基于计算几何和遗传算法的多波束测深和测线布设模型

摘　　要

本文主要通过建立**多波束测深模型**来测量海底深度等数据，通过**遗传算法**等方法，对海洋测线进行布设。

针对问题一，我们运用**平面几何**的定理，分别计算出测量海底深度、测深覆盖宽度和重叠率等公式，并由题中所给数据，分别算出三者的真实数值。通过分析数据，我们发现，测线由左向右分布时，测量海水深度逐渐变小，覆盖宽度逐渐变小，与前一条测线的重叠率不断减小，在 **600m** 和 **800m** 处的测线重叠率小于 0，即出现**漏测**的情况。

针对问题二，由于角度具有对称性，我们只需要考虑 0 度，45 度，90 度，135 度和 180 度这**五种情形**。我们将测量船和海底坡面的位置关系抽象成三棱锥体，运用立体几何的定理，分类讨论在不同的测线夹角中，测深覆盖宽度与测量船距中心点距离间的关系。经过计算，我们发现 0 度和 45 度时，测深覆盖宽度随测量船距中心距离的增加而**增加**；90 度时，测深覆盖宽度随距离增加**保持不变**；135 度和 180 度时，测深覆盖宽度随测量船距中心距离的增加而**减少**。

针对问题三，我们设计了两种测线布设的方式，即测线**由中间向两侧对称分布**和测线**由东向西均匀分布**。运用几何公式和 **Bisinhopping 算法**，我们分别算出两种方式测线布设的最优解。当测线由中间向两侧对称分布时，测线布设的最优解为 **35** 条测线，总长度 **127663** 米，测线方向与海底坡面的法向在水平面上投影的夹角为 **63.64** 度；当测线由东向西均匀分布时，测线布设的最优解为 **23** 条测线，总长度为 **84070** 米，测线方向与海底坡面的法向在水平面上投影的夹角为 **79.87** 度. 经过比较，我们选择后一种测线布设的方案。

针对问题四，根据所给数据，我们画出海底地形的仿真图。为了满足题目中三个要求，我们运用**贪心算法**，将三个要求函数归一化，写出加权目标函数，为了算出最优解，我们运用**遗传算法**，多次迭代，最终得到的最优解：测线数目为 **55** 根，测线总长度为 **343793.56** 米，漏测海区占总待测海域面积的百分比为 **8.63%**，重叠率超过 20% 部分的总长度为 **3148.25** 米。

关键字：**多波束测深　测线布设　几何　Bisinhopping 算法　贪心算法　遗传算法**

一、问题重述

1.1 背景知识

在传统的海洋测绘中，一般采用单波束测深系统。单波束测深系统利用声波在水中传播的特性，测量海水的深度，但由于这种方式在地形复杂处需要多次测量以及对测量人员的要求较高，在现代海洋测绘中，高效率、高精度、全覆盖的多波束测深系统被更加广泛地应用。为了使得多波束测深系统能够更加准确地测出数据，设计合适的测线显得极其重要。

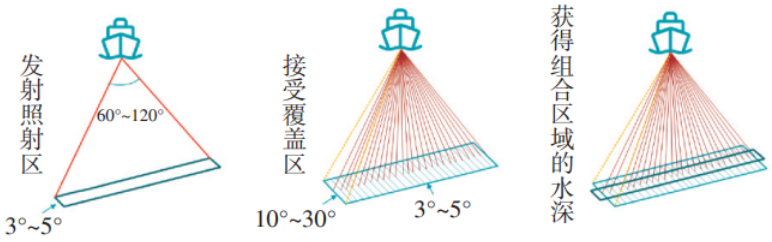

图 1. 多波束测深系统工作原理 [6]

1.2 要解决的问题

通过题目所给条件和数据，我们要解决以下四个问题：

问题一：在多波束换能器开角固定和海底坡度为 α 的条件下，建立多波束测深的覆盖宽度及相邻条带之间重叠率的数学模型，并根据给定换能器开角和海底坡度算出海水深度，覆盖宽度和侧线间的重叠率。

问题二：在一个矩形待测海域中，测线方向与海底坡面的法向在水平面上投影的夹角为 β，建立多波束测深覆盖宽度的数学模型，并根据给定数据，计算覆盖宽度。

问题三：在一个南北长 2 海里、东西宽 4 海里的矩形海域内，海域中心点处的海水深度为 110 m，海底坡度为 1.5 度。多波束换能器的开角为 120 度。如何建立数学模型，设计一组测量长度最短、可完全覆盖整个待测海域的测线，且相邻条带之间的重叠率满足 10%~20% 的要求。

问题四：在若干年前的某海域（南北长 5 海里、东西宽 4 海里），曾运用单波束测量海底测深数据，现需要利用这组数据为多波束测量船的测量布线提供帮助。在设计测线时，有如下要求：(1) 沿测线扫描形成的条带尽可能地覆盖整个待测海域；(2) 相邻条带之间的重叠率尽量控制在 20% 以下；(3) 测线的总长度尽可能短。在设计出具体的测线后，请计算如下指标：(1) 测线的总长度；(2) 漏测海区占总待测海域面积的百分比；(3) 在重叠区域中，重叠率超过 20% 部分的总长度。

二、问题分析

2.1 问题一

问题一中固定了多波束换能器的开角以及海底的坡度，运用几何中的正弦定理和相似关系，我们可以分别计算出海水深度、覆盖宽度、测线间的重叠率与测线距中心距离的公式，运用题目所给出的数据，可以分别计算出具体数据。

2.2 问题二

问题二在问题一的基础上进一步从三维思考，通过分析，我们发现问题二中测线的方向夹角具有对称特征，因此在模型建立时，我们只需要考虑 0 度，45 度，90 度，135 度和 180 度这几种情况。通过建立坐标系，给出平面的方程和波束在水平面的投影方程，通过数学分析，0 度，45 度时随着测线离中心距离的增加，覆盖宽度逐渐增加；90 度时覆盖宽度与测线离中心距离无关；135 度和 180 度时随着测线离中心距离的增加，覆盖宽度逐渐减少。在 45 度和 135 度时，船与海底坡面的关系可以抽象成三棱锥体，运用几何知识计算；在 90 度时，可以运用问题一所推出的公式; 在 0 度与 180 度时，与测线垂直的面和海底坡面的交线与海底水平平行，可运用等腰三角形的知识，根据水深高度计算。

2.3 问题三

问题三在问题二的基础上给定了海域形状和面积，问题三中需要满足以下两个要求：可完全覆盖整个待测海域的测线，且相邻条带之间的重叠率满足 10%~20%。在这个基础上设计出长度最短的测线。假定测线间的距离不变，在矩形海域，测线布设需要考虑中心测线和边缘测线，为满足覆盖整个待测海域，最靠近边缘测线需要覆盖边界区域，中心测线之间的距离不能过大；为满足相邻条带重叠率满足 10%~20%，在问题二公式的基础上，推导出重叠率公式，使得重叠率最低处高于 10%，最高处低于 20%。满足以上两个要求后，运用几何关系，可以列出计算长度的公式，以两个要求为约束函数，长度公式为目标函数，求出最优解，即为所求。为进一步求出更好的解，考虑两种测线布置的方式：一种是由中心向外布设，一种是边缘向中心布设。分别算出两种布设方式的最优解，通过比较，最终可以确定测线布设的最优方案。

2.4 问题四

在问题四中，我们已经知道单波束测绘的数据，由此，通过编程软件，我们可以画出海底地形。根据题目中的三个要求：沿测线扫描形成的条带尽可能地覆盖整个待测海域、相邻条带之间的重叠率尽量控制在 20% 以下和测线的总长度尽可能短，赋予这三个要求不同的权重，列出对应的目标函数和约束条件，将三个要求归一化，而后运用算法不断迭代，找出三个指标的最佳权重，代入到问题四中，求出最优解。

三、模型假设

- 假设海水均匀，每一点处海水深度确定。
- 假设测量船沿着直线运动。
- 假设海底坡度与法向投影夹角恒定。
- 假设测线布设中，测线两两平行且距离相等。
- 假设海底地形，海面天气等因素对单波束数据测量影响较小。

四、符号说明

符号	含义
n_k	第 k 个换能器到中心点的距离
h_k	第 k 个换能器处水深深度
α	海底斜面的坡度
$s_{1,k}$	第 k 个换能器重锤线左侧覆盖宽度
$s_{2,k}$	第 k 个换能器重锤线右侧覆盖宽度
s	第 k 个换能器覆盖宽度
η	重叠率
β	测线方向与海底坡面的法向在水平面上投影的夹角
α_1	与测线方向垂直的面与海底坡面的交线和水平面的夹角
α_2	测线与海底坡面构成的线面角
d	相邻两条测线之间相隔的距离
d_0	距离最右侧测线最近的矩形顶点到该测线的距离
m	矩形海域的宽
n	矩形海域的长
L_k	第 k 个部分测线的数目
w_k	遗传算法权重

五、模型的建立和求解

5.1 对问题一的求解

在问题一中，直接对图形使用正弦定理，计算出每个换能器所处方位的海洋深度及其覆盖宽度，并且可以得到每个换能器重锤线左侧和右侧的覆盖宽度。

$$h_k = 70 - n_k * tan\alpha$$

$$s_{1,k} = \frac{\sqrt{3}h_k}{2\sin(\frac{\pi}{6} - \alpha)}$$

$$s_{2,k} = \frac{\sqrt{3}h_k}{2\sin(\frac{5\pi}{6} - \alpha)}$$

$$s_k = s_{1k} + s_{2k}$$

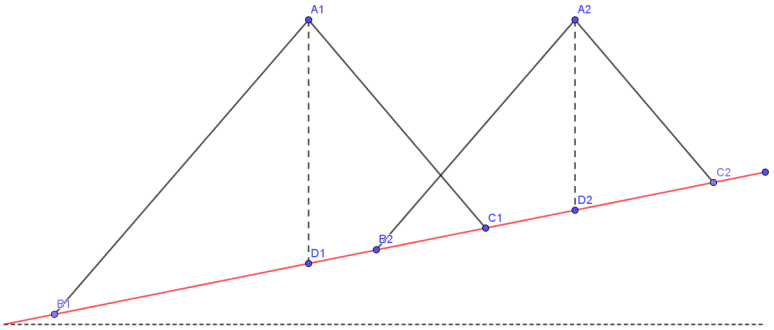

图 2. 相邻两个换能器工作示意图

如图 2，通过在斜坡上建立一维数轴，以中心点处为原点，向浅处为正方向，将每个换能器在数轴上的坐标记为 x_k, 同时计算出每个换能器重锤线的左侧覆盖宽度 $s_{1,k}$ 与右侧覆盖宽度 $s_{2,k}$, 可以得到前一个换能器覆盖范围的右侧端点 $C1$ 的坐标 s_1 和后一个换能器覆盖范围的左侧端点 $B2$ 的坐标 s_2, 相减即可得到重叠宽度 w，再除以后一个换能器覆盖宽度的总长 s_k，得到重叠率 η。

$$x_k = \frac{n_k}{cos\alpha}$$

$$s_1 = x_k - s_{1,k}$$

$$s_2 = x_{k-1} + s_{2,k-1}$$

$$w = s_2 - s_1$$

$$\eta = \frac{w}{s_k}$$

由此可以得到问题 1 的计算结果如下表：

测线距中心点处的距离/m	-800	-600	-400	-200	0	200	400	600	800
海水深度/m	90.95	85.71	80.47	75.24	70	64.76	59.53	54.29	49.05
覆盖宽度/m	315.81	297.63	279.44	261.26	243.07	224.88	206.70	188.51	170.33
与前一条测线的重叠率/%	——	35.7	31.51	26.74	21.26	14.89	7.41	-1.53	-12.36

表 1. 问题 1 的计算结果

通过分析表格中的数据可以发现，测线由左向右分布时，测量海水深度逐渐变小，覆盖宽度逐渐变小，与前一条测线的重叠率不断减小，在 $600m$ 和 $800m$ 处的重叠率小于 0，即出现漏测的情况。

5.2 对问题二的求解

对于问题二，计算出不同的角度 β 下对应的与测线垂直的面和海底坡面相交构成的交线的坡角 α_1。首先考虑 β 是钝角的情况：

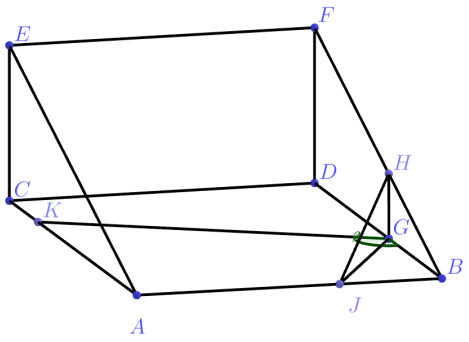

图 3. 海底坡面示意图

将需要计算的角度单独放在一个三棱锥中拿出来，并且重新标点，如下图所示：

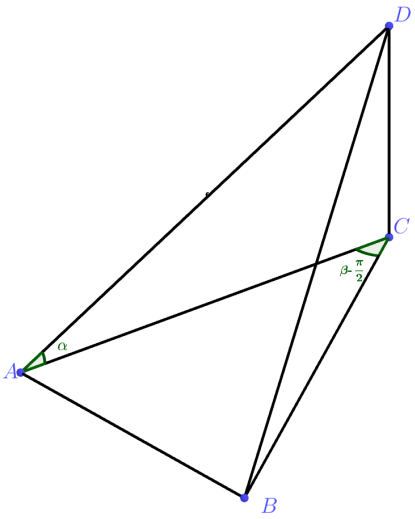

图 4. 包含 α_1 的三棱锥

容易知道 $\angle DBC = \alpha, DC \perp$ 面 $ABC, \angle ACB = \beta - \frac{\pi}{2}, AB \perp BC$, 使用正弦定理和正余弦的定义:

$$\angle CAB = \pi - \beta \qquad \sin(\pi - \beta) = \frac{BC}{AC}$$

$$\tan\alpha = \frac{DC}{BC}$$

$$\tan\alpha_1 = \tan\angle DAC = \frac{DC}{AC} = \sin(\pi - \beta)\tan\alpha$$

可以得到 $\alpha_1 = \arctan(\tan\alpha \cdot \sin\beta)$。

同样的，在 β 为锐角时，同样的，$\alpha_1 = \arctan(\tan\alpha \cdot \sin\beta)$。由于坡角必为正角，所以将 α_1 记为 $arctan(|\tan\alpha \cdot \sin\beta|)$。

由于覆盖宽度与水深呈正比关系，因此需要计算出测线与坡面的线面角 α_2，用于水深的计算。同样的，将所求的角置于三棱锥中, 如下图:

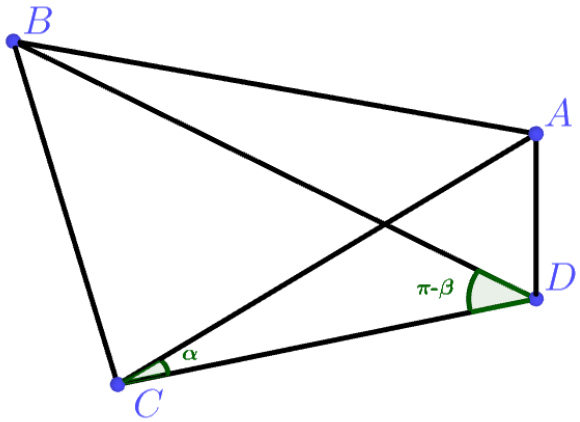

图 5. 包含 α_2 的三棱锥

其中，$\angle ACD=\alpha, AD\perp$ 面 $BCD, \angle BCD=\pi-\beta, DC\perp BC$。而 $\angle ABD$ 即是我们需要的角，对此使用正余弦的定义进行计算：

$$\begin{aligned}\tan\alpha &= \frac{AD}{CD}\\ \cos(\pi-\beta) &= \frac{CD}{BD}\\ \tan\alpha_2 = \tan\angle ABD &= \frac{AD}{BD}\\ &= \tan\alpha\cdot\cos(\pi-\beta) = -\tan\alpha\cdot\cos(\beta)\end{aligned}$$

由于 α_2 与 α 的大小较为相近，所以 α_2 为正角，因此可以得到 $\alpha_2 = arctan(|\tan\alpha\cdot\cos\beta|)$

在 β 确定的情况下，当测量船距海域中心点处的距离为 n 时，测量船所处的位置距离海底的深度为：

$$h = h_0 + n\cdot\tan\alpha_2\cdot\cos\beta$$

其中 $h_0 = 120m$。由此可以计算出测量船在距离海域中心点 n 处的覆盖宽度:

$$\begin{aligned}t_1 &= \frac{\sqrt{3}h}{2\sin(\frac{\pi}{6}-\alpha)}\\ t_2 &= \frac{\sqrt{3}h}{2\sin(\frac{5\pi}{6}-\alpha)}\\ t &= t_1+t_2\end{aligned}$$

其中 t_1, t_2 分别为测量船对海底水平面作垂线下来，左右两侧的覆盖宽度，t 为测量船在距离海域中心点 n 处的覆盖宽度。

将得出的结果画成变化图如下：

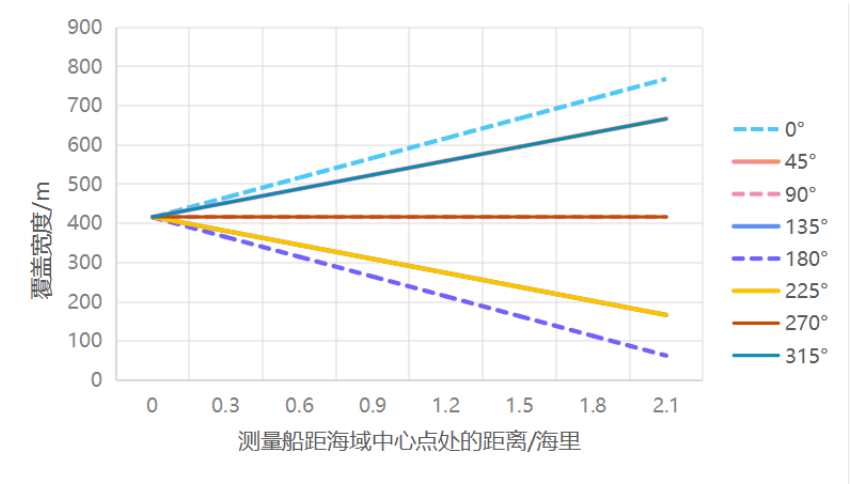

图 6. 不同测线方向夹角下的覆盖宽度变化图

最后将得出的结果填入表中，得到表 2：

覆盖宽度/m		测量船距海域中心点处的距离/海里							
		0	0.3	0.6	0.9	1.2	1.5	1.8	2.1
测线方向夹角/°	0	415.69	466.09	516.49	566.89	617.29	667.69	718.09	768.48
	45	416.65	452.37	488.09	523.81	559.53	595.25	630.97	666.69
	90	416.69	416.69	416.69	416.69	416.69	416.69	416.69	416.69
	135	416.65	380.93	345.21	309.49	273.77	238.05	202.33	166.61
	180	415.69	365.29	314.89	264.50	214.10	163.70	113.30	62.90
	225	416.65	380.93	345.21	309.49	273.77	238.05	202.33	166.61
	270	416.69	416.69	416.69	416.69	416.69	416.69	416.69	416.69
	315	416.65	452.37	488.09	523.81	559.53	595.25	630.97	666.69

表 2. 问题 2 的计算结果

5.3 对问题三的求解

对于问题三，先画出问题三的海底坡面示意图：

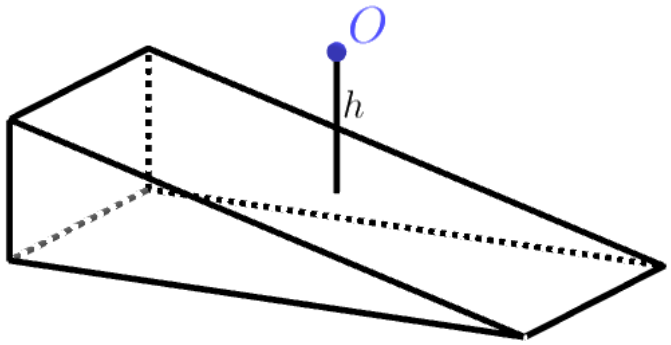

图 7. 海底坡面示意图

由此，可以计算得到在该矩形海域右边界处水深为 $h_1 = 99.54m$，左边界处水深为 $h_2 = 120.46m$。

在本题中，应用 Basinhopping 算法解决该题，下面是对该算法的介绍：

Basinhopping 算法又叫盆地跳跃算法，是一种全局优化算法，他将单个全局最优解定位在多个局部最优解之中，其特征有以下三个：1. 随机改变坐标位置。2. 在新坐标附近进行局部极小化搜索。3. 根据目标函数值决定是否接受或拒绝新的坐标。

由于 Basinhopping 算法采用了随机性的元素，可以跳出局部最优解，更有可能找到全局最

优解，而且该算法适用于非线性问题，因此在解答本题的过程中，我们选用了该算法。

对于问题三，需要先设计测线的排列方式，再根据相邻测线的重叠率和完全覆盖的条件运用 Basinhopping 算法求最优解。

下面介绍两种不同的测线布设方式，及其结果的对比。

5.3.1 测线由中间向两侧对称分布

第一种测线分布设计如图 8：

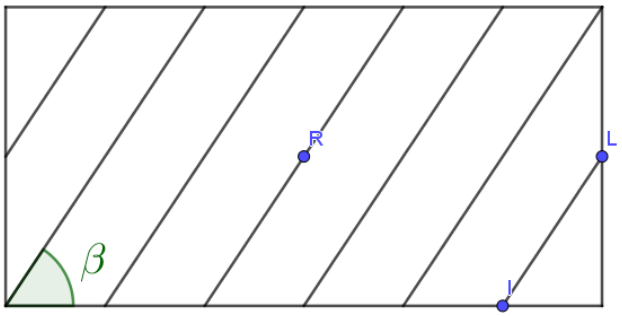

图 8. 第一种测线设计方案

在矩形的正中间放置一条过其中心点的测线，其余测线以相同的密度向两边分布，由于其具有对称性，所以计算测线总长的时候，只需计算其中一半测线的长度。

m 为矩形海域的宽，n 为矩形海域的长。不妨称过中心点的测线为中心线，经过计算可以知道中心线右侧的测线数目为 $L = \left[\frac{n|\sin(\pi-\beta)|+m|\cos(\pi-\beta)|}{2d}\right]$。

将测线长度分为两部分计算，第一部分是两个端点都在矩形的长上的测线，第二部分是有一个端点在矩形的宽上的测线。经过计算可以知道，第一部分测线的数量是 $L_1 = 2\left[\frac{n|\sin(\pi-\beta)|-m|\cos(\pi-\beta)|}{2d}\right]+1$, 每一条测线的长度均为 $\frac{m}{\sin(\pi-\beta)}$,$d$ 为两条相邻测线之间的距离。第二部分测线的长度随着向两边分布而不断缩短，具体每条的长度为：

$$s_k = \frac{n|\sin(\pi-\beta)|+m|\cos(\pi-\beta)|-2k}{2|\sin(\pi-\beta)\cos(\pi-\beta)|},$$
$$\left[\frac{n|\sin(\pi-\beta)|-m|\cos(\pi-\beta)|}{2d}\right]+1 \le k \le \left[\frac{n|\sin(\pi-\beta)|+m|\cos(\pi-\beta)|}{2d}\right]$$

s_k 指的是从中心线开始往右数第 k 条测线，其中中心线是第 0 条。

有了上述条件，即可列出测线总长度的公式：

$$f(\beta,d) = \frac{m}{\sin(\pi-\beta)}\cdot\left(2\left[\frac{n|\sin(\pi-\beta)|-m|\cos(\pi-\beta)|}{2d}\right]+1\right)+2\sum_{\frac{L_1-1}{2}}^{L} s_k$$

而根据题意，要求相邻两条测线的重叠率在 10%~20%, 由问题一知，当测线所处的位置越深，相邻两条测线的重叠率越高，当测线所处的位置越浅，相邻两条测线的重叠率越低，因此只需约束最深处和最浅处测线测量宽度的重叠率在 10%~20%，即可将所有测线的重叠率均约束在 10%~20%。

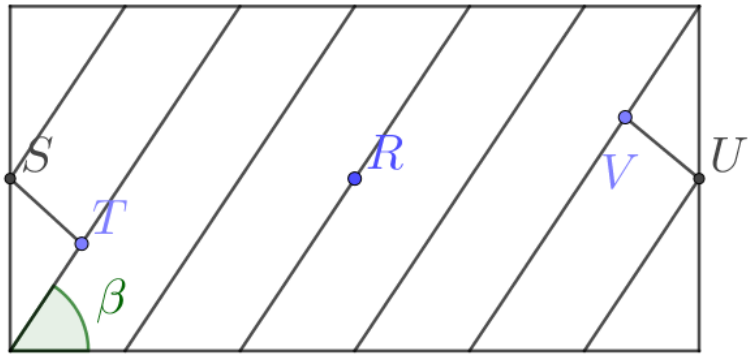

图 9. 测线重叠率最小与最大处示意图

在 S 点处，深度最大，U 点处深度最小，所以只需控制 S,T 两点的重叠率 $\leq$20%,U,V 两点的重叠率 $\geq$10%。记 S 点的深度为 h_1',T 点的深度为 h_2'，U 点的深度为 h_1'',V 点的深度为 h_2''，则，

$$
\begin{aligned}
h &= 110m \\
h_1' &= h - \frac{m\sin(\alpha_2)}{2\sin\beta} + \frac{d\cdot L\tan\alpha}{\sin\beta} + s_L\tan\alpha_2 \\
h_2' &= h_1' - d\cdot\tan\alpha_1 \\
h_1'' &= h + \frac{m\sin(\alpha_2)}{2\sin\beta} - \frac{d\cdot L\tan\alpha}{\sin\beta} - s_L\tan\alpha_2 \\
h_2'' &= h_1'' + d\cdot\tan\alpha_1
\end{aligned}
$$

接下来使用与问题一相同的算法，计算 S,T 处重叠率：

$$
\begin{aligned}
x &= \frac{d}{\cos\alpha_1} \\
x_{1,1} &= \frac{\sqrt{3}h_1^{'}}{2\sin(\frac{\pi}{6}-\alpha)} \\
x_{2,1} &= \frac{\sqrt{3}h_2^{'}}{2\sin(\frac{5\pi}{6}-\alpha)} \\
x_1 &= \frac{\sqrt{3}h_1^{'}}{2\sin(\frac{\pi}{6}-\alpha_1)} + \frac{\sqrt{3}h_1^{'}}{2\sin(\frac{5\pi}{6}-\alpha_1)} \\
x_1^{'} &= x - x_{1,1} \\
x_2^{'} &= x_{2,1} \\
w^{'} &= x_2^{'} - x_1^{'} \\
\eta^{'} &= \frac{w^{'}}{x_1}
\end{aligned}
$$

同样的，得到 U,V 处的重叠率为 $\eta^{''}$。

为了保证对海域的全覆盖，最右侧的测线的测量宽度必须将角落全涵盖。由于海底坡面的坡角很小，所以测量船在最右侧测线移动时覆盖宽度的变化率远小于边界上点到最右侧测线的变化率，因此，选择矩形右侧顶角是否被覆盖作为是否漏测的依据。

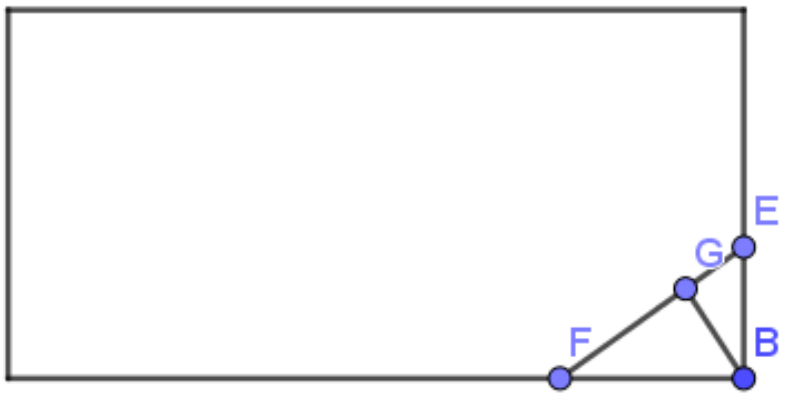

图 10. 判断全覆盖依据的示意图

如图 11，只需保证 G 点处的测量宽度包含了 B 点即可，也即：

$$
\begin{aligned}
h_0 = h &+ \frac{m \cdot \tan\alpha_2}{2\sin\beta} - \frac{d \cdot L \cdot \tan\alpha}{\sin\beta} \\
&- (\frac{n}{2} + \frac{m}{2\tan\beta} - \frac{d \cdot L}{\sin\beta}) \cdot \cos\beta \cdot \tan\alpha_2 \\
x_2 &= \frac{\sqrt{3}h_0}{2\sin(\frac{5\pi}{6} - \alpha_1)} \\
y &= \frac{\frac{n \cdot \sin\beta}{2} + \frac{m}{2\cos\beta} - L \cdot d}{\cos\alpha_1}
\end{aligned}
$$

问题二就转化成了以下问题：

$$
\begin{aligned}
\min \quad & f(\beta, d) \\
& \eta^{'} \le 20\% \\
& \eta^{''} \ge 10\% \\
& x_2 \ge y
\end{aligned}
$$

得到最优解为 $\beta = 63.64°, d = 221.72m$, 测线数目是 35 条，测线总长度为 $127662.53m$。

5.3.2 测线由东向西均匀分布

第二种测线分布设计如图 11：

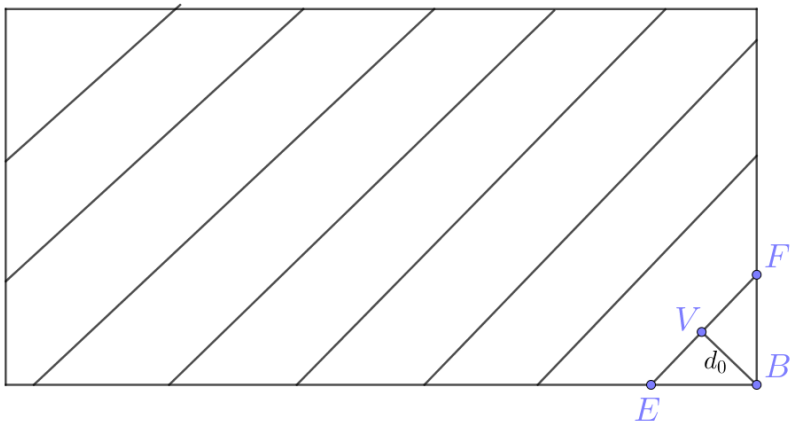

图 11. 第二种测线设计方案

首先确定最右边的测线的位置，如图 11，设 B 点距最右边测线的距离为 d_0, 已知右边界处的水深为 $h_1 = 99.54m$, 左边界处的水深为 $h_2 = 120.46m$, 可以得到 V 点处的水深为 $h_V = h_1 + d_0 \cdot \tan\alpha_1$,V 点右侧的覆盖宽度为 $x_2 = \frac{\sqrt{3}h_V}{2\sin(\frac{5\pi}{6} - \alpha_1)}$。

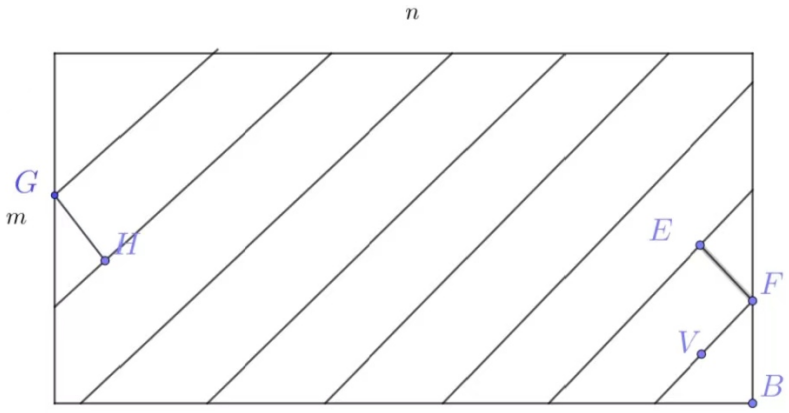

图 12. 对重叠率进行约束的点示意图

同样的，对于题目要求的 $10\% \sim 20\%$ 的重叠率，在水深最深处和水深最浅处进行约束，同样得到水最浅处的两个深度为 $h_F = h_1^{'} = 99.54m, h_E = h_2^{'} = h_1^{'} + d \cdot \tan\alpha_1$，水最深处的两个深度为 $h_G = h_1^{''} = 120.46, h_H = h_2^{''} = h_1^{''} - d \cdot \tan\alpha_1$。与情形一类似的，得到最浅处的重叠率 $\eta^{'}$ 和最深处的重叠率 $\eta^{''}$。

对于测线的总长度，总共分为三部分来计算，第一部分为有一端点在矩形右边界上的测线，第二部分为两端点均在矩形上下边界上的测线，第三部分为有一端点在左边界上的测线。

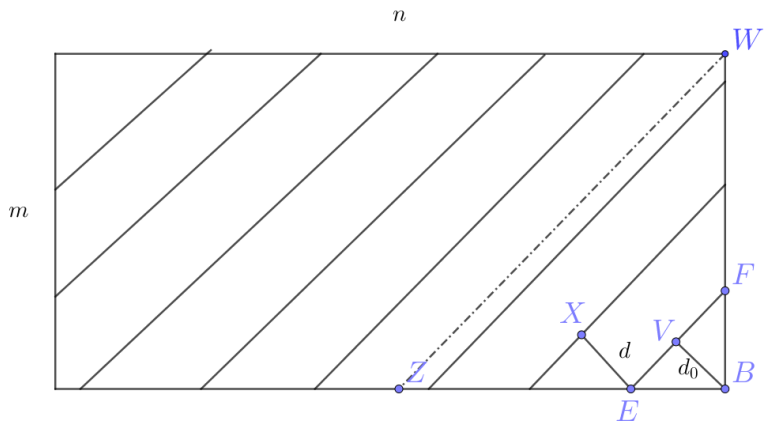

图 13. 测线示意图

如图 13，第一部分的测线的数目应为 $L_1 + 1 = \left[\frac{m\cos\beta - d_0}{d}\right] + 1$, 第二部分测线的数目为 $L_2 = \left[\frac{n\sin\beta - d_0}{d}\right] - \left[\frac{m\cos\beta - d_0}{d}\right]$, 第三部分测线的数目为 $L_3 = \left[\frac{m\cos\beta}{d}\right]$。

而我们的目标函数应该如下：

$$
\begin{aligned}
s_1 &= \sum_{k=0}^{L_1} \frac{d_0 + kd}{\sin\beta\cos\beta} \\
s_2 &= L_2 \cdot \frac{m}{\sin\beta} \\
s_3 &= \sum_{T=1}^{L_3} \frac{n - \left[\frac{n\sin\beta - d_0}{d}\right] \cdot \frac{d}{\sin\beta} - \frac{d_0}{\sin\beta} + \frac{m}{\tan\beta} - \frac{d\cdot T}{\sin\beta}}{\cos\beta} \\
f(\beta, d, d_0) &= s_1 + s_2 + s_3
\end{aligned}
$$

由此，可以将该问题转化为如下数学问题：

$$
\begin{aligned}
min \quad & f(\beta, d, d_0) \\
& \eta^{'} \geq 10\% \\
& \eta^{''} \leq 20\% \\
& x_2 \geq \frac{d_0}{\cos\alpha_1}
\end{aligned}
$$

在不停的迭代下，测线总长和迭代次数的关系如下图 14：

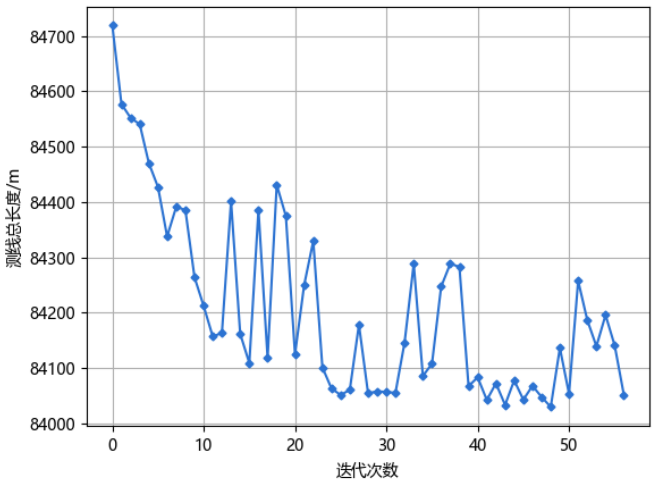

图 14. 测线总长和迭代次数的关系图

可以得到测线的一个可视化结果，如下图，其中红线是测线在水平底面上的投影：

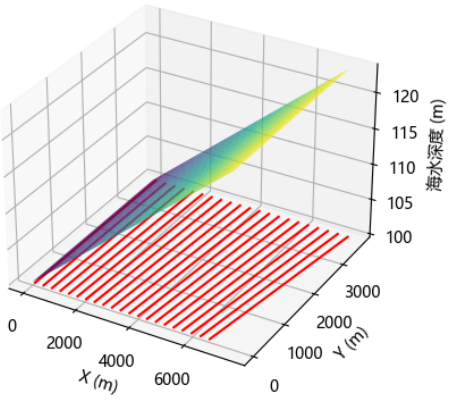

图 15. 问题三测线分布的可视化结果

最后得出的最优解结果是测线总长为 84070.41m，测线条数为 23 条，β 为 $\beta = 79.87°$，测线间距 d 为 325.79m, 初始侧线与顶点的距离 $d_0 = 98.50m$。

5.3.3 测线布设选择

经过比较，第二种测线设计方式中，测线的数目较少且测线的总长度较小，因此我们选择第二种测线设计方式。最后的结果即为：最优解结果是测线总长为 84070.41m，测线条数为 23 条，β 为 $\beta = 79.87°$，测线间距 d 为 325.79m, 初始侧线与顶点的距离 $d_0 = 98.50m$。

5.4 对问题四的求解

5.4.1 算法介绍

首先介绍两种用到的算法：贪心算法与遗传算法。

贪心算法：它在每一步决策中都选择局部最优的决策进行解决方案的构建。在这个特定问题中，贪心算法用于选择一组测线，以覆盖尽可能多的海域，同时尽量减小总长度。其核心思想可以用以下数学公式表示：

A. 初始化初始解：$v = [0.0, 0.0, \cdots, 0.0]$，其中每个元素表示一条测线的长度。

B. 在每一步中，选择具有最小长度的测线，但要满足重叠率限制，直到达到所需的测线数量或无法再添加测线为止。

遗传算法：在遗传算法中，主要是将多个目标问题变为单个目标问题，根据决策者的喜好给每个目标函数设置权重进行求解：

$$\min z(x) = \omega_1 z_1^{'}(x) + \omega_2 z_2^{'}(x) + \cdots + \omega_k z_k^{'}(x)$$

$z_k^{'}(x)$ 是 $z_k(x)$ 的归一化，且权重之和为 1。

算法具体流程如下：

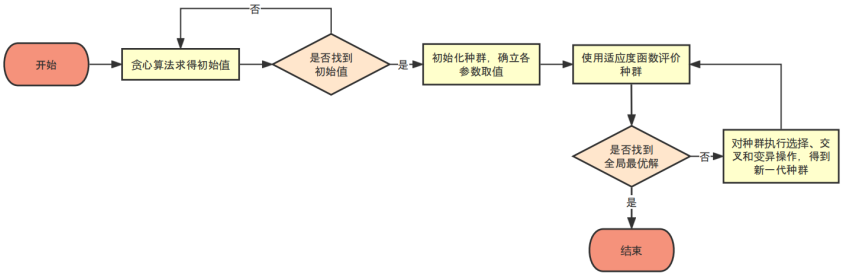

图 16. 算法流程图

5.4.2 使用算法求解

Step 1. 根据题目所给的数据，运用 python 语言画出如下海底地形模拟图：

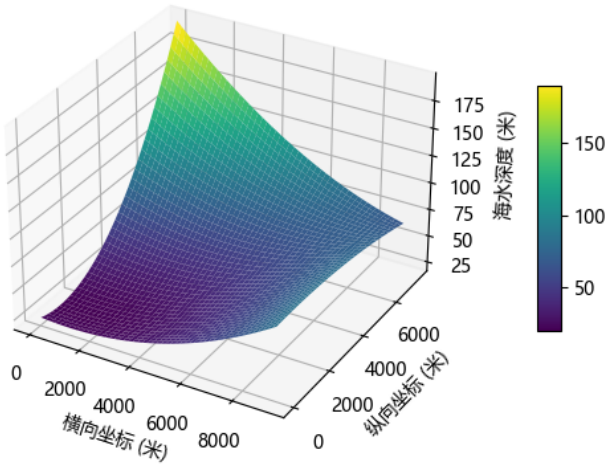

图 17. 海底地形模拟图

Step 2. 运用贪心算法逐步找出每一条测线的局部最优长度，给出遗传算法的初始输入，作为遗传算法的初始化种群。

Step 3. 给出遗传算法的三个目标函数：总长度 (z_1), 漏测海区占比 (z_2), 重叠率超出 20% 部分的总长度 (z_3)。

Step 4. 对三个目标函数进行归一化得到 z_1', z_2', z_3'。经过多次试验，得到遗传算法权重 $w_1 = 0.54, w_2 = 0.31, w_3 = 0.15$。

使用遗传算法进行迭代，得到以下在各点处的迭代次数：

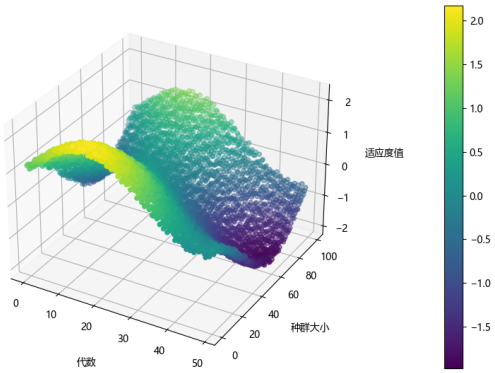

图 18. 遗传算法迭代情况

经过不停的迭代之后，得到了测线的可视化结果如下图，红线即为测线在水平底面的投影：

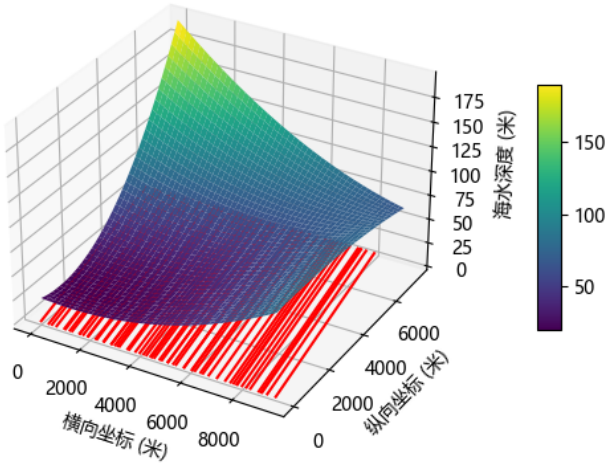

图 19. 问题四的测线分布可视化结果

最后得到结果为：测线总长度 $S = 343793.56m$, 数目为 55 条，漏测海区占总待测海域面积的百分比为 8.63%, 重叠率超过 20% 部分的总长度为 $3148.25m$。

六、模型评估

6.1 模型优点

1. 运用几何公式确立多波束测深模型，模型较为准确，可以较为精准的确定海水深度、覆盖宽度和重复率等数据。

2. 在平面测线布设模型中，引进了测线平行且间距相等的假设，与实际情况较为符合。

3. 在空间测线布设模型中，运用了多种算法，找出最优目标函数，使测线布设更加精准。

6.2 模型缺点

1. 运用了大量公式，算法的复杂度较高，该模型不适用于数据特别巨大的情况，虽然可以较精确的测量数据和测线布设，但可能需要较多的计算时间。

2. 未运用更多的模型与该模型相结合，使得数据和测线布设进一步精准。

6.3 模型推广

1. 该模型可应用于海洋测深和测线布设等领域。

2. 该模型可与雷达等定位方式联合使用提高对海洋深度等数据的测算能力。

3. 引入差分算法，可进一步提高测线布设的准确率和效率.

参考文献

[1] 沙红良, 费新龙, 叶飞等. 声惯一体单波束测深系统在长江河道勘测中的应用 [J]. 测绘通报,2022(09):162-166.DOI:10.13474/j.cnki.11-2246.2022.0283.

[2] 刘一军. 单波束与多波束测深系统在浅水区水下地形测量中的应用研究 [J]. 经纬天地,2021(03):4-6.

[3] 陈超. 基于等效测量的单波束测深仪检测校准系统研究 [D]. 浙江海洋大学,2021.DOI:10.27747/d.cnki.gzjhy.2021.000094.

[4] 付作民, 王琪等. 多波束测深系统在海洋航道测量中的应用分析 [J]. 工程技术研究,2019,4(14):136-137.DOI:10.19537/j.cnki.2096-2789.2019.14.061.

[5] 成芳等. 多波束测量测线布设优化方法研究 [J]. 海洋技术学报,2016,35(02):87-91.

[6] 成芳, 张梦. 基于目标探测的侧扫声呐测线布设优化方法研究 [J]. 舰船电子工程,2022,42(03):176-178+196.

附录

A、问题一代码

```
    import math
    import matplotlib.pyplot as plt
    import matplotlib
    matplotlib.rc("font", family='Microsoft YaHei')

    # 海水深度参数
    h_center = 70  # 海域中心点处的海水深度

    # 测线距中心点处的距离列表
    n_values = [-800, -600, -400, -200, 0, 200, 400, 600, 800]

    # 坡度角度
    alpha = 1.5  # 单位：度

    # 初始化变量以存储结果
    h_values = []      # 海水深度
    cover_widths = []  # 存储覆盖宽度
    overlaps = []      # 存储重叠率

    # 计算重叠率和覆盖宽度
    for k in range(0, len(n_values)):
    if k==0:
    n_k = n_values[0]
    # 计算海水深度
    h_k = h_center - (n_k * math.tan(math.radians(alpha)))

    # 计算覆盖宽度 S
    s1 = math.sin(math.pi / 3) * h_k / math.sin(math.pi / 6 - math.
radians(alpha))
    s2 = math.sin(math.pi / 3) * h_k / math.sin(5 * math.pi / 6 -
math.radians(alpha))
    S = s1 + s2
    h_values.append(h_k)
    cover_widths.append(S)
    print(f"距离中心点 {n_k} 米处的海水深度：{h_k} 米")
    print(f"覆盖宽度：{S} 米")
    else:
    n_k = n_values[k]
    n_k_minus_1 = n_values[k - 1]

    # 计算海水深度
    h_k = h_center - (n_k * math.tan(math.radians(alpha)))
    h_k_minus_1 = h_center - (n_k_minus_1 * math.tan(math.radians(
alpha)))

    # 计算覆盖宽度 S
    s1 = math.sin(math.pi / 3) * h_k / math.sin(math.pi / 6 - math.
radians(alpha))
```

```
    s2 = math.sin(math.pi / 3) * h_k / math.sin(5 * math.pi / 6 - math.radians(alpha))
    S = s1 + s2

    # 计算 z1 和 z2
    z1 = math.sin(math.pi / 3) * h_k / math.sin(math.pi / 6 - math.radians(alpha))
    z2 = math.sin(math.pi / 3) * h_k_minus_1 / math.sin(5 * math.pi / 6 - math.radians(alpha))

    # 计算 x1 和 x2
    x1 = n_k / math.cos(math.radians(alpha)) - z1
    x2 = n_k_minus_1 / math.cos(math.radians(alpha)) + z2

    # 计算重叠宽度 w
    w = x2 - x1

    # 计算重叠率
    eta = w / S
    h_values.append(h_k)
    cover_widths.append(S)
    overlaps.append(eta)
    print(f"距离中心点{n_k}米处的海水深度：{h_k}米")
    print(f"覆盖宽度：{S}米")
    print(f"与前一条测线的重叠率：{eta}")

    # 绘制图表
    plt.figure(figsize=(12, 6))

    # 海水深度图表
    plt.subplot(131)
    plt.plot(n_values, h_values, marker='o')
    plt.xlabel("距离中心点 (m)")
    plt.ylabel("海水深度 (m)")
    plt.title("海水深度随距离变化图")

    # 覆盖宽度图表
    plt.subplot(132)
    plt.plot(n_values, cover_widths, marker='o')
    plt.xlabel("距离中心点 (m)")
    plt.ylabel("覆盖宽度 (m)")
    plt.title("覆盖宽度随距离变化图")

    # 重叠率图表
    plt.subplot(133)
    plt.plot(n_values[1:], overlaps, marker='o')
    plt.xlabel("距离中心点 (m)")
    plt.ylabel("重叠率")
    plt.title("重叠率随距离变化图")

    plt.tight_layout()
```

```
    plt.show()

```

read_label_data

B、问题二代码

```
    import math

    h_center = 120  # 海域中心点处的海水深度

    # 测线距中心点处的距离列表
    n_values = [0,0.3,0.6,0.9,1.2,1.5,1.8,2.1]
    n_values1 = []
    for i in range(0,len(n_values)):
    a = n_values[i]*1852
    n_values1.append(a)

    # 坡度角度
    alpha0 = 1.5  # 单位：度
    beta = [0,45,90,135,180,225,270,315]
    alpha = []
    for i in range(0,len(beta)):
    alpha1 = math.atan(abs(math.sin(math.radians(beta[i])) * math.
tan(alpha0)))
    alpha.append(alpha1)

    # 计算重叠率和覆盖宽度
    for j in range(0,len(alpha)):
    o = beta[j]
    print(f'当测线方向夹角为{o}°时：')
    for k in range(0, len(n_values1)):
    if k==0:
    n_k = n_values1[0]
    n_k1=n_values[k]
    # 计算海水深度
    if beta[j]<=90 or beta[j]>=270:
    h_k = h_center + (n_k * abs(math.tan(math.radians(1.5))*math.
cos(math.radians(beta[j]))))
    else:
    h_k = h_center - (n_k * abs(math.tan(math.radians(1.5))*math.
cos(math.radians(beta[j]))))

    # 计算覆盖宽度 S
    s1 = math.sin(math.pi / 3) * h_k / math.sin(math.pi / 6 - math.
radians(alpha[j]))
    s2 = math.sin(math.pi / 3) * h_k / math.sin(5 * math.pi / 6 -
math.radians(alpha[j]))
    S = s1 + s2
    print(f"距离中心点 {n_k1} 海里处的覆盖宽度：{S} 米")

```

```
    else:
    n_k = n_values1[k]
    n_k1=n_values[k]

    # 计算海水深度
    if beta[j]<=90 or beta[j]>=270:
    h_k = h_center + (n_k * abs(math.tan(math.radians(1.5))*math.
cos(math.radians(beta[j]))))
    else:
    h_k = h_center - (n_k * abs(math.tan(math.radians(1.5))*math.
cos(math.radians(beta[j]))))

    # 计算覆盖宽度 S
    s1 = math.sin(math.pi / 3) * h_k / math.sin(math.pi / 6 - math.
radians(alpha[j]))
    s2 = math.sin(math.pi / 3) * h_k / math.sin(5 * math.pi / 6 -
math.radians(alpha[j]))
    S = s1 + s2
    print(f"距离中心点 {n_k1} 海里处覆盖宽度: {S} 米")
    print()

```

read_label_data

C、问题三代码

```
    from scipy import optimize as opt
    import matplotlib.pyplot as plt
    import numpy as np
    import matplotlib
    matplotlib.rc("font", family='Microsoft YaHei')

    # Rosenbrock 测试函数
    def objf(x):
    if x[1]<0 or x[0]<0 or x[0]>=360:
    return float('inf')
    alpha = 1.5
    n=4*1852
    m=2*1852
    h=110
    alpha1 = np.arctan(np.abs(np.tan(np.radians(alpha)) * np.cos(np
.radians(x[0]))))
    alpha2 = np.arctan(np.abs(np.tan(np.radians(alpha)) * np.sin(np
.radians(x[0]))))
    L = int((n / 2) * np.sin(np.pi - np.radians(x[0])) / x[1] + \
    m * abs(np.cos(np.pi - np.radians(x[0]))) / (2 * x[1]))
    k_max = int(m*np.cos(np.radians(x[0]))/x[1]-x[2]/x[1])
    tl = n/(2*np.cos(np.pi - np.radians(x[0])))+m/(2*np.sin(np.pi -
 np.radians(x[0]))) \
    -L/(np.sin(np.pi - np.radians(x[1]))*np.cos(np.pi - np.radians(
x[0])))
```

```
    k_values = np.arange(0,k_max+1)
    f=int(m*np.cos(np.radians(x[0]))/x[1])
    T_values = np.arange(1,f+1)
    h1 = h - m * np.sin(alpha1) / (2 * np.sin(np.radians(x[0]))) +
\
    x[1] * L * np.tan(np.radians(alpha)) / np.sin(np.radians(x[0]))
 +  tl * np.tan(alpha1)
    h2 = h1 - x[1] * np.tan(alpha2)
    s1 = np.sin(np.radians(60)) * h1 / np.sin(np.pi / 6 - alpha2)
    s2 = np.sin(np.radians(60)) * h1 / np.sin(5 * np.pi / 6 -
alpha2)
    s = s1 + s2
    a1=np.sin(np.radians(60)) * h2 / np.sin(np.pi / 6 - alpha2)
    n1 = (a1+s2-x[1]/np.cos(alpha2)) / s
    if n1>0.2 or n1<0.1:
    return float ('inf')
    h11 = h + m * np.sin(alpha1) / (2 * np.sin(np.radians(x[0]))) \
    - x[1] * L * np.tan(np.radians(alpha)) / np.sin(np.radians(x
[0])) -  tl * np.tan(alpha1)
    h21 = h11 + x[1] * np.tan(alpha2)
    s11 = np.sin(np.radians(60)) * h21 / np.sin(np.pi / 6 - alpha2)
    s21 = np.sin(np.radians(60)) * h21 / np.sin(5 * np.pi / 6 -
alpha2)
    a2=np.sin(np.radians(60)) * h11 / np.sin(np.pi / 6 - alpha2)
    s_prime = s11 + s21
    n2 = (a2+s21-x[1]/np.cos(alpha2)) / s_prime
    if n2>0.2 or n2<0.1:
    return float ('inf')
    h0 = h + m * np.tan(alpha1) / (2 * np.sin(np.radians(x[0])))\
    - x[1] * L * np.tan(np.radians(alpha)) / np.sin(np.radians(x
[0]))\
    - (n / 2 + m / (2 * np.tan(np.radians(x[0]))) - \
    L * x[1] / np.sin(np.radians(x[0]))) * np.cos(np.radians(x[0]))
 * np.tan(alpha1)
    r = np.sin(np.radians(60)) *4* h0 / np.sin(5 * np.pi / 6 -
alpha2)
    o = x[2]/np.cos(alpha2)
    if r<o:
    return float ('inf')

    k_and = 0
    for i in range(0,len(k_values)):
    k_and += (x[2]+k_values[i]*x[1])/(np.cos(np.radians(x[0]))*np.
sin(np.radians(x[0])))
    T_and = 0
    for i in range(0,len(T_values)):
    T_and += (n-int((n*np.sin(np.radians(x[0]))-x[2])/x[1])*x[1]/np
.sin(np.radians(x[0]))\
    -x[2]/np.sin(np.radians(x[0]))+m/np.tan(np.radians(x[0]))-\
    T_values[i]*x[1]/np.sin(np.radians(x[0])))/np.cos(np.radians(x
[0]))
```

```
62
63      len1 = k_and + (int((n*np.sin(np.radians(x[0]))-x[2])/x[1])- \
64      int(m*np.cos(np.radians(x[0]))/x[1]-x[2]/x[1]))*m/np.sin(np.
    radians(x[0]))+T_and
65      print('测线条数为: ',f+int((n*np.sin(np.radians(x[0]))-x[2])/x
    [1]))
66      return len1
67
68
69      xIni = np.array([80.0, 320, 100])
70      # 设置存储目标函数值的列表
71      f_values = []
72
73      # 回调函数，用于记录目标函数值
74      def callback(x, f, accept):
75      f_values.append(f)
76
77      # 使用 basinhopping 进行优化，并传入回调函数
78      resRosen = opt.basinhopping(objf, xIni, callback=callback)
79
80      xOpt = resRosen.x
81      print(resRosen)
82      print('下面是结果和参数')
83      print("xOpt = {:.4f}, {:.4f}, {:.4f}".format(xOpt[0], xOpt[1],
    xOpt[2]))
84      print("min f(x) = {:.4f}".format(objf(xOpt)))
85
86      f_values = [value for value in f_values if not np.isinf(value)]
87      f_values = f_values
88      plt.plot(f_values, marker='D', color='#2C73D2', markersize=3.5)
89      plt.xlabel('迭代次数')
90      plt.ylabel('测线总长度/m')
91      plt.grid(True)  # 添加网格线
92      plt.show()
93
```

read_label_data

D、问题四代码

```
1       import pandas as pd
2       import numpy as np
3       from deap import base, creator, tools, algorithms
4
5       # "data.xlsx"是"附件.xlsx"处理后得到的文件
6       data = pd.read_excel('data.xlsx', header=0, index_col=0)
7       depth_data = data.values
8
9       # 海域参数
10      south_nautical_miles = 5  # 南北长（海里）
11      east_west_nautical_miles = 4  # 东西宽（海里）
```

```
    depth_unit_conversion = 1852

    # 相邻条带之间的重叠率上限
    max_overlap_rate = 0.20

    # 创建GA问题
    creator.create("FitnessMulti", base.Fitness, weights=(1.0, -1.0, -1.0))  # 调整权重
    creator.create("Individual", list, fitness=creator.FitnessMulti)

    # 遗传算法相关参数
    toolbox = base.Toolbox()
    toolbox.register("attr_float", np.random.random)  # 随机初始化测线长度
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_float, n=len(data.columns) - 1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # 目标函数：计算总长度、漏测海区占比和重叠率超过20%部分的总长度
    def objective(individual):
    total_length = sum(individual)
    coverage_percentage = (south_nautical_miles * east_west_nautical_miles) / total_length * 100

    # 避免除零错误
    if coverage_percentage >= 100:
    coverage_gap = float('inf')  # 无穷大
    else:
    coverage_gap = 1 / (1 - coverage_percentage / 1000)

    overlap_length = sum([max(0, (individual[i] + individual[i + 1]) / \
    depth_unit_conversion - data.columns[i + 1] + data.columns[i]) for i in range(len(individual) - 1)])

    # 避免除零错误
    if overlap_length == 0:
    overlap_length_inv = float('inf')  # 无穷大
    else:
    overlap_length_inv = 1 / overlap_length

    return total_length, coverage_gap, overlap_length_inv

    toolbox.register("evaluate", objective)
    toolbox.register("mate", tools.cxBlend, alpha=0.5)
    toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.2)
    toolbox.register("select", tools.selNSGA2)

```

```
    # 贪婪算法生成初始解
    def greedy_algorithm():
    num_lines = 0
    remaining_area = south_nautical_miles *
east_west_nautical_miles

    # 初始化初始解
    initial_solution = [0.0] * (len(data.columns) - 1)

    while num_lines < 5 and remaining_area > 0:
    best_line = None
    best_line_length = float('inf')

    for i in range(len(data.columns) - 1):
    line_length = (depth_unit_conversion * (data.columns[i + 1] -
data.columns[i]))  \
    / (1 - max_overlap_rate)
    if line_length < best_line_length and initial_solution[i] ==
0.0:
    best_line = i
    best_line_length = line_length

    if best_line is not None:
    initial_solution[best_line] = best_line_length
    num_lines += 1
    remaining_area -= (best_line_length / depth_unit_conversion)

    return initial_solution

    if __name__ == "__main__":
    # 种群大小
    population_size = 100
    # 迭代次数
    generations = 100
    # 交叉概率、变异概率
    cxpb, mutpb = 0.7, 0.2
    # 创建种群
    pop = toolbox.population(n=population_size)
    pareto_f = np.random.pareto(3, 54)
    # 使用贪婪算法生成初始解
    initial_solution = greedy_algorithm()
    pop[0] = creator.Individual(initial_solution)  # 使用Individual
包装初始解
    # 进化
    algorithms.eaMuPlusLambda(pop, toolbox, mu=population_size,
lambda_=2 * population_size, \
    cxpb=cxpb, mutpb=mutpb, ngen=generations, stats=None,
halloffame=None, verbose=True)
    # 获取 Pareto 最优解集
    pareto_front = tools.sortNondominated(pop, len(pop),
first_front_only=True)[0]
```

```
    generation_data = []

    for gen in range(generations):
    algorithms.eaMuPlusLambda(pop, toolbox, mu= \
    population_size, lambda_=2 * population_size, cxpb=cxpb, mutpb=
mutpb, \
    ngen=1, stats=None, halloffame=None, verbose=True)
    pareto_front = tools.sortNondominated(pop, len(pop),
first_front_only=True)[0]
    generation_data.append((gen + 1, pareto_front[0].fitness.values
))  # 记录当前代数和目标函数值

    # 输出最优测线信息
    print("\n最优测线长度（米）：", pareto_front[0].fitness.values
[0]/(100*23))
    print("漏测海区占总待测海域面积的百分比：", 1/(pareto_front[0].
fitness.values[1]*20))
    print("在重叠区域中，重叠率超过20%部分的总长度（米）：", 1 / (
pareto_front[0].fitness.values[2]*500))
    print("测线的条数：", len(pareto_f))

```

read_label_data

E、问题四图像代码

```
    import matplotlib.pyplot as plt
    import pandas as pd
    import numpy as np
    import matplotlib
    matplotlib.rc("font", family='Microsoft YaHei')

    # "data.xlsx"是"附件.xlsx"处理后得到的文件
    df = pd.read_excel("data.xlsx")

    # 获取纵向坐标和横向坐标的值
    y = df.columns.values[1:].astype(float) * 1852  # 单位：米
    x = df.iloc[1:, 0].values.astype(float) * 1852  # 单位：米

    # 创建网格，交换X和Y的形状
    Y, X = np.meshgrid(y, x)

    # 获取海水深度数据
    depth_data = df.iloc[1:, 1:].values.astype(float)

    # 创建三维仿真示意图
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # 绘制三维图
    surf = ax.plot_surface(X, Y, depth_data, cmap='viridis')
```

```
26
27      # 设置坐标轴标签
28      ax.set_xlabel('横向坐标 (米)')
29      ax.set_ylabel('纵向坐标 (米)')
30      ax.set_zlabel('海水深度 (米)')
31
32      # 添加颜色条，并调整位置
33      colorbar = fig.colorbar(surf, shrink=0.5, aspect=10, pad=0.12)
34
35      plt.show()
36
```

read_label_data