X0(5)) / 0.0000001 + 5000);
    Y_1 = round((X(6) - X0(6)) / 0.0000001 + 5000);
    Y_2 = round((X(7) - X0(7)) / 0.0000001 + 5000);
    Y_3 = round((X(8) - X0(8)) / 0.0000001 + 5000);
    Y_4 = round((X(9) - X0(9)) / 0.0000001 + 5000);
    Y_5 = round((X(10) - X0(10)) / 0.0000001 + 5000);
    N_1(X_1, Y_1) = N_1(X_1, Y_1) + 1;
```

```
    N_2(X_2, Y_2) = N_2(X_2, Y_2) + 1;
    N_3(X_3, Y_3) = N_3(X_3, Y_3) + 1;
    N_4(X_4, Y_4) = N_4(X_4, Y_4) + 1;
    N_5(X_4, Y_4) = N_5(X_4, Y_4) + 1;
end

%逐个画五个未知节点的图，只需将N_1改为N_2~N_5即可

N = zeros(4004002, 3);
k = 1;

for i = 4000:6000

    for j = 4000:6000
        N(k, :) = [i * 0.0000001, j * 0.0000001, N_1(i, j)];
        k = k + 1;
    end

end

[X, Y, Z] = griddata(N(:, 1), N(:, 2), N(:, 3), linspace(min(N(:, 1)), max(N(:, 1)
    ))', linspace(min(N(:, 2)), max(N(:, 2))), 'natural');
surf(X, Y, Z)
colormap(hsv)
```

附录十四：问题三计算信标节点 O 缺失时概率的 Matlab 代码

文件名：function_P_oNone.m

```
function P = function_P_oNone(X)
% 计算概率密度P
sigma = 0.0063;
P1_2 = function_Gauss(RSSI_distance(-67.4), sqrt((X(1) - X(2))^2 + (X(6) - X
    (7))^2), sigma);
P1_3 = function_Gauss(RSSI_distance(-67.3), sqrt((X(1) - X(3))^2 + (X(6) - X
    (8))^2), sigma);
P1_4 = function_Gauss(RSSI_distance(-69.2), sqrt((X(1) - X(4))^2 + (X(6) - X
    (9))^2), sigma);
P1_5 = function_Gauss(RSSI_distance(-68.9), sqrt((X(1) - X(5))^2 + (X(6) - X
    (10))^2), sigma);

P1_7 = function_Gauss(RSSI_distance(-59.1), sqrt((X(1) - 0)^2 + (X(6) - 5)^2),
     sigma);
P1_8 = function_Gauss(RSSI_distance(-70.8), sqrt((X(1) - 10)^2 + (X(6) - 0)^2)
```

```
    , sigma);
P1_9 = function_Gauss(RSSI_distance(-70.2), sqrt((X(1) - 10)^2 + (X(6) - 5)^2)
    , sigma);

P2_1 = function_Gauss(RSSI_distance(-68.0), sqrt((X(2) - X(1))^2 + (X(7) - X
    (6))^2), sigma);
P2_3 = function_Gauss(RSSI_distance(-64.4), sqrt((X(2) - X(3))^2 + (X(7) - X
    (8))^2), sigma);
P2_4 = function_Gauss(RSSI_distance(-60.7), sqrt((X(2) - X(4))^2 + (X(7) - X
    (9))^2), sigma);
P2_5 = function_Gauss(RSSI_distance(-65.6), sqrt((X(2) - X(5))^2 + (X(7) - X
    (10))^2), sigma);

P2_7 = function_Gauss(RSSI_distance(-68.4), sqrt((X(2) - 0)^2 + (X(7) - 5)^2),
     sigma);
P2_8 = function_Gauss(RSSI_distance(-67.6), sqrt((X(2) - 10)^2 + (X(7) - 0)^2)
    , sigma);
P2_9 = function_Gauss(RSSI_distance(-64.8), sqrt((X(2) - 10)^2 + (X(7) - 5)^2)
    , sigma);

P3_1 = function_Gauss(RSSI_distance(-66.7), sqrt((X(3) - X(1))^2 + (X(8) - X
    (6))^2), sigma);
P3_2 = function_Gauss(RSSI_distance(-64.7), sqrt((X(3) - X(2))^2 + (X(8) - X
    (7))^2), sigma);
P3_4 = function_Gauss(RSSI_distance(-64.8), sqrt((X(3) - X(4))^2 + (X(8) - X
    (9))^2), sigma);
P3_5 = function_Gauss(RSSI_distance(-62.7), sqrt((X(3) - X(5))^2 + (X(8) - X
    (10))^2), sigma);

P3_7 = function_Gauss(RSSI_distance(-67.9), sqrt((X(3) - 0)^2 + (X(8) - 5)^2),
     sigma);
P3_8 = function_Gauss(RSSI_distance(-67.7), sqrt((X(3) - 10)^2 + (X(8) - 0)^2)
    , sigma);
P3_9 = function_Gauss(RSSI_distance(-68.4), sqrt((X(3) - 10)^2 + (X(8) - 5)^2)
    , sigma);

P4_1 = function_Gauss(RSSI_distance(-68.8), sqrt((X(4) - X(1))^2 + (X(9) - X
    (6))^2), sigma);
P4_2 = function_Gauss(RSSI_distance(-61.0), sqrt((X(4) - X(2))^2 + (X(9) - X
    (7))^2), sigma);
P4_3 = function_Gauss(RSSI_distance(-64.7), sqrt((X(4) - X(3))^2 + (X(9) - X
    (9))^2), sigma);
```

```
    P4_5 = function_Gauss(RSSI_distance(-63.9), sqrt((X(4) - X(5))^2 + (X(9) - X
        (10))^2), sigma);

    P4_7 = function_Gauss(RSSI_distance(-70.0), sqrt((X(4) - 0)^2 + (X(9) - 5)^2),
         sigma);
    P4_8 = function_Gauss(RSSI_distance(-66.1), sqrt((X(4) - 10)^2 + (X(9) - 0)^2)
        , sigma);
    P4_9 = function_Gauss(RSSI_distance(-64.1), sqrt((X(4) - 10)^2 + (X(9) - 5)^2)
        , sigma);

    P5_1 = function_Gauss(RSSI_distance(-69.2), sqrt((X(5) - X(1))^2 + (X(10) - X
        (6))^2), sigma);
    P5_2 = function_Gauss(RSSI_distance(-65.3), sqrt((X(5) - X(2))^2 + (X(10) - X
        (7))^2), sigma);
    P5_3 = function_Gauss(RSSI_distance(-62.6), sqrt((X(5) - X(4))^2 + (X(10) - X
        (9))^2), sigma);
    P5_4 = function_Gauss(RSSI_distance(-63.3), sqrt((X(5) - X(5))^2 + (X(10) - X
        (10))^2), sigma);

    P5_7 = function_Gauss(RSSI_distance(-69.5), sqrt((X(5) - 0)^2 + (X(10) - 5)^2)
        , sigma);
    P5_8 = function_Gauss(RSSI_distance(-64.4), sqrt((X(5) - 10)^2 + (X(10) - 0)
        ^2), sigma);
    P5_9 = function_Gauss(RSSI_distance(-67.0), sqrt((X(5) - 10)^2 + (X(10) - 5)
        ^2), sigma);

    P = P1_2 * P1_3 * P1_4 * P1_5 * P1_7 * P1_8 * P1_9 * P2_1 * P2_3 * P2_4 * P2_5
         * P2_7 * P2_8 * P2_9 * P3_1 * P3_2 * P3_4 * P3_5 * P3_7 * P3_8 * P3_9 *
        P4_1 * P4_2 * P4_3 * P4_5 * P4_7 * P4_8 * P4_9 * P5_1 * P5_2 * P5_3 * P5_4
        * P5_7 * P5_8 * P5_9;
end
```

附录十七：问题三计算信标节点 O 缺失时未知节点坐标的 Matlab 代码

文件名：main_oNone.m

```
clc; clear;
X0 = [1.7913 6.1619 4.7172 7.4835 6.5297 4.2959 3.9327 1.7051 2.7283 1.1745]; %初
    值
X = X0;
N_1 = zeros(10000, 10000); %出现次数
N_2 = zeros(10000, 10000); %出现次数
N_3 = zeros(10000, 10000); %出现次数
N_4 = zeros(10000, 10000); %出现次数
```

```
N_5 = zeros(10000, 10000); %出现次数

for i = 1:1000000
    new_X = disturb(X); %扰动得到新的X
    E_before = -log(function_P_oNone(X));
    E_after = -log(function_P_oNone(new_X));

    if (E_after <= E_before) %X更新
        X = new_X;
    end

    if (E_after > E_before)
        xx = rand();

        if (xx < exp(-(E_after - E_before))) %X一定概率更新
            X = new_X;
        end

    end

    X_1 = round((X(1) - X0(1)) / 0.0000001 + 5000); %加5000，使得平均值在5000处
    X_2 = round((X(2) - X0(2)) / 0.0000001 + 5000);
    X_3 = round((X(3) - X0(3)) / 0.0000001 + 5000);
    X_4 = round((X(4) - X0(4)) / 0.0000001 + 5000);
    X_5 = round((X(5) - X0(5)) / 0.0000001 + 5000);
    Y_1 = round((X(6) - X0(6)) / 0.0000001 + 5000);
    Y_2 = round((X(7) - X0(7)) / 0.0000001 + 5000);
    Y_3 = round((X(8) - X0(8)) / 0.0000001 + 5000);
    Y_4 = round((X(9) - X0(9)) / 0.0000001 + 5000);
    Y_5 = round((X(10) - X0(10)) / 0.0000001 + 5000);
    N_1(X_1, Y_1) = N_1(X_1, Y_1) + 1;
    N_2(X_2, Y_2) = N_2(X_2, Y_2) + 1;
    N_3(X_3, Y_3) = N_3(X_3, Y_3) + 1;
    N_4(X_4, Y_4) = N_4(X_4, Y_4) + 1;
    N_5(X_4, Y_4) = N_5(X_4, Y_4) + 1;
end

%逐个画五个未知节点的图，只需将N_1改为N_2~N_5即可
N = zeros(4004002, 3);
k = 1;

for i = 4000:6000
```

```
50
51      for j = 4000:6000
52          N(k, :) = [i * 0.0000001, j * 0.0000001, N_1(i, j)];
53          k = k + 1;
54      end
55
56  end
57
58  [X, Y, Z] = griddata(N(:, 1), N(:, 2), N(:, 3), linspace(min(N(:, 1)), max(N(:, 1)
        ))', linspace(min(N(:, 2)), max(N(:, 2))), 'natural');
59  surf(X, Y, Z); colormap(hsv)
```