西南大学

数学建模校内竞赛论文

论文题目：

组号：4

成员：刘康、刘明华、彭美妮

选题：B

姓名	学院	年级	专业	学号	联系电话	建模获奖经历
刘康	计算机与信息科学学院 软件学院	2022	计算机科学与技术（普通）	222022321182035	13193810723	无
刘明华	计算机与信息科学学院 软件学院	2023	计算机科学与技术（普通）	222023321182062	13095088965	2023 年美赛 S 奖
彭美妮	数学与统计学院	2023	数学类	222023314210077	19936400086	2023 年美赛 S 奖

日期：2024 年 8 月 11 日

基于覆盖宽度模型的海底测线设计

摘要

随着多波束测深系统的发展,侧线设计问题是海域地形测量中的重要研究课题。本文针对多波束探测海底地形的测线设计问题，基于解析几何思想,通过确定覆盖宽度、两相邻测线重叠率等指标，以测线总长短、漏测率低、重叠率低于20%为目标，建立了覆盖宽度模型和测线设计模型。

针对问题一，我们根据题目所给的覆盖宽度和重叠率的初始定义，重新定义坡面上的覆盖宽度和重叠率，通过**解析几何**和正弦定理等数学知识，找到各变量之间的函数关系式，建立了**覆盖宽度和重叠率之间的数学模型**。最后带入题目所给数据，求解离中心处各距离的海水深度、覆盖宽度和重叠率，将结果存放在result1.xlsx。

针对问题二，我们以海域中心处为原点建立了空间直角坐标系，同样利用解析几何知识和**向量分析**，求出每点的海水深度、侧线垂直方向与水平面的夹角，再带入题目所给参数，根据问题一中定义的覆盖宽度，则可求得不同的测线方向和海底坡面法向量在水平面的投影的夹角β下，离中心处不同距离的点的覆盖宽度，建立了**覆盖宽度模型**。将求解结果存放在result2.xlsx中。

针对问题三，则是对问题二覆盖宽度模型的实际应用。对于一片矩形海域，首先，根据问题二的求解结果，我们证明测线方向**与等深线方向共线时**设计侧线最佳。其次，考虑到要求测线最短，则重叠率应固定为**10%**。然后，从海域边缘开始计算第一条测线，依次递推出覆盖整片海域的所有测线，建立出**测线设计模型**。最终求解得到设计的测线为**34条**，测线总长度为**68海里**。

针对问题四，我们首先用插值画出该海域**等深线图**，发现海底坡度变化大，故根据等深线进行海域划分，再对每个小区域单独设计测线。将区域映射到平面上，利用**最小二乘法**求坡面直线的斜率，取绝对值后求出坡度α，再求拟合平面中心处的海水深度，则可利用问题三的测线设计模型。最后计算设计的这组测线总长度为**241.0海里**，漏测率为**0**，重叠率高于20%的长度为**3.5海里**。

最后,我们对建立的模型进行了**灵敏度分析**，发现模型稳定性好。然后进行全面的评价：本文的模型贴合实际,能合理解决提出的问题,具有实用性强,算法简单等特点,该模型在建筑测绘、地形探测等方面也能使用。

关键词：　解析几何　向量分析　等深线　最小二乘法　测线设计

一、问题重述

1.1 问题背景

单波束测深技术通过声波在水中的传播特性来测量水体深度，具有单点连续测量的特点，但在测线间没有数据，限制了其测量的全面性和效率。为了克服这些缺点，多波束测深技术应运而生，通过一次发射多个波束，实现了对海底地形的全覆盖测量，显著提高了测量效率和数据完整性。然而，在复杂地形或大范围海域的测量中，如何合理设计测线，确保测量数据既全面又高效，成为了一个重要的研究课题。

1.2 问题提出

问题一：在给定换能器开角、坡度及海水深度的条件下，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。旨在计算并展示不同位置处的覆盖宽度及与前一条测线的重叠率。

问题二：考虑一个矩形待测海域，建立多波束测深覆盖宽度的数学模型，考虑测线方向与海底坡面法向的夹角。旨在计算并展示不同位置及测线方向夹角下的覆盖宽度。

问题三：在给定海域大小、水深分布及坡度条件下，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。旨在提供一组优化的测线设计方案。

问题四：用某海域的单波束测深数据，为多波束测量船的测量布线提供帮助，确保测线设计满足覆盖全面、重叠率控制及测线长度最短的要求。目的是计算测线的总长度；计算漏测海区占总待测海域面积的百分比；计算重叠区域中重叠率超过 20%部分的总长。

二、模型假设与符号说明

2.1 模型基本假设

(1) 所给数据真实有效，可以用于建立模型。

(2) 声波传播过程中无物体遮挡，沿直线匀速传播。

(3) 船运动的海面水平，没有不可到达的地方。

(4) 一海里为 1852m。

2.2 符号说明

表 1 符号说明

符号	含义	单位
D_i	海水深度	m
D_0	海域中心点处的海水深度	m
d_i	第i条测线距中心点处的距离	m
α	海底坡面的坡度	°
W_i	第i条测线的覆盖宽度	m
W_{i1}	第i条测线左半部分的覆盖宽度	m
W_{i2}	第i条测线右半部分的覆盖宽度	m
θ	多波束换能器的开角	°
η_i	相邻条带的重叠率	
Δd	相邻条带之间的距离	m
β	测线与坡面法向量在水平面投影的夹角	°
a_i	第i条测线在海底坡面的投影到中心点处的距离	m
L	区域最后一条测线的条带与区域边缘相差部分	m

三、问题分析

问题一：要求建立多波束测深的覆盖宽度和相邻条带之间重叠率的数学模型。我们先根据题目所给的初始覆盖宽度和重叠率的定义，给出二者新的定义式，再通过解析几何与正弦定理等数学知识，找出各变量之间的函数关系。再将题目所给的开角、中心处海水深度和坡度值，求解离中心处各距离的海水深度、覆盖宽度和重叠率，再填入表格。

问题二：给了一个矩形海域，海域坡面与水平面有夹角，测线方向与海底坡面法向量在水平面的投影的夹角为β。题目要求我们建立覆盖宽度的数学模型，则我们仍旧利用问题一所定义的覆盖宽度和重叠率，那么只需求发射波束的该点的海水深度、测线垂直方向与水平面的夹角，再带入题目所给的开角、坡度和中心处海水深度，则可求得不同β下，离中心处不同距离的点的覆盖宽度。

问题三：给了一片矩形海域，要求我们设计一组重叠率在 10%到 20%、可覆盖整片海域且总长度最短的测线。是问题二模型的实际应用。对于测线的方向，我们需先证明测线方向与等深线方向共线时最好。然后为了让总长度最短，我们应取重叠率为 10%，从第一条测线的条带与海域边缘重叠开始计算，运用问题二的模型，一条条确定，再对最后一条测线计算能否覆盖海域的边缘，从而得到题目所需的一组测线。

问题四：给出了一片海域的单波束测量出的海底深度数据，要求我们设计一组用于多波束测深的测线，同时保证尽量少漏测、重叠率低于 20%、尽量覆盖整个海域且总

长度最短。因为海水深度变化大，则我们应根据等深线对海域进行矩形划分，再对各个小区域近似拟合为平面，则可利用问题二所建立模型和问题三的实际应用，对该海域进行测线设计。求出拟合平面与水平面的夹角和平面中心处的海水深度，再通过插值求出各点所需海水深度，即可得到每个区域的测线设计。考虑到各个相交区域的交界处的重叠率会过高，故再合理删除几条测线，最后计算漏测率、重叠率高于 20%的长度和测线总长度，得到最优的一组测线。

四、模型建立与求解

4.1 问题一模型建立与求解

4.1.1 问题一求解思路

由于海底不再是平面而是坡面，则原本题目定义的多波束测深条带的覆盖宽度和相邻条带之间的重叠率不再适用，因为侧线左右的覆盖宽度不相同了。并且海水深度也在变化，故在原先定义的基础上，进行新的定义。由此建立覆盖宽度与重叠率的数学模型，再根据题目所给开角、坡度和海域中心点处的水深度，利用建立的模型计算所需指标值。

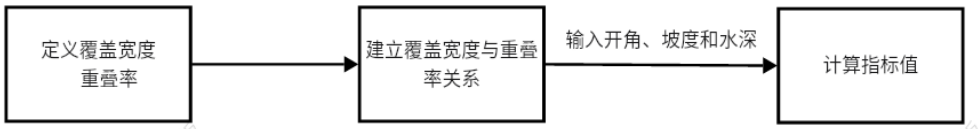

图1 问题一流程图

4.1.2 问题一模型建立

(1) 定义海水深度D_i

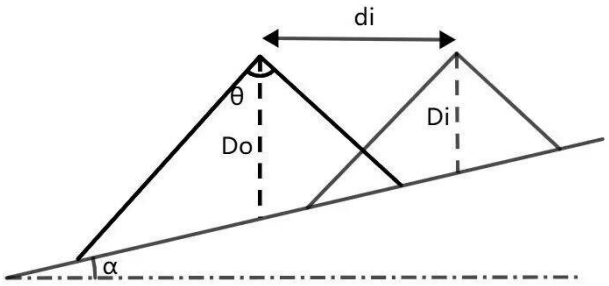

图2

$$D_i = D_0 - d_i tan\alpha \tag{1}$$

如图 1 其中，i表示第i条测线，我们令海域中心点处的侧线为第 0 条测线，则$i = -4, -3, \dots, 3, 4$。D_0表示海域中心点处的海水深度，在问题一中值为 70m。d_i为第i条测线距中心点处的距离。α为与测线方向垂直的平面和海底坡面的交线与水平面的夹角，即坡度，在问题一中值为1.5°。

(2) 定义覆盖宽度W_i

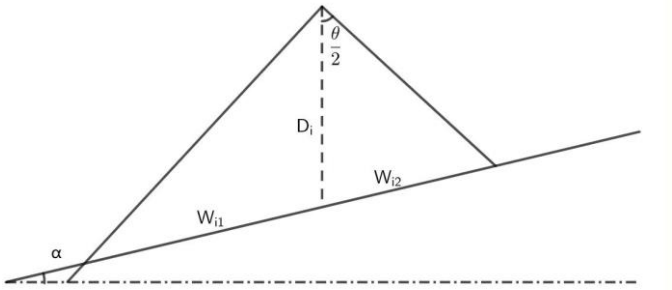

图3

令测线左半部分的覆盖宽度为W_{i1}，右半部分的覆盖宽度为W_{i2}，则测线的覆盖宽度为

$$W_i = W_{i1} + W_{i2} \tag{2}$$

由图 2 ，根据正弦定理，可得

$$\frac{Di}{sin\left(\frac{\pi}{2} - \frac{\theta}{2} - \alpha\right)} = \frac{W_{i1}}{sin\frac{\theta}{2}}$$

$$\frac{Di}{sin\left(\frac{\pi}{2} - \frac{\theta}{2} + \alpha\right)} = \frac{W_{i2}}{sin\frac{\theta}{2}}$$

其中θ为多波束换能器的开角，在问题一中值为120°。故两式化简可得

$$W_{i1} = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)} \tag{3}$$

$$W_{i2} = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)} \tag{4}$$

(3) 定义重叠率η

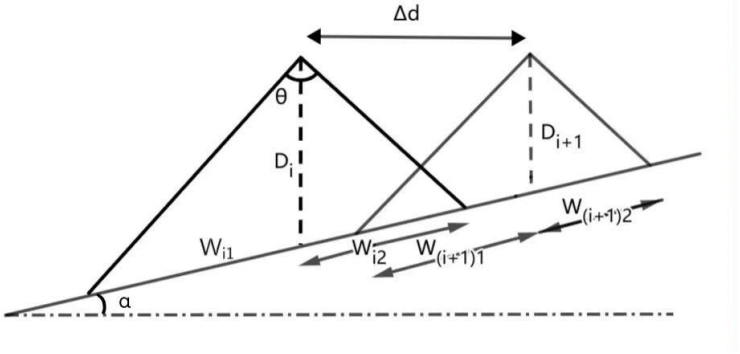

图4

由题目可知，初始的相邻条带的重叠率定义为$\eta = 1 - \frac{d}{W}$，我们在初始定义的基础上，考虑到海底为坡面时测线左右部分的覆盖宽度不同，因此引入了新的重叠率定义，即

$$\eta_{i+1} = \frac{W_{i2} + W_{(i+1)1} - \frac{\Delta d}{\cos\alpha}}{W_i} \qquad i = -4, \ldots, 3$$

其中，Δd表示两条相邻测线的距离，即 200m。将$W_{i2}, W_{(i+1)1}$带入，得到

$$\eta_{i+1} = \frac{\frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)} + \frac{D_{i+1} \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)} - \frac{\Delta d}{\cos\alpha}}{W_i} \tag{5}$$

综上，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为：

$$D_i = D_0 - d_i tan\alpha$$

$$W_i = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)} + \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)}$$

$$\eta_{i+1}=\frac{\frac{D_i\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)}+\frac{D_{i+1}\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)}-\frac{\Delta d}{\cos\alpha}}{W_i}\quad i=-4,\ldots,4$$

4.1.3 问题一模型求解与分析

根据建立的模型，运用 Python 编写代码求解。多波束换能器的开角为 120°，坡度为 1.5°，海域中心点处的海水深度为 70m，求得结果如表 2 所示，并保存到了 result1.xlsx 文件中。

表 2

测线距中心点处的距离/m	-600	-400	-200	0	200	400	600
海水深度/m	85.71	80.47	75.24	70	64.76	59.53	54.29
覆盖宽度/m	297.61	279.43	261.25	243.07	224.89	206.71	188.53
与前一条测线的重叠率/%	33.64	29.58	25	19.78	13.78	6.81	-1.38

由结果可见，随着测线距中心点处的距离增加，海水深度在变浅，导致开角不变的情况下，波束的覆盖宽度减小，因此相邻测线的重叠率减小，在测线距中心点处-600m 到-200m，重叠率超过 20%，测量效率低。而在 400m 及 600m 时，重叠率低于 10%甚至为负数，即有漏测的部分，影响了测量质量。

4.2 问题二模型建立与求解

4.2.1 问题二求解思路

因为沿同一测线方向，覆盖宽度在变化，同时海水深度也在变化。也就是说，在问题一二维的基础上，变为了三维。因此要建立多波束测深覆盖宽度的数学模型，我们先作测线在海底坡面上的投影，求出此投影与测线在水平面上的投影所成角，然后可求得测线方向上每点的海水深度。再做测线垂直方向在坡面的投影，该求出投影与测线垂直方向在水平面的投影的所成角，则可求出该点的波束的覆盖宽度。由此成功建立了多波束测深覆盖宽度的数学模型。

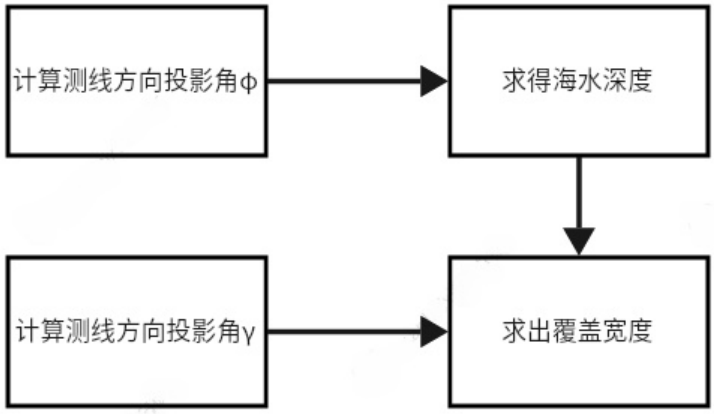

图5 问题二流程图

4.2.2 问题二模型建立

以海域中心在水平面上的点为原点，坡面法向量$\vec{n}$在水平面的投影向量为 x 轴，x 轴逆时针旋转 90°为 y 轴，垂直 xOy 平面向上作 z 轴，建立空间直角坐标系。再作测线在海底坡面上的投影，与测线在水平面的投影所成角为φ。由坡度α，易得坡面法向量$\vec{n}$与 z 轴所成角也为α，不妨设$\vec{n} = (sin\alpha, 0, cos\alpha)$。如图 4。

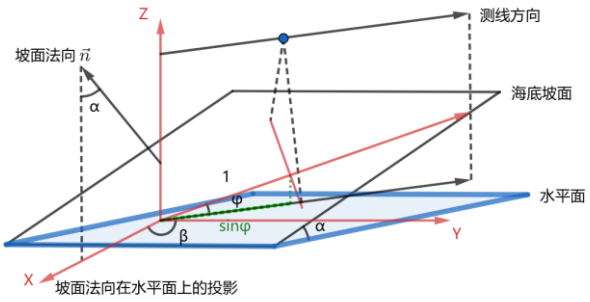

图6

(2) 求测线在坡面的投影与在水平面的投影的所成角φ

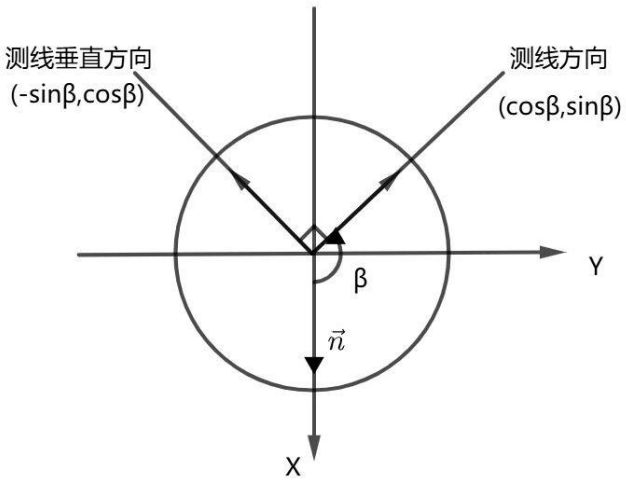

图7

x0y 平面如图 5。易得测线在水平面上投影的方向向量为$(cos\beta, sin\beta)$，测线垂直方向的方向向量为$(-sin\beta, cos\beta)$。不妨令测线在坡面上的投影的方向向量$\vec{a}$长度为 1，则测线在坡面上的投影的方向向量$\vec{a} = (cos\beta cos\varphi, sin\beta cos\varphi, sin\varphi)$。因为$\vec{n} \perp \vec{a}$，故得

$$sin\alpha cos\beta cos\varphi + cos\alpha sin\varphi = 0$$

解得

$$tan\varphi = -tan\alpha cos\beta \tag{6}$$

(3) 求海水深度D_i

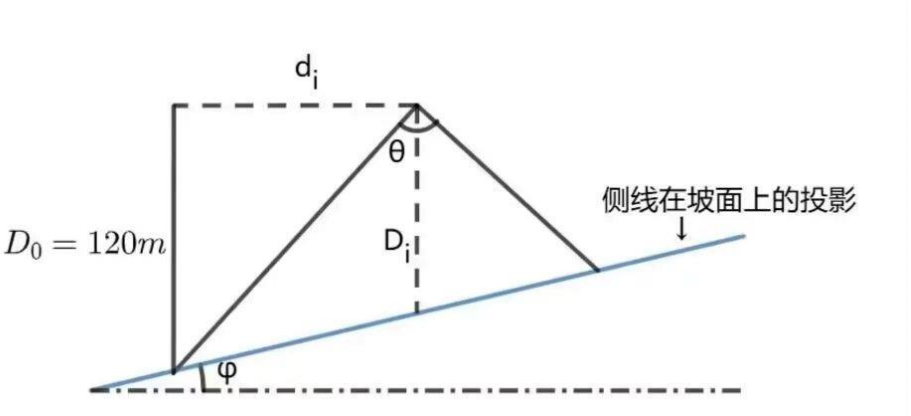

图8

由图 6 知，在与海域中心点处距离为d_i的点上，海水深度为

$$D_i = D_0 - d_i \tan\varphi$$

将(6)式带入，得

$$D_i = D_0 + d_i tan\alpha cos\beta \quad i = 0,1,\dots,7 \tag{7}$$

其中D_0为海域中心点处的海水深度，在问题二中的值为 120m。

(4) 求测线垂直方向在坡面的投影与水平面的投影的所成角γ

如图 5。因为测线垂直方向的方向向量为$(-sin\beta, cos\beta)$，同理可设测线垂直方向在坡面的投影的方向向量$\vec{b}$的长度为 1，则$\vec{b} = (-cos\gamma sin\beta, cos\gamma cos\beta, sin\gamma)$。因为$\vec{n} \perp \vec{b}$，可得

$$-cos\gamma sin\beta sin\alpha + cos\alpha sin\gamma = 0$$

解得

$$tan\gamma = tan\alpha sin\beta \tag{8}$$

(5) 求覆盖宽度W_i

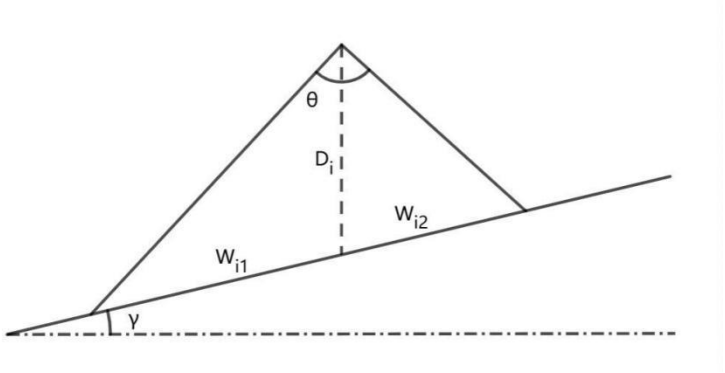

图9

由 4.1.2 所建立的模型可知：

$$W_i = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\gamma\right)} + \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\gamma\right)} \tag{9}$$

综上，问题二中所建立的多波束测深覆盖宽度的数学模型为：

$$D_i = D_0 + d_i tan\alpha cos\beta$$

$$tan\gamma = tan\alpha sin\beta$$

$$W_i = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\gamma\right)} + \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\gamma\right)} \quad i = 0,1,\ldots,7$$

4.2.3 问题二模型求解与分析

根据题目所给的条件，多波束换能器的开角θ为 120°，坡度α为 1.5°，海域中心点处的海水深度D_0为 120m，通过 Python 进行求解，得到结果如表 3 所示，同时保存到 result2.xlsx 文件中。

表 3

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	0	415.69	466.09	516.49	566.89	617.29	667.69	718.09	768.48
	45	416.19	451.87	487.55	523.23	558.91	594.59	630.27	665.95

夹角/°	90	416.69	416.69	416.69	416.69	416.69	416.69	416.69	416.69
	135	416.19	380.51	344.83	309.15	273.47	237.79	202.11	166.43
	180	415.69	365.29	314.89	264.50	214.10	163.70	113.30	62.90
	225	416.19	380.51	344.83	309.15	273.47	237.79	202.11	166.43
	270	416.69	416.69	416.69	416.69	416.69	416.69	416.69	416.69
	315	416.19	451.87	487.55	523.23	558.91	594.59	630.27	665.95

由结果分析可知，测线方向夹角为 90°和 180°时，即测线方向和坡面法向量在水平面上的投影垂直时，也就是船的测线在海域的等深线上时，海水深度不变，波束的覆盖宽度不变，则相邻两测线的重叠率也不变。而其他夹角的情况，覆盖宽度都在明显的变大或变小，则相邻测线的重叠率明显变大或变小，很有可能出现漏测的情况，或是重叠率不在 10%—20%的范围内。

4.3 问题三模型建立与求解

4.3.1 问题三求解思路

问题三本质上是问题二所建模型的实际应用，要求相邻条带的重叠率在 10%—20%内的条件下，求出一组可以覆盖整个海域并且总长度最短的测线。根据问题一和问题二的结果可知，随着船距中心点处的距离变化，海水深度变化，重叠率的变化很大，几百米的变化就会导致重叠率变化 10%。也就是说，在 2 海里（3704m）的长度范围内，就算最浅的深度满足了最低的 10%重叠率，那么最深的深度处的重叠率一定会超过 20%，不满足题目要求。而若是最深的深度满足了最高的 20%重叠率，则最浅的深度处的重叠率也无法满足 10%，甚至可能漏测。因此，最合理的方法应该是让船沿海域的等深线移动，也就是等深线作测线，即问题二中测线方向垂直的情况，此时海水深度不变，覆盖宽度不变，重叠率不变，可以很好的保持在 10%—20%的范围内，防止了漏测问题和数据冗杂问题。而要求总测线长度最短，那么就可取重叠率为 10%。

此时，问题三就变成了以等深线作测线，在重叠率为 10%的情况下，找一组覆盖全部海域的测线。我们就可先求出波束边缘与海域东侧重合的一条测线，再通过问题二的多波束测深覆盖宽度的数学模型和问题一中覆盖宽度与重叠率之间的数学模型，求得与第一调测线相邻的第二条测线的覆盖宽度和海水深度，依次递推，直到求到最后

一条测线的波束覆盖了海域西侧再停止，此时即求得一组满足条件的总长度最短的测线。

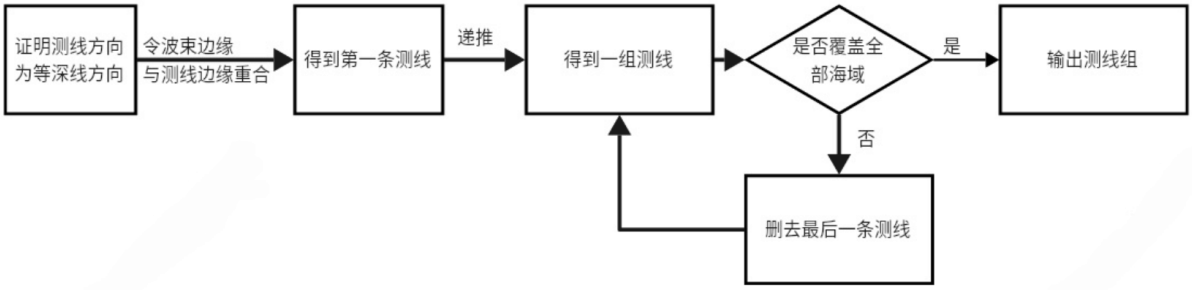

图10 问题三流程图

4.3.2 问题三模型建立

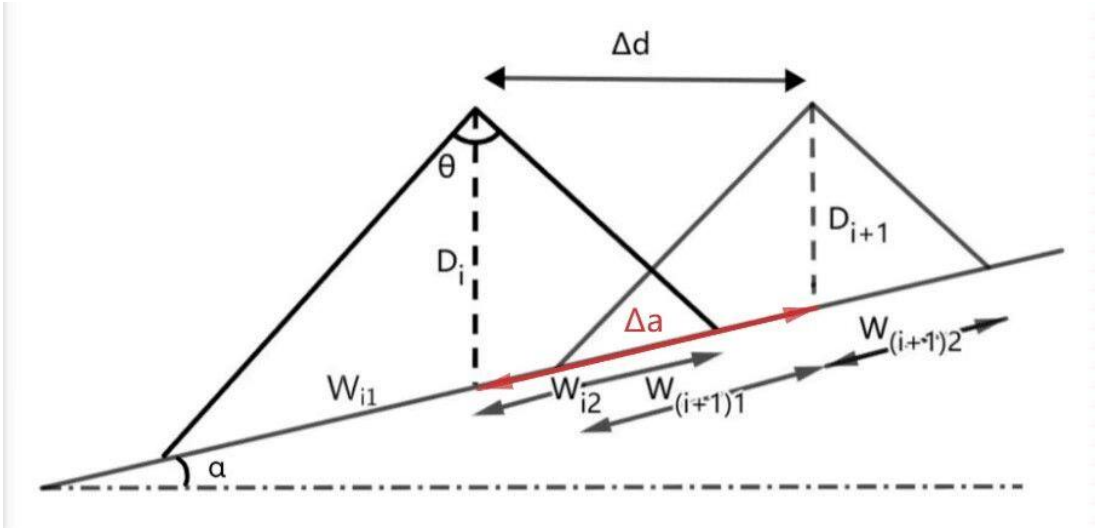

图11

以海底的海域中心点为原点，南北为 x 轴，东西为 y 轴，令向东为 x 轴正方向。南北长x为 2 海里，即 3704m，东西长y为 4 海里，即 7408m。i 为自西向东的第 i 条测线。由 4.1.2 中所定义的海底深度：

$$D_i = D_0 - d_i \tan\alpha$$

D_i表示第 i 条测线所在海域的海水深度。D_0表示海域中心点处的海水深度，在问题三中值为 110m。d_i表示第 i 条测线与中心点处的距离。α为坡度，在问题三中值为 1.5°。a_i表示第i条测线在海底坡面的投影到中心点处的距离，即有

$$a_i = \frac{d_i}{\cos\alpha} \tag{10}$$

因为第 1 条测线的西侧与海域的西侧重合，则有

$$\frac{D_i \sin\theta}{\cos\left(\frac{\theta}{2}+\alpha\right)} = w_{i1} = \frac{y}{2} + a_i \tag{11}$$

令$i=1$，可求得$a_1 = -3345.41$

再根据所定义的重叠率为 10%，由 4.1.2 中的(2)(3)(4)(5)式，有

$$\eta_{i+1} = \frac{\frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)} + \frac{D_{i+1}\sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)} - \Delta a}{W_i} = 10\% \tag{12}$$

$$W_i = \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}+\alpha\right)} + \frac{D_i \sin\frac{\theta}{2}}{\cos\left(\frac{\theta}{2}-\alpha\right)}$$

其中$\Delta a = a_{i+1} - a_i$，可求得a_2。以此类推，可求得$a_3, \ldots, a_n$。当$a_n \geqslant$3704m 时，停止迭代。

4.3.3 问题三模型求解与分析

通过 Python 进行求解，得到结果如表 4 。

表 4

测线	距离/m	测线	距离/m
a_1	-3345.41	a_{18}	2331.59
a_2	-2750.81	a_{19}	2478.96
a_3	-2203.04	a_{20}	2614.71
a_4	-1698.43	a_{21}	2739.78
a_5	-1233.58	a_{22}	2854.98
a_6	-805.34	a_{23}	2961.12
a_7	-410.85	a_{24}	3058.89
a_8	-47.43	a_{25}	3148.96
a_9	287.36	a_{26}	3231.93
a_{10}	595.77	a_{27}	3308.37
a_{11}	879.89	a_{28}	3378.78
a_{12}	1141.62	a_{29}	3443.65
a_{13}	1382.74	a_{30}	3503.41

a_{14}	1604.85	a_{31}	3558.46
a_{15}	1809.47	a_{32}	3609.17
a_{16}	1997.97	a_{33}	3655.89
a_{17}	2171.62	a_{34}	3698.92

对第 34 条测线的东侧部分的覆盖宽度进行计算，由 4.1.2 中(1)(4)式可得

$$W_{34,2} = \frac{D_{34}\, sin\frac{\theta}{2}}{cos\left(\frac{\theta}{2} - \alpha\right)}$$

$$D_{34} = D_0 - d_{34} tan\alpha$$

$$d_{34} = a_{34}\, cos\,\alpha$$

解得$W_{34,2} = 21.83$。又因为$a_{34} + W_{34,2} = 3720.75m > \frac{y}{2} = 3704m$，所以第 34 条测线的东侧可以覆盖海域的东侧，故 34 条测线已足够覆盖整个海域，由此求出了满足条件的总长度最短的一组侧线，总长度为$34x$，即 68 海里，取一海里为 1852m，故总长度为 125936m。

最后绘制出所求的这组测线，如图 9 所示。

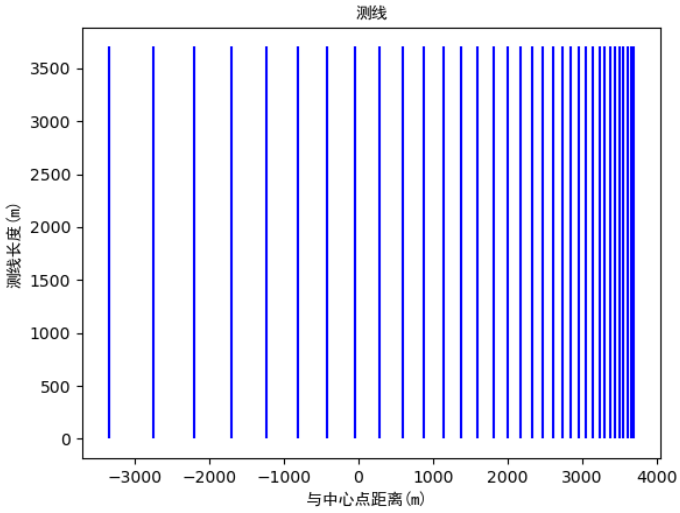

图12

横轴为测线与中心点的距离，以向东为正方向。纵轴为侧线长度，都为 3704m。由图中可见，测线在西侧分布稀疏，东侧分布密集，这与海水深度越深则重叠率越高相契合，可见我们的设计是合理的。

综上，我们成功设计了一组重叠率为 10%，沿等深线方向，长度最短为 125936m 且覆盖整个海域的一组测线，满足题目要求。

4.4 问题四模型建立与求解

4.4.1 问题四求解思路

根据题目所给的单波束测量的某海域的海水深度数据，利用插值补充未知点数据，画出等深线图。在通过等深线分布将海域进行矩形划分，得到几个矩形区域。用最小二乘法将这几个区域拟合为平面，求出平面与水平面的夹角和中心处的海水深度。然后，合理假设一个区域内的相邻测线重叠率为 10%，来进行测线设计。对相邻区域的两条测线重叠率过高的情况，适当删除几条测线后，得到最终测线的分布，计算漏测率、重叠率高于 20%的长度、测线总长度。

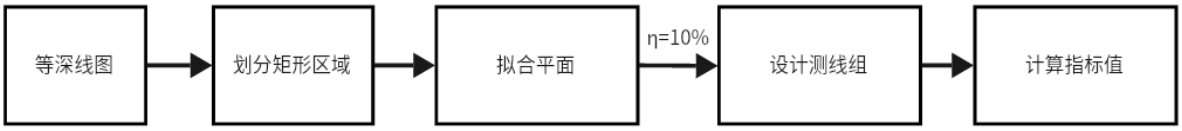

图13 问题四流程图

4.4.2 问题四模型建立

(1) 海域区域划分

我们根据附件.xlsx 中的坐标点，以附件中的(0,0)为原点，自西向东为 x 轴正方向，自南向北为 y 轴正方向，垂直于 xOy 向上为 z 轴，建立空间直角坐标系。

利用 Python 的 scipy.interpolate.griddata，通过已知数据点对该海域进行插值，求得未知点的海水深度，再绘制出海域的等深线图如图 10 所示，便于计算与分析。

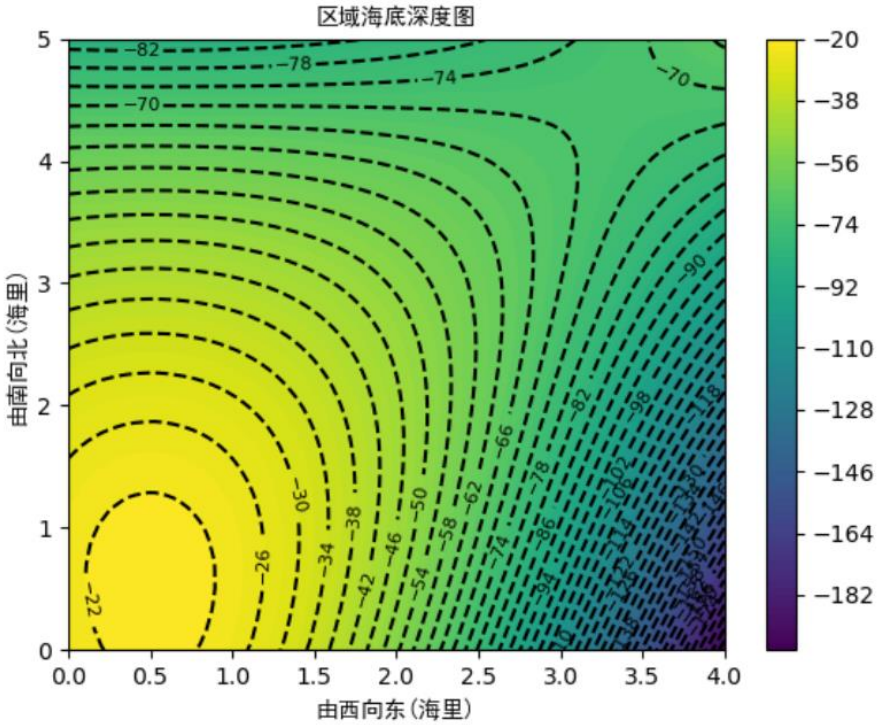

图14

对等深线图分析可知，海域的深度变化很大，那么对一整个海域进行测线设计的话，很容易出现在深度大的地方重叠率过高，而在深度小的地方出现漏测。因此我们对整片海域进行矩形划分，得到几个小区域，再分别进行如问题三的测线设计。为了达到划分出矩形区域的效果，我们尽量让划分的区域内的等深线呈平行状态且与矩形的边平行，由此划分出如图 11 所示的 6 块区域。

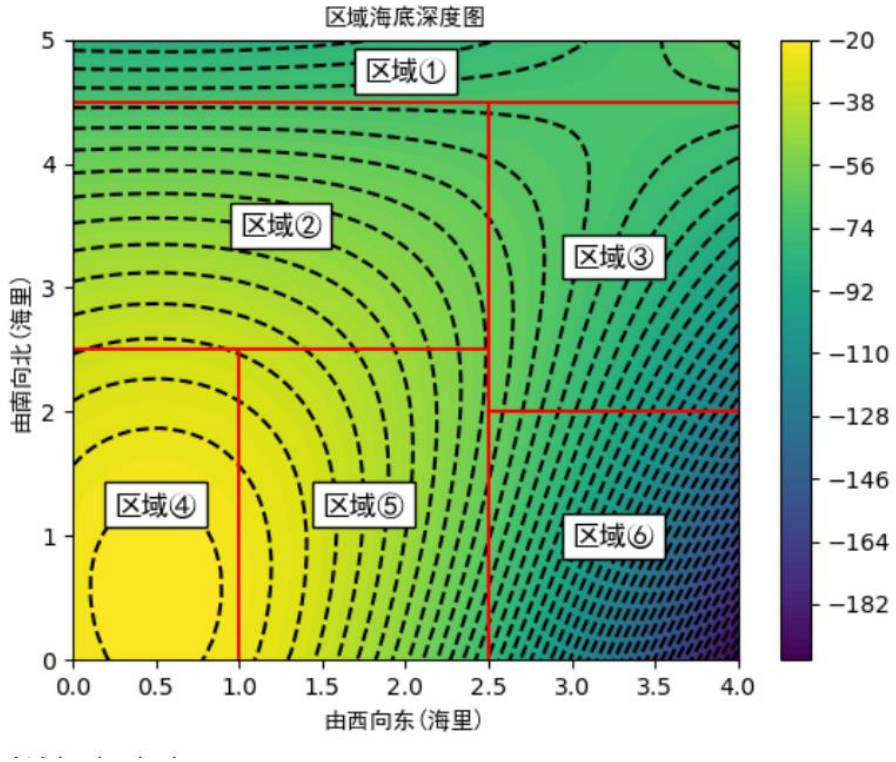

图15

(2) 区域拟合平面

我们将每个区域映射到 x0y 平面和 x0z 平面上，运用最小二乘法求坡面直线的斜率，求出坡面与水平面的坡度α和该平面中心点的海水深度D_0。然后可按照问题三的同样思路进行测线设计。我们依旧设定一个区域内的相邻测线的重叠率为 10%，这样放到实际里也不容易出现太多漏测或重叠率过高导致数据冗杂的问题。

(3) 漏测面积百分比 k、相邻区域边缘的测线重叠率η'

由前面的区域划分和测线设计思路可知，侧线方向不同的区域不会重叠，故重叠率只需计算侧线方向相同的两相邻区域。

对两个测线方向相同的相邻区域如图 12 所示，分别命名为区域 1 和区域 2。令区域 1 边最后一条的测线到中心点的距离为a，该测线左右部分的覆盖宽度分别为W_1', W_2'，垂

直测线方向的区域长度的一半为S。区域 1 的坡度为α_1，区域 2 的坡度为α_2。区域 2 的第一条的覆盖宽度为W。

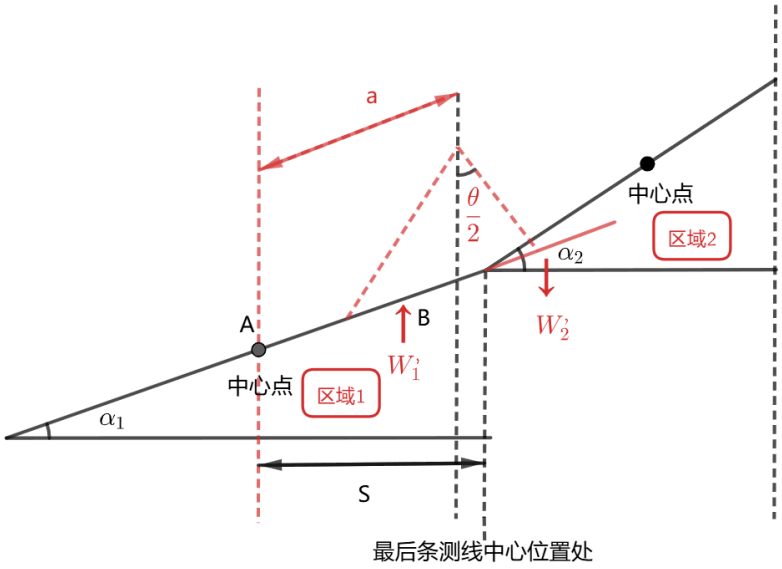

图16

则区域 1 最后一条测线的条带与区域 1 的边缘相差部分，称为相差长度L，有

$$L = a + W_2' - \frac{S}{cos\alpha_1} \tag{13}$$

若$L < 0$，说明最后一条测线的覆盖宽度没能覆盖该区域的边缘，出现了漏测部分。则漏测海区占总待测海域面积的百分比k有

$$k = \frac{L \cdot H}{S_0} \tag{14}$$

其中H为漏测区域的测线方向上的长度。S_0为总待测海域面积，在问题四中值为 4*5=20 海里。

若$L > 0$，说明最后一条测线的覆盖宽度超过了该区域，此时就可求重叠率了。实际中溢出部分的方向与区域 2 的测线覆盖宽度的方向并不共线，为了简化模型，我们假设其共线。则相邻区域边沿的测线重叠率η'为

$$\eta' = \frac{L}{W} \tag{15}$$

对重叠率η'大于 20%的长度求出来既可。此时，满足需要的一组测线已成功设计，且测线条数、测线总长度、漏测面积百分比、重叠率超 20%部分的总长度都可求出。

4.4.3 问题四模型求解与分析

对每个区域运用问题三的测线设计方案，通过 Python 进行求解出测线后计算，得到测线组总计 133 条测线，总长度为 241.0 海里。测线分布图如图 13 所示。

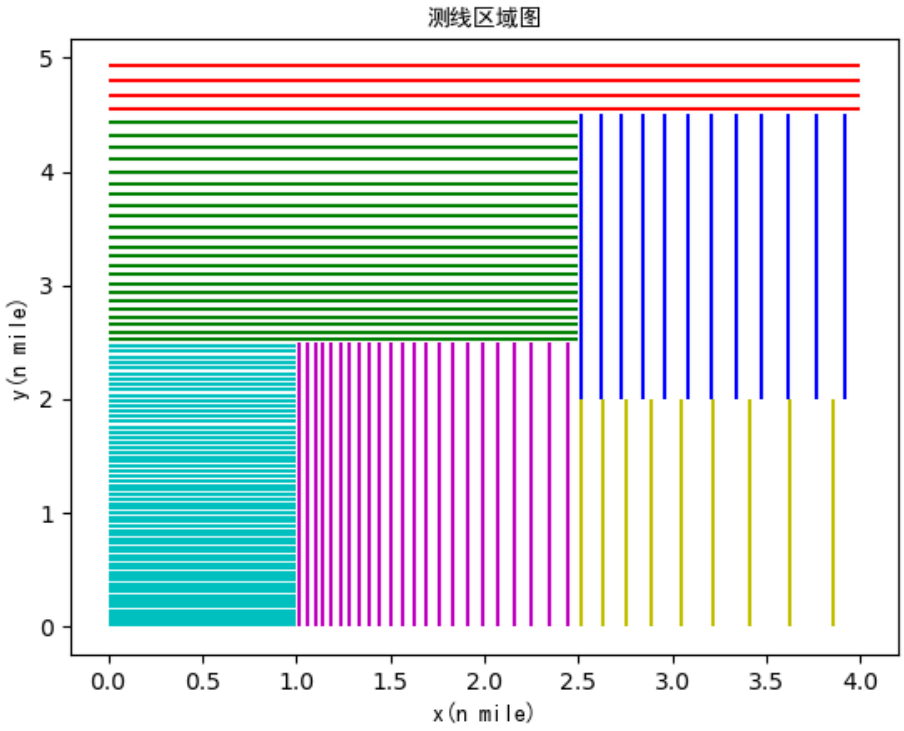

图17 测线区域图

由图 13 可见每个区域的测线分布都是由稀疏到密集，与问题三中设计的测线分布相同，说明我们的侧线设计正确。

相邻区域的测线的重叠率如表 5 所示。

表 5

两相邻区域	重叠率η'/%
①与②	14.23
②与④	23.87
③与⑤	33.21
⑤与⑥	38.64

其中，区域①与区域②的重叠率未超过 20%，区域②与区域④、区域③与区域⑤、区域⑤与区域⑥之间的重叠率超过了 20%。并且求解得出没有漏测面积，而重叠率超 20%的总长度为 3.5 海里。整体的重叠率为 10%，少数部分超出 20%，故获取的数据不会过于冗杂，并且没有漏测数据，适合实际应用。

综上，我们设计的一组测线满足题目要求，且比较合理。并求得测线总长度为 241.0 海里，漏测面积百分比为 0，重叠率超 20%的总长度为 3.5 海里。

五、灵敏度分析

我们对于模型中的参数：多波束换能器的开角θ、海底坡面的坡度α，在适当范围内浮动，看覆盖宽度W_i随这两个参数浮动的变化情况。

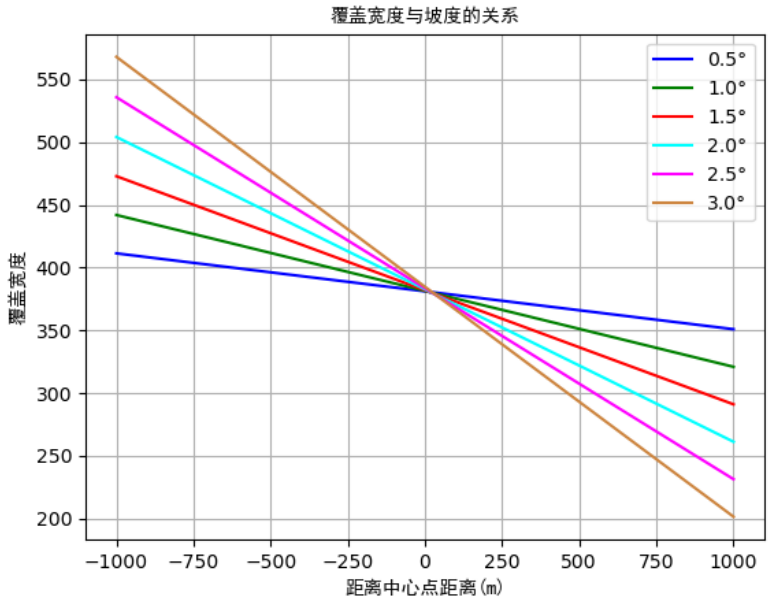

图18

如图 14 所示，对固定的坡度，覆盖宽度随着距离中心点距离增加而减少，符合越深的地方覆盖深度越大，越浅的地方覆盖深度越小的实际情况。而坡度越小，则覆盖宽度变化程度越小。而坡度取 1.5° 时，覆盖宽度的变化程度也很合理。

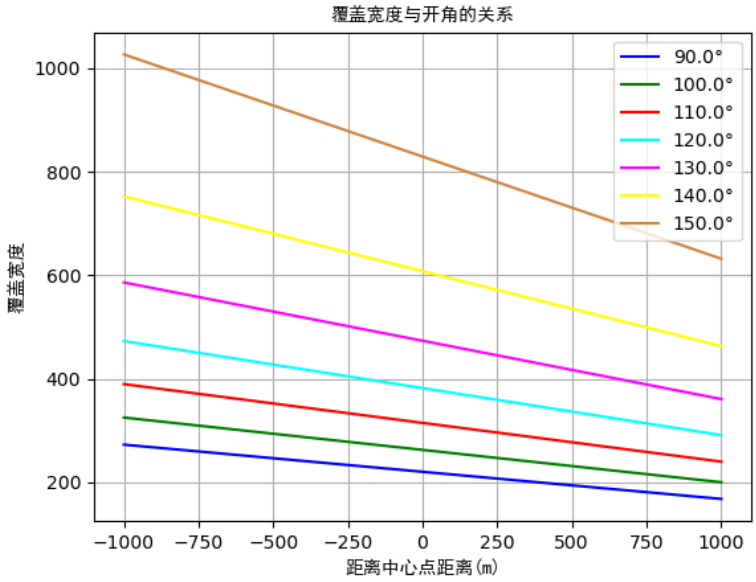

图19

如图 15 所示，在开角固定时，覆盖宽度随距离中心点距离增加而减少。开角越小，覆盖宽度的变化程度就越小，与实际中开角越大，波束就探测的范围越广的情况相符。同时，开角取 120° 时，覆盖宽度的变化程度合理。说明这两个参数的取值合理，此时模型稳定性好，适用于实际情况。

六、模型评价与推广

6.1 模型的优点

(1) 模型充分结合实际，简化海底地形条件，忽略其他因素对声波的影响，同时考虑了诸多重要因素，如:海底的坡度、在某点的覆盖宽度变化、不同重叠率的计算。这样得到的模型贴合实际，具有较高的应用价值。

(2) 模型的建立全部基于解析几何知识，运用数学思维，严谨地建立模型，并且进行灵敏度分析，分析了设定参数的合理性。同时结合实际分析，得到了严谨且合理的结果。

(3) 本文得到的测线设计方案具有总长度短、漏测率低、重叠率适中等特点，，在现有条件下能有效提高生产效率，具有科学性和可行性，可以运用到实际当中。

6.2 模型的不足

(1) 地形曲面的简化处理：尽管我们的模型在许多方面表现优秀，但在处理海底地形时，将原本复杂的曲面地形简化为平面，将等深线简化为直线，这种做法在一定程度上牺牲了模型的精度。在实际的海洋环境中，地形往往呈现出复杂的曲面形状，

而我们的模型未能充分考虑到这一点，这可能会导致模型预测与实际情况之间存在一定的误差。

(2) 区域边界处理的挑战：在进行跨区域测线设计时，相邻区域之间的测线衔接需要额外的处理，这涉及到对重复测线的识别和删减，以避免数据冗余。虽然这是模型设计中不可避免的一部分，但在实际操作中，这一过程可能比较复杂，需要消耗更多的时间和资源，增加了实施难度。

6.3 模型的推广

推广：我们的模型不仅适用于海底地形的多波束测深，还具有广泛的推广潜力。例如，可以将其应用于地面地形的探测，为地质勘探、地形测绘等领域提供支持；同时，模型也可以用于建筑扫描，为建筑设计和维护提供精确的数据。

改进：为了进一步提升模型的准确性和实用性，我们建议结合相关领域的最新研究成果和文献资料，对模型进行迭代升级。特别地，应该深入研究如何在考虑海底地形为曲面、等深线为曲线的情况下优化测线设计，以期构建一个更加贴近真实海洋环境、更为合理的模型。通过这种方式，我们不仅能够减少模型误差，还能拓展其应用范围，使其在更多的场景下发挥更大的作用。

参考文献

[1]李苏豪,董书琴,刘小虎.基于多波束测深技术的航线规划模型[C]//中国指挥与控制学会（Chinese Institute of Command and Control）.第十二届中国指挥控制大会论文集（上册）.信息工程大学;,2024:6.DOI:10.26914/c.cnkihy.2024.006496.

[2]杨卓,刘炳凯,刘彤,等.多波束测深系统在水下构筑物探测领域的应用研究[J].广州建筑,2023,51(06):84-86.

[3]赵保成,徐健,徐坚,等.基于无人船的多波束系统水下地形测量应用[J].地理空间信息,2023,21(09):65-68.

附录

附录 1

计算 1.py

```
import math
sei_ta = 2*math.pi/3
alpha_1=0.008471924589085633
alpha_2=0.008480735721641814
alpha_3=0.011878523269372773
alpha_4=0.002441995130245888
alpha_5=0.014037973141299983
alpha_6=0.033249036801760126
def cal_W(sei_ta,alpha,former_D,d, index):
      D = former_D - d * math.tan(alpha)
      if index == 1:
            return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 + alpha)
      else:
            return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 - alpha)

def cal(index1,index2,alpha1,alpha2,D1,D2,for-
mer_2,back_1,back_2,len_1,len_2):
      W2=cal_W(sei_ta,alpha1,D1,former_2*math.cos(alpha1),2)
      result_1=-(len_1/2)/math.cos(alpha1)*1852+W2+former_2
      重叠率
=result_1/(cal_W(sei_ta,alpha2,D2,back_1*math.cos(al-
pha2),1)+cal_W(sei_ta,alpha2,D2,back_1*math.cos(alpha2),2))
      if result_1>0:
            print(f'区域{index1}未漏测,与区域{index2}重叠率为:{重叠率}')
      else:
            print(f'区域{index1}漏测{abs(result_1)}米')
      result_2=(-len_2/2/math.cos(alpha2))*1852+back_2+cal_W(sei_ta,al-
pha2,D2,back_2*math.cos(alpha2),2)
      if result_2>0:
            print(f'区域{index2}未漏测')
      else:
            print(f'区域{index2}漏测{abs(result_2)}米')
      return result_1,result_2

res_1,_=cal(1,2,alpha_1,al-
pha_2,74.99469383849981,53.381320917806065,373.96280743024556,-
1732.3327140442464,1811.916036147846,0.5,2)
```

```
res_2,_=cal(2,4,alpha_2,al-
pha_4,53.381320917806065,24.394733893557422,1811.916036147846,-
2262.95527395032,2290.417370868393,2,2.5)
#3 号去最后一个
#倒数第六个
res_3,_=cal(3,5,alpha_3,al-
pha_5,76.91331140350877,41.74538429406851,1354.6606972844043,-
1282.9127132985072,1352.7708490617877,1.5,1.5)
#倒数第四个
res_4,_=cal(6,5,alpha_6,al-
pha_5,108.74402944241791,41.74538429406851,1366.0295588016102,-
1282.9127132985072,1352.7708490617877,1.5,1.5)
```

灵敏度——开角和重叠率.py

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

sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                 130 * math.pi / 180,  140*math.pi/180,150 *
math.pi / 180]
alpha_values=[10 * math.pi / 180,20 * math.pi / 180,30 * math.pi / 180,40
* math.pi / 180,50 * math.pi / 180,60 * math.pi / 180,70 * math.pi / 180]

def cal_depth(x):
    former_D=70
    return former_D-x*math.sin(1.5*math.pi/180)

def cal_width(x,alpha,sei_ta):
    return cal_depth(x)*math.sin(sei_ta/2)*(1/math.cos(sei_ta/2 + al-
pha) + 1/math.cos(sei_ta/2 - alpha))

def cal_eta(x,i,D,alpha,sei_ta):
    former_W2=cal_depth(D[i-1])*math.sin(sei_ta/2)/math.cos(sei_ta/2 -
alpha)
    cur_W1=cal_depth(x)*math.sin(sei_ta/2)/math.cos(sei_ta/2 + alpha)
    up=cur_W1+former_W2-200/math.cos(alpha)
    down=cal_width(D[i-1],alpha,sei_ta)
    return up/down *100
```

```
# 定义参数
former_D = 110
sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                        130 * math.pi / 180, 140*math.pi/180,150 *
math.pi / 180]
alpha_values=[0.5 * math.pi / 180,1 * math.pi / 180,1.5 * math.pi / 180,2
* math.pi / 180,2.5 * math.pi / 180,3*math.pi/180]
d_list = np.arange(-1000, 1000+1, 200)

# 创建一个空列表来存储所有计算结果
eta_results = []

# 对于每个 alpha 计算结果
count=1
for sei_ta in sei_ta_values:
      eta_results.append([cal_eta(d_list[1], count, d_list, alpha_val-
ues[2],sei_ta) for d in d_list[1:]])
      count+=1

# 绘制曲线
#colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow','peru']
colors = ['blue', 'green', 'red', 'cyan', 'magenta','peru','yellow']
for i, result in enumerate(eta_results):
      plt.plot(d_list[1:], result, color=colors[i], label=f'{sei_ta_val-
ues[i]*180/math.pi:.1f}°')

# 设置图例
plt.legend(loc='upper right')
# 设置坐标轴标签
plt.title('重叠率与开角的关系',fontproperties="SimHei")
plt.xlabel('距离中心点距离(m)',fontproperties="SimHei")
plt.ylabel('重叠率(%)',fontproperties="SimHei")
plt.xlim(-1000,1200)
plt.rcParams['font.family']=['SimHei']#关键是这句
# 显示图形
plt.grid(True)
plt.show()
```

灵敏度——坡度和重叠率.py

```
import math
import matplotlib.pyplot as plt
```

```
import numpy as np

sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                        130 * math.pi / 180,  140*math.pi/180,150 *
math.pi / 180]
alpha_values=[10 * math.pi / 180,20 * math.pi / 180,30 * math.pi / 180,40
* math.pi / 180,50 * math.pi / 180,60 * math.pi / 180,70 * math.pi / 180]

def cal_depth(x):
      former_D=70
      return former_D-x*math.sin(1.5*math.pi/180)

def cal_width(x,alpha,sei_ta):
      return cal_depth(x)*math.sin(sei_ta/2)*(1/math.cos(sei_ta/2 + al-
pha) + 1/math.cos(sei_ta/2 - alpha))

def cal_eta(x,i,D,alpha,sei_ta):
      former_W2=cal_depth(D[i-1])*math.sin(sei_ta/2)/math.cos(sei_ta/2 -
alpha)
      cur_W1=cal_depth(x)*math.sin(sei_ta/2)/math.cos(sei_ta/2 + alpha)
      up=cur_W1+former_W2-200/math.cos(alpha)
      down=cal_width(D[i-1],alpha,sei_ta)
      return up/down *100

# 定义参数
former_D = 110
sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                        130 * math.pi / 180,  140*math.pi/180,150 *
math.pi / 180]
alpha_values=[0.5 * math.pi / 180,1 * math.pi / 180,1.5 * math.pi / 180,2
* math.pi / 180,2.5 * math.pi / 180,3*math.pi/180]
d_list = np.arange(-1000, 1000+1, 200)

# 创建一个空列表来存储所有计算结果
eta_results = []

# 对于每个 alpha 计算结果
count=1
for alp in alpha_values:
      eta_results.append([cal_eta(d_list[1], count, d_list,
alp,sei_ta_values[3]) for d in d_list[1:]])
```

```
        count+=1

# 绘制曲线
#colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow','peru']
colors = ['blue', 'green', 'red', 'cyan', 'magenta','peru']
for i, result in enumerate(eta_results):
        plt.plot(d_list[1:], result, color=colors[i], label=f'{alpha_val-
ues[i]*180/math.pi:.1f}° ')

# 设置图例
plt.legend(loc='upper right')
# 设置坐标轴标签
plt.title('重叠率与坡度的关系',fontproperties="SimHei")
plt.xlabel('距离中心点距离(m)',fontproperties="SimHei")
plt.ylabel('重叠率(%)',fontproperties="SimHei")
plt.xlim(-1000,1200)
plt.rcParams['font.family']=['SimHei']#关键是这句
# 显示图形
plt.grid(True)
plt.show()
```

灵敏度——坡度与覆盖宽度.py

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

def calculate_and_plot_lines(alpha, former_D, sei_ta, d):
        def cal_W(d, index):
                D = former_D - d * math.tan(alpha)
                if index == 1:
                        return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 + al-
pha)
                else:
                        return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 - al-
pha)

        return cal_W(d, 1) + cal_W(d, 2)

# 定义参数
alpha = 1.5 * math.pi / 180
former_D = 110
```

```
sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                 130 * math.pi / 180, 140*math.pi/180,150 *
math.pi / 180]
alpha_values=[0.5 * math.pi / 180,1 * math.pi / 180,1.5 * math.pi / 180,2
* math.pi / 180,2.5 * math.pi / 180,3*math.pi/180]
d_list = np.arange(-1000, 1000+1, 200)

# 创建一个空列表来存储所有计算结果
results = []

# 对于每个 alpha 计算结果
for alp in alpha_values:
    results.append([calculate_and_plot_lines(alp, former_D,
120*math.pi/180, d) for d in d_list])

# 绘制曲线
#colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow','peru']
colors = ['blue', 'green', 'red', 'cyan', 'magenta','peru']
for i, result in enumerate(results):
    plt.plot(d_list, result, color=colors[i], label=f'{alpha_val-
ues[i]*180/math.pi:.1f}° ')

# 设置图例
plt.legend()

# 设置坐标轴标签
plt.title('覆盖宽度与坡度的关系',fontproperties="SimHei")
plt.xlabel('距离中心点距离(m)',fontproperties="SimHei")
plt.ylabel('覆盖宽度',fontproperties="SimHei")
plt.rcParams['font.family']=['SimHei']#关键是这句
# 显示图形
plt.grid(True)
plt.show()
```

灵敏度分析.py

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

def calculate_and_plot_lines(alpha, former_D, sei_ta, d):
    def cal_W(d, index):
```

```
            D = former_D - d * math.tan(alpha)
            if index == 1:
                  return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 + al-
pha)
            else:
                  return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 - al-
pha)

      return cal_W(d, 1) + cal_W(d, 2)

# 定义参数
alpha = 1.5 * math.pi / 180
former_D = 110
sei_ta_values = [90 * math.pi / 180,100*math.pi/180,110 * math.pi / 180,
120*math.pi/180,
                        130 * math.pi / 180, 140*math.pi/180,150 *
math.pi / 180]
alpha_values=[10 * math.pi / 180,20 * math.pi / 180,30 * math.pi / 180,40
* math.pi / 180,50 * math.pi / 180,60 * math.pi / 180,70 * math.pi / 180]
d_list = np.arange(-1000, 1000+1, 200)

# 创建一个空列表来存储所有计算结果
results = []

# 对于每个 sei_ta 计算结果
for sei_ta in sei_ta_values:
      results.append([calculate_and_plot_lines(alpha, former_D, sei_ta,
d) for d in d_list])

# 绘制曲线
colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow','peru']
for i, result in enumerate(results):
      plt.plot(d_list, result, color=colors[i], label=f'{sei_ta_val-
ues[i]*180/math.pi:.1f}°')

# 设置图例
plt.legend()

# 设置坐标轴标签
plt.title('覆盖宽度与开角的关系',fontproperties="SimHei")
plt.xlabel('距离中心点距离(m)',fontproperties="SimHei")
plt.ylabel('覆盖宽度',fontproperties="SimHei")
plt.rcParams['font.family']=['SimHei']#关键是这句
```

```
# 显示图形
plt.grid(True)
plt.show()
```

问题二.py

```
import openpyxl
import math

workbook = openpyxl.Workbook()
worksheet = workbook.active

distance = [0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1]
for i in range(len(distance)):
    distance[i] = distance[i] * 1852
angle = [0, math.pi / 4, math.pi / 2, math.pi * 3 / 4, math.pi, math.pi * 5 / 4, math.pi * 3 / 2, math.pi * 7 / 4]
original_D = 120
alpha = 1.5 * math.pi / 180
beta = 120 * math.pi / 180
sei_ta = 2 * math.pi / 3

def cal_gama(alp, bet):
    gama = math.atan(math.tan(alp) * math.sin(bet))
    return gama

row = 1
for ang in angle:
    col = 1
    for dis in distance:
        gama = cal_gama(alpha, ang)
        D = original_D + dis * math.tan(alpha) * math.cos(ang)
        W = D * math.sin(sei_ta / 2) * (1 / math.cos(sei_ta / 2 + gama) + 1 / math.cos(sei_ta / 2 - gama))
        worksheet.cell(row=row, column=col).value = f'{W:.2f}'
        col += 1
    row += 1

workbook.save("output.xlsx")
```

问题三.py

```
import math
import openpyxl
```

```
from sympy.solvers import solve
from sympy import Symbol
alpha = 1.5 * math.pi / 180
sei_ta = 2 * math.pi / 3
former_D=110
length=4*1852
width=2*1852
dis=[0 for _ in range(114)]
workbook = openpyxl.Workbook()
worksheet = workbook.active
x = Symbol("x")
dis[0]=float(solve(length/2+x/math.cos(alpha)-(former_D-x*math.tan(al-
pha))*math.sin(sei_ta)/math.cos(sei_ta/2+alpha),x)[0])

def cal_W(d,index):
      D=former_D-d*math.tan(alpha)
      W=None
      if index==1:
            W=D*math.sin(sei_ta/2)/math.cos(sei_ta/2+alpha)
      else:
            W=D*math.sin(sei_ta/2)/math.cos(sei_ta/2-alpha)
      return W

ind=0
for i in range(1,114):
      d=Symbol('d')
      dis[i]=float(solve(cal_W(d,1)-(d-dis[i-1])/math.cos(al-
pha)+cal_W(dis[i-1],2)-(1/10) *(cal_W(d,1)+cal_W(d,2)),d)[0])
      if dis[i]>=length/2:
            break
      ind+=1
for i in range(ind+1):
      print(dis[i]/math.cos(alpha),end=' ')
      worksheet.cell(row=i+1,column=1).value = f'{dis[i]/math.cos(al-
pha):.2f}'
workbook.save("问题三.xlsx")
print(f'\n总的线条数为:{ind+1},长度总和为:{(ind+1)*width}')

import matplotlib.pyplot as plt

data = [dis[i]/math.cos(alpha) for i in range(ind+1)]
# 绘制垂直线
plt.vlines(data,ymin=0,ymax=width, colors='blue')
```

```
# 添加标题和标签
plt.title('测线',fontproperties="SimHei")
plt.xlabel('与中心点距离(m)',fontproperties="SimHei")
plt.ylabel('测线长度(m)',fontproperties="SimHei")
# 显示图表
plt.show()
```

问题四——分块拟合平面.py

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

##########
import math
from sympy import solve
import openpyxl
import matplotlib.pyplot as plt
from sympy import Symbol
def calculate_and_plot_lines(alpha, length, width, former_D, filename='问题三.xlsx'):
    sei_ta = 2 * math.pi / 3
    dis = [0 for _ in range(300)]
    workbook = openpyxl.Workbook()
    worksheet = workbook.active
    x= Symbol('x')
    # 计算第一个点的距离
    dis[0] = float(solve(length/2 + x/math.cos(alpha) - (former_D - x*math.tan(alpha)) * math.sin(sei_ta) / math.cos(sei_ta/2 + alpha), x)[0])

    def cal_W(d, index):
        D = former_D - d * math.tan(alpha)
        if index == 1:
            return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 + alpha)
        else:
            return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 - alpha)

    ind = 0
    for i in range(1, 300):
```

```
            d = Symbol('d')
            dis[i] = float(solve(cal_W(d, 1) - (d - dis[i-1]) /
math.cos(alpha) + cal_W(dis[i-1], 2) - (1/10) * (cal_W(d, 1) + cal_W(d,
2)), d)[0])
            if dis[i] >= length/2:
                  break
            ind += 1

      # 打印并保存到 Excel
      for i in range(ind+1):
            print(dis[i] / math.cos(alpha), end=' ')
            worksheet.cell(row=i+1, column=1).value = f'{dis[i] /
math.cos(alpha):.2f}'

      workbook.save(filename)
      print(f'\n总的线条数为: {ind+1}, 长度总和为: {(ind+1)*width}')

      # # # 绘制线条
      data = [dis[i] / math.cos(alpha) for i in range(ind+1)]
      plt.vlines(data, ymin=0, ymax=width, colors='blue')

      # # 添加标题和标签
      plt.title('测线', fontproperties="SimHei")
      plt.xlabel('与中心点距离(m)', fontproperties="SimHei")
      plt.ylabel('测线长度(m)', fontproperties="SimHei")
      plt.show()

      return ind+1,(ind+1)*width
##########
# 海里到米的转换系数
nautical_mile_to_meter = 1852

# 读取 CSV 文件
data = pd.read_csv('111111.csv', names=['x', 'y', 'z'], skiprows=1)

# 将 x, y 列转换为米
data['x'] = data['x'] * nautical_mile_to_meter
data['y'] = data['y'] * nautical_mile_to_meter

def fit_region(data, x_min, x_max, y_min, y_max, x_or_y):
      # 筛选出符合条件的数据
      filtered_data = data[
            (data['x'] >= x_min) & (data['x'] <= x_max) &
```

```
            (data['y'] >= y_min) & (data['y'] <= y_max)
        ]

        if x_or_y == 'x':
            # 如果 x 是自变量
            x_filtered = filtered_data['x'].values.reshape(-1, 1)
            y_filtered = filtered_data['z'].values
        elif x_or_y == 'y':
            # 如果 y 是自变量
            x_filtered = filtered_data['y'].values.reshape(-1, 1)
            y_filtered = filtered_data['z'].values
        else:
            raise ValueError("x_or_y must be either 'x' or 'y'")
        y_filtered = -filtered_data['z'].values

        # 使用线性回归模型进行拟合
        model = LinearRegression()
        model.fit(x_filtered, y_filtered)

        # 计算斜率
        slope = model.coef_[0]

        # 计算中心点
        center_x = np.mean(x_filtered)
        center_y = model.predict([[center_x]])[0]

        # 计算 R^2
        r_squared = model.score(x_filtered, y_filtered)

        return math.atan(abs(slope)),-center_y, model, r_squared ,
(abs(x_max-x_min),abs(y_max-y_min))

regions = [
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 4.5 * nautical_mile_to_meter, 'y_max': 5 *
nautical_mile_to_meter, 'x_or_y': 'y'},
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 2.5 * nauti-
cal_mile_to_meter, 'y_min': 2.5 * nautical_mile_to_meter, 'y_max': 4.5 *
nautical_mile_to_meter, 'x_or_y': 'y'},
        {'x_min': 2.5 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 2 * nautical_mile_to_meter, 'y_max': 4.5 *
nautical_mile_to_meter, 'x_or_y': 'x'},
```

```
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 1 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2.5 *
nautical_mile_to_meter, 'x_or_y': 'y'},
        {'x_min': 1 * nautical_mile_to_meter, 'x_max': 2.5 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2.5 *
nautical_mile_to_meter, 'x_or_y': 'x'},
        {'x_min': 2.5 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2 * nau-
tical_mile_to_meter, 'x_or_y':'x'},
        # 添加更多区域...
]

sum_line=0
sum_length=0
# 循环拟合每个区域
results = []
for region in regions:
        alpha, center, model, r_squared,ran= fit_region(data, **region)
        results.append({
                'alpha': alpha,
                'center': center,
                'model': model,
                'r_squared': r_squared
        })
        print(f"Region: x in [{region['x_min'] / nautical_mile_to_me-
ter:.2f}, {region['x_max'] / nautical_mile_to_meter:.2f}] nautical miles,
y in [{region['y_min'] / nautical_mile_to_meter:.2f}, {region['y_max'] /
nautical_mile_to_meter:.2f}] nautical miles")
        print(f"alpha: {alpha}")
        print(f"Center: {center}")
        print(f"R^2: {r_squared}\n")
        if center=='x':
                line,length=calculate_and_plot_lines(alpha,ran[0],ran[1],cen-
ter)
                sum_line+=line
                sum_length+=length
        else:
                line,length=calculate_and_plot_lines(alpha,ran[1],ran[0],cen-
ter)
                sum_line+=line
                sum_length+=length

# 结果存储在 results 列表中
```

```
# 保存结果到 CSV 文件
results_df = pd.DataFrame(results, columns=['alpha', 'center', 'mod-
el' ,'r_squared'])
results_df.to_csv('fit_results.csv', index=False)

# 输出保存成功的消息
print(f'总的线数为:{sum_line}, 总的长度为:
{sum_length/nautical_mile_to_meter}海里')
print("Results saved to fit_results.csv")
```

问题四——简化.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata

# 读取 CSV 文件
df = pd.read_csv('111111.csv')

# 将 x 和 y 坐标从海里转换成米
df['x'] = df['x']   # 将 x 从海里转换为米
df['y'] = df['y'] # 将 y 从海里转换为米

# 选取数据的一个子集（例如，每 10 个点取一个）
df_subset = df.iloc[::10, :]

# 提取 x, y, z 值
x = df_subset['x'].values
y = df_subset['y'].values
z = df_subset['z'].values
z=-z
# 创建规则网格
grid_x, grid_y = np.mgrid[min(x):max(x):100j, min(y):max(y):100j]

# 使用 griddata 进行插值
grid_z = griddata((x, y), z, (grid_x, grid_y), method='linear')

# 使用 matplotlib 绘制等高线图
fig, ax = plt.subplots()
cs = ax.contourf(grid_x, grid_y, grid_z, levels=100, cmap='viridis')
c = ax.contour(grid_x, grid_y, grid_z, colors='k', levels=cs.levels[::2])
plt.clabel(c, inline=True, fontsize=8)
```

```
# 设置坐标轴标签和标题
ax.set_xlabel('由西向东(海里)',fontproperties="SimHei")
ax.set_ylabel('由南向北(海里)',fontproperties="SimHei")
ax.set_title('区域海底深度图',fontproperties="SimHei")

# 添加颜色条
plt.colorbar(cs)

# 显示图形
plt.show()
```

问题四——数据处理.py

```
import pandas as pd
import numpy as np

# 文件路径
file_path = r"D:\数学建模\题目\B 题\附件.xlsx"

# 读取数据
data = pd.read_excel(file_path)

# 提取数据并进行单位转换
z_values = data.iloc[1:253, 2:204].values   # 转换为米

# 创建新的坐标系统
x_values = np.arange(0.00, 4.01, 0.02).tolist()
y_values = np.arange(0.00, 5.01, 0.02).tolist()

# 将转换后的数据写入新文件
output_file_path = "111111.csv"
with open(output_file_path, 'w') as f:
      # 写入表头
      f.write("x,y,z\n")

      # 写入数据
      for y in y_values:
            for i, x in enumerate(x_values):
                  z = z_values[y_values.index(y)][i]
                  f.write(f"{x:.2f},{y:.2f},{z:.2f}\n")

print("Data has been written to the file.")
```

问题一.py

```
import math

D=[-800,-600,-400,-200,0,200,400,600,800]

def cal_depth(x):
	former_D=70
	return former_D-x*math.sin(1.5*math.pi/180)

def cal_width(x):
	alpha=1.5*math.pi/180
	sei_ta=2*math.pi/3
	return cal_depth(x)*math.sin(sei_ta/2)*(1/math.cos(sei_ta/2 + alpha) + 1/math.cos(sei_ta/2 - alpha))

def cal_eta(x,i,D):
	alpha=1.5*math.pi/180
	sei_ta=2*math.pi/3
	former_W2=cal_depth(D[i-1])*math.sin(sei_ta/2)/math.cos(sei_ta/2 - alpha)
	cur_W1=cal_depth(x)*math.sin(sei_ta/2)/math.cos(sei_ta/2 + alpha)
	up=cur_W1+former_W2-200/math.cos(alpha)
	down=cal_width(D[i-1])
	return up/down

depth=[]
width=[]
eta=[-100]
for i,x in enumerate(D):
	depth.append(cal_depth(x))
	width.append(cal_width(x))
	if i==0:
		continue
	else:
		eta.append(cal_eta(x,i,D))
for data in depth:
	print(f'{data:.2f}',end=' ')

print('\n')
for data in width:
	print(f'{data:.2f}',end=' ')
```

```
print('\n')
for data in eta:
    print(f'{data*100:.2f}',end=' ')
```

绘制.py

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np
import math
from sympy import solve, Symbol
import matplotlib.pyplot as plt

# 定义颜色列表
colors = ['r', 'g', 'b', 'c', 'm', 'y']

# 海里到米的转换系数
nautical_mile_to_meter = 1852

# 读取 CSV 文件
data = pd.read_csv('111111.csv', names=['x', 'y', 'z'], skiprows=1)

# 将 x, y 列转换为米
data['x'] = data['x'] * nautical_mile_to_meter
data['y'] = data['y'] * nautical_mile_to_meter

def calculate_and_plot_lines(alpha, length, width, former_D):
    re = []
    sei_ta = 2 * math.pi / 3
    dis = [0 for _ in range(300)]
    x = Symbol('x')
    # 计算第一个点的距离
    dis[0] = float(solve(length/2 + x/math.cos(alpha) - (former_D - x*math.tan(alpha)) * math.sin(sei_ta) / math.cos(sei_ta/2 + alpha), x)[0])
    re.append(dis[0])

    def cal_W(d, index):
        D = former_D - d * math.tan(alpha)
        if index == 1:
            return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 + alpha)
        else:
```

```
                return D * math.sin(sei_ta/2) / math.cos(sei_ta/2 - al-
pha)

    ind = 0
    for i in range(1, 300):
        d = Symbol('d')
        dis[i] = float(solve(cal_W(d, 1) - (d - dis[i-1]) /
math.cos(alpha) + cal_W(dis[i-1], 2) - (1/10) * (cal_W(d, 1) + cal_W(d,
2)), d)[0])
        if dis[i] >= length/2:
            break
        re.append(dis[i])
        ind += 1

    for i in range(ind+1):
        print(dis[i] / math.cos(alpha), end=' ')

    return ind+1, (ind+1)*width, re

def fit_region(data, x_min, x_max, y_min, y_max, x_or_y, id):
    # 筛选出符合条件的数据
    filtered_data = data[
        (data['x'] >= x_min) & (data['x'] <= x_max) &
        (data['y'] >= y_min) & (data['y'] <= y_max)
    ]

    if x_or_y == 'x':
        # 如果 x 是自变量
        x_filtered = filtered_data['x'].values.reshape(-1, 1)
        y_filtered = filtered_data['z'].values
    elif x_or_y == 'y':
        # 如果 y 是自变量
        x_filtered = filtered_data['y'].values.reshape(-1, 1)
        y_filtered = filtered_data['z'].values
    else:
        raise ValueError("x_or_y must be either 'x' or 'y'")
    y_filtered = -filtered_data['z'].values

    # 使用线性回归模型进行拟合
    model = LinearRegression()
    model.fit(x_filtered, y_filtered)

    # 计算斜率
```

```
        slope = model.coef_[0]

        # 计算中心点
        center_x = np.mean(x_filtered)
        center_y = model.predict([[center_x]])[0]

        # 计算 R^2
        r_squared = model.score(x_filtered, y_filtered)

        return math.atan(abs(slope)), -center_y, model, r_squared,
(abs(x_max-x_min), abs(y_max-y_min))

regions = [
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 4.5 * nautical_mile_to_meter, 'y_max': 5 *
nautical_mile_to_meter, 'x_or_y': 'y', 'id': 1},
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 2.5 * nauti-
cal_mile_to_meter, 'y_min': 2.5 * nautical_mile_to_meter, 'y_max': 4.5 *
nautical_mile_to_meter, 'x_or_y': 'y', 'id': 2},
        {'x_min': 2.5 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 2 * nautical_mile_to_meter, 'y_max': 4.5 *
nautical_mile_to_meter, 'x_or_y': 'x', 'id': 3},
        {'x_min': 0 * nautical_mile_to_meter, 'x_max': 1 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2.5 *
nautical_mile_to_meter, 'x_or_y': 'y', 'id': 4},
        {'x_min': 1 * nautical_mile_to_meter, 'x_max': 2.5 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2.5 *
nautical_mile_to_meter, 'x_or_y': 'x', 'id': 5},
        {'x_min': 2.5 * nautical_mile_to_meter, 'x_max': 4 * nauti-
cal_mile_to_meter, 'y_min': 0 * nautical_mile_to_meter, 'y_max': 2 * nau-
tical_mile_to_meter, 'x_or_y': 'x', 'id': 6},
        # 添加更多区域...
]

sum_lines = 0
sum_length = 0
# 循环拟合每个区域
plt.figure()
plt.title('测线区域图', fontproperties="SimHei")
plt.xlabel('x(n mile)', fontproperties="SimHei")
plt.ylabel('y(n mile)', fontproperties="SimHei")

for i, region in enumerate(regions):
```

```
        alpha, center, model, r_squared, ran = fit_region(data, **region)

        print(f"Region: x in [{region['x_min'] / nautical_mile_to_me-
ter:.2f}, {region['x_max'] / nautical_mile_to_meter:.2f}] nautical miles,
y in [{region['y_min'] / nautical_mile_to_meter:.2f}, {region['y_max'] /
nautical_mile_to_meter:.2f}] nautical miles")
        print(f"alpha: {alpha}")
        print(f"Center: {center}")
        print(f"R^2: {r_squared}\n")

        color = colors[i % len(colors)]   # 获取颜色

        if region['x_or_y'] == 'x':
            line, length, res = calculate_and_plot_lines(alpha, ran[0],
ran[1], center)
            sum_lines += line
            sum_length += length
            print(f'测线数为{line}')
            if region['id'] == 5:
                # res=res[:-5]
                res = res
            elif region['id'] == 6:
                # res=res[:-3]
                res = res
            for i, da in enumerate(res):
                res[i] = res[i] / nautical_mile_to_meter + (re-
gion['x_min'] / 1852 + region['x_max'] / 1852) / 2
                res[i] = (region['x_min'] + region['x_max']) / 1852 -
res[i]

            plt.vlines(res, region['y_min'] / 1852, region['y_max'] /
1852, colors=color, linestyles='solid', label='')
        else:
            line, length, res = calculate_and_plot_lines(alpha, ran[1],
ran[0], center)
            sum_length += length
            sum_lines += line
            print(f'测线数为{line}')
            for i, da in enumerate(res):
                res[i] = res[i] / nautical_mile_to_meter + (re-
gion['y_min'] / 1852 + region['y_max'] / 1852) / 2
                res[i] = (region['y_min'] + region['y_max']) / 1852 -
res[i]
```

```
            plt.hlines(res, region['x_min'] / 1852, region['x_max'] /
1852, colors=color, linestyles='solid', label='')

plt.show()
print(f'总数为：{sum_lines},总长度为：{sum_length/1852}')
```

划分图——线条.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata

# 读取 CSV 文件
df = pd.read_csv('111111.csv')

# 选取数据的一个子集（例如，每 10 个点取一个）
df_subset = df.iloc[::10, :]

# 提取 x, y, z 值
x = df_subset['x'].values
y = df_subset['y'].values
z = df_subset['z'].values
z = -z

# 创建规则网格
grid_x, grid_y = np.mgrid[min(x):max(x):100j, min(y):max(y):100j]

# 使用 griddata 进行插值
grid_z = griddata((x, y), z, (grid_x, grid_y), method='linear')

# 使用 matplotlib 绘制等高线图
fig, ax = plt.subplots()
cs = ax.contourf(grid_x, grid_y, grid_z, levels=100, cmap='viridis')
c = ax.contour(grid_x, grid_y, grid_z, colors='k', levels=cs.levels[::2])
# plt.clabel(c, inline=True, fontsize=8)

# 设置坐标轴标签和标题
ax.set_xlabel('由西向东(海里)', fontproperties="SimHei")
ax.set_ylabel('由南向北(海里)', fontproperties="SimHei")
ax.set_title('区域海底深度图', fontproperties="SimHei")

# 添加颜色条
```

```
plt.colorbar(cs)

# 在地形图上绘制分割线
plt.hlines(4.5, 0, 4, colors='r', linestyles='solid')
plt.hlines(2.5, 0, 2.5, colors='r', linestyles='solid')
plt.hlines(2, 2.5, 4, colors='r', linestyles='solid')
plt.vlines(1, 0, 2.5, colors='r', linestyles='solid')
plt.vlines(2.5, 0, 4.5, colors='r', linestyles='solid')

# 设置坐标轴范围
plt.xlim(0, max(x))
plt.ylim(0, max(y))

# 在左上角矩形内添加文本
plt.text(2, 4.75, '区域①', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))
plt.text(1.25, 3.5, '区域②', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))
plt.text(3.25, 3.25, '区域③', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))
plt.text(0.5, 1.25, '区域④', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))
plt.text(1.75, 1.25, '区域⑤', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))
plt.text(3.25, 1, '区域⑥', fontsize=12, ha='center', va='center',
fontproperties="SimHei", bbox=dict(facecolor='white', alpha=1))

# 显示图形
plt.show()
```