基于解析几何及优化算法的测量船多波束测线问题

摘要

针对测量船的多波束测线问题，本文从波束条带覆盖理论入手，给出了测量船处于不同位置时的水深测量模型，并根据此模型给出了满足不同优化目标的测线方案。

针对问题一，我们在二维平面内建立数学模型，将问题转化为**解三角形**问题。我们通过边角间的关系进行未知角的转化求解，再通过**正弦定理**列出三角方程，得到未知边的解。实际建立了一个二维平面内只需知道测线所处位置深度、坡度角及相应探测条件就能结合计算机进行精确求解的模型，解出波束覆盖宽度、测线间的重叠率。

针对问题二，我们通过建立空间直角坐标系，将直线、平面关系通过**向量运算**进行求解，思路简洁，计算部分可通过计算机实现。通过三维建模我们可得出同一测线不同位置测量船所处地点的深度及斜切坡度。核心在于得到剖面后可将问题转化为问题一模型进行求解，解出不同点波束覆盖宽度。

针对问题三，我们首先通过建立空间直角坐标系得出测量船所处位置深度、斜切坡度等基础信息条件。再聚焦于测线组平面进行研究。以测线间距、测线方向为决策变量，以波束覆盖程度、重叠率要求及一定边界条件为约束条件，以最短测线组总长为目标函数，建立测线组总长最短的**优化模型**，求解过程使用了 Python 中的 scipy 库，应用 optimize 模块中的 **minimize 函数**对优化函数进行求解，method=**‘SLSQP’**。最终得出优化结果。

针对问题四，由于需要优化的目标不止一个，我们可以通过建立**多目标优化模型**进行求解。我们首先通过附件表所给出数据，结合**双线性插值法**，将此海域内海底地形全面进行模拟。再将海域分割为多个矩形小块，对每个小块单独分析时可通过拟合将不规则海底地形近似为规则坡面，再运用问题一问题二所建立的模型进行覆盖宽度、重复率的求解。将决策变量定义为测线间距及测线方向，根据题目条件，应用帕累托最优解进行线性加权求出目标方程，此目标方程的最小值即满足题目所给的多个约束条件：测线波束所形成的条带对海域覆盖越多越好，尽可能覆盖全海域；相邻条带间重叠率尽量控制。

关键词： 覆盖宽度　重叠率　优化模型　解析几何

一　问题的背景与重述

1.1 问题背景

随着雷达测绘技术的应用与发展，海底地形地貌有了更为精准、高效的测量方法。其中，波束测深技术是海水深度测量的最主要的技术之一，核心在于通过声波在海水中的传播速度和传播时间计算出海水的深度。

波束测深技术一共分为单波束测深与多波束测深两种。“海底地形测量的核心是水深测量,长期以来,水深测量经历了从人工到电子化,再到单波束技术的变革。近年来,又从基于单波束的点线测量模式发展到基于多波束的面状全覆盖测量模式,极大地提高了测量精度和效率。”[1]

较单波束测深而言，多波束测深具有覆盖性强、测量效率高等优点。测量过程中，沿船体移动方向波束可形成具有一定宽度的全覆盖水深条带，测出条带内的海水深度。

1.2 问题重述

问题一 已知海底坡度为 1.5°，多波束换能器开角为 120°，海域中心点处海水深度为70m。船体沿坡面走向方向行进，形成测线。以距中心点距离为0的测线为基准线，得出距中心点处水平距离不同的测线所对应的距海底的深度、覆盖宽度，进而得出测线组的重叠率。

问题二 测线方向与海底坡面法向在水平面上的投影成β，海底坡面坡度已知为α，起始点深度 D。需要建立数学模型，得出沿着此测线方向上不同位置的海水深度、条带覆盖宽度。

下一步，已知多波束换能器开角120°，坡度1.5°，海域中心点处海水深度120m，需要在建立上述模型的基础上得出不同方向夹角的测线上不同位置的条带覆盖宽度。

问题三 海域形状为矩形，南北长 2 海里，东西宽 4 海里。海底地形已知，为一坡度为 1.5° 的斜坡，西深东浅。需要设计出具体测线布局，满足相邻条带间的重叠率大于 10%并小于 20%，且实现海域全覆盖，使得测线组总测量长度最短。

问题四 海域形状为矩形，南北长 5 海里，东西宽 4 海里。海底地形形状不规则，附件中给出了此海域中一定具有一定间隔的每个点的深度。现欲对此海域设计测线组进行测量，且需要满足三个设定：测线波束所形成的条带对海域覆盖越多越好，尽可能覆盖全海域；相邻条带间重叠率尽量控制在 20%以下；尽可能缩短测线总长度。满足如上设定后，需要求出测线总长度、漏测海区占总待测海域面积的百分比以及重叠区域中重叠率超过 20%部分的总长度。

二　问题分析

2.1 总体分析

问题一模型的建立为解决问题二、问题三、问题四的基础。四个问题的设置逐层递进。问题二是问题一模型的三维拓展。问题三、问题四(问题四优化目标更多)为问题一、问题

二建立模型的具体应用。四个问题建立的模型具有实际应用价值。

2.2 问题一分析

2.2.1 二维平面覆盖宽度、重叠率模型的建立

问题一要求给出在海底地形有一定坡度条件下多波束测深覆盖宽度及相邻条带之间重叠率的数学模型，核心在于求出不同测线在同一水平面上时的覆盖宽度、重叠率。于波束测深条带与坡面所形成的三角形，在三角形内通过角度变换，将题目简化为一个解三角形问题。再通过正弦定理将各角关系转变为各边关系，最终得到波束覆盖宽度。这里我们假设坡度、多波束换能器开角不变，利用相邻测线间的距离，通过边、角转换，得到相邻条带之间的重叠率。

2.2.2 测线距中心点距离发生改变时数据的计算

题目给出多波束换能器开角、坡度、中心点海水深度具体数值，要求计算距中心点处距离不同的测线处的海水深度、覆盖宽度、与前一条测线的重叠率。可以通过 2.2.1 所建立的数学模型，带入多波束换能器开角、坡度、深度数值，得出结果。

2.3 问题二分析

2.3.1 三维空间覆盖宽度、重叠率模型的建立

问题二要求给出矩形待测海域多波束测深覆盖宽度的数学模型，关键在于通过测线垂面、测线竖直投影面将问题转化为问题一模型进行求解。问题二与问题一的区别在于：（1）测线方向与坡面法向在水平面上的投影的夹角不再固定在 90°。（2）问题一聚焦于求出不同测线在同一水平面上时的覆盖宽度、重叠率；问题二聚焦于求出同一测线不同位置的波束覆盖宽度。

我们建立空间直角坐标系，这样做可以清晰表示不同平面、不同直线的方向向量，易得到相互之间的角度关系。再通过向量计算，先得到与测线垂直平面交坡面所得的交线与水平面的夹角，将此夹角与问题一的坡度相联系，不防称此角大小为斜切坡度。再通过计算得到测线在坡面上的投影与水平面形成的夹角，以此求出不同点处的深度。已知问题一模型中波束覆盖宽度与深度、坡度有关，故我们求出同一条测线不同位置上深度、测线斜切坡度后，即可将三维模型中的剖面转移至二维平面，通过问题一所得结论求解。

2.3.2 测量船距海域中心点处不同距离覆盖宽度的计算

题目给出测量船距海域中心点处的距离及测线方向夹角，要求计算不同条件下的波束覆盖宽度。可以通过 2.3.1 所建立的数学模型，代入深度、局部坡度数值，得出结果。

2.4 问题三分析

问题三要求给出满足一定条件下测线组总测量长度最短的测线设计方案。故考虑建立优化模型求解。我们通过建立空间直角坐标系得出不同直线、平面的方向向量，并设出测

线间距、测线方向，以此作为决定测线总长的决策变量。不同位置的深度、覆盖宽度、覆盖率可在问题一、二模型的基础上进行求解，步骤与问题一、二类似。接下来我们得出测线总长式子，并以此作为目标函数。再根据题目对覆盖率、相邻条带之间的重叠率的要求，对决策变量的范围进行确定，得到约束条件。

2.5 问题四分析

问题四要求求出满足一定条件下的：测线总长度、漏测海区占总待测海域面积的百分比；重叠区域中重叠率超过 20%部分的总长度。目标在于建立多目标优化模型以满足以下条件的测线设计：测线波束所形成的条带对海域覆盖越多越好，尽可能覆盖全海域；相邻条带间重叠率尽量控制在 20%以下；尽可能缩短测线总长度。

我们首先通过附件表所给出数据，结合双线性插值法，将此海域内海底地形全面进行模拟，可得出任意一点的海底深度，形成水深“字典”。再将矩形海域进行分格，利用测线与过划分点所作测线垂线将海域分为许多个矩形小块，对每个小块先单独分析，利用问题一、问题二模型得出相应覆盖宽度、相邻条带重叠率，再计算漏测、重测情况。最后累加全部小块的漏测、重测情况，得出总的计算结果。

定义决策变量为测线间距 d、测线方向 β 。根据题目条件，应用帕累托最优解进行线性加权求出一变量为漏测面积、测线总长度、相邻条带间重叠率大于 20%的测线长度的目标方程。此目标方程的最小值即满足题目所给的多个约束条件：测线波束所形成的条带对海域覆盖越多越好，尽可能覆盖全海域；相邻条带间重叠率尽量控制。

三　模型假设

为了使数学模型更加精确，本文建立如下合理假设：

假设一：在海底地形有一定坡度的情况下，对覆盖宽度与条带重叠率为海底地形水平时的定义进行补充延伸：

（1） 覆盖宽度 W 为多波束测深条带面与坡面形成的交线长度。

（2） 若定义 x 为两多波束测深条带重叠部分长度，前一条测线（水深更深）覆盖宽度为 W0，则 η=x/W0。

如图 1 所示：

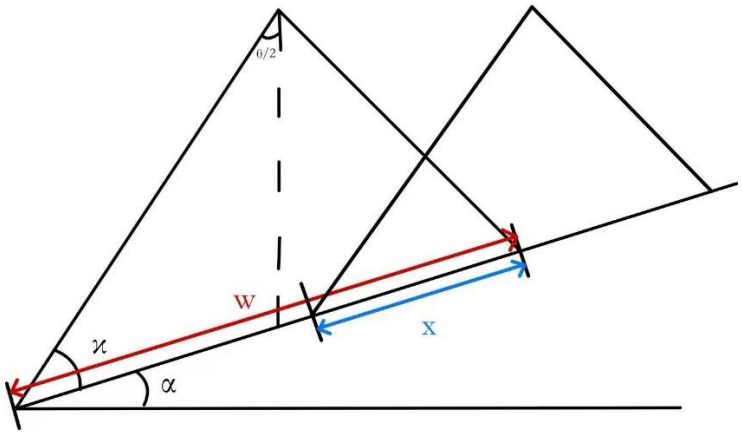


图 1：二维平面测线船波束覆盖范围及重叠率示意图

假设二：测量船可进行精准定位，测量过程中可以得到测量船的精确位置。

假设三：测量船总能按设计出的测线方向进行运动，不会因客观条件（如风浪）而形成的误差。

四　符号说明

本文中涉及的符号如下表所示：

表 1：本文主要涉及的符号说明

符号	说明	单位
Di	第 i 条测线所处位置的深度	m
d	测线间距	m
I,E	方向向量	/
α、$\alpha 1$	坡度	°
$\alpha 2$	测线坡面的投影直线与海平面所成角	°
μ	测线方向角	°
F	漏测区域的面积	㎡

五　模型的建立与求解

5.1 问题 1——二维平面内测深模型的建立

5.1.1 二维平面多波束测深模型的建立

已知水深为 D_i，不同测线位置的水深可与测线间距的距离有关。下以第 i 条测线与第 i+1 条测线为例。

不妨设：

$$D_{i+1} < D_i \tag{1}$$

可以由以下式子得出不同测线位置处对应的海水深度：

$$D_{i+1} = D_i - tan\alpha \cdot d \tag{2}$$

过角度关系及正弦定理得出覆盖宽度 W。

应用**正弦定理**，求出 x1，

$$\frac{D}{\sin\left(\frac{\pi}{2}-\frac{\theta}{2}-\alpha\right)} = \frac{x1}{\sin\left(\frac{\theta}{2}\right)} \tag{3}$$

$$x1 = D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\left(\frac{\pi}{2}-\frac{\theta}{2}-\alpha\right)} \tag{4}$$

下求出 x2，

$$\gamma = \pi - \theta - \beta \tag{5}$$

$$\beta = \frac{\pi}{2} - \frac{\theta}{2} + \alpha \tag{6}$$

$$\frac{D}{sin\gamma} = \frac{x2}{\sin\left(\frac{\theta}{2}\right)} \tag{7}$$

$$x2 = D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\gamma} \tag{8}$$

对于距中心处距离为 D 的测线，求得

$$w1 = x1 + x2 \tag{9}$$

代入 x1，x2，即有

$$w1 = D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{cos\left(\frac{\theta}{2} + \alpha\right)} + D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\gamma} \tag{10}$$

又因为 α，θ，γ 保持不变。故同理，得出 D_{i+1} 后，

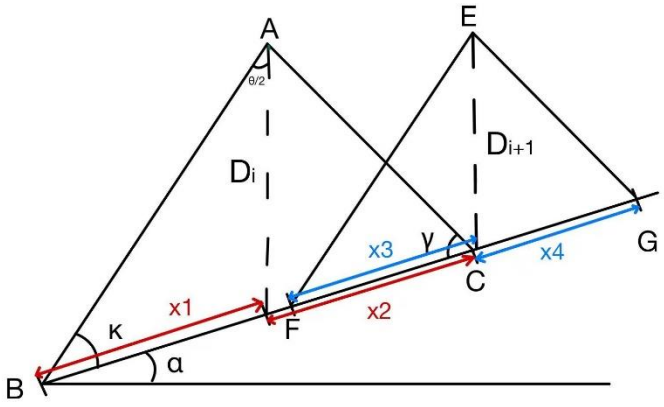
$$x3 = Di + 1 \cdot \frac{sin\left(\frac{\theta}{2}\right)}{cos\left(\frac{\theta}{2} + \alpha\right)} \tag{11}$$

$$x2 = Di + 1 \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\gamma} \tag{12}$$

得出两条测线重叠部分长度，

$$x5 = x2 + x3 - \frac{d}{cos\alpha} \tag{13}$$

$$\eta = \frac{x5}{w1} = \frac{D_{i+1} \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\gamma} + D_{i+1} \cdot \frac{sin\left(\frac{\theta}{2}\right)}{cos\left(\frac{\theta}{2}+\alpha\right)} - \frac{d}{cos\alpha}}{D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{cos\left(\frac{\theta}{2}+\alpha\right)} + D \cdot \frac{sin\left(\frac{\theta}{2}\right)}{sin\gamma}} \tag{14}$$



5.1.2 计算结果填表

已知 D_5=70m，代入上述模型，运行附录 1 程序，可得出距中心点不同距离测线对应水深。

表 1：问题 1 的计算结果

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	90.9487	85.7116	80.4744	75.2372	70.0000	64.7628	59.5256	54.2885	49.0513
覆盖宽度/m	315.8133	297.6276	279.4418	261.2560	243.0703	224.8845	206.6987	188.5130	170.3272
与前一条测线的重叠率/%	——	0.3364	0.2959	0.2500	0.1978	0.1378	0.0681	-0.0139	-0.1117

5.2 问题二——三维空间内测深模型的建立

本题核心在于通过建立数学模型，将三维空间转化为二维平面问题求解，即将问题二转化为问题一模型进行求解。核心在于求出：

（1）斜切坡度

（2）同一测线上距海域中心点处不同距离的测量船所处位置处海的深度。

5.2.1 三维空间多波束测深模型的建立

step1 建立平面直角坐标系，并求出基础向量

作坡面法向在水平面的投影向量，记为 $\boldsymbol{x}$。于水平面内，在 $\boldsymbol{x}$ 逆时针旋转 90° 方向，作向量 $\boldsymbol{y}$。作一向量垂直于水平面，记为 $\boldsymbol{z}$。

以海域中心点在水平面的投影点为原点 0，$\boldsymbol{x}$ 向量为 x 轴，$\boldsymbol{y}$ 向量为 y 轴，$\boldsymbol{z}$ 向量为 z 轴。建立空间直角坐标系 O-xyz。

得出相应向量表示，

$$I1=\begin{pmatrix}cos\beta\\sin\beta\\0\end{pmatrix},I2=\begin{pmatrix}sin\alpha\\0\\cos\alpha\end{pmatrix}$$

记水平面法向量为 I4，

$$I4=\begin{pmatrix}0\\0\\1\end{pmatrix}$$

step2 求同一测线上测量船距海域中心点不同距离时的深度

记测线在坡面的投影向量为 I5，故 I5 与 I1 横纵坐标相同。

再由

$$I5\cdot I2=0 \tag{15}$$

得 $I5=\begin{pmatrix}cos\beta\\sin\beta\\-tan\alpha\cdot cos\beta\end{pmatrix}$

在二维平面内作出测线垂平面 EFGH，如下图。下求 I5 与水平面夹角 ω：

利用线面向量夹角公式，

$$sin\alpha 2=\frac{I5\cdot I4}{|I5|\cdot|I4|} \tag{16}$$

$$\alpha 2=arcsin\alpha 2=arcsin\frac{I5\cdot I4}{|I5|\cdot|I4|} \tag{17}$$

下求测量船距海域中心点处距离为 d 时的海底深度 D

$$D=120-d\cdot tan\alpha 2 \tag{18}$$

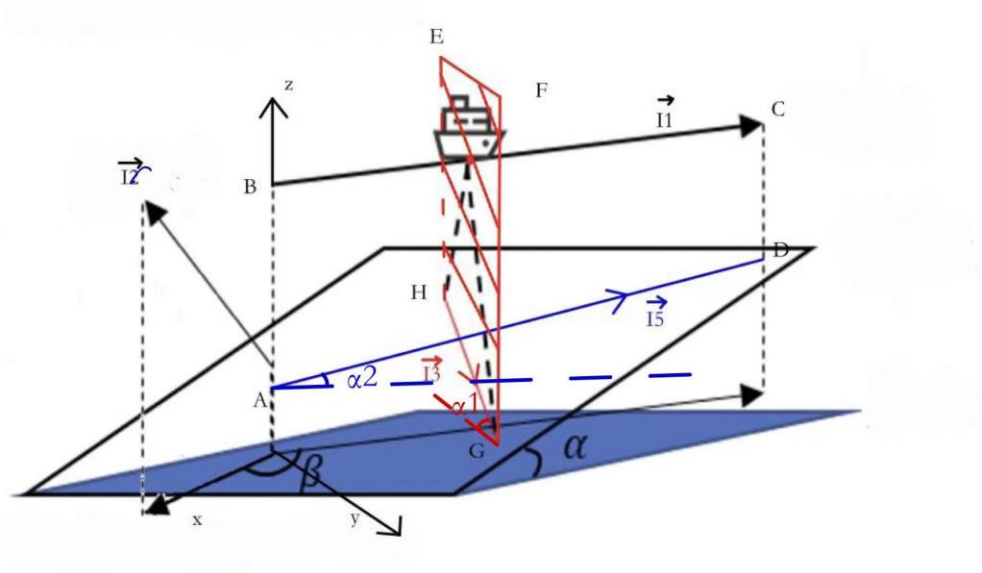


图 2：三维空间内各向量及测线垂面示意图

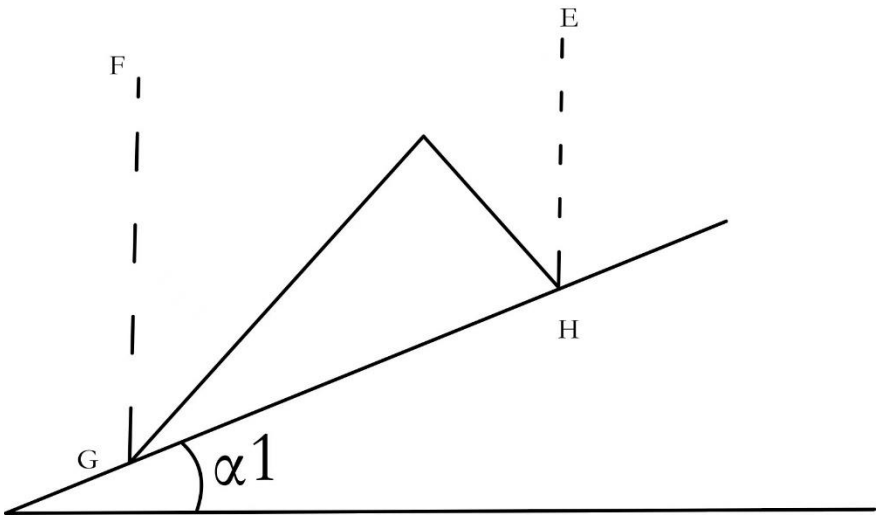


图 3：测线垂面（图 2 红色阴影面）转移至二维平面示意图

step3 求斜切坡度

因为 I1 垂直于 I3，I2 垂直于 I3。利用 I1×I2，即可得出同时垂直于 I1、I2 的向量的表达式:

$$I3 = I1 \times I2 = \begin{pmatrix} cos\beta \\ sin\beta \\ 0 \end{pmatrix} \times \begin{pmatrix} sin\alpha \\ 0 \\ cos\alpha \end{pmatrix} = \begin{pmatrix} sin(\beta)cos(\alpha) \\ -cos(\alpha)cos(\beta) \\ -sin(\alpha)sin(\beta) \end{pmatrix} \tag{19}$$

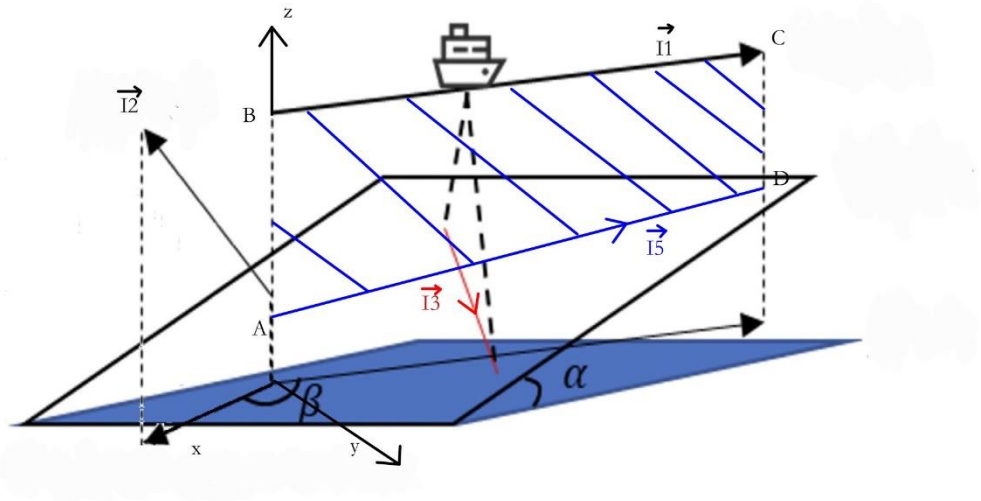


图 4：三维空间内测线投影面的示意图

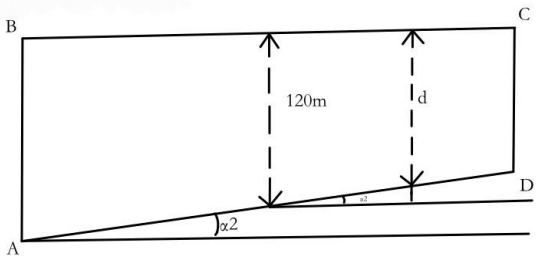


图 5：测线投影面（图 4 蓝色阴影面）转移至二维平面示意图

I3 向量与水平面夹角即问题 1 处坡面与水平面夹角。

将 ABCD 面在二维平面内画出，如图。利用线面向量夹角公式求出 I3 与水平面所成角 $\alpha 1$:

$$sin\alpha 1 = cos\left(\frac{\pi}{2} - \alpha 1\right) = \frac{I3 \cdot I4}{|I3| \cdot |I4|} \tag{20}$$

$$\alpha 1 = arcsin\alpha 1 = arcsin\frac{I3 \cdot I4}{|I3| \cdot |I4|} \tag{21}$$

如上图，已知$\alpha 1$，D，即可利用问题一所建立模型求解。

5.2.2 计算结果填表

由于测线方向夹角相差 180° 的一组测线向量方向相反，故讨论测量船距海域中心点处不同距离的覆盖宽度时，在沿测线方向上确定测量船距海域中心点处的距离即可，不必沿相反方向找覆盖宽度，就能保证数据的完整性。

运行附录 2 程序，可得如下结果，

表 2：问题 2 的计算结果

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	415.6922	466.0911	516.4899	566.8888	617.2876	667.6865	718.0854	768.4842
	45	416.1915	451.8717	487.5519	523.2321	558.9123	594.5924	630.2726	665.9528
	90	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919
	135	416.1915	380.5113	344.8312	309.1510	273.4708	237.7906	202.1104	166.4302
	180	415.6922	365.2933	314.8945	264.4956	214.0967	163.6979	113.2990	62.9002
	225	416.1915	380.5113	344.8312	309.1510	273.4708	237.7906	202.1104	166.4302
	270	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919	416.6919
	315	416.1915	451.8717	487.5519	523.2321	558.9123	594.5924	630.2726	665.9528

5.3 问题三——海底地形规则时优化模型的建立与求解

已知多波束换能器开角 120°，坡度 1.5°，海域中心点处海水深度为 120m。

解决问题三我们的总思路为通过建立数学模型，定义决策变量、约束条件，得到测线总距离目标函数。通过优化模型的求解得出测线设计方案。具体如下：

5.3.1 优化模型的建立

step1 建立空间直角坐标系，并求出基础向量。

以海域海底西北角为原点，向正南方向作 x 轴，向正东方向作 y 轴，向正北方向作 z 轴，如图，建立空间直角坐标系。

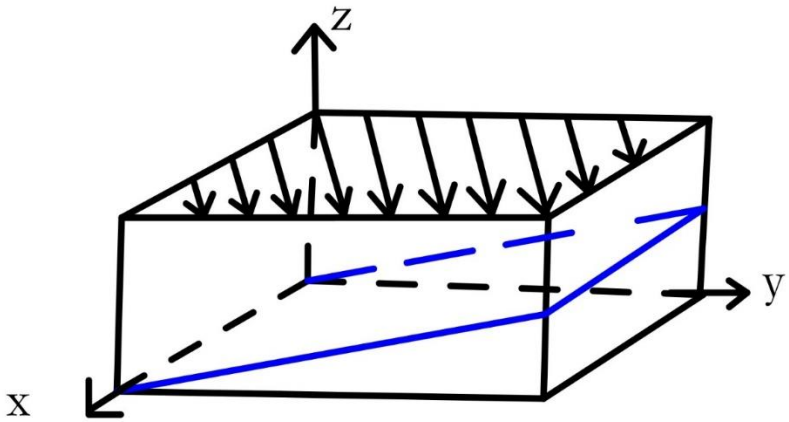


图 6：问题 3 海域空间直角坐标系示意图

可得坡面法向量

$$E1 = \begin{pmatrix} 0 \\ -sin1.5^\circ \\ cos1.5^\circ \end{pmatrix},$$

测线方向向量

$$E2 = \begin{pmatrix} cos\mu \\ sin\mu \\ 0 \end{pmatrix},$$

水平面法向量

$$E5 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

step2 建立模型

聚焦于海域上平面矩形，设组测线成平行关系，且等距，距离为 d。设测线与南北方向线成 μ 角。作出矩形内测线模型，如图（注：图中未画完全部测线，每条测线间有多条测线未画出）：

将全部测线分为两类：

（1） 第一类测线全部等长，与矩形的两个交点分别位于相互平行的两矩形边上。长度与|AE|相等。如 CD 测线。

（2） 第二类测线长度不相等，与矩形的两个交点分别位于相互垂直的两矩形边上但由于测线相互平行且测线间距相等，故随着位置的单向推移测线长度成等差数列。如 IJ 测线。第二类测线组首项为 0，末项为 2|OA|，项数为 n-i+1.

为使计算简便，在误差较小的前提下，我们不妨设第一类测线最左侧测线过 D 点，最右侧测线过 G 点。

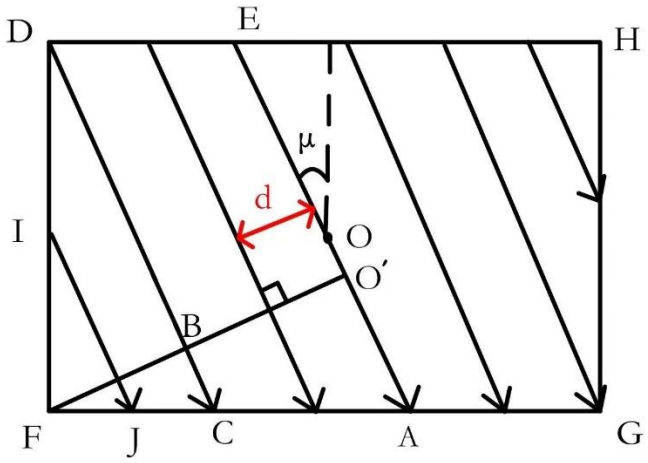


图 7：海域顶面示意图

O 点为矩形中点，AE 为过矩形中点测线。自 AE 将矩形分为左右两部分。为使计算简便，先对左侧进行分析，右侧情况可由对称性近似得出。通过查阅资料，误差较小，所得结果偏差不大。

作 O′F 垂直于测线交 DC 测线于 B 点。设 O′B 与 AE 左侧（不包含 AE）、BD 右侧（包含 BD）的测线共形成 i 个交点，OC 共与 AE 左侧（不包含 AE）测线形成 n 个交点。考虑极端情况，最左侧测线与 F 点相交。故对 i、n 的理解可定为：AE 左侧共有 n 条测线，其中共有 i 条第一类等长测线。

即

$$\begin{cases} i \cdot d < |O'B| \\ (i+1) \cdot d > |O'B| \end{cases} \tag{22}$$

又通过角度、线段长关系，得到

$$|OA| = \frac{|OI|}{cos\mu} \tag{23}$$

$$|O'F| = \frac{2}{cos\mu} \tag{24}$$

设测线于坡面上的投影向量为 E3，设过测线作垂平面交坡面交线方向为 E4，

$$E4 = E2 \times E1 \tag{25}$$

可通过 E4 向量得到 E4 与水平面的夹角为$\alpha 1$，E3 与水平面的夹角为$\alpha 2$。

此时应用第一问的模型。其他条件确定的情况下，已知测线垂平面与坡面的交线向量即可知道此交线与水平面的夹角。即求出 E4，得出$\alpha 1$。$\alpha 1$在此问中类似于问题一的α角。再应用第二问的模型，已知$\alpha 2$即可得出对应位置的深度 D。有了深度、$\alpha 1$、测线距离 d，即可通过 W、η 公式进行计算。

例如，给定 OC 方向向量、测线间距 d，求出 OC 向量在坡面上的投影与水平面所形成的角，即可得到与 OC 线相交的一系列测线的相邻条带间的重叠率。

$$\eta_n = \left(D_{n-1} \cdot \frac{sin60^\circ}{sin(30^\circ + \alpha 1)} + D_n \cdot \frac{sin60^\circ}{\sin(30^\circ - \alpha 1)} - \frac{d}{\cos \alpha 1} \right) \cdot \left(\frac{1}{w_n} \right) \tag{26}$$

将$\alpha 1$、d 代入第一问模型，可得 OC 上 n 个点的覆盖宽度与重叠率。wn 为最左侧测线波束覆盖宽度。

由于要求海域全覆盖，即要求最左侧测线覆盖宽度覆盖西南角：

$$n \cdot d + \frac{w_n}{2} > |O'F| \tag{27}$$

为使计算简便，又由于误差较小，此处将最左侧测线覆盖宽度左右两边近似相等。

矩形左侧测线总长

$$l = 2|OA| \cdot i + 2|OA| \cdot \frac{n - i + 1}{2} = |OA| \cdot (n + i + 1) \tag{28}$$

为使计算简便，不防假设左右两侧测线数目近似相等。通过查阅资料得知这种做法误差较小，故近似成立。求出测线组总长

$$L = 2|OA| \cdot (n + i + 2) \tag{29}$$

决策变量：

由于可以通过选择合适的测线间距及测线方向来使测线组总长度最短，已知测线间距、测线方向可通过不等式组得出测线组中测线数量等需要知道的变量。因此以测线间距、测线与 x 轴夹角为决策变量，即决策变量为

μ，d

目标函数：

由于最终目标是设计一组测量长度最短、可完全覆盖整个待测海域的测线，因此以测线组总长度最小为目标函数，即目标函数为

$$\min L = 2|OA| \cdot (n + i + 2) \tag{30}$$

约束条件：

①重叠率约束：由于海域地形西低东高，通过问题一，我们可以知道重叠率随海域深度的加深而增大。故欲满足题目所要求的相邻条带间重叠率满足 10%-20%，我们只需控制最左侧测线与相邻条带重叠率小于百分之二十，最右侧测线与相邻条带重叠率大于百分之十即

可。

η_n 表示最左侧测线重叠率，η_{-n} 表示最右侧测线重叠率。

$$\begin{cases} \eta_n < 20\% \\ \eta_{-n} > 10\% \end{cases} \tag{31}$$

②覆盖率约束：要求保证实现海域全覆盖，故

$$n \cdot d + \frac{w_n}{2} > |O'F| \tag{32}$$

③边界约束：

$$d \in （0，|FH|）$$

④测线方向约束：

$$\mu \in [0，\ \pi/2]$$

综上所述，建立测线组总长度最小优化模型

$$minL = 2|OA| \cdot (n + i + 2) \tag{33}$$

$$\text{s.t.} \begin{cases} \eta_n < 20\% \\ \eta_{-n} > 10\% \\ n \cdot d + \frac{w_n}{2} > |O'F| \\ d \in （0，|FH|） \\ \mu \in \left[0，\ \frac{\pi}{2}\right] \end{cases}$$

5.3.2 优化模型求解

求解过程使用了 Python 中的 scipy 库，应用 optimize 模块中的 minimize 函数对优化函数进行求解，求解方法为‘SLSQP’。程序代码见附录 3。

得到：测线与南北方向夹角 μ 约为 79.20°，测线间距 d 为 567.43m 时，在满足全海域覆盖、重叠率 10%-20%的条件下，能使得测线组总长度最小，最小值为 158136.87m。

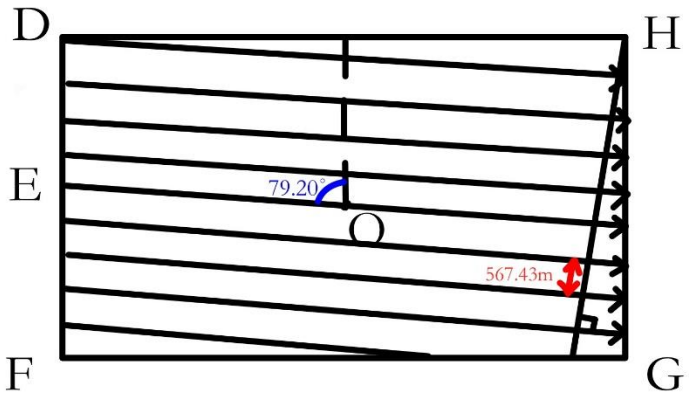


图 8：优化模型最终解示意图

表 3：最终解各测线详细数据

覆盖率	——	0.1	0.11345	0.126504	0.139179	0.151491	0.165673	0.177438	0.188876	0.2
波束覆盖宽度	626.1525	635.7983	645.444	655.0897	664.7354	674.3812	684.0269	693.6726	703.3184	712.9641
海水深度	180.7394	183.5237	186.3079	189.0922	191.8764	194.6607	197.4449	200.2292	203.0134	205.7977
测线距离中心点处的距离	-2837.15	-2269.72	-1702.29	-1134.86	-567.43	0	567.4297	1134.859	1702.289	2269.719

5.4 问题四——海底地形不规则时优化模型的建立与求解

首先通过已给附件数据大致作出如下海水深度分布图，判断出海底地形为不规则形状。画图程序为附录 4。

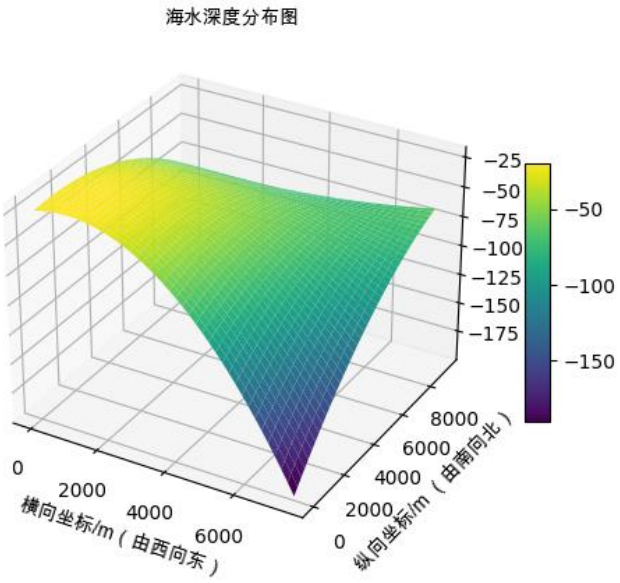


图 9:问题 4 中海水深度分布图

由于所给限制条件较多，需满足的条件也不止一个。故此题我们通过建立多目标优化

模型进行求解。根据参考文献所提，由实践得知，航线与江流方向平行即为最优解[2] 。故本题中， $\beta \in \left[0, \frac{\pi}{2}\right]$

以海域西北角为原点，建立空间直角坐标系。设测线方向与 x 轴成 β 角，测线距离为 d。设中心测线（过矩形中心点的测线）上的点为 O_{0i}。O_{oo}（2.5,2），不妨取同一测线上每一个点间隔为 T。则 Oji（$2.5+i\cdot T\cdot\cos\beta+ j\cdot T\cdot\sin\beta$，$2-j\cdot\cos\beta\cdot d\cdot i\cdot T\cdot\sin\beta$）

至此，可将海域内任意测线上任意一点的坐标进行表示。

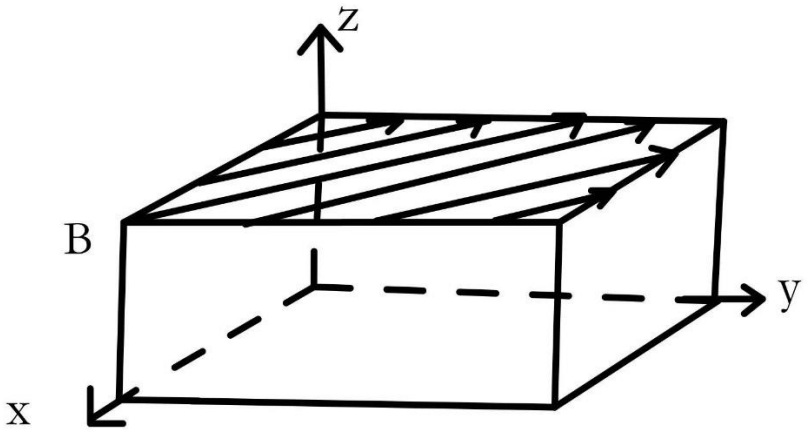


图 10：问题 4 海域空间直角坐标系示意图

利用附件所给数据，通过双线性插值法，可将每一侧线上不同划分点的深度得出。

利用测线与过划分点所作测线垂线将海域分为许**多个矩形小块**，单独研究每个矩形小块所涉及的测线的条带覆盖宽度及重叠率。

下选取一过海域中心点的矩形 $O_{00}O_{01}O_{10}O_{11}$ 为例进行分析。其他矩形块同理，可用相同方法类似求出。

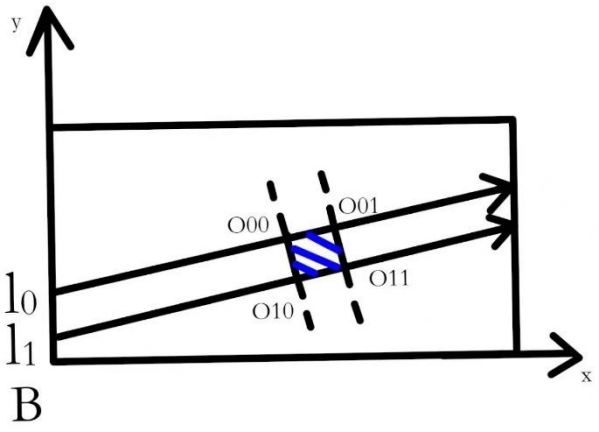


图 11：问题 4 海域顶面及矩形小块示意图

先研究中心测线 I0，$O_{00}O_{01}O_{10}O_{11}$ 矩形区域内在误差范围内可视为平坦坡面。

得到 $O_{00}O_{10}$ 直线方程，

$$y = tan(\beta + \pi/2)\cdot(x - 2.5 - i\cdot T\cdot cos\beta) + 2 + i\cdot T\cdot sin\beta \quad (34)$$

找研究海域内横纵坐标符合上述直线的点（如下图），进行**拟合**，求其**二次回归直线**方向。

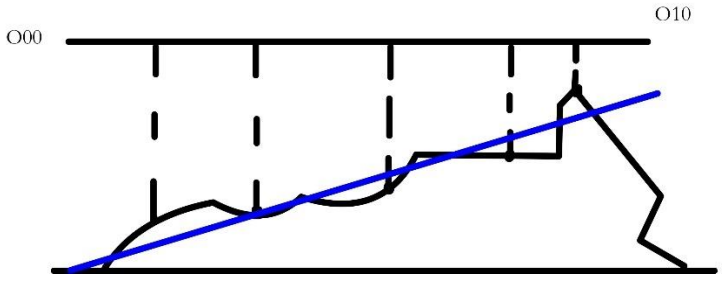


图 12：底面坡度拟合示意图

将二次回归直线方向向量与海平面的夹角视为坡度角。即可对 10、11 在矩形 $O_{00}O_{01}O_{10}O_{11}$ 内的部分应用问题一、问题二模型进行求解。可以得到条带覆盖宽度 W0，W1 及两条相邻测线这部分的重复率 η 。

若 η <0，记 $O_{00}O_{01}O_{10}O_{11}$ 四边形为漏测区域 F_0；若 n>20%，记 $O_{00}O_{10}$ 为重叠率超过 20%部分的测线长度 L_0。

至此对矩形 $O_{00}O_{01}O_{10}O_{11}$ 区域分析结束。将此分析拓展至海域全部矩形小块中，对漏测区域进行分类累加。得到漏测总面积 S、重叠率超过 20%部分的测线总长度 L1、测线组总长度 L2。

此处应用**帕累托最优解**进行线性加权。

（1） 求出不考虑 L1，L2 取值大小的前提下 S 的最小值，记为S'。

（2） 求出不考虑 S，L2 取值大小的前提下 L1 的最小值，记为$L1'$。

（3） 求出不考虑 L1，S 取值大小的前提下 L2 的最小值，记为$L2'$。

决策变量：

$$\beta,\ d$$

目标函数：

$$minH = \sqrt{\left(\frac{(S'-S)}{S}\right)^{2+}\left(\frac{(L1'-L1)}{L1}\right)^{2+}\left(\frac{(L2'-L2)}{L2}\right)^{2}} \tag{35}$$

通过计算机多目标优化程序进行求解，得出优化结果，确定 β，d，结合问题一、二、三的模型，求出 L1、L2，得到测线总长度、重叠率超过 20%部分的总长度。

通过面积计算得海域总面积 S_0，则漏测海区占总待测海域面积百分比：

$$\frac{S}{S0}\cdot 100\% \tag{36}$$

附录 5 给出了建立此模型的代码框架。

六　模型的评价

6.1 模型优点

（1）　采用建系方法将全海域进行数据覆盖，进行向量运算，求解简便，思路清晰。
（2）　问题一模型适用于二维平面，问题二模型适用于三维空间，具有广泛性。
（3）　模型较为简洁，具有普适性。应用时已知一定关键数据即可得到相应的条带覆盖宽度、覆盖率等结果。
（4）　选择了 minimize 函数使用‘SLSQP’法进行优化模型的求解，较为简便，有效满足了要求条件。

6.2 模型缺点

（1）问题三建立模型得到最短测线设计过程中使用了近似求解，存在一定误差
（2）当水深数据较少时无法较为精确地得到海底地形全数据，具有一定局限性。

参考文献

[1] 余启义.基于多波束测深技术的海底地形测量.测绘与空间地理信息,2022,45（09）.
[2] 付勇先.多波束测深系统在大洋河涨潮段水下地形测量的应用.红水河,2021,40(04).

附录

附录 1：问题一模型建立及数据计算 python 程序

```
import math
import pandas as pd

# 坡度角
alpha = math.radians(1.5)
# 多波束换能器的开角
theta = math.radians(120)
# 波束与坡面近坡底夹角
kappa = math.pi/2 - theta/2 - alpha
# 波束与坡面远坡底夹角
gamma = math.pi/2 - theta/2 + alpha

# 侧线距离中心点处的距离
distance = [-800., -600., -400., -200., 0., 200., 400., 600., 800.]
```

```
# 计算波束覆盖宽度
def length_W(D):
    x1 = D * (math.sin(theta/2)/math.sin(kappa))
    x2 = D * (math.sin(theta/2)/math.sin(gamma))
    return x1 + x2,x1,x2

# 计算深度的函数
def depth(distance):
    return 70 - math.tan(alpha) * distance

# 计算不同波束之间的重叠宽度
def overlap_width(x2,x3,d):
    return x2 + x3 - d/math.cos(alpha)

# 自己提出的重叠率计算公式
def overlap_rate(overlap,w):
    return overlap/w

# 题目给出的重叠率计算公式
def overlap_rate2(d,w):
    return 1 - d/w

# 海水深度
D = [depth(dis) for dis in distance]
# 波束覆盖宽度
W = [length_W(d)[0] for d in D]
# 9 个波束之间的 8 条重叠宽度
overlap = []
for i,d in enumerate(D):
    if i < len(D)-1:
        x2 = length_W(d)[2]
        x3 = length_W(D[i+1])[1]
        overlapped_width = overlap_width(x2,x3,distance[i+1] - distance[i])
        overlap.append(overlapped_width)

# 计算重叠（自己提出的公式）
overlap_rate = [overlap_rate(overlap[i],W[i]) for i in range(len(overlap))]

data = {
    "海水深度": D,
    "波束覆盖宽度": W,
    "重叠宽度": overlap + [None],
    "重叠率 1": overlap_rate + [None],
}
```

```
# 创建 DataFrame
df = pd.DataFrame(data)
df_transposed = df.transpose()
# 保存为 xlsx 文件
df_transposed.to_excel("question1_data.xlsx", header=False)

# print(f"海水深度{D}")
# print(f"波束覆盖宽度{W}")
# print(f"重叠宽度{overlap}")
# print(f"重叠率{overlap_rate}")
```

附录 2：问题二模型建立及数据计算 python 程序

```
import math
import pandas as pd

beta_degree = [0, 45, 90, 135, 180, 225, 270, 315]
beta = [math.radians(degree) for degree in beta_degree] # 弧度
alpha = math.radians(1.5)
# 测量船距离海域中心点处的海里
distance_nmile = [0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1]
# 测量船距离海域中心点处的距离（米）
distance = [d * 1852 for d in distance_nmile]

# 计算波束面与坡面的交线与水平面的夹角角度的函数
def angle(beta):
    # I1 向量
    I1 = [math.cos(beta), math.sin(beta), 0]
    # I2 向量
    I2 = [math.sin(alpha), 0, math.cos(alpha)]
    # 求 I1 和 I2 的叉乘 I3
    I3 = [math.fabs(I1[1] * I2[2] - I1[2] * I2[1]),
          math.fabs(I1[2] * I2[0] - I1[0] * I2[2]),
          math.fabs(I1[0] * I2[1] - I1[1] * I2[0])]
    # 水平面法向量 I4
    I4 = [0, 0, 1]

    # I3 和 I4 的夹角
    cos_angle = (I3[0] * I4[0] + I3[1] * I4[1] + I3[2] * I4[2]) / \
                (math.sqrt(I3[0] ** 2 + I3[1] ** 2 + I3[2] ** 2) *
                    math.sqrt(I4[0] ** 2 + I4[1] ** 2 + I4[2] ** 2))
    # 弧度制
```

```
    return math.asin(cos_angle)

# 六个侧线方向夹角情况下的波束面与坡面的交线与水平面的夹角
PHI = [angle(b) for b in beta]

# 计算 phi 的函数
def cal_omega(beta):
    # 水平面法向量 I4
    I4 = [0, 0, 1]
    # 侧线在坡面上的投影向量 I5
    I5 = [math.cos(beta), math.sin(beta),
          (-math.cos(beta) * math.sin(alpha)) / math.cos(alpha)]
    sin_omega = (I5[0] * I4[0] + I5[1] * I4[1] + I5[2] * I4[2]) / \
                        (math.sqrt(I5[0] ** 2 + I5[1] ** 2 + I5[2] ** 2) *
                        math.sqrt(I4[0] ** 2 + I4[1] ** 2 + I4[2] ** 2))
    omega = math.asin(sin_omega)
    return omega

# 根据 distance 计算深度的函数
def depth(distance,beta):
    omega = cal_omega(beta)
    return 120 - distance * math.tan(omega)

# 计算覆盖宽度
def cover_width(distance, beta, phi):
    D = depth(distance, beta)
    # print(D)
    # 多波束换能器的开角
    theta = math.radians(120)
    # 波束与斜坡面近坡底夹角
    kappa = math.pi / 2 - theta / 2 - phi
    # 波束与斜坡面远坡底夹角
    gamma = math.pi / 2 - theta / 2 + phi
    # 波束覆盖宽度
    x1 = D * (math.sin(theta / 2) / math.sin(kappa))
    x2 = D * (math.sin(theta / 2) / math.sin(gamma))
    return x1 + x2

cover_wid_2 = []
# 遍历角度 beta
for i in range(len(beta)):
    cover_wid_1 = []
    # 遍历距离 distance
    for j in range(len(distance)):
```

```
            cover_wid_1.append(cover_width(distance[j], beta[i], PHI[i]))
        cover_wid_2.append(cover_wid_1)

# 创建一个 DataFrame
df = pd.DataFrame(cover_wid_2, columns=[str(d) + '海里' for d in distance_nmile],
index=[str(b) + '°' for b in beta_degree])
# 保存到 xlsx 文件
df.to_excel('question2_data.xlsx')

# 计算得到的侧坡面角度与侧线与平面角度
# print([round(math.degrees(PHI[i]),4) for i in range(len(PHI))])
# print([round(math.degrees(cal_omega(b)),6) for b in beta])

# 深度数据
# for i in range(len(beta)):
#     print([round(depth(d,beta[i]),6) for d in distance])
```

附录 3：问题三 python 建模及优化模型求解程序

```
import math
import numpy as np
from scipy.optimize import minimize
import pandas as pd

# 波速探测器开角
theta = math.radians(120)
# 坡面坡度
alpha = math.radians(1.5)
# 平面长宽
Height = 2*1852
Length = 4*1852

# 决策变量
# mu = params[0]
# d = params[1]

def compute_alpha_angles(mu, alpha):
    # 坡面法向量
    E1 = [0, -math.sin(alpha), math.cos(alpha)]

    # 测线方向
    def cal_E2(mu):
        return [math.cos(mu), math.sin(mu), 0]
```

```
    # 过测线作平面交坡面的交线方向
    def cal_E3(E1, E2):
        return [E2[0], E2[1], -E1[1] * E2[1] / E1[2]]

        # 过测线作垂平面交坡面交线方向

    def cal_E4(E1, E2):
        return np.cross(E2, E1)

    # 水平面法向量
    E5 = [0, 0, 1]

    # 计算 alpha_1 和 alpha_2
    E2 = cal_E2(mu)
    E3 = cal_E3(E1, E2)
    E4 = cal_E4(E1, E2)

    def cal_alpha_1(E4, E5):
        sin_alpha_1 = np.dot(E4, E5) / (np.linalg.norm(E4) * np.linalg.norm(E5))
        return math.asin(sin_alpha_1)

    def cal_alpha_2(E3, E5):
        sin_alpha_2 = np.dot(E3, E5) / (np.linalg.norm(E3) * np.linalg.norm(E5))
        return math.asin(sin_alpha_2)

    alpha_1 = cal_alpha_1(E4, E5)
    alpha_2 = cal_alpha_2(E3, E5)

    return alpha_1, alpha_2

# 第一问的模型，给定 alpha_1 和测线间隔 d，可以求出 OC 上的一系列测线的重叠率
def calculate_overlap_rate(alpha_1=None, alpha_2=None,theta_degree=120, mu=None,
d=None):

    X3 = np.abs(Length/2 + (Height/2)*math.tan(mu))
    X2 = np.abs(X3*math.sin(mu))
    X1 = np.abs(X3*math.cos(mu))
    OA = np.abs((Height/2)/math.cos(mu))
    X4 = OA - X2
    X5 = Length/2 - (Height/2)*math.tan(mu)
    X6 = X5*math.cos(mu)

    # 共有 n 条线
```

```
n = np.abs(int(X1 // d))
# n 条测线距离中心点处的距离
distance_list = [i * d for i in range(0, n+1)]

# X6 内总共有 i 条线
I = np.abs(int(X6 // d))

# 坡度角
alpha = alpha_1
# 多波束换能器的开角
theta = math.radians(theta_degree)
# 波束与坡面近坡底夹角
kappa = math.pi / 2 - theta / 2 - alpha
# 波束与坡面远坡底夹角
gamma = math.pi / 2 - theta / 2 + alpha

# 计算波束覆盖宽度
def length_W(D):
    x1 = D * (math.sin(theta / 2) / math.sin(kappa))
    x2 = D * (math.sin(theta / 2) / math.sin(gamma))
    return x1 + x2, x1, x2

# 起始点深度
D0 = 110 - X4*math.tan(alpha_2)
# 计算深度的函数
def depth(distance):
    return D0 - math.tan(alpha) * distance

# 计算不同波束之间的重叠宽度
def overlap_width(x2, x3, d):
    return x2 + x3 - d / math.cos(alpha)

# 重叠率计算公式
def overlap_rate(overlap, w):
    return overlap / w

# 海水深度
D = [depth(dis) for dis in distance_list]
# 波束覆盖宽度
W = [length_W(depth)[0] for depth in D]
# n 个波束之间的 n-1 条重叠宽度
overlap = []
for i, depth in enumerate(D):
    if i < len(D) - 1:
```

```
            x2 = length_W(depth)[2]
            x3 = length_W(D[i + 1])[1]
            overlapped_width = overlap_width(x2, x3, np.abs(distance_list[i + 1]
- distance_list[i]))
            overlap.append(overlapped_width)

    # 计算重叠
    overlap_rate_list = [overlap_rate(overlap[i], W[i]) for i in
range(len(overlap))]

    return overlap_rate_list, W, D, distance_list, X1, OA, I , n

# 要求：X1 上最后一条覆盖率不能超过 20%
# 也即 overlap_rate_list_X1[-1] <= 0.2 #约束条件 1

# 要求：X1'上最后一条覆盖率不能低于 10%
# 也即 overlap_rate_list_X1_[-1] >= 0.1 #约束条件 2

# 约束条件 3:n 条测线的宽度总和加上 OC'上最后一条测线的左半波束覆盖宽度必须大于
X1
# Distance_list[-1] + W[-1]/2 >= X1 #约束条件 3

# 约束条件 3:mu 的范围为[0,pi/2]

# 目标方程
def objective_function(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1, alpha_2=alpha_2, theta_degree=120,
mu=mu, d=d)
    return  OA * (2*I + 2*n + 4)

# 约束条件 1
def constraint1(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1, alpha_2=alpha_2, theta_degree=120,
mu=mu, d=d)
    return -overlap_rate_list[-1] + 0.2
```

```
# 约束条件 2
def constraint2(params):
    mu = params[0]
    d = -params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list,    W,    D,    distance_list,    X1,    OA,    I    ,n=
calculate_overlap_rate(alpha_1=alpha_1,    alpha_2=alpha_2,    theta_degree=120,
mu=mu, d=d)
    return overlap_rate_list[-1] - 0.1

# 约束条件 3
def constraint3(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list,    W,    D,    distance_list,    X1,    OA,    I    ,n=
calculate_overlap_rate(alpha_1=alpha_1,    alpha_2=alpha_2,    theta_degree=120,
mu=mu, d=d)
    return distance_list[-1] + W[-1]/2 - X1

# 边界条件
bounds = [(0, math.pi/2), (0, 1.5*1852)]

# 初始条件
initial_guess = [math.radians(70), 0.3 * 1852]

# 约束条件
constraint1 = {'type': 'ineq', 'fun': constraint1}
constraint2 = {'type': 'ineq', 'fun': constraint2}
constraint3 = {'type': 'ineq', 'fun': constraint3}
constraints = [constraint1,  constraint2, constraint3]

# 求解
solution    =    minimize(objective_function,    initial_guess,    method='SLSQP',
bounds=bounds, constraints=constraints, options={'maxiter': 100000})
print(solution)
print(math.degrees(solution.x[0]), solution.x[1])

# 给出最优解的详细数据
def show_detail(mu,d,index):
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list,    W,    D,    distance_list,    X1,    OA,    I,    n    =
calculate_overlap_rate(alpha_1=alpha_1,
```

```
alpha_2=alpha_2,

theta_degree=120,

mu=mu, d=d)
    print(f"覆盖率：{overlap_rate_list}")
    print(f"波束覆盖宽度：{W}")
    # print(f"海水重叠宽度：{overlap}")
    print(f"海水深度：{D}")
    print(f"测线距离中心点处的距离：{distance_list}")

    # data = {
    #     "覆盖率": overlap_rate_list+[None],
    #     "波束覆盖宽度": W,
    #     "海水深度": D,
    #     "测线距离中心点处的距离": distance_list
    # }
    #
    # # 创建一个 DataFrame
    # df = pd.DataFrame(data)
    # df= df.transpose()
    #
    # # 保存到 xlsx 文件
    # df.to_excel(f"question2_data{index}.xlsx", header=False)


show_detail(solution.x[0], solution.x[1],0) # d>0 的一侧
print("-----------------------------------------------")
show_detail(solution.x[0], -solution.x[1],1) # d<0 的一侧
```

附录 4 图 9python 画图程序

```
from pandas import read_excel
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.font_manager import FontProperties

font = FontProperties(fname='/Library/Fonts/Arial Unicode.ttf')

# 读取 Excel 文件
data = read_excel('附件.xlsx')

# 提取横向坐标和纵向坐标
```

```
x_coords = data.iloc[0, 2:].values.astype(np.float64) * 1852
y_coords = data.iloc[1:, 1].values.astype(np.float64) * 1852

# 提取海水深度数据
depth_data = data.iloc[1:, 2:].values.astype(np.float64)

# 创建 X 和 Y 的网格
x, y = np.meshgrid(x_coords, y_coords)

# 创建 3D 图形
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(x, y, -depth_data, cmap='viridis')

# 设置标签
ax.set_xlabel('横向坐标/m（由西向东）', fontproperties=font)
ax.set_ylabel('纵向坐标/m （由南向北）', fontproperties=font)
ax.set_zlabel('深度 (m)', fontproperties=font)

# 设置标题
ax.set_title('海水深度分布图', fontproperties=font)

# 添加颜色条
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)

# 显示图形
plt.show()
```

附录 5 问题 4 模型框架程序

```
from pandas import read_excel
import math
import numpy as np
from scipy.interpolate import RegularGridInterpolator
from scipy.optimize import minimize

# 读取 Excel 文件
data = read_excel('附件.xlsx')

# 提取横向坐标和纵向坐标
x_coords = data.iloc[0, 2:].values.astype(np.float64) * 1852
y_coords = data.iloc[1:, 1].values.astype(np.float64) * 1852
```

```
# 提取海水深度数据
depth_data = data.iloc[1:, 2:].values.astype(np.float64)

# 使用双线性插值得到深度值
def get_depth(x, y):
    interp_func = RegularGridInterpolator((y_coords, x_coords), depth_data,
method='linear')
    return interp_func([[y, x]])[0]

# 测试 get_depth 函数
print(get_depth(2*1852, 2.5*1852))

# 海平面法向量
I_z = [0, 0, 1]

# 中心点 O_0_0
O_0_0 = [2.5 * 1852, 2 * 1852]
# 间隔 T
T = 0.02 * 1852

alpha = math.radians(1.5)
# 海底区域长宽
Height = 5*1852
Length = 4*1852


# 计算 j_i 点参数
def cal_j_i(i, j, mu, d):
    # 计算 O_j_i 位置的函数
    def cal_O_j_i(i, j, mu, d):
        O_j_i = [2.5 * 1852 + j * d * math.sin(mu) + i * T * math.cos(mu),
                 2 * 1852 - j * d * math.cos(mu) + i * T * math.sin(mu)]
        return O_j_i

    O_j_i = cal_O_j_i(i, j, mu, d)
    D_j_i = get_depth(O_j_i[0], O_j_i[1])
    # 计算过 O_j_i 的垂线
    def cal_y(mu, O_j_i, x):
        y = math.tan(mu+math.pi/2) * (x - O_j_i[0]) + O_j_i[1]
        return y

    # 横坐标初始为 O_j_i[0]，在其左右范围[-d*math.sin(mu),d*math.sin(mu)]内取 10
个点
    delta_O_j_i_x = np.linspace(O_j_i[0] - d * math.sin(mu), O_j_i[0] + d *
```

```
math.sin(mu), 10)
    # 得到这十个点的纵坐标
    delta_0_j_i_y = [cal_y(mu, 0_j_i, x) for x in delta_0_j_i_x]
    # 得到这十个点的深度值
    delta_0_j_i_depth = [get_depth(delta_0_j_i_x[i], delta_0_j_i_y[i]) for i in
range(len(delta_0_j_i_x))]
    # 对这十个点拟合出一条向量
    coefficients = np.polyfit(delta_0_j_i_x, delta_0_j_i_depth, 1)
    a = coefficients[0]
    # 得到拟合曲线的向量
    I_o_j_i = [1, -1/math.tan(mu), a]
    # 第 i 条线第 j 个点作垂线与坡面的交线和水平面的夹角
    alpha_j_i      =      math.asin(np.abs(np.dot((I_o_j_i),      I_z)      /
(np.linalg.norm(I_o_j_i) * np.linalg.norm(I_z))))

    return 0_j_i, D_j_i, I_o_j_i, alpha_j_i

# 计算所有区块的重叠率
def cal_overlap_rate(mu, d, theta):

    Eta_list = []

    def find_j_values(d , mu):
        a = 2.5 * 1852;b = 5 * 1852;c = 2 * 1852;e = 4 * 1852
        # 计算两个不等式的 j 的取值范围
        j_range1 = [(0 - a) / (d * math.sin(mu)), (b - a) / (d * math.sin(mu))]
        j_range2 = [(c - e) / (d * math.cos(mu)), (c - 0) / (d * math.cos(mu))]

        # 找到 j 的取值范围的交集
        j_min = max(min(j_range1), min(j_range2))
        j_max = min(max(j_range1), max(j_range2))

        # 找到所有在取值范围内的整数 j
        j_values = [j for j in range(int(j_min), int(j_max) + 1)]

        return j_values

    j_values = find_j_values(d, mu)

    # 求出对应第 j 条测线上的 i 的范围
    def find_i_values(d, mu, T, j):
        a = 2.5 * 1852;b = 5 * 1852;c = 2 * 1852;e = 4 * 1852

        # 计算两个不等式的 i 的取值范围
```

```
        i_range1 = [(0 - a - j * d * math.sin(mu)) / (T * math.cos(mu)), (b - a
- j * d * math.sin(mu)) / (T * math.cos(mu))]
        i_range2 = [(c - e + j * d * math.cos(mu)) / (T * math.sin(mu)), (c - 0
+ j * d * math.cos(mu)) / (T * math.sin(mu))]

        # 找到 i 的取值范围的交集
        i_min = max(min(i_range1), min(i_range2))
        i_max = min(max(i_range1), max(i_range2))

        # 找到所有在取值范围内的整数 i
        i_values = [i for i in range(int(i_min), int(i_max) + 1)]

        return i_values

    # 从 j 最小的开始，到倒数第二个最大的
    for j in j_values[0:-1]:
        i_values_0 = find_i_values(d, mu, T, j)
        i_values_1 = find_i_values(d, mu, T, j+1)

        # 找出较大的最小值和较小的最大值
        min_0 = np.min(i_values_0)
        max_0 = np.max(i_values_0)
        min_1 = np.min(i_values_1)
        max_1 = np.max(i_values_1)

        start_value = np.max([min_0, min_1])
        end_value = np.min([max_0, max_1])
        i_values = np.arange(start_value, end_value + 1).tolist()
        for i in i_values:
            O_j_i, D_j_i, I_o_j_i, alpha_j_i = cal_j_i(i, j, mu, d)
            O_j1_i, D_j1_i, I_o_j1_i, alpha_j1_i = cal_j_i(i, j+1, mu, d)

            W_j_i = D_j_i * (math.sin(theta/2)/math.sin(math.pi/2 - theta/2 -
alpha_j_i)) +\
                    D_j_i * (math.sin(theta/2)/math.sin(math.pi/2 + alpha_j_i))

            Eta_j_i = (D_j_i * (math.sin(theta/2)/math.sin(math.pi/2 - theta/2
- alpha_j_i)) + \
                       D_j1_i   *   (math.sin(theta/2)/math.sin(math.pi/2   +
alpha_j1_i)) - \
                       d/math.cos(alpha_j_i))/ W_j_i
            Eta_list.append(Eta_j_i)

    return Eta_list
```

```
# 目标函数
def objective_function1(params):
    mu = params[0]
    d = params[1]
    theta = params[2]
    Eta_list = cal_overlap_rate(mu, d, theta)
    # 漏测块数
    F = 0
    # 超长长度的条数
    L = 0
    for k in Eta_list:
        if k < 0:
            F += 1
        elif k > 0.2:
            L += 1
    # 计算漏测的总面积
    S_F = F * d * T
    return S_F

# 求解 L2 最小值
def objective_function2(params):
    mu = params[0]
    d = params[1]
    theta = params[2]
    Eta_list = cal_overlap_rate(mu, d, theta)
    # 超长长度的条数
    L = 0
    for k in Eta_list:
        if k > 0.2:
            L += 1
    # 计算超长的总长度
    L2 = L * T
    return L2

initial_guess = [math.radians(20), 0.02 * 1852, math.radians(120)]

# 求解 S_F 最小值
solution1 = minimize(objective_function1, initial_guess,
method='SLSQP',options={'maxiter': 100000})
Min_S_F = solution1.fun

# 求解 L_2 最小值
```

```
solution2 = minimize(objective_function2, initial_guess, method='SLSQP',options={'maxiter': 100000})
Min_L2 = solution2.fun


# 计算总长度的最小值（在 question_4_total_length.py 中已经计算出来）
Min_L1 = 1016267
# 求解帕累托最优解
def objective_function3(params1):
    S = params1[0]
    L1 = params1[1]
    L2 = params1[2]

    pareto = np.sqrt(((Min_S_F - S)/S)**2 + ((Min_L1 - L1)/L1)**2 + ((Min_L2 - L2)/L2)**2)
    return pareto

initial_guess = [Min_S_F/2, Min_L1/2, Min_L2/2]
# 求解 L_2 最小值
solution3 = minimize(objective_function3, initial_guess, method='SLSQP',options={'maxiter': 100000})

pareto = solution3.fun

print(pareto)
```

附录 6 求解 L1 最小值程序

```
import math
import numpy as np
from scipy.optimize import minimize
from pandas import read_excel

# 波速探测器开角
theta = math.radians(120)

# 读取 Excel 文件
data = read_excel('附件.xlsx')

# 提取横向坐标和纵向坐标
x_coords = data.iloc[0, 2:].values.astype(np.float64) * 1852
y_coords = data.iloc[1:, 1].values.astype(np.float64) * 1852
```

```
# 提取海水深度数据
depth_data = data.iloc[1:, 2:].values.astype(np.float64)

# 更正参数顺序并重新计算梯度和坡度
gradient_y, gradient_x = np.gradient(depth_data, y_coords, x_coords)

# 计算每点的坡度
slope = np.sqrt(gradient_x**2 + gradient_y**2)

# 计算平均坡度
average_slope = np.mean(slope)

# 计算坡度角
#alpha = math.radians(np.arctan(average_slope) * (180/np.pi))

alpha = math.radians(0.83)

# 平面长宽
Height = 4*1852
Length = 5*1852

# 决策变量
# mu = params[0]
# d = params[1]

def compute_alpha_angles(mu, alpha):
    # 坡面法向量
    E1 = [0, -math.sin(alpha), math.cos(alpha)]

    # 测线方向
    def cal_E2(mu):
        return [math.cos(mu), math.sin(mu), 0]

    # 过测线作平面交坡面的交线方向
    def cal_E3(E1, E2):
        return [E2[0], E2[1], -E1[1] * E2[1] / E1[2]]

        # 过测线作垂平面交坡面交线方向

    def cal_E4(E1, E2):
        return np.cross(E2, E1)

    # 水平面法向量
```

```
    E5 = [0, 0, 1]

    # 计算 alpha_1 和 alpha_2
    E2 = cal_E2(mu)
    E3 = cal_E3(E1, E2)
    E4 = cal_E4(E1, E2)

    def cal_alpha_1(E4, E5):
        sin_alpha_1 = np.dot(E4, E5) / (np.linalg.norm(E4) * np.linalg.norm(E5))
        return math.asin(sin_alpha_1)

    def cal_alpha_2(E3, E5):
        sin_alpha_2 = np.dot(E3, E5) / (np.linalg.norm(E3) * np.linalg.norm(E5))
        return math.asin(sin_alpha_2)

    alpha_1 = cal_alpha_1(E4, E5)
    alpha_2 = cal_alpha_2(E3, E5)

    return alpha_1, alpha_2

# 第一问的模型，给定 alpha_1 和测线间隔 d，可以求出 OC 上的一系列测线的重叠率
def calculate_overlap_rate(alpha_1=None, alpha_2=None,theta_degree=120, mu=None,
d=None):

    X3 = np.abs(Length/2 + (Height/2)*math.tan(mu))
    X2 = np.abs(X3*math.sin(mu))
    X1 = np.abs(X3*math.cos(mu))
    OA = np.abs((Height/2)/math.cos(mu))
    X4 = OA - X2
    X5 = Length/2 - (Height/2)*math.tan(mu)
    X6 = X5*math.cos(mu)

    # 共有 n 条线
    n = np.abs(int(X1 // d))
    # n 条测线距离中心点处的距离
    distance_list = [i * d for i in range(0, n+1)]

    # X6 内总共有 i 条线
    I = np.abs(int(X6 // d))

    # 坡度角
    alpha = alpha_1
    # 多波束换能器的开角
    theta = math.radians(theta_degree)
```

```
    # 波束与坡面近坡底夹角
    kappa = math.pi / 2 - theta / 2 - alpha
    # 波束与坡面远坡底夹角
    gamma = math.pi / 2 - theta / 2 + alpha

    # 计算波束覆盖宽度
    def length_W(D):
        x1 = D * (math.sin(theta / 2) / math.sin(kappa))
        x2 = D * (math.sin(theta / 2) / math.sin(gamma))
        return x1 + x2, x1, x2

    # 起始点深度
    D0 = 47.2 - X4*math.tan(alpha_2)
    # 计算深度的函数
    def depth(distance):
        return D0 - math.tan(alpha) * distance

    # 计算不同波束之间的重叠宽度
    def overlap_width(x2, x3, d):
        return x2 + x3 - d / math.cos(alpha)

    # 重叠率计算公式
    def overlap_rate(overlap, w):
        return overlap / w

    # 海水深度
    D = [depth(dis) for dis in distance_list]
    # 波束覆盖宽度
    W = [length_W(depth)[0] for depth in D]
    # n 个波束之间的 n-1 条重叠宽度
    overlap = []
    for i, depth in enumerate(D):
        if i < len(D) - 1:
            x2 = length_W(depth)[2]
            x3 = length_W(D[i + 1])[1]
            overlapped_width = overlap_width(x2, x3, np.abs(distance_list[i + 1]
- distance_list[i]))
            overlap.append(overlapped_width)

    # 计算重叠
    overlap_rate_list = [overlap_rate(overlap[i], W[i]) for i in
range(len(overlap))]

    return overlap_rate_list, W, D, distance_list, X1, X2, X3, X4, X5, X6, OA,
```

```
I , n

# 要求：X1 上最后一条覆盖率不能超过 20%
# 也即 overlap_rate_list_X1[-1] <= 0.2 #约束条件 1

# 要求：X1'上最后一条覆盖率不能低于 0%
# 也即 overlap_rate_list_X1_[-1] >= 0 #约束条件 2

# 约束条件 3:n 条测线的宽度总和加上 OC'上最后一条测线的左半波束覆盖宽度必须大于 X1
# Distance_list[-1] + W[-1]/2 >= X1 #约束条件 3

# 约束条件 6:mu 的范围为[0,pi/2]

# 目标方程
def objective_function(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, X2, X3, X4, X5, X6, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1,    alpha_2=alpha_2,    theta_degree=120,
mu=mu, d=d)
    return  OA * (2*I + 2*n + 4)

# 约束条件 1
def constraint1(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, X2, X3, X4, X5, X6, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1,    alpha_2=alpha_2,    theta_degree=120,
mu=mu, d=d)
    return -overlap_rate_list[-1] + 0.2

# 约束条件 2
def constraint2(params):
    mu = params[0]
    d = -params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, X2, X3, X4, X5, X6, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1,    alpha_2=alpha_2,    theta_degree=120,
mu=mu, d=d)
    return overlap_rate_list[-1]
```

```
# 约束条件 3
def constraint3(params):
    mu = params[0]
    d = params[1]
    alpha_1, alpha_2 = compute_alpha_angles(mu=mu, alpha=alpha)
    overlap_rate_list, W, D, distance_list, X1, X2, X3, X4, X5, X6, OA, I ,n=
calculate_overlap_rate(alpha_1=alpha_1,   alpha_2=alpha_2,   theta_degree=120,
mu=mu, d=d)
    return distance_list[-1] + W[-1]/2 - X1

# 边界条件
bounds = [(0, math.pi/2), (0, 1.5*1852)]

# 初始条件
initial_guess = [math.radians(79.5), 315]

# 约束条件
constraint1 = {'type': 'ineq', 'fun': constraint1}
constraint2 = {'type': 'ineq', 'fun': constraint2}
constraint3 = {'type': 'ineq', 'fun': constraint3}
constraints = [constraint1, constraint2, constraint3]

# 求解
solution   =   minimize(objective_function,   initial_guess,   method='SLSQP',
bounds=bounds, constraints=constraints, options={'maxiter': 100000})
print(solution)
print(math.degrees(solution.x[0]), solution.x[1])
```