中国工业与应用数学学会
China Society for Industrial and Applied Mathematics

CUMCM 中国大学生数学建模竞赛

北京师范大学

学　　生：董雨铮
　　李昊霖
　　高延子鹏

指导老师：

荣获二零二三年高教社杯全国大学生数学建模竞赛本科组二等奖。

中国工业与应用数学学会
全国大学生数学建模竞赛组织委员会
二零二三年十一月

基于贪心策略和自适应遗传算法的多波束测线排布模型

摘要

多波束测线布设问题的核心在于如何布设测线。为了合理、精确地设计测线，并满足测线排布方案对总长度、重叠率和漏测区域的要求，本文通过分析测量时海底、测线和测量条带的几何特征，并使用多种优化策略，最终给出满足题目要求的最优测线布设方案。

问题 1 是一个平面问题，目标是建立多波束测深覆盖宽度和相邻条带重叠率的数学模型。为此，我们研究了测线、条带和海底坡面之间的几何关系，通过几何关系得到各点水深与离中心点距离成比例，进而确定计算不同位置的水深信息的表达式；再通过**解三角形**得到度量多波束测深覆盖宽度的公式；为了描述海域全覆盖测量，我们参照国家标准，合理定义了**相邻条带的重叠率**,并据此计算每一条测线与前一条测线的重叠率;最后建立起计算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,并计算出题目所列的指标值。

问题 2 是对问题 1 的推广，要求改为确定海底坡面上任意方向测线的覆盖宽度，只需要确定水深和测量条带与水平面夹角即可转化为问题 1。水深易通过解三角形求得;我们通过建立空间直角坐标系，将求测量条带与水平面夹角的问题向量化进行解决。进而转化为问题 1 已求出的模型计算出**测线上各点的覆盖宽度**。

对于问题 3，目标是为一个矩形海底坡面设计最优的测线排布方案。我们使用了两种不同的方法，首先，我们从**理论上推导出了测线平行于海底坡面时为最优设计**，然后可以在此基础上完成测线的设计。第二种方法是采用**贪心算法**在排布方向角的可行域进行遍历，将重叠率尽可能的减小，找到在满足全覆盖测量和重叠率在 10%~20% 的条件下，使得总测线长度最短的测线布设方案，最终最短测线总长度为 **68 海里**。

问题 4 聚焦于实际问题，将海底变为由离散先验深度数据确定的形状不规则的海床，并要求针对这一海底形貌设计满足一系列要求的最优测线方法。难点在于矩形海域并不是规则斜面，无法如前 3 问利用海底曲面方程进行计算。我们将矩形海域划分为网格进行离散计算，并利用二分查找优化计算速度。针对本问题，我们基于**遗传算法**开发了**自适应遗传算法**，将测线的角度、若干条测线坐标纵截距作为决策变量，迭代可求得优化结果。对比设置不同覆盖面积最小值的约束条件下得到的多组优化结果，我们得到了最终的测线方案，并得出 3 种要求指标，其中最短测线长度为 **402.968 海里**。

关键字：多波束测线方案设计　贪心策略　遗传算法　二分查找算法

1 问题背景与问题重述

1.1 问题背景

水深测量是海底地形测量工作的重要环节。水深测量的方法和手段多种多样[1]，其中单波束测深是一种利用声波在水中的传播特性来测量水体深度的技术[2]，能够完成沿航迹线上的断面测量。随着人类对海洋资源的开发和利用，从单波束测深技术的基础上发展而来的多波束测深系统被广泛地应用，多波束测量系统可以一次获取在航向垂直平面内且以测量船测线为轴线的一系列测量条带，从而快速、可靠地测算海底深度并绘制其它海底地貌特征。为了保证测量的准确性，相邻条带通常有 10%-20% 的重叠率，且为了数据的全面性，测线要尽可能地能覆盖整个海域，同时要兼顾测线总长度尽可能短. 因此，如何设计和规划测线是提高多波束测量技术测量效率和提升准确性的关键问题。

1.2 问题重述

问题让我们计算测深条带覆盖宽度和相邻条带重叠率，并为多波束测量船设计测线，兼顾总测线长度尽可能短、重叠率和漏测区域的要求，各问题具体要求如下：

问题 1：假定海底为坡面，在测线与海底坡面始终平行情况下，要求我们建立计算离海域中心点不同位置处的测深条带覆盖宽度和相邻条带之间的重叠率的数学模型，并计算题目表格所列的指标值。

问题 2：仍然假定海底为坡面，在已知测线方向与海底坡面的法向在水平面上投影的夹角的前提下，建立多波束覆盖宽度的数学模型，并计算题目表格所列的指标值。

问题 3：考虑一个海底为坡面的矩形海域，已知中心点海水深度、坡度和换能器开角，设计一组测量长度最短的测线，同时要求可以完全测量海域, 且相邻测深条带之间的重叠率在 10%∼20% 之间。

问题 4：现有若干年前单波束测量的一个矩形海域的深度信息，结合已有的测量结果为多波束测量船规划测线，要求：(1) 沿测线扫描的条带尽可能的完全覆盖海域，(2) 相邻条带的重叠率尽量低于 20%，(3) 测线总长度尽可能短。并在规划出具体测线后，计算总长度、重叠率超过 20% 的长度和漏测区域比例。

2 问题分析

该问题最终目标为按要求优化一组任意形状不平整海面上的多波束测线，题目层层深入，首先考虑一个倾斜海底坡面，在问题 1 中，测线平行于海底坡面，可简单通过几何方法求解；而在问题 2 中，测线方向不再与海底斜面垂直，而是可以更一般的变化。

前两个问题构成了对海底坡面上确定测线的多波束测量的数学模型，而问题 3 提出了对于海底坡面上最优测线的设计问题。最终，问题 4 聚焦于实际问题，将海底变为由离散先验深度数据确定的形状不规则的海床，并要求针对这一海底形貌设计满足一系列要求的最优测线。

2.1 问题 1 的分析

问题 1 要求我们考虑一个已知坡角的海底斜面，在已知测线与海底坡面平行（即测线与海底坡面法向在水平面上的投影垂直）的前提下，建立计算条带覆盖宽度和相邻条带重叠率的模型。作出剖面图，就把该问题转化成一个平面几何问题。我们**先确定各点的水深信息**，在已知中心点水深后，其余位置的水深根据与中心点的距离成比例的变化，故确定任意位置的水深信息；**再计算各测深条带的覆盖宽度**，换能器发出的最边缘两束波束会与海底坡面形成一个三角形（其顶角为换能器开角），在此三角形中应用正弦定理可以解出最边缘两束波束与海底接触点的距离，进而把这段距离投影到水平面上得到条带覆盖宽度；**然后计算出相邻条带重叠部分的宽度，从而算出相邻条带之间的重叠率**。最后针对问题 1 所列位置的指标值，代入相应数值即可精准计算出结果。

2.2 问题 2 的分析

问题 2 仍然要求我们考虑一个已知坡角的海底斜面，但此时测线与海底坡面法向在水平面上的投影夹角不在是 90°，这导致不同测深条带与海平面的夹角不再等于坡角，因此不能简单地使用问题 1 的模型. 为此，我们以海域中心点在水平面的投影点为原点建立空间直角坐标系。之后的建模思路类似问题 1，**先计算各点水深信息**，为此我们将测线投影到 x 轴上，利用问题 1 模型计算各方向上的各点水深，只需注意投影后离中心点的距离发生了改变，而由于此方向的投影不改变水深，故根据投影后的等效离海域中心点距离可以算出原位置的水深；然后通过分析线面之间的投影关系以及方向向量之间的夹角，**再计算出每条测线方向上的等效坡角**，进而修正问题 1 的模型，**得到问题 2 中计算测深条带覆盖宽度的模型**。最后代入问题 2 所列位置的指标值，计算出结果。

2.3 问题 3 的分析

根据问题 1、2，我们已经建立了计算多波束测量覆盖宽度和相邻条带重叠率的数学模型，在本问题中，我们要设计一组总长度尽可能小的测线，并满足两个条件：（1）无漏测区域，（2）相邻条带重叠率满足 10%—20%。问题优化变量为排布方向角和一系列测线的间距。

在同一测线布线模式下，相邻条带重叠率越小，可测量的有效面积越大因此我们使用贪心算法，从矩形区域的浅水区开始，依次排布测线。对于每条新排布的测线，令其

浅水区重叠率为最低值,然后计算深水区测线重叠率，若小于上限值，则说明该测线合理，继续排布下一条；反正，说明该排布方向不合理，应该更换排布方向角。最后我们对排布方向角进行参数化扫描遍历，从中找出符合约束条件且总长度最小的测线方案，即得到最优布线策略。

2.4 问题 4 的分析

问题 4 需根据已知的离散网格深度数据，设计在 3 个指标下最优测线方案。此问题可拆解成两部分，一方面是已知一种具体测线方案，求此方案对应的 3 个指标值；另一方面是根据得出的指标值，对测线方案进行优化。

对于计算某测线方案的指标值，实际上是利用已知的离散深度数据对测线方案进行评价，这里难点在于矩形海域并不是规则斜面，无法如前 3 问利用海底曲面方程进行计算。我们将矩形海域划分为**网格**进行离散计算，并利用二**分查找**优化计算速度。

- 对于指标 1（测线总长度），可直接由平面**几何方式**求得；
- 对于指标 2（测量覆盖面积占比），将矩形海域按一定步长划分为均匀网格，对于每个网格，计算网格中心点是否可被相邻两条测线覆盖（使用二分查找相邻测线），若可覆盖，则将此网格面积计入被覆盖面积。
- 对于指标 3（重叠率超过 20% 的总长度），将每条测线按一定步长划分为若干段，对于每个测线子段，计算段中心点与前一条测线对应位置构成的截面的覆盖范围（使用二分搜索覆盖边界位置），得到此截面的重叠比例，若比例超过 20% 阈值，则将此段测线长度计入重叠超过 20% 总长度。

对于测线方案的优化。考虑 3 个指标间关系，实际上指标 1、2 间的制约关系已经能够反映指标 3（重叠长度）的要求，因此重点考虑对指标 1、2（测线总长、测线覆盖面积占比）进行优化。我们将指标 2（覆盖面积占比）最小值作为约束条件，遍历不同的指标 2 最小值，以指标 1（测线总长度）作为优化目标，求得多组优化结果。使用**遗传算法**，将测线的角度、每条测线坐标纵截距作为决策变量，迭代可求得优化结果。对比设置不同覆盖面积最小值的约束条件下得到的多组优化结果，我们得到了最终的测线方案，并求出 3 种要求指标。

3 模型假设和解释

为了便于建立模型和精准计算结果，本文结合实际情况，并参考国家相关标准和已有文献研究，作出了一些合理的假设，具体假设如下：

假设一：假设所有测线是相互平行的直线 [3]。

测线是直线可以维持测量的稳定性。同时，测线相互平行可以最大限度地增大海底覆盖率，同时保持有效扫测宽度在一定的区间内 [3]，从而提高测量的效率。

假设二：当海底不平坦时，覆盖宽度 W 定义为波束信号与海底的交线在水平面上的投影宽度 [4]。

由于真实海底凹凸不平，按此定义考虑投影的宽度，可以把宽度均统一在海平面内，便于后续合理定义并计算相邻条带的重叠率。

假设三：测量船在从一条测线移动到另一条测线时移动的距离不计入测线总长度。

在实际测量中，为保持测线的稳定性，测量船需要提前在测线方向上行驶一定距离后再开始测量 [5]，因此忽略测量船调整方向并移动的这部分行驶距离。

4 符号说明

表 1　符号说明

符号	符号含义	单位
d	相邻两条测线的间距	m
W	多波束测深条带的覆盖宽度	m
θ	多波束测量船的换能器开角	°
η	相邻条带的重叠率	%
α	坡度	°
β	测线方向夹角	°
D_0	区域中心点海水深度	m
L	某位置距离海域中心点的距离	m
$Loss$	漏测面积与总面积之比，即漏测率	%
L_{total}	测线总长度	m

5 问题 1 模型的建立和求解

5.1 问题 1 模型建立

在问题 1 当中，海底地形并不平坦，不同位置处的海水深度不同，因此多波束测量船在不同位置发射声波信号时，其条带覆盖宽度和相邻条带重叠率也不完全相同，故需要我们重新考虑如何测算条带覆盖宽度以及重叠率。对于问题一所述情形，作出如下示

意图（如图1）：与测线方向垂直的平面和海底坡面的交线与水平面夹角为 α（即坡度

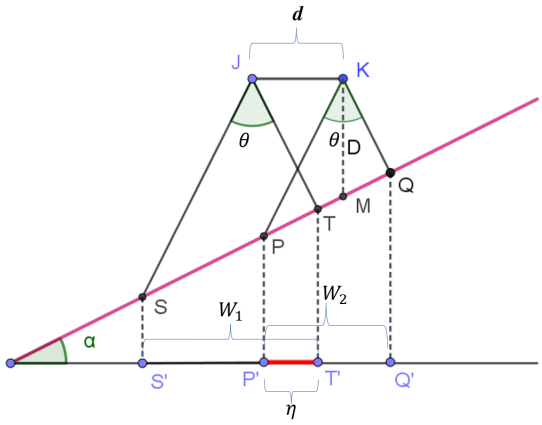


图 1　问题 1 示意图

为 α）；多波束测量船先后在海平面上位置 J、K 发射声波信号，J 和 K 的间距记为 d，第一次信号与海底斜坡的两个交点为 S 和 T；第二次信号与海底斜坡的两个交点为 P 和 Q；两次测量时换能器的开角均为 θ，

由假设 1，ST 和 PQ 在水平面上的投影 $S'T'$、$P'Q'$ 是条带的覆盖宽度。为了计算 $S'T'$ 和 $P'Q'$, 我们需要先算出 ST 和 PQ 的长度，再根据坡度算出覆盖宽度, 进而算出相邻条带的重叠率。下面以 K 处发射信号为例，推导计算条带覆盖宽度 W 的公式.

Step1. 计算条带覆盖宽度

设 K 点水深为 D，考虑 $\triangle KPQ$，把它分为 $\triangle KPM$ 和 $\triangle KQM$, 在 $\triangle KPM$ 中，$\angle KPM = \frac{\pi}{2} - \frac{\theta}{2} - \alpha$, 由正弦定理得:

$$\frac{|PM|}{\sin(\theta/2)} = \frac{D}{\sin\angle KPM}$$

由此得，

$$|PM| = \frac{D\cdot\sin(\theta/2)}{\cos(\theta/2+\alpha)} =: l_1 \tag{1}$$

同理可得，

$$|QM| = \frac{D\cdot\sin(\theta/2)}{\cos(\theta/2-\alpha)} =: l_2 \tag{2}$$

因此，

$$\begin{aligned} W &= |PQ|\cdot\cos\alpha \\ &= (l_1+l_2)\cdot\cos\alpha \\ &= D\cdot\sin\frac{\theta}{2}\cdot[\frac{1}{\cos(\theta/2-\alpha)}+\frac{1}{\cos(\theta/2+\alpha)}]\cdot\cos\alpha \end{aligned} \tag{3}$$

Step2. 计算水深

若已知 J 点处的水深，则可以类似地算出 J 点条带覆盖宽度。现设 K 点处水深为 D，J、K 点距离为 L，且 J 处在比 K 点水更深的地方，画出剖面图 (如图2)。

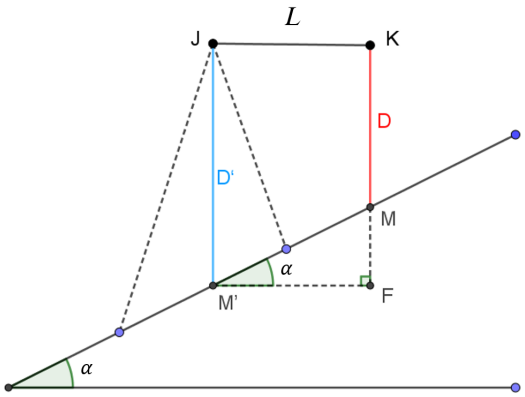


图 2　计算海水深度

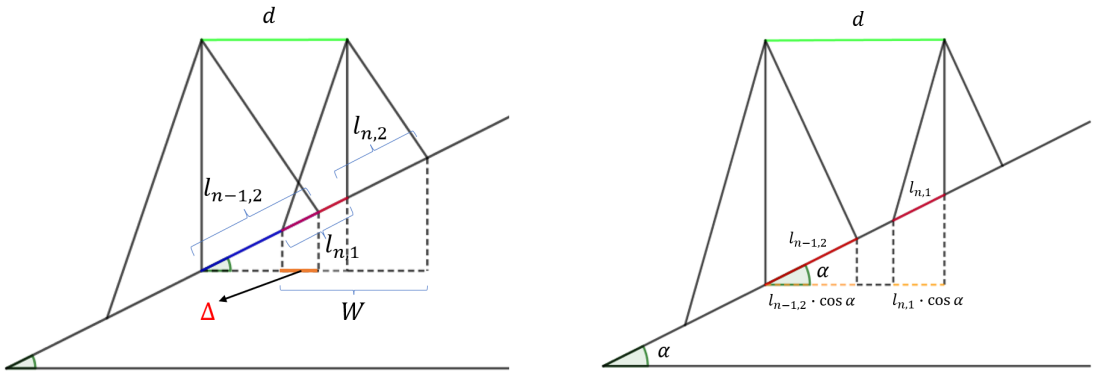
则 J 点处水深 D' 计算公式为:

$$D' = D + |MF| = D + L \cdot \tan\alpha \tag{4}$$

现在，如果我们已知海域中心点的水深以及某点离海域中心的距离，就可以算出该点的水深和条带覆盖宽度。接下来，我们需要建立计算相邻条带的重叠率的模型。

Step3. 计算相邻条带的重叠率

现设有 n 条互相平行的测线，它们会形成 n 条平行的测深条带（编号为 1,2,3,……,n），它们各自的覆盖宽度计算公式由 (3) 给出。基于假设二对覆盖宽度的定义，我们定义**海底不平坦时的相邻条带重叠率** η 为: 当前条带 (编号为 n) 与前一条条带 (编号为 n-1) 重叠部分在水平面上投影宽度 Δ 占当前条带覆盖宽度 W_n 的百分比（如图3a所示）。



(a) 重叠情形　　**(b) 漏测情形**

图 3　坡面上相邻条带的重叠率示意图

其具体计算公式为:

$$\eta_n = \frac{\Delta}{W_n} = \frac{(l_{n-1,2} + l_{n,1}) \cdot \cos\alpha - d}{W_n} \tag{5}$$

其中，$l_{n,1}$ 表示编号为 n 的条带的深水侧测量范围，$l_{n-1,2}$ 表示编号为 n-1 的条带的浅水

侧测量范围，则 W_n 表示编号为 n 的条带的覆盖宽度，而 $\Delta = (l_{n-1,2} + l_{n,1}) \cdot \cos\alpha - d$ 表示重叠部分在水平面的投影宽度，即重叠宽度。

同时由图3b所示，此时 $\Delta = (l_{n-1,2} + l_{n,1}) \cdot \cos\alpha - d < 0$, 所以 $\eta < 0$，因此这种定义重叠率的方法也可以反映出漏测的情形。

5.2 问题 1 的计算模型

综上所述，我们建立了一个计算条带宽度和相邻条带测量宽度的模型，若已知斜坡坡度 α、换能器开角 θ、海域中心点处水深 D_0、相邻测线的间距 d 和测线距中心点的间距 L, 就可以算出多波束系统的覆盖宽度和相邻条带的重叠率。结合公式 (3)(4)(5), 具体计算公式汇总如下：

$$\begin{cases} D_n = D_0 - L \cdot \tan\theta \\ W_n = D_n \cdot \sin\dfrac{\theta}{2} \cdot [\dfrac{1}{\cos(\theta/2+\alpha)} + \dfrac{1}{\cos(\theta/2-\alpha)}] \cdot \cos\alpha \\ \eta_n = \dfrac{(l_{n-1,2} + l_{n,1}) \cdot \cos\alpha - d}{W_n} \end{cases} \tag{6}$$

其中，W_n 表示编号为 n 的测深条带的覆盖宽度。

5.3 问题 1 计算结果

利用上述计算模型，代入换能器开角 $\theta = 120°$, 坡度 $\alpha = 1.5°$, 海域中心点水深 $D_0 = 70$ 米，计算题目中表格所列位置指标所得结果如下 (表2)。

表 2　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	54.2884	49.0513
覆盖宽度/m	315.7051	297.5256	279.3460	261.1665	242.9870	224.8074	206.6279	188.4484	170.2688
与前一条测线的重叠率/ %	——	35.70	31.51	26.74	21.26	14.89	7.41	-1.53	-12.36

注：这里我们认为离中心点距离为负值的位置是水更深的位置。

分析结果知，水更深的位置测深条带的覆盖宽度更大，与前一条测线的重叠率也更高；在水浅的位置甚至出现了一定程度的漏测情况，这符合海底是一个坡面的设定。

6 问题 2 模型的建立和求解

6.1 问题 2 模型建立

在问题 2 中，测线方向不在始终平行于海底坡面，因此问题 1 中的模型需要进行一定的修正。为此，我们以海域中心点在水平面上的投影点为原点 O，以坡面法向 $\overrightarrow{n_1}$ 在水平面上的投影的方向为 x 轴正方向，过点 O 且垂直水平面向上为 z 轴正方向，y 轴正向保证右手系建立空间直角坐标系（如图4）。记海域中心点水深为 D_0 海底斜面的坡度为 α, 测线与 x 轴正向的夹角为 β; 海域中心点在海底坡面上的投影为 E, 测线在海底坡面上的投影为 EF，测线在水平面的投影为 OT。

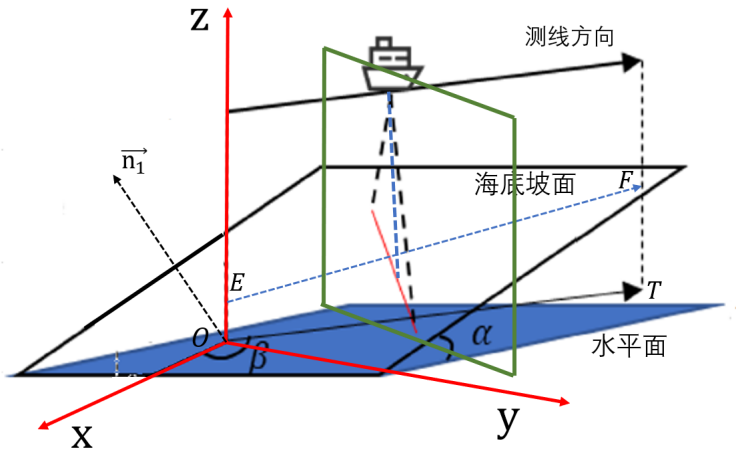
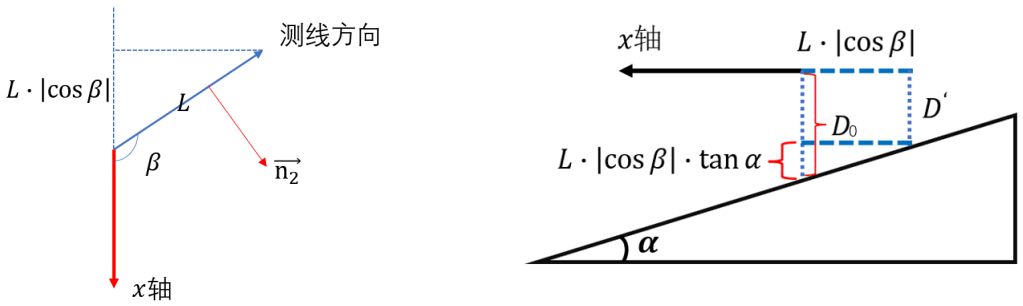


图 4　问题 2 示意图

接下来我们将推导测线、条带和坡面之间的几何关系，进而得到问题 2 的计算模型。

Step1. 计算水深 D

将 OT 向 x 轴平行投影, 并画出俯视图 (如图5a，图中 $\overrightarrow{n_2}$ 是与 OT 的投影正交的单位向量)；同时, 作出过 x 轴且与海底坡面垂直的平面剖面图 (如图5b).



(a) 俯视图　　**(b) 沿 y 轴视图**

图 5　计算水深的示意图

由图5a所示，离海域中心点的等效距离 L' 为

$$L' = L \cdot |\cos\beta| \tag{7}$$

由于沿 x 轴方向投影不改变水深，故现在可以利用问题 1 的模型来计算水深，结合图5b，在测线方向夹角为 β, 离海域中心点距离为 L 位置处的水深计算公式为:

$$D = D_0 + L \cdot \cos\beta \cdot \tan\alpha \tag{8}$$

其中，β 的取值范围为 [0°，360°), 此时可以去掉 $|\cos\beta|$ 的绝对值来计算真实水深。

Step2. 计算等效坡角 α'

作出图4中绿色框标记的截面 (图6), 即垂直于测线和水平面且过测量船位置的平面，记该截面与海底坡面的交线 (即图4中的细红线) 与水平面所成夹角为 α'；坡面的单位法向向量记为 $\overrightarrow{n_1}$, 交线在水平面上投影的单位方向向量记为 $\overrightarrow{n_2}$。写出坡面的单位法向向量

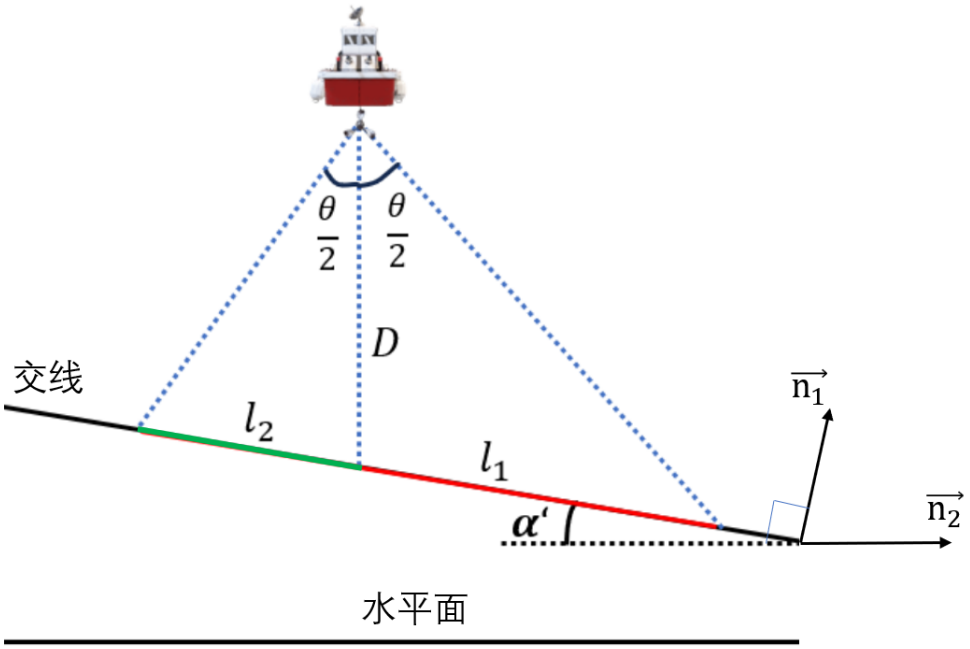


图 6

$\overrightarrow{n_1}$=(sin α,0,cos α)，交线在水平面上投影的单位方向向量 $\overrightarrow{n_2}$=(cos($\beta - \frac{\pi}{2}$), sin($\beta - \frac{\pi}{2}$), 0), 由线面角的定义，得到如下计算等效坡角 α' 的公式:

$$\sin\alpha' = \cos(\frac{\pi}{2} - \alpha') = \frac{\langle\overrightarrow{n_1}, \overrightarrow{n_2}\rangle}{|\overrightarrow{n_1}||\overrightarrow{n_2}|} = \sin\alpha \cdot \sin\beta \tag{9}$$

Step3. 计算覆盖宽度 W

算出沿 β 方向、距海域中心点距离为 L 的位置处的水深 D 和测深条带等效坡角 α' 后，我们便可以利用问题 1 的模型计算覆盖宽度，具体计算公式如下:

$$W = D \cdot \sin\frac{\theta}{2} \cdot [\frac{1}{\cos(\theta/2 + \alpha')} + \frac{1}{\cos(\theta/2 - \alpha')}] \cdot \cos\alpha' \tag{10}$$

6.2 问题 2 的计算模型

至此，我们已经建立了计算沿任意测线方向的测深覆盖宽度模型。若给定斜坡的坡度 α、换能器开角 θ、海域中心点处水深 D_0，并且已知测线方向夹角 β 和测量船距海域中心点的距离 L, 就可以算出多波束测深的覆盖宽度。综合公式 (7)(8)(9)(10), 得到具体计算公式如下：

$$\begin{cases} D = D_0 + L \cdot \cos\beta \cdot \tan\alpha \\ \alpha' = \arcsin(\sin\alpha \cdot \sin\beta) \\ W = D \cdot \sin\dfrac{\theta}{2} \cdot [\dfrac{1}{\cos(\theta/2+\alpha')} + \dfrac{1}{\cos(\theta/2-\alpha')}] \cdot \cos\alpha' \end{cases} \tag{11}$$

6.3 问题 2 计算结果

利用上述模型，代入换能器开角 $\theta = 120°$, 坡度 $\alpha = 1.5°$, 海域中心点水深 $D = 120$ 米, 并依次代入题目所列的测线方向夹角 β 和距海域中心点距离 L, 计算结果如下 (表3)。

表 3　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	416.1200	451.7941	487.4682	523.1422	558.8163	594.4903	630.1644	665.8384
	90	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491
	135	416.1200	380.4460	344.7719	309.0979	273.4238	237.7498	202.0757	166.4017
	180	415.6922	365.2933	314.8945	264.4956	214.0967	163.6976	113.2990	62.9002
	225	416.1200	380.4460	344.7719	309.0979	273.4238	237.7498	202.0757	166.4017
	270	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491	416.5491
	315	416.1200	451.7941	487.4682	523.1422	558.8163	594.4903	630.1644	665.8384

分析结果知，沿 0° 方向，离海域中心点越远，条带覆盖宽度越宽；沿 180° 方向，离海域中心点越远，条带覆盖宽度越窄，这与我们假设的斜坡延伸方向相一致。同时，以坡面法向在水平面投影上的投影所在的直线为轴，发现关于该轴对称的测线在距海域中心相同距离处所测条带的覆盖宽度相同，这也验证了我们的结果可以真实反映海底情况。

7 问题 3：海底坡面测线设计问题

7.1 海底坡面上重叠率的计算

问题 2 中已经计算了海底坡面上各位置的多波束覆盖宽度，可以进一步计算出相邻条带的重叠率。将图6中的单条带扩展为相邻条带，绘制于图7。

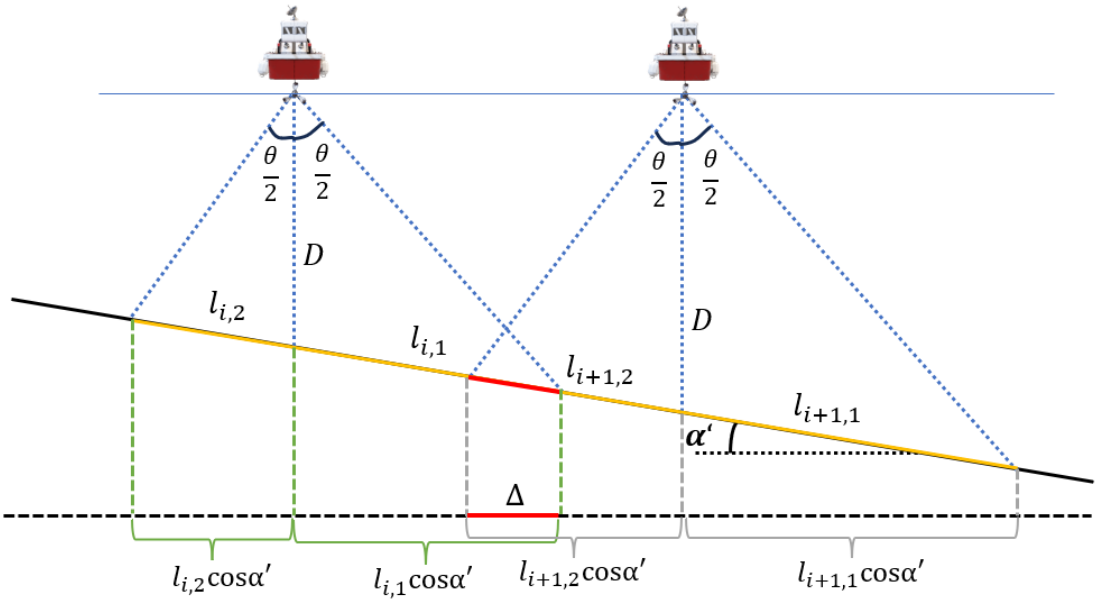


图 7　相邻条带重叠计算图

与上文一致，**相邻条带重叠率的定义为重叠宽度与覆盖宽度的比值**。图7将坡面上的所有区域投影到了水平面上，$l_{i,1}, l_{i,2}$ 分别为船在深水侧和浅水侧的单侧测量范围，所以 $l_{i,1}+l_{i,2}$ 为条带测量范围，等效倾角 α' 可以使用问题 2 建立的模型求出，所以可以计算出条带覆盖宽度 $W=(l_{i,1}+l_{i,2})\cdot\cos\alpha'$ 和重叠宽度 Δ.

由于平行投影不改变线段比例的性质，因此重叠率为：

$$\eta_{i+1}=\frac{\Delta}{W}=\frac{\Delta}{(l_{i+1,1}+l_{i+1,2})\cdot\cos\alpha'} \tag{12}$$

由图7可见，重叠部分

$$\Delta=(l_{i,1}+l_{i+1,2})\cdot\cos\alpha'-d_{i+1} \tag{13}$$

其中，d_{i+1} 为第 i 条测线与第 i+1 条的间隔。所以：

$$\eta_{i+1}=\frac{(l_{i,1}+l_{i+1,2})\cdot\cos\alpha'-d_{i+1}}{(l_{i+1,1}+l_{i+1,2})\cdot\cos\alpha'} \tag{14}$$

$$\sin\alpha'=\sin\alpha\cdot\sin\beta \tag{15}$$

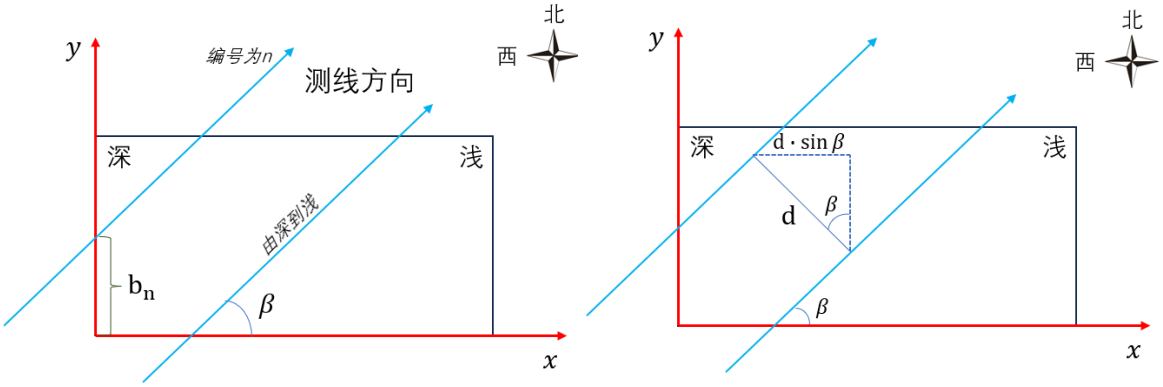
方程 (14)、(15) 即为海底坡面上相邻条带重叠率的计算模型。

7.2 海底坡面测线设计的理论分析

7.2.1 数学准备

我们将矩形待测海域的西南角设为坐标原点，向东为 x 轴正向，向北为 y 轴正向，如图8a。我们将单位统一为米，区域中心坐标为 (3704m，1852m)，中心水深 $D_0 = 110\,m$，沿 y 轴方向为等深度线，深度与 x 坐标的关系可以由简单的几何关系求得：

$$D(x) = D_0 - (x - 3704) \cdot \tan\alpha \tag{16}$$



(a) 海域坐标系示意图　　(b) 相邻测线几何关系图

图 8　海面坐标系及测线几何关系示意图

性质 1： 间距给定的情况下，海底坡面上的相邻测线，浅水区的重叠率低于深水区。

证明： 实际上这一关系可通过图7获知，但为了严谨，我们在此给出证明。由 (14) 知，相邻测线的重叠率计算公式为：

$$\eta_{i+1} = \frac{(l_{i,1} + l_{i+1,2})\cos\alpha' - d_{i+1}}{(l_{i,1} + l_{i,2})\cos\alpha'}$$

其中，

$$l_{i,1} = \frac{D_i \cdot \sin(\theta/2)}{\cos(\theta/2 + \alpha')} \tag{17}$$

$$l_{i,2} = \frac{D_i \cdot \sin(\theta/2)}{\cos(\theta/2 - \alpha')} \tag{18}$$

对于两条给定间隔为 d 的测线，在计算重叠率时需要选取两条测线上各一点，从而确定一条垂直于测线方向的直线。设在测线 i 上确定点的 x 坐标为 x_i，则在测线 i+1 上点的 x 坐标为 $x_{i+1} = x_i - d \cdot \sin\beta$。$x_i - x_{i+1}$ 为一定值，所以当 x_i 增加时，D_i 与 D_{i+1} 减小，水深变浅，符合题目西深东浅的设计；$l_{i,1}, l_{i,2}, l_{i+1,2}$ 随之减小，η 增大。 □

性质 2： β 的有效设计范围为 [0°,90°]

首先，β 的自由变化范围为 [0°,360°)，但由于 $tan\beta$ 表示的是直线的斜率，所以 [0°,180°] 与 [180°,360°] 完全重复；因此将 β 的范围限制在 [0°,180°]，又由于海底坡面在南北方向具有平移不变性，互补的测线方向角（β 和 180° -β）关于东西方向对称，在设计上完全相同，具有相同的最优目标，即目标函数关于 $\beta = 90°$ 对称，所以可以将 β 的变化范围限制在 [0°,90°]。

7.2.2 最优测线排布方式的理论分析

对于问题 3，我们从理论上计算出最短测线总长度对应的 β 为 90°，分析过程展示如下：

首先，**影响测线总长度的因素**主要有两个，第一个是**测量效率**，定义为测量船沿测线走过单位距离测量波束扫描过的覆盖面积；第二个是**相邻条带的重叠率**。显然，当测量效率尽可能高，重叠率尽可能低时，测线总长度最小。

其次，对于**重叠率**，题目给出了 10%∼20% 的限制，当且仅当 $\beta = 90°$ 时，此时的测线族实际上是平行于海底坡面的一族平行线，所以重叠率可以很容易的控制在任意恒定值，但对于 $\beta \neq 90°$，一条测线上的重叠率显然处处不相同，无法同时控制在某一值。

再次，对于**测量效率**，可以证明，过海上同一点，$\beta = 90°$ 的测线有最高的测量效率，证明如下：

定理：过海上同一点，$\beta = 90°$ 的测线有最高的测量效率。

证明：测量船沿测线走过距离 s 时，条带扫过的面积（覆盖面积）记为 SA，已知第 i 条测线的方程为

$$y = \tan\beta \cdot x + b_i \tag{19}$$

所以，

$$\mathrm{d}s = \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2} = \sqrt{1 + \tan^2\beta}\,\mathrm{d}x \tag{20}$$

因此，

$$\mathrm{SA} \overset{def}{=} \int W\mathrm{d}s = \int W\sqrt{1 + \tan^2\beta}\,\mathrm{d}x \tag{21}$$

其中，W 为覆盖宽度，由方程 (11) 可得：

$$W = D \cdot \sin\frac{\theta}{2} \cdot [\frac{1}{\cos(\theta/2 + \alpha')} + \frac{1}{\cos(\theta/2 - \alpha')}] \cdot \cos\alpha' \tag{22}$$

令

$$A(\alpha, \beta, \theta) \overset{def}{=} \sin\frac{\theta}{2} \cdot [\frac{1}{\cos(\theta/2 + \alpha')} + \frac{1}{\cos(\theta/2 - \alpha')}] \cdot \cos\alpha' \tag{23}$$

所以，

$$W = D \cdot A = [D_0 - (x - x_c)\tan\alpha] \cdot A \tag{24}$$

其中，x_c 是中心点的横坐标。对 (21) 式微分得：

$$\mathrm{dSA} = [D_0 - (x - x_c)\tan\alpha] \cdot A\ \mathrm{d}s \tag{25}$$

式中的 A 可以通过三角函数运算进行一些简化：

$$\begin{aligned} A(\theta,\alpha,\beta) &= \sin\frac{\theta}{2}\cdot[\frac{\cos\alpha'}{\cos(\theta/2+\alpha')}+\frac{\cos\alpha'}{\cos(\theta/2-\alpha')}] \\ &= \sin\frac{\theta}{2}\cdot\frac{2\cos(\theta/2)\cdot\cos^2\alpha'}{\cos^2(\theta/2)\cdot\cos^2\alpha'-\sin^2(\theta/2)\sin^2\alpha'} \\ &= \frac{\sin\theta\cdot(1-\sin^2\alpha\cdot\sin^2\beta)}{\cos^2(\theta/2)-\sin^2\alpha\cdot\sin^2\beta} \\ &= \sin\theta\cdot[\frac{\sin^2(\theta/2)}{\cos^2(\theta/2)-\sin^2\alpha\cdot\sin^2\beta}+1] \end{aligned} \tag{26}$$

将 (26) 式代入 (25) 式得到：

$$\mathrm{dSA} = [D_0-(x-x_c)\tan\alpha]\cdot\sin\theta\cdot[\frac{\sin^2(\theta/2)}{\cos^2(\theta/2)-\sin^2\alpha\cdot\sin^2\beta}+1]\mathrm{d}s \tag{27}$$

即:

$$\frac{\mathrm{dSA}}{\mathrm{d}s} = [D_0-(x-x_c)\tan\alpha]\cdot\sin\theta\cdot[\frac{\sin^2(\theta/2)}{\cos^2(\theta/2)-\sin^2\alpha\cdot\sin^2\beta}+1] \tag{28}$$

$\frac{\mathrm{dSA}}{\mathrm{d}s}$ 是测量船走过单位距离时条带的覆盖面积，即测量效率。从 (28) 式可知，当 $\sin\beta$ 增大时，测量效率单调递增。所以，当 $\sin\beta=1$，即 $\beta=90°$ 时，测量效率最高。 □

7.3 最优测线排布方式的算法

7.3.1 总体思想

我们已经知道海水深度与重叠率的单调关系，我们依此设计了一套贪心算法。贪心的目的是在满足题目要求的前提下，使重叠率尽可能的小; 因为在同样的 β 下，重叠率越小，间距越大，测线覆盖的海域面积越大。

已知海底西深东浅，我们从区域东南角开始排设，因为 $\beta\in$ [0°,90°]，所以测线指向为从西南指向东北方向，对于同一条测线，我们首先考虑测线的东北方向，再考虑西南方向，也就是先浅后深。我们将浅水区域测线重叠率设为最小值 10%，然后考虑深水区测线重叠率，如果小于等于 20%，则测线符合要求，继续设计下一条测线；如果大于 20%，该条侧线不符合要求，说明该 β 值下无法设计出符合要求的测线族，因为存在一条侧线，它的两端无法同时满足 10% $\sim$ 20% 的要求。

由性质 1，浅水端重叠率最低，所以按照贪心的算法思想，我们将浅水端重叠率设为 10%，并期望此时在深水端的重叠率也能达到最小。所以此时整体重叠率最小，可以用相同的条带覆盖尽可能多的区域。因此，此问题贪心法的局部最优可以导致全局最优的结果。

7.3.2 整体优化模型

我们的优化模型描述如下：

$$\min L_{total}(\beta, b_n) = \sum_{i=1}^{n} L_i$$

$$s.t. \begin{cases} \eta \in [10\%, 20\%] \ , i = 1, 2, 3... \\ Loss \equiv 0 \end{cases} \tag{29}$$

即满足相邻条带的重叠率 $\eta \in [10\%, 20\%]$ 和漏测率 $Loss = 0$，要求测线总长度 L_{total} 尽可能短，其中优化变量为 β 和一组测线的截距 b_n。

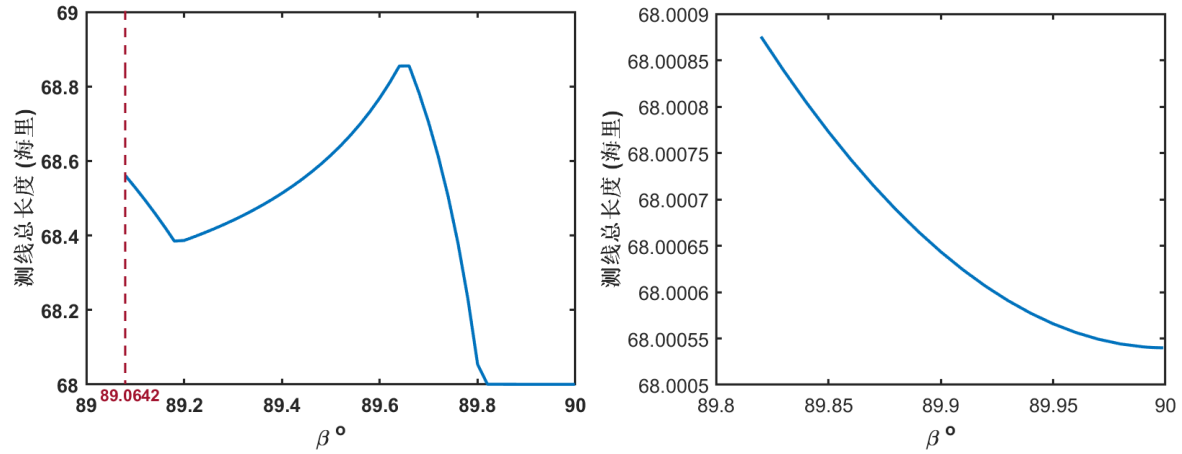
7.3.3 优化结果

首先，我们要确定测线方向角 β 的可行域，也就是相邻测线的重叠率满足 10%～20% 要求的 β 的范围。因为待测海域范围很大，而且测线只能为直线。加上上文已经证明了 β =90° 时, 测量效率最高，且此时容易控制重叠率。考虑一定的算法搜索偏差，满足要求的 β 应该是靠近 90° 的一段狭窄的区间。

我们利用程序对 β 进行遍历，找到了满足条件的 β 可行域为

$$\beta_{\textbf{possible}} \in [\textbf{89.0642}^\circ, \textbf{90}^\circ] \tag{30}$$

然后，我们在可行域中对 β 遍历，每一个 β 都按照贪心策略进行测线布设，将 β 与最短测线总长度的变化关系绘制于图9a。



(a) 测线总长度与 β 的关系　　**(b) 精细扫描结果**

图 9　不同 β 下的最短测线总长度

从图中可见，最短测线总长度全局最优值出现在 $\beta \in [89.82^\circ, 90^\circ]$ 之间的一个平台上，对该段进行精细扫描如图9b所示，可见，**全局最优值对应的 β 为** 90°，相应的最

优值计算为为 **125.936 千米**，即 **68 海里**，且满足完全覆盖待测矩形海域以及重叠率在 10%~20% 的要求，此时对应的最优测线示意图如图10。

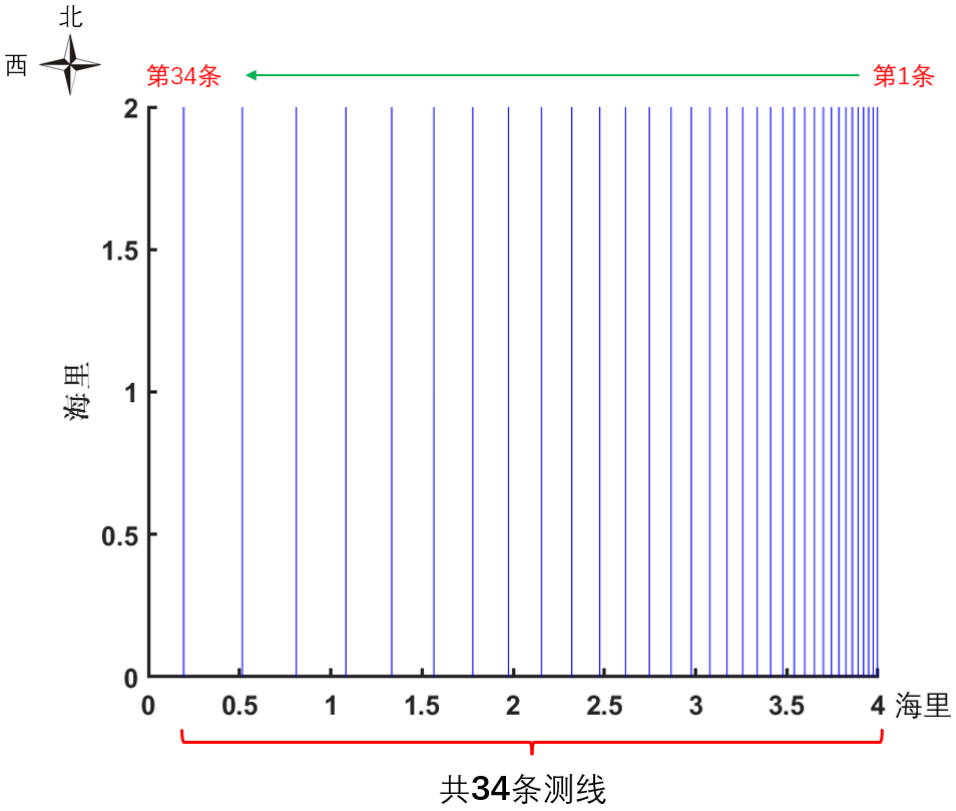


图 10　测线分布示意图

如图10所示，**共有 34 条测线，测线自东向西平行排布，东密西疏；各测线的坐标和测线间隔如下表**（由于篇幅原因，这里只列出前三条测线和最后两条测线，全部的测线信息可见支撑文件）

	测线编号					
	1	2	3	……	33	34
测线横坐标/海里	3.996531	3.973297	3.948076	……	0.51459	0.193586
与前一条测线的间距/海里	——	0.023234	0.025221	……	0.295715	0.321004

8 问题 4: 不规则海底坡面测线布局设计

此问面向实际情况，需根据已知的离散网格深度数据，设计在 3 个指标下较优测线方案。我们首先对测线布局的表示方式进行约定，接着分析测线三个指标的求解方式，而后依据测线指标求解最优布局，最后进行结果分析。

8.1 测线布局约定

坐标系约定

如图 11所示，以东西方向（4 海里）为 x 轴、南北方向（5 海里）为 y 轴，对此问矩形海域建立平面直角坐标系。

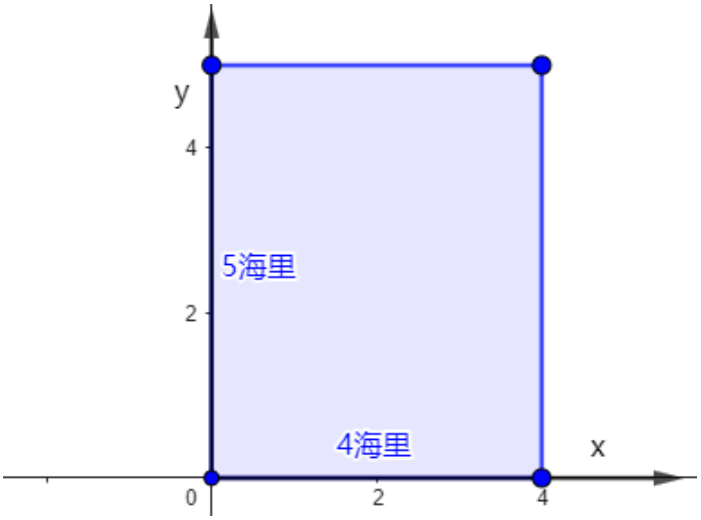


图 11　问题四矩形海域坐标系

测线布局表示约定

测线布局的表示分为三部分，测线条数、测线角度、测线位置（间距）。

测线条数：约定某测线布局的测线条数为 n 条。

测线角度：由**假设**一可知所有测线平行，有相同的角度，因而约定所有测线与 x 轴正半轴夹角为 α　$(\alpha \in [0, 180°))$。则当直线斜率存在时，斜率 $k = \tan\alpha$。

测线位置：当直线斜率存在时，使用直线斜截式方程 $y = kx + b$ 确定直线位置。由于所有直线有共同的斜率 k，因此对于每条直线仅需提供纵截距 b_i 即可确定测线位置。

取上述约定得到的直线在矩形海域内的线段为一个测线布局，即决策变量为

$$\begin{cases} n & n \geq 0 \\ \alpha & 0 \leq \alpha < 180° \\ b_i & i = 1, 2, \cdots n \end{cases} \tag{31}$$

其中 n 为测线条数，α 为测线与 x 正半轴夹角，b_i 为 n 条平行测线的纵截距。

8.2 测线指标求解

8.2.1 指标 1：测线总长度

已知一个测线族中 n 条测线的角度 α、第 i 条测线纵截距 b_i，求测线总长度。

求出每条测线所在直线与矩形的两个交点，将所有测线在矩形内线段长度求和即可。求直线与矩形的两交点的方法为：如图 12，求出直线 $y = kx + b_i$ $(k = \tan\alpha)$ 与矩形的下、左、右、上四侧所在直线的交点 $B_j\,(x_j, y_j)$ $(j = 1, 2, 3, 4)$，找到其中在矩形边界上的两点（判断坐标是否范围即可）。

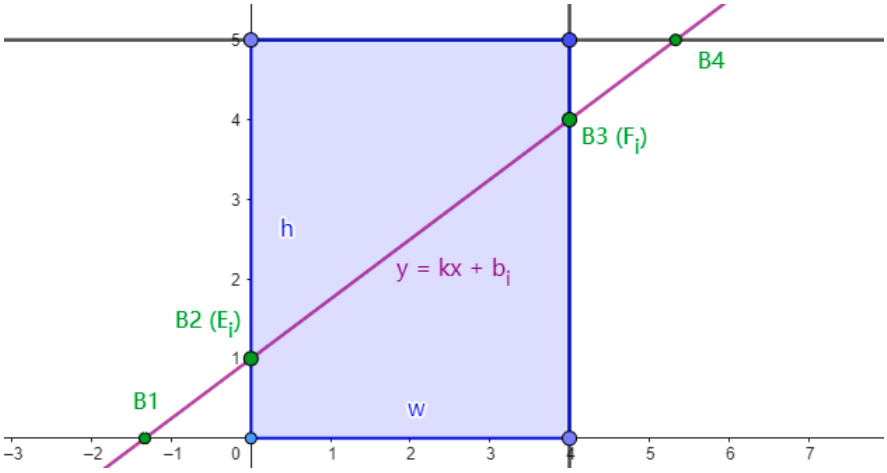


图 12　直线与矩形交点

B_j 可直接联立两直线方程得出，这里略去计算过程，得到 4 个交点坐标如下。

$$\begin{cases} B_1\left(-\frac{b_i}{k}, 0\right) \\ B_2\,(0, b) \\ B_3\,(w, kw + b) \\ B_4\left(\frac{h - b_i}{k}, h\right) \end{cases} \tag{32}$$

得到四个交点后，需要对每个 B_j 判断是否在矩形边界上，找到找到测线在矩形边界上的两个端点（如图 12 B2、B3）。判断在矩形边界的条件如下：

$$0 \le x_j \le w \quad \wedge \quad 0 \le y_j \le h \tag{33}$$

满足如上条件的两个 B_j 即为第 i 条测线在矩形边界上的两个端点，记为点 E_i, F_i（如图 12）。

因此，一个测线族的总长度 L_{total} 为

$$L_{total} = \sum_{i=1}^{n} |E_i F_i| = \sum_{i=1}^{n} \sqrt{(x_{E_i} - x_{F_i})^2 + (y_{E_i} - y_{F_i})^2} \tag{34}$$

8.2.2 指标 2：漏测海区占待测海域面积百分比

求漏测面积占比，则需计算测线覆盖总面积。由于待测海域海底并不规则，无法通过海底斜面方程计算测线覆盖范围，因此考虑将矩形海域划分为网格离散计算。

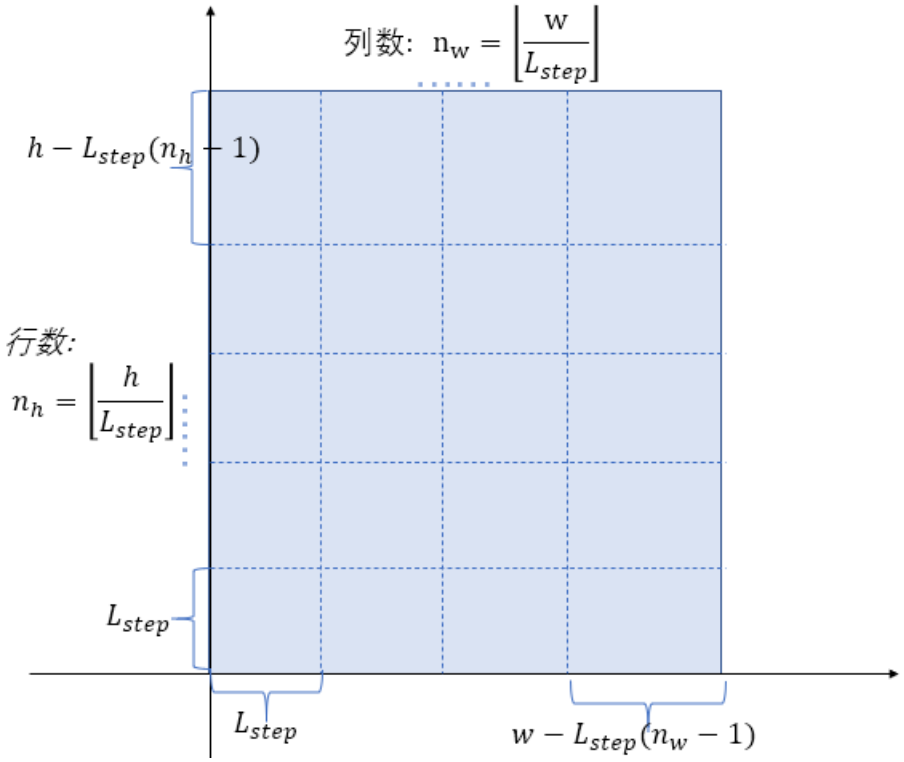


图 13　按照步长划分为均匀网格

如图 13所示，以 L_stepm 为步长将矩形海域划分为均匀的正方形网格，不整除的多出部分计入最后（上侧和右侧）的网格。共划分成 $n_h = \left\lfloor \frac{h}{L_{step}} \right\rfloor$ 行、$n_w = \left\lfloor \frac{w}{L_{step}} \right\rfloor$ 列;

共有 4 种面积的小网格:

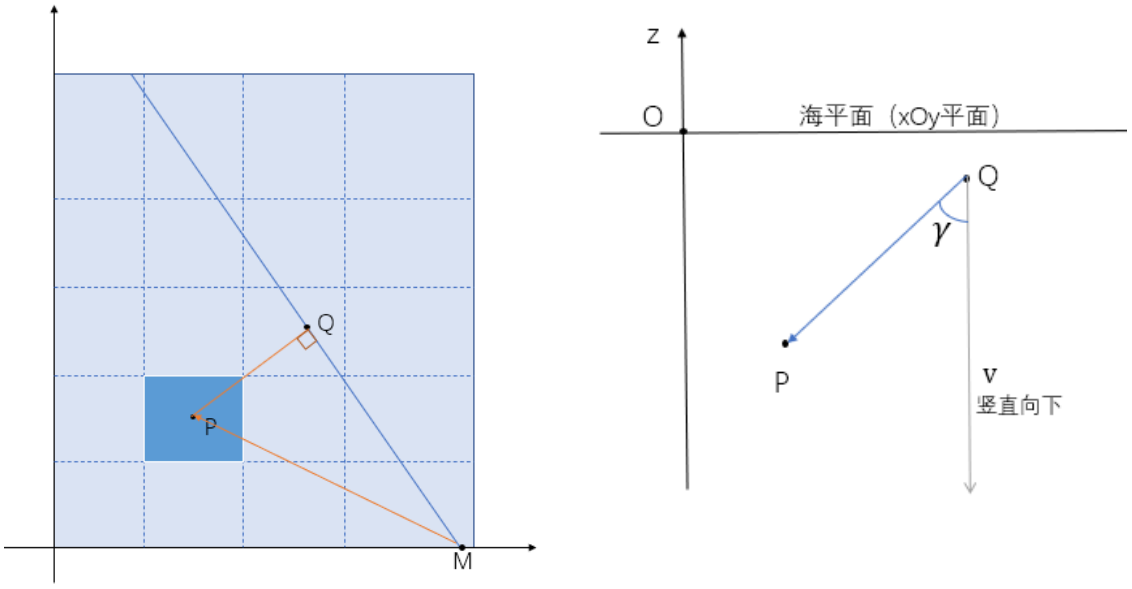
1. $S_1 = L_{step}^2$; 为大多数小网格面积;
2. $S_2 = L_{step} \cdot (w - (n_w - 1)L_{step})$；为最右侧一列网格（除右上角）面积。
3. $S_3 = L_{step} \cdot (h - (n_h - 1)L_{step})$；为最上侧一行网格（除右上角）面积。
4. $S_2 = (w - (n_w - 1)L_{step}) \cdot (h - (n_h - 1)L_{step})$；为右上角网格面积。

对于每个小网格，计算其**中心位置**点 P(x_P, y_P) 能否被**相邻**两条测线辐射到，若其中心点 P 可被其两侧的测线任意之一覆盖到，则**认为此网格可被完全覆盖**，将此网格的面积计入总覆盖面积。方法为计算点 P 到测线的垂线与竖直向下方向夹角是否小于换能器开角的一半（$\frac{\theta}{2}$）。具体计算如下。

从网格中心点 P 向侧限作垂线，垂足为 Q，如图14a所示为在 xOy 平面的俯视图。以原平面直角坐标系 xOy 点 O 在海平面位置为原点，以竖直向上方向为 z 轴，建立空间直角坐标系，沿直线 PQ 做纵截面可得到图 14b，其中 γ 为待求解角。

依据问题 4 提供的离散数据，可插值得出点 P 处深度，记为 d；则点 P 的空间坐标为 $P(x, y, -d)$；取测线上任意一点 $M(x_M, y_M, 0)$，则有向量 $\vec{MP}$

$$\vec{MP} = (x - x_M, y - y_M, -d) \tag{35}$$



(a) 点 P 与测线位置关系（俯视图）　　(b) PQ 连线处的纵截面图

图 14　P 点是否在测线辐射范围内的计算

向量 $\vec{MP}$ 向测线方向 $\vec{l}=(\cos\alpha,\sin\alpha,0)$ 投影即可得到向量 $\vec{MQ}$

$$\vec{MQ}=\frac{\vec{MP}\cdot\vec{l}}{|\vec{l}|}\cdot\frac{\vec{l}}{|\vec{l}|} \tag{36}$$

可计算得出点 Q 坐标

$$Q=Q+\vec{MQ} \tag{37}$$

因此得到向量 $\vec{QP}$

$$\vec{QP}=P-Q \tag{38}$$

至此，如图 14b所示，已知向量 $\vec{QP}$ 和竖直向下方向向量 $\vec{v}$，可计算的测线辐射到点 P 与竖直方向夹角 γ 为

$$\gamma=\arccos(\frac{\vec{QP}\cdot\vec{v}}{|\vec{QP}|\,|\vec{v}|}) \tag{39}$$

题中给出多波束换能器开角 $\theta=120°$，计算得出的 γ 若满足：

$$\gamma\ \leq\ \frac{\theta}{2}=60° \tag{40}$$

则**点 P 可此测线覆盖**到。

对于每个小网格，若其中心点 P 可被其两侧的测线任意之一覆盖到，则认为此小网格可被完全覆盖，将**此网格的面积计入总覆盖面积**。

综上，计算漏测海区占总面积百分比的流程为算法 1。

算法 1 计算漏测海区占总面积百分比

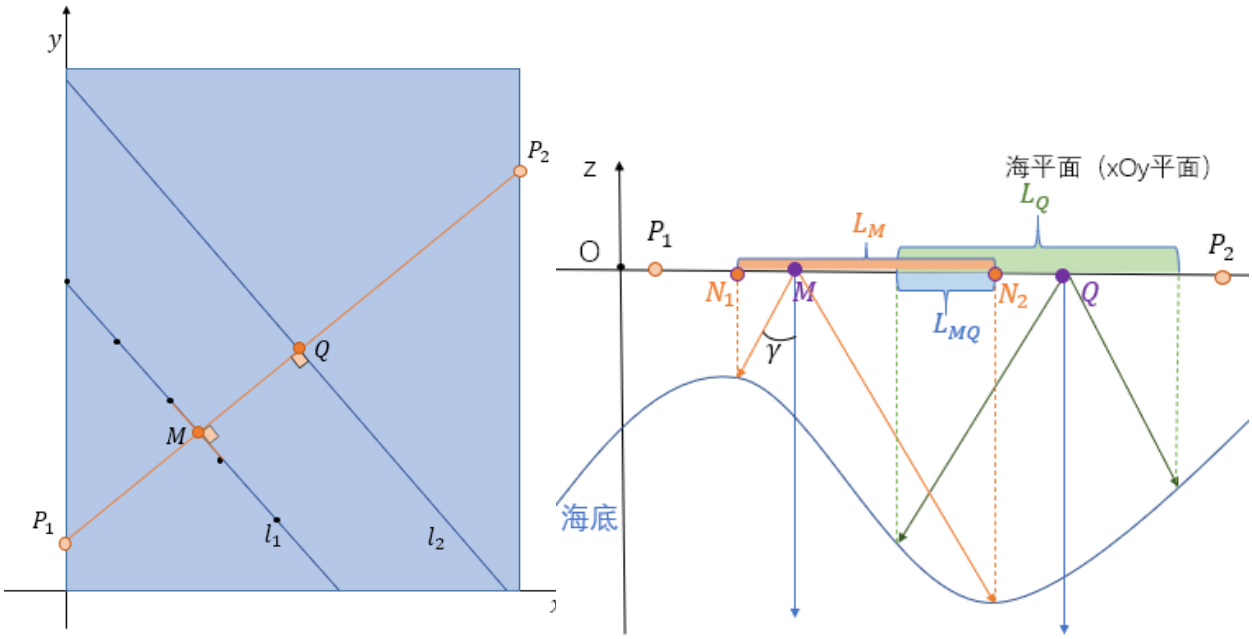
输入: 划分网格得到的小网格几何 $Grids$；待计算测线族；

输出: 返回漏测海区占总面积百分比

function GetMissAreaPercentage
    $CoverageArea \leftarrow 0$
    for $grid$ in $Grids$ **do**
        if grid 可被其相邻两侧测线任意之一覆盖 **then**
            $CoverageAera \leftarrow CoverageAera + grid.area$
        end if
    end for
    $MissAreaPercentage \leftarrow 1 - \dfrac{CoverageArea}{AllArea}$
    return $MissAreaPercentage$
end function

8.2.3 指标 3：重叠率超过 20% 部分的总长度

因为海底地形不规则，故仍考虑划分小单元的方式离散计算重叠超过阈值的长度。将每条测线按一定步长划分为若干段，对于每个测线子段，计算段中心点与前一条测线对应位置构成的截面的覆盖范围，得到此截面的重叠比例，若比例超过 20% 阈值，则将此段测线长度计入重叠超过 20% 总长度。总体流程如算法 2所示。



(a) 测线分段计算重叠长度（俯视水平面）　　(b) 相邻测线截面重叠率计算（MQ 纵截面）

图 15　相邻测线重叠长度计算

算法 2 计算重叠率超过 20% 总长度

输入: 待计算的测线族集合 $Lines$

输出: 重叠率超过 20% 总长度

function GetOverlapLength
    $OverlapLength \leftarrow 0$
    for $l_1 \leftarrow Lines$ **do**
        for $segment \leftarrow l_1$ **do**
            $M \leftarrow segment.midpoint$
            $Q \leftarrow$ 与 l_1 相邻的前一条测线 l_2 上的与 M 的对应点
            $L_M, L_Q, L_{MQ} \leftarrow l_1$ 覆盖长度，l_2 覆盖长度，l_1 l_2 覆盖重叠部分水平长度
            $ratio \leftarrow \dfrac{L_{MQ}}{L_M}$
            if $ratio \geq 20\%$ **then**
                $OverlapLength = OverlapLength + segment.length$
            end if
        end for
    end for
    return $OverlapLength$
end function

下面对算法流程进行分析。

(a) 划分子段计算

如图 15a所示，当计算测线 l_1 与其相邻测线 l_2 的重叠率时，将 l_1 按照步长 L_{step} 划分为若干均匀子段。对于每个子段，若其中心点（如图 15a M 点）与相邻测线对应位置连线（如图 15a Q 点）所在截面的覆盖重叠率超过 20% 阈值，则 M 点所在子段重叠率全部超过阈值，将此段长度计入“重叠率超过 20% 的总长度”。

(b) 计算测线纵截面重叠率

如图 15b所示，此图对应为图 15a中 MQ 所在直线的纵截面。在此截面中，测线 $l1$ 在点 M 处向两侧测深，其覆盖范围投影到水平面上为线段 N_1N_2（长度为 L_M，如图 15b）；同理在此截面内，测线 l_2（与 l_1 相邻）在点 Q 处向两侧测深，覆盖线段长度为 L_Q；如图 15b中标注，两条测线在截面内重叠部分在水平面上投影长度为 L_{MQ}。

根据 l_1 覆盖长度 L_M、l_2 覆盖长度 L_Q、两测线重叠长度 L_{MQ} 可计算此截面测线 l_1 重叠比例 $ratio$：

$$ratio = \frac{L_{MQ}}{L_M} \tag{41}$$

关键在于如何计算上述覆盖长度（如 L_M），也就是计算测线 l_1 在点 M 处的覆盖边界点 N_1、N_2。具体计算方式如下。

(c) 计算测线截面覆盖范围边界点

计算各边界点的原理相同，这里以计算 l_1 在此界面的覆盖范围 N_1N_2 的边界点 N_1 为例。，我们采用二**分搜索**的方式计算边界点 N_1。如图 15b所示，左侧边界点一定在 M 左侧，在点 P_1（截面在矩形海域的边界交点）右侧，因而边界点 N_1 的搜索空间为线段 P_1M。易知线段 P_1M 满足二分性质，即一定存在点 N_1，在 N_1 左侧（靠近点 P_1 一侧）的所有点都不能被测线 l_1 覆盖；在 N_1 左侧（靠近点 M 一侧）的所有点都可被测线 l_1 覆盖。

二分搜索还需要判断任意一点 $T(x, y, z)$ 是否可被测线 l_1 覆盖，这里仅需判断向量 $\vec{MT}$ 与数值向下方向夹角是否小于 $\frac{\theta}{2}$（θ 为换能器开角），即是否满足条件：

$$\arccos(\frac{\vec{MT}\cdot(0,0,-1)}{|\vec{MT}|}) \leq \frac{\theta}{2} \tag{42}$$

二分查找流程如算法 3所示。

算法 3 二分查找覆盖范围边界点

输入: 测线截面信号发射点 M；截面与矩形边界交点 P

输出: 返回覆盖范围边界点坐标 N

function BinarySearchBoundPoint
    $L \leftarrow P$
    $R \leftarrow M$
    while $||PL| - |PR|| < eps$ **do**
        $Mid \leftarrow (L + R)/2$
        if 点 Mid 可被测线在 M 点发出的信号覆盖 **then**
            $R \leftarrow Mid$
        else
            $L \leftarrow Mid$
        end if
    end while
    $N \leftarrow L$
    return N
end function

综上 (a)(b)(c), 将每条测线划分为若干段，对每个测线子段，计算段中心点与前一条测线对应位置构成截面的覆盖范围，计算重叠比例，若比例超过 20%，则将此段测线长度计入重叠超过 20% 总长度，即可得到一个测线族的指标 3(重叠超过 20% 总长度)。

8.3 测线布局优化

此模型的优化变量为 β 以及所有测线纵截距组成的数组 b_n，优化维度为 $n+1$，这是一个高维空间的优化问题，我们选择使用遗传算法（Genetic Algorithm，GA）来进行最优解的搜寻。

遗传算法适用于高维度优化问题的原因在于其独特的搜索和优化策略，使其能够在大型搜索空间中寻找全局最优解或者近似最优解。遗传算法的基本思想是模拟生物进化过程中的自然选择、交叉和变异机制，以逐代改进种群中的个体，最终找到问题的最优解或者接近最优解。通常的遗传算法都有如下参数：

1. 种群大小：这是种群中个体数量的设置。较大的种群可提高全局搜索的能力，但也会降低计算效率。
2. 交叉概率：控制交叉操作发生的概率，较大的交叉概率可提高搜索到全局最优值的概率，但会导致收敛速度减慢。
3. 变异概率：控制变异操作发生的概率，较大的变异概率也可提高搜索到全局最优值的概率，但也会导致收敛速度减慢。

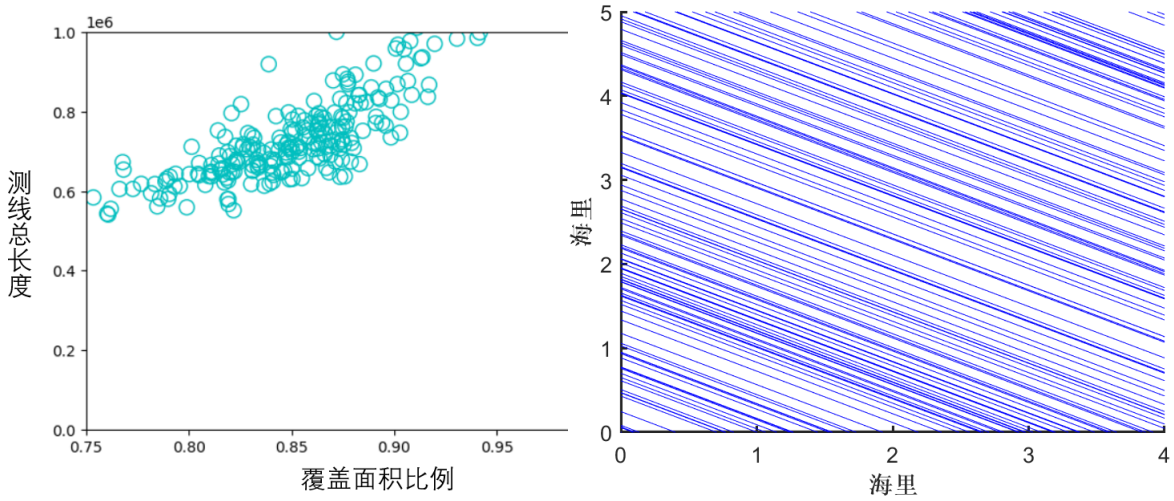
在我们的问题中，实际上测线的条数也是不确定的，如果简单的将测线条数设置为优化变量，则优化变量的维度不确定，并会在优化过程中频繁改变，这将严重影响遗传算法的效率。为解决此问题，我们提出了**自适应遗传算法**。**该算法将优化变量维度和求解域同时进行扩展，求解域中包括了可行域和不可行域，当优化变量的某一维落在该维度的不可行域时，我们将它称为隐藏维度；反正，落在可行域时，称为正常维度。在求目标函数的值时，仅使用正常维度的变量，这样就将可变维度的优化变量转化为了固定维度的优化变量**。

在该问题中，具体解决方法为，我们提前将优化变量中的n 的维度进行提高，预计测线数量应为 100 左右，我们将b_n 的维度设为 10000，在求目标函数值前，我们先通过斜率和截距判断该直线是否与待测矩形海域相交，我们仅保留能相交产生测线的截距值，并利用它们计算目标函数。虽然我们没有直接定义测线的数量，但该算法可以自动通过与矩形相交直线的数量来优化测线的数量，这体现了该算法的**自适应性**，所以，我们将该算法称为自适应遗传算法。

在该问题中，我们将种群大小设为 100，变异概率与交叉概率均设为 0.001，最大迭代次数为 1000 次，模型设置完成后，开始计算求解。

8.4 结果展示与分析

通过迭代求解，绘制所有可能解于图16a，我们将覆盖面积比例限制在 90% 以上，在所有可能的解中选择测线总长度最小的解，如图16b。最终解为测线总长度 746296.81m，即 **402.968 海里，测线族倾角为 148.56°，覆盖面积比例为 90.28%，重叠区域中，重叠率超过 20% 的测线总长度为 220.4220 海里**。

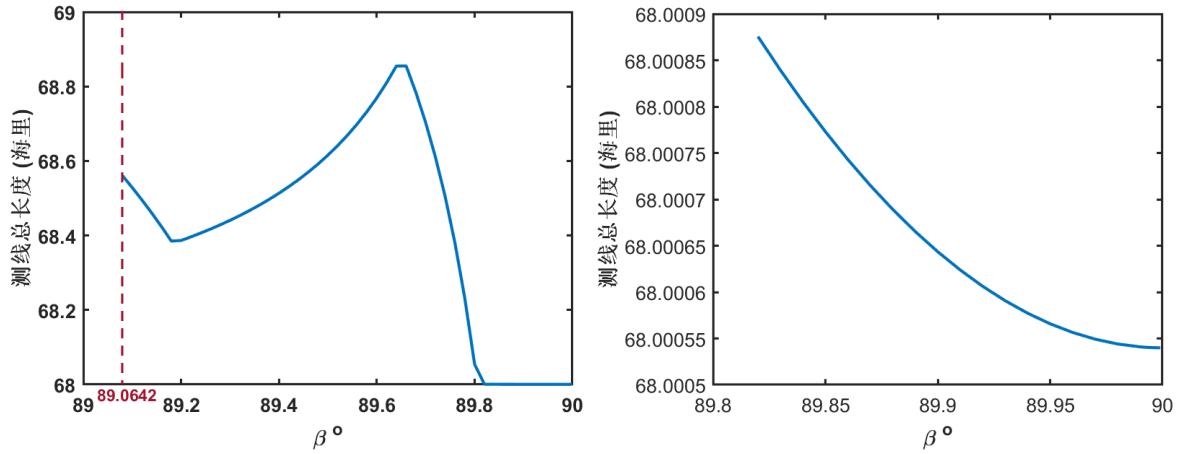


(a) 解的散点图　　**(b) 测线排布图**

图 16　优化结果示意图

9 灵敏度分析

在问题 1 和问题 2 中，我们的模型可以精确地测量出多波束测量条带的覆盖宽度和相邻条带的覆盖率. 而问题 3 中对排布方位角 β 可行域的搜索鲁棒性强, 由下图所示:



(a) 测线总长度与 β 的关系　　**(b) 精细扫描结果**

图 17　不同 β 下的最短测线总长度

由图可知，在保证全覆盖测量的前提下，即使 β 有一个微小的偏差，我们的模型仍能给出一个总测量长度较短的测线布设方案。

10 模型评价

10.1 问题 3 中的贪心优化算法

10.1.1 优点

该算法利用贪心的策略，从局部最优出发，找到了全局最优，算法简单高效，可以很容易的找到全局最优解。而且由于在灵敏度分析中提到的原因，该算法鲁棒性极强，非常适合于现实情境中测量船位置和移动方向存在偏差的情况。

10.1.2 缺点

该算法专门为海底坡面设计，无法扩展到任意形状的海底上，但在实用中，实际上没有完美的海底坡面，所以该算法只能为某些近似坡面的海底形貌提供测线布线指导，实用价值有一定的局限性。

10.2 问题 4 中的遗传算法

10.2.1 优点

我们使用的遗传算法全局搜索能力强、鲁棒性强，适用于解决复杂的高维优化问题。该算法思路清晰，且对于约束的处理简单明了，具有较强的可解释性。对于问题 4，有 3 个优化目标，遗传算法在解决多目标优化问题上也有较高的效率，我们通过将部分优化目标转化为约束的方法，使遗传算法能更高效的处理我们的问题。

10.2.2 缺点

作为一种迭代优化算法，该算法具有效率低、运算速度慢、对计算机硬件要求高等特点，该算法的时间复杂度和空间复杂度均与待测海域面积呈线性关系，对于大面积的测线布线问题需要较长时间的计算。且该算法容易陷入到局部最优中，比如在该问题中，该算法只找到了一个可行解族。如果增加变异概率和种群数量，收敛速度又会大大减慢。

10.3 未来展望

为了达到更好的优化效果，可以选用其他的群优化算法，对于高维优化问题，可以尝试如粒子群算法 (PSO)、模拟退火算法 (SA)、蚁群算法 (ACA)、免疫优化算法 (IA)、人工鱼群算法 (AFSA) 等。对于其中的**粒子群算法和模拟退火算法**，已经包含在附录 B 问题 4 的求解源代码中，可以很容易的调用，但本文并未给出对这些算法应用的详细分析。而对于其他的算法，我们使用的 Python 库 scikit-opt 也可以很好的支持。

参考文献

[1] 周立. 海洋测量学 [M]. 北京：科学出版社, 2013

[2] 赵建虎. 现代海洋测绘 [M]. 武汉大学出版社,2007.

[3] 魏然. 多波束测深系统导航软件的设计与实现 [D]. 哈尔滨工程大学,2008.

[4] GB/T 42640-2023. 多波束水下地形测量技术规范 [S].

[5] 张旭, 叶小心, 洪德玫. 多波束系统在长江航道测量中的测线布设方法研究 [J]. 中国水运. 航道科技,2017(01):52-55.DOI:10.19412/j.cnki.42-1395/u.2017.01.011.

附录

附录 A　支撑材料目录

支撑文件目录	
文件名列表	文件说明
result1.xlsx	问题 1 的计算结果
result2.xlsx	问题 2 的计算结果
问题 3 的测线设计方案.xlsx	问题 3 最优方案中各测线的信息
问题 4 的测线间隔.xlsx	问题 4 最优方案中各测线之间的间隔
Problem1.m	问题 1 计算模型
Problem2.m	问题 2 计算模型
Problem3.m	问题 3 的测线设计模型
Problem3 测线.m	问题 3 最优测线方案
Problem4.ipynb	问题 4 的测线设计模型
Problem4 测线.m	问题 4 最优测线方案

附录 B　算法源代码

```
问题1的源代码
clear;
L = -800:200:800;
alpha=1.5
D = 70 - L*tand(alpha)
L1 = D*sind(60)/cosd(60+alpha)
L2 = D*sind(60)/cosd(60-alpha)
W = (L1+L2)*cosd(alpha)
for i=2:9
    eta(i) = ((L2(i-1)+L1(i))*cosd(alpha)-200)/((L2(i-1)+L1(i))*cosd(alpha));
end
ANS = [D;W;eta]
\\
问题2的源代码
```

```
clear;
beta=0:45:315;
L = 0:0.3:2.1;
L=L*1852;
alpha = 1.5;
theta = 120;
h0 = 120;
for i=1:8
    for j=1:8
        alpha1 = asind(sind(alpha)*sind(beta(i)));
        D = h0+L(j)*cosd(beta(i))*tand(alpha);
        L1 = D*sind(theta/2)/cosd(theta/2+alpha1);
        L2 = D*sind(theta/2)/cosd(theta/2-alpha1);
        W(i,j) = (L1+L2)*cosd(alpha1);
    end
end
W
\\
```

问题3的源代码

```
clear;
theta=120;
global h0
global alpha
h0=110;
alpha=1.5;
xl = 4*1852;
yl = 2*1852;
eta_max = 0.2;
eta_min = 0.1;
RES=[;];
Al=[89.9999];
for beta=Al
    alert=0;
    b=[];
    k=tand(beta);
    b_min = min(0,-xl*tand(beta));
    b_max = max(yl,yl-4*tand(beta));
    alpha1 = asind(sind(alpha)*sind(beta));

    %求初始b1
    b(1) = tand(theta/2)*Deep(xl,0)*sqrt(k^2+1)-k*xl;

    n=1; %n表示b数组长度，即测线条数
    while b(n) < tand(theta/2)*Deep(0,yl)*sqrt(k^2+1)+yl
        if length(b) ~= n
            disp("ERROR(1): length of b cannot match")
        end
```

```
syms B
dis = (B-b(n))/sqrt(k^2+1);
L2_pre = (h0-(b(n)/k+2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2-alpha1);
if -b(n)/k > 0
    x_omx = -b(n)/k-dis*sind(beta);
else
    x_omx = 0;
end

L1 = (h0+(x_omx-2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2+alpha1);
L2 = (h0+(x_omx-2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2-alpha1);
f=@(B) ((L2_pre+L1)*cosd(alpha1)-dis)/((L1+L2)*cosd(alpha1))-eta_min;
B_sol =solve(f,B);
B_sol = vpa(B_sol);
b(n+1) = B_sol;

%test restriction
if k*xl+b(n) < yl
    x_omi = min(xl,(yl-b(n+1))/k);
    x_pre = xl;
else
    x_omi = (yl-b(n+1))/k;
    x_pre = (yl-b(n))/k;
end

L2_pre = (h0+(x_pre-2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2-alpha1);
L1 = (h0+(x_omi-2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2+alpha1);
L2 = (h0+(x_omi-2*1852)*tand(alpha))*sind(theta/2)/cosd(theta/2-alpha1);
dis = (b(n+1)-b(n))/sqrt(k^2+1);
eta_omi = ((L2_pre+L1)*cosd(alpha1)-dis)/((L1+L2)*cosd(alpha1));

if b(n+1) >= yl
    disp('stop')
    if abs(b(n)-yl)/sqrt(k^2+1)>L2_pre
        dl=1; %补足测线长度
        b(n+1) = yl-dl*sind(beta);
        n = n+1;
    else
        b=b(1:n)
    end
    break
end
if eta_omi <0
    S = sprintf("Warning: beta = %f cannot measure all data",beta);
    alert=1;
    disp(S)
    break
```

```
        end
        if eta_omi > eta_max
            S = sprintf("Warning: beta = %f cannot fit restriction on eta",beta);
            alert=1;
            eta_omi
            disp(S)
            break
        end
        if eta_omi < eta_min
            eta_omi
            S = sprintf("ERROR(2): Data out of expection!");
            disp(S)
            break
        end
        n=n+1; %b数组长度加一
    end

    if alert==0
        Length =0;
        for i=1:n
            x1 = max(0,-b(i)/k);
            x2 = min(xl,(yl-b(i))/k);
            Length = Length + (x2-x1)*sqrt(k^2+1);
            %(x2-x1)*sqrt(k^2+1)
        end
        RES=[RES,[beta;Length]];
    end
end
b
RES
plot(RES(1,:),RES(2,:)./1852,'-')
xlabel("\beta ^o",Interpreter="tex")
ylabel("测线总长度 (海里)")

function D = Deep(x,y)
    global h0
    global alpha
    D = h0+(x-2*1852)*tand(alpha);
end
```

问题4源代码

```
################################## 模块导入 ##################################
from math import *
import numpy as np
import pandas as pd
import time
```

```python
import matplotlib.pyplot as plt
from sko.GA import GA
from sko.PSO import PSO
from sko.SA import SA
################################## 加载数据 ##################################
file_path = "./附件.xlsx"
sheet_name = 'Sheet1'
df = pd.read_excel(file_path, sheet_name=sheet_name, engine='openpyxl')
# best_x = np.load("P4_best_x.npy")
################## 导入并处理深度数据，使得可用xy坐标索引 ####################
depth_df = df.iloc[1:252, 2:203]
col_name = df.iloc[0, 2:203].tolist()
row_name = df.iloc[1:252, 1].tolist()
depth_df.index = row_name
depth_df.index = depth_df.index.round(2)
depth_df.columns = col_name
depth_df.columns = depth_df.columns.round(2)
length = depth_df.index[-1] # 海域南北长度
width = depth_df.columns[-1] # 海域东西宽度

################################## 工具函数 ##################################
def timing(func):
    '''
    函数计时工具
    '''
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        elapsed_time = end_time - start_time
        print(f"{func.__name__} 执行时间: {elapsed_time} 秒")
        return result
    return wrapper

def get(x, y, depth_df):
    '''
    description: 插值得到坐标为(x,y)处的深度值
    note: xy单位为海里，返回深度单位为m
    '''
    if (x < depth_df.index[0] or x > depth_df.index[-1] or
            y < depth_df.columns[0] or y > depth_df.columns[-1]):
        return None
    def bilinear_interpolation(x, y, x1, y1, x2, y2, q11, q12, q21, q22):
        # 计算长方形的宽度和高度
        width = x2 - x1
        height = y2 - y1
        # 将目标点的坐标映射到 [0, 1] 的标准化坐标空间中
```

```
        x_nor = (x - x1) / width
        y_nor = (y - y1) / height
        # 计算四个顶点的权重
        weight1 = (1 - x_nor) * (1 - y_nor)
        weight2 = x_nor * (1 - y_nor)
        weight3 = (1 - x_nor) * y_nor
        weight4 = x_nor * y_nor
        # 使用权重和顶点的数值进行插值
        res = (
            q11 * weight1 +
            q21 * weight2 +
            q12 * weight3 +
            q22 * weight4
        )
        return res

    dx = depth_df.index[1] - depth_df.index[0]
    x1 = round((int(x * 100) // 2 * 2) / 100.0, 2)
    x2 = round(x1 + dx, 2)
    dy = depth_df.columns[1] - depth_df.columns[0]
    y1 = round((int(y * 100) // 2 * 2) / 100.0, 2)
    y2 = round(y1 + dy, 2)

    return bilinear_interpolation(x, y, x1, y1, x2, y2,
                                  depth_df.loc[x1, y1], depth_df.loc[x1, y2],
                                   depth_df.loc[x2, y1], depth_df.loc[x2, y2])

def getm(x, y, depth_df = depth_df):
    '''
    description: 插值得到坐标为(x,y)处的深度值
    note: xy单位为m, 返回深度单位为m
    '''
    x /= 1852
    y /= 1852
    # ! 注意这里交换 x,  y
    res = get(y, x, depth_df)
    return res

def eq(x, y):
    '''
    description: 判断浮点数 x y 是否相等, 误差为eps
    '''
    eps = 1e-8
    return abs(x - y) < eps

def get_distance(a, b):
    '''
```

```
    description: 计算平面中两点 a b 的距离 (a b 为 numpy 格式)
    '''
    x1, y1 = a
    x2, y2 = b
    x1 -= x2
    y1 -= y2
    return sqrt(x1 * x1 + y1 * y1)

def proj(vec, proj_des):
    '''
    Description:  向量投影,  计算向量 vec 在向量 proj_des 上的投影向量
    '''
    proj_norm = np.dot(vec, proj_des) / np.linalg.norm(proj_des)
    proj_vector = proj_norm * (proj_des / np.linalg.norm(proj_des))
    return proj_vector

def unique_point(lst):
    eps = 0.1
    n = len(lst)
    if n == 0 or n == 2:
        return lst
    elif n > 2:
        res = [lst[0]]
        for i in range(1, n):
            l2 = lst[i]
            flag = True
            for l1 in res:
                if get_distance(l1, l2) < eps:
                    flag = False
                    break
            if flag:
                res.append(l2)
        return res



################ 主要部分 (计算某测线布局的总长、覆盖面积、重叠长度) #################
class Line:
    '''
    Description:  测线数据结构, 用于存储特定矩形海域测线相关信息
    note: 所有长度单位使用 m
    '''
    def __init__(self, k, b, points, length):
        self.k = k
        self.b = b
        self.points = points
        self.low_point = points[0]
```

```
        self.high_point = points[1]
        self.length = length

    def __str__(self):
        return f"k= {self.k}, b= {self.b}, points= {self.points}, length= {self.length}"

class Layout:
    '''
    Description: 创建一种测线布局 (测线个数、测线角度、测线位置)
    '''
    def __init__(self, alpha, lst, w, h):
        '''
        n: 平行测线条数
        alpha: # 测线与x轴正半轴夹角 [0, pi) (单位为弧度)
        lst: 测线位置列表, y = kx + b 中的纵截距 b (m)
        w: 待测海域宽度 (沿x正半轴) (m)
        h: 待测海域长度 (沿y正半轴) (m)
        '''
        self.alpha = alpha
        self.w = w
        self.h = h
        self.beta = atan(self.h / w) # 过原点对角线与x正半轴夹角
        self.theta = radians(120)
        if eq(self.alpha, pi/2):
            self.k = inf
        else:
            self.k = tan(self.alpha)
        assert self.alpha >= 0 and self.alpha < pi, "测线角度错误"
        self.lst = [b for b in lst if self.check_line_rectangle_intersections(self.k, b)]
        self.lst.sort()
        self.n = len(self.lst)

    def in_rectangle(self, x, y):
        return 0 <= x <= self.w and 0 <= y <= self.h

    def check_line_rectangle_intersections(self, k, b):
        if not k < inf:
            # 斜率不存在情况
            return 0 <= b <= self.w
        if eq(k, 0):
            # 斜率为 0 的情况
            return 0 <= b <= self.h
        else:
            # 与待测海域右侧边界交点(w, y1)
            y1 = k * self.w + b
            # 与待测海域上侧边界交点(x2, h)
            x2 = (self.h - b) / k
```

```
        # 与待测海域下层边界交点(x3, 0)
        x3 = -b / k
        tmp = [[0, b], [self.w, y1], [x2, self.h], [x3, 0]]
        tmp = [np.array(i) for i in tmp if self.in_rectangle(*i)]
        tmp = unique_point(tmp)
        if len(tmp) > 2:
            print(tmp, self.w, self.h)
        assert len(tmp) <= 2, f"计算直线与矩形交点个数非法 num = {len(tmp)}"
        return len(tmp) == 2

    def get_length(self):
        # 计算测线总长度
        # 返回值以m为单位
        if self.n == 0:
            self.length_all = 1e6
            return self.length_all
        self.length_all = 0
        self.lines = []
        k = self.k
        for b in self.lst:
            l = 0
            # 测线与x轴垂直，斜率不存在情况（特判）
            if not k < inf:
                l = self.h
                tmp = [[b, 0], [b, self.h]]
            # 斜率存在的情况
            elif eq(k, 0):
                tmp = [[0, b], [self.w, b]]
                l = self.w
            else:
                # 测线方程 y = kx + b
                # 与待测海域右侧边界交点(w, y1)
                y1 = k * self.w + b
                # 与待测海域上侧边界交点(x2, h)
                x2 = (self.h - b) / k
                # 与待测海域下层边界交点(x3, 0)
                x3 = -b / k
                # 计算测线与矩形边界交点
                tmp = [[0, b], [self.w, y1], [x2, self.h], [x3, 0]]
                tmp = [np.array(i) for i in tmp if self.in_rectangle(*i)]
                tmp = unique_point(tmp)
                assert len(tmp) == 2, f"计算测线与矩形交点个数非法 num = {len(tmp)}"
                l = get_distance(*tmp)
            self.lines.append(Line(k, b, tmp, l))
            self.length_all += l
        return self.length_all
```

```
def get_adjacent_line_indices(self, x, y):
    '''
    描述：计算点(x,y)两侧的测线下标
    特殊情况：当(x,y)位于边缘时，可能仅一侧有测线
    '''
    k = self.k
    if eq(self.alpha, pi/2):
        b0 = x
    else:
        b0 = y - k * x
    l = 0
    r = self.n - 1
    while l < r:
        mid = (l + r + 1) // 2
        b = self.lines[mid].b
        if b <= b0:
            l = mid
        else:
            r = mid - 1
    if l == 0 and self.lines[l].b > b0:
        l = -1
    r = l + 1
    return (l, r)

def check_coverage(self, line_index, x, y):
    '''
    description: 判断编号位line_index的测线是否可覆盖坐标(x,y)
    theta: 为信号源张角（单位：弧度）
    note：(x,y) 长度单位为m，以下计算使用弧度作为角度单位
    '''
    if line_index < 0 or line_index >= self.n:
        return False
    # 点(x,y) 位置的三维坐标 （以海平面为xoy平面）
    P = np.array([x, y, -getm(x, y)])
    # 待判断测线
    line = self.lines[line_index]
    # 测线上一点 （这里选择测线与矩形的左下侧交点）
    Q = np.array([*line.low_point, 0])
    # 测线三维方向向量
    v = np.array([cos(self.alpha), sin(self.alpha) ,0])
    # 计算向量QP在测线上的投影，垂足为 M
    QP = P - Q
    QM = proj(QP, v)
    M = Q + QM
    # 则垂足M指向点P的向量为MP
    MP = P - M
    MP = MP / np.linalg.norm(MP)
```

```
        # 计算MP与竖直向下方向的夹角gemma
        down = np.array([0, 0, -1])
        gemma = acos(np.dot(down, MP))
        return gemma <= self.theta / 2

    # @timing
    def get_coverage_area(self, step_length):
        '''
        Description: 计算覆盖面积
        step_length: 划分网格步长 (单位: m)
        note: 以下计算, 长度单位均化为m, 角度计算均为弧度
        '''
        # 覆盖面积统计变量
        self.coverage_area = 0
        self.coverage_ratio = 0
        if self.n == 0:
            return self.coverage_area
        if not hasattr(self, 'lines'):
            self.get_length()
        # 将矩形海域长宽换位m为单位
        h = self.h
        w = self.w
        # 按照步长参数计算划分网格数 (下取整, 多余部分计入右上边缘)
        nw = int(w / step_length)
        nh = int(h / step_length)
        # 4种情况小方格面积
        areas = [
            step_length * step_length,
            step_length * (h - (nh - 1) * step_length),
            step_length * (w - (nw - 1) * step_length),
            (h - (nh - 1) * step_length) * step_length * (w - (nw - 1) * step_length)
        ]
        # 创建覆盖网格矩阵 (计算面积不必须, 但可用于可视化)
        g = np.zeros((nw, nh), dtype=bool)
        # 遍历所有小网格, 判断每个格点是否可被覆盖, 对被覆盖的网格面积求和
        for i in range(nw):
            for j in range(nh):
                # 计算格点中心坐标 (x, y)
                x = (i + 0.5) * step_length
                y = (j + 0.5) * step_length
                # 查找格点中心两侧的测线下标
                l1, l2 = self.get_adjacent_line_indices(x, y)
                # 判断测线l1,l2是否可覆盖当前格点中心位置 (x,y)
                is_covered = self.check_coverage(l1, x, y) or self.check_coverage(l2, x, y)
                if not is_covered:
                    area = 0
                else:
```

```
                # 计算该格点面积
                if i < nw-1 and j < nh-1:
                    area = areas[0]
                elif i < nw-1:
                    area = areas[1]
                elif j < nh-1:
                    area = areas[2]
                else:
                    area = areas[3]
            g[i][j] = is_covered
            self.coverage_area += area
    # print(f"网格覆盖比例 --> {g.sum()} / {int(nh * nw)} 个")
    # print( "面积覆盖比例 --> {:5.4f} \n".format(self.coverage_area / (w * h)))
    self.coverage_ratio = self.coverage_area / (self.h * self.w)
    return self.coverage_area

def find_line_rectangle_intersections(self, P, v):
    if eq(v[0], 0):
        # 斜率不存在情况
        p1 = np.array([P[0], 0])
        p2 = np.array([P[0], self.h])
    elif eq(v[1], 0):
        # 斜率为0的情况
        p1 = np.array([0, P[1]])
        p2 = np.array([self.w, P[1]])
    else:
        k = v[1] / v[0]
        b = -k * P[0] + P[1]
        # 与待测海域右侧边界交点(w, y1)
        y1 = k * self.w + b
        # 与待测海域上侧边界交点(x2, h)
        x2 = (self.h - b) / k
        # 与待测海域下层边界交点(x3, 0)
        x3 = -b / k
        tmp = [[0, b], [self.w, y1], [x2, self.h], [x3, 0]]
        tmp = [np.array(i) for i in tmp if self.in_rectangle(*i)]
        tmp = unique_point(tmp)
        assert len(tmp) == 2, f"计算直线与矩形交点个数非法 num = {len(tmp)}"
        tmp.sort(key=lambda x: (x[1], x[0]))
        p1 = np.array(tmp[0])
        p2 = np.array(tmp[1])
    return p1, p2

def get_boundary(self, Q, v):
    '''
    Description:  查找以点Q为中心, 二维向量v为辐射方向 的辐射边界
    return: 返回左右边界 (val_l, val_r)
```

```
        这里仅返回实数，表示边界点距矩形边界的距离
    v: 这里的方向一定是从l2指向l1的
    '''
    # 单位化v
    v = v / np.linalg.norm(v)
    # 二分精度（单位：m）
    eps = 1
    # v 方向直线与矩形交点 （左右边界点）
    p1, p2 = self.find_line_rectangle_intersections(Q, v)
    # 二分查找左侧边界
    len1 = get_distance(Q, p1)
    l = 0
    r = len1
    while abs(r - l) > eps:
        mid = (l + r) / 2
        P = Q + v * mid
        d = getm(*P)
        gemma = atan(mid / d)
        if gemma <= self.theta / 2:
            l = mid
        else:
            r = mid
    bound1 = len1 - l
    # 二分查找右侧边界
    len2 = get_distance(Q, p2)
    l = 0
    r = len2
    while abs(r - l) > eps:
        mid = (l + r) / 2
        P = Q - v * mid
        d = getm(*P)
        gemma = atan(mid / d)
        if gemma <= self.theta / 2:
            l = mid
        else:
            r = mid
    bound2 = len1 + l
    # 注意这里bound指的是两个边界点距最左侧的距离
    res = (bound1, bound2)
    return bound1, bound2

# @timing
def get_overlap_length(self, threshold_ratio, step_length):
    '''
    Description:  求重叠比例超过 threshold_ratio 的测线总长度
    threshold_ratio : 计入重叠长度的比例阈值， 0~1
    step_length : 沿测线划分小段的步长
```

```
        note: 长度单位为m,  角度单位为rad
        '''
        self.overlap_length = 0
        if self.n < 2:
            return self.overlap_length
        # 因为需要个测线数据，所以需要调用get_length
        if not hasattr(self, 'lines'):
            self.get_length()
        # 遍历第 2~n 条测线,  每条测线求与其前一条测线重写超过阈值的长度
        alpha = self.alpha
        k = self.k
        for i in range(1, self.n):
            # 每次迭代判断l2与其前一条测线l1的重叠超阈值长度
            l2 = self.lines[i]
            l1 = self.lines[i-1]
            # 计算切分段数
            step_num = int(l2.length / step_length)
            # 最后一段长度（因为均匀分段可能不整除，因而将对于部分划入最后一段）
            last_seg_length = l2.length - step_length * (step_num - 1)
            # 对于l2，与矩形下部分边缘交点为P，方向向量为 v
            P = np.array(l2.low_point)
            v = np.array([cos(alpha), sin(alpha)])
            if (self.alpha > pi / 2):
                v = -v
            for j in range(step_num):
                seg_len = step_length if j < step_num-1 else last_seg_length
                try:
                    # 取l2每段的终点Q
                    Q = P + (j + 0.5) * step_length * v
                    # 求过Q点垂直于l2的直线与l1的交点M
                    if eq(k, 0):
                        M = np.array([Q[0], l1.b])
                    elif not k < inf:
                        M = np.array([l1.b, Q[1]])
                    else:
                        kn = -1 / k
                        bn = -kn * Q[0] + Q[1]
                        xm = (l1.b - bn) / (kn - k)
                        ym = k * xm + l1.b
                        M = np.array([xm, ym])

                    # 如果M不在矩形内，则不可能重叠
                    if not self.in_rectangle(*M):
                        continue
                    # 求点l2上点Q 和 l1上点M 能够辐射到的范围的边界点（二分查找）
                    QM = Q - M
```

```
                Q1, Q2 = self.get_boundary(Q, -QM)
                M1, M2 = self.get_boundary(M, -QM)
                # 计算重叠率
                if M2 <= Q1:
                    ratio = 0
                else:
                    ratio = (M2 - Q1) / (Q2 - Q1)
                # 判断ratio是否达到阈值，并计算当前段长度
                if ratio >= threshold_ratio:
                    self.overlap_length += seg_len
            except:
                self.overlap_length += seg_len
        return self.overlap_length

##############################################################################

@timing
def goalfunc(par):
    w = 4 * 1852 # 矩形海域跨度（x轴正半轴）
    h = 5 * 1852 # 矩形海域长度（y轴正半轴）
    # 测线与x轴正半轴夹角
    angle = par[0] # 输入角度，将对应弧度赋值给angle
    n = len(par)-1
    lst = par[1:n+1]
    num = 20000
    # for i in range(num):
    #     t = 100 + i * 20
    #     lst[i] = t
    # print(lst)
    n = len(lst)

    # # 创建测线布局 lyt
    lyt = Layout(angle, lst, w, h)

    # 总测线长度get_length_all()计算测试
    # print("测线总长度：", lyt.get_length())
    Tot_length = lyt.get_length()

    # 测试布局中测线参数计算及保存
    # for line in lyt.lines:
    #     print(line)

    # # 测试get_adjacent_line_indices
    # print(lyt.get_adjacent_line_indices(1, 3.9))

    # 测试覆盖面积
    # print(f"覆盖面积 = {lyt.get_coverage_area(100)} m^2")
```

```
    Area_ratio = lyt.get_coverage_area(50)/(20*1852*1852)

    # 测试重叠长度
    # threshold_ratio = 0.2
    # step_length = 100
    # over_length = lyt.get_overlap_length(threshold_ratio, step_length)
    global goal_cnt
    global output
    output.append(lyt)
    goal_cnt += 1
    print("\nNo.{:<3} num ={:3} alpha ={:6.2f}° length_all ={:11.2f}m area_ratio =
        {:6.4f}".format(goal_cnt, lyt.n, degrees(angle), Tot_length, Area_ratio))
    return Tot_length

dim=20001
lbb=-1000000*np.ones(dim)
ubb=1000000*np.ones(dim)
lbb[0]=0.00001
ubb[0]=pi-0.00001


def Ccconstr(par):
    w = 4 * 1852 # 矩形海域跨度 (x轴正半轴)
    h = 5 * 1852 # 矩形海域长度 (y轴正半轴)
    # 测线与x轴正半轴夹角
    angle = par[0] # 输入角度, 将对应弧度赋值给angle
    n = len(par)-1
    lst = par[1:n+1]
    num = 20000
    n = len(lst)

    # # 创建测线布局 lyt
    lyt = Layout(angle, lst, w, h)
    Area_ratio = lyt.get_coverage_area(50)/(20*1852*1852)
    return 0.9-Area_ratio

constraint_ueq = (
    lambda par: Ccconstr(par)
    ,
)

goal_cnt = 0
output = []
#遗传算法
ga = GA(func=goalfunc, n_dim=dim,prob_mut=0.001, size_pop=100, max_iter=300,lb=lbb,ub=ubb,
    constraint_ueq=constraint_ueq)
best_x, best_y = ga.run()
```

```
print('best_x:', best_x, '\n', 'best_y:', best_y)

# #粒子群
# pso = PSO(func=goalfunc, n_dim=dim, pop=40, max_iter=150, lb=lbb, ub=ubb, w=0.8, c1=0.5,
    c2=0.5, constraint_ueq=constraint_ueq)
# pso.run()
# print('best_x is ', pso.gbest_x, 'best_y is', pso.gbest_y)

#免疫优化算法
# ia = IA(func=goalfunc, n_dim=dim,lb=lbb,ub=ubb, size_pop=500, max_iter=800, prob_mut=0.2,
#              T=0.7, alpha=0.95)
# best_x, best_y= ia.run()
# print('best_x:', best_x, '\n', 'best_y:', best_y)

#模拟退火算法
# num = 20000
# x0=np.array([])
# x0=np.append(x0,0)
# for i in range(num):
#    t = 100 + i * 10
#    x0=np.append(x0,t)

# sa = SA(func=goalfunc, x0=x0, T_max=1, T_min=1e-9, L=300,
    max_stay_counter=150,lb=lbb,ub=ubb,constraint_ueq=constraint_ueq)
# best_x, best_y = sa.run()
# print('best_x:', best_x, 'best_y', best_y)

#遗传算法结果绘图
Y_history = pd.DataFrame(ga.all_history_Y)
fig, ax = plt.subplots(2, 1)
ax[0].plot(Y_history.index, Y_history.values, '.', color='red')
Y_history.min(axis=1).cummin().plot(kind='line')
plt.show()

#粒子群算法结果绘图
# plt.plot(pso.gbest_y_hist)
# plt.show()

#模拟退火
# plt.plot(pd.DataFrame(sa.best_y_history).cummin(axis=0))
# plt.show()

def print_result(best_x):
    w = 4 * 1852
    h = 5 * 1852
    angle = best_x[0]
    lst = best_x[1::]
```

```
    lyt = Layout(angle, lst, w, h)
    length_all = lyt.get_length()
    num = lyt.n
    area = lyt.get_coverage_area(100)
    Area_ratio = area / (h * w)
    threshold_ratio = 0.2
    step_length = 100
    overlap_length = lyt.get_overlap_length(threshold_ratio, step_length)

    print("num ={:3} alpha ={:6.2f}° length_all ={:11.2f}m area_ratio = {:6.4f} overlay_length
        = {:11.2f}".format(lyt.n, degrees(angle), length_all, Area_ratio, overlap_length))

'''
w = 4 * 1852 # 矩形海域跨度 (x轴正半轴)
h = 5 * 1852 # 矩形海域长度 (y轴正半轴)
angle = best_x[0] # 输入角度, 将对应弧度赋值给angle
n = len(best_x)-1
lst = best_x[1:n+1]
num = 1000
n = len(lst)
# # 创建测线布局 lyt
lyt = Layout(angle, lst, w, h)

# 总测线长度get_length_all()计算测试
# print("测线总长度: ", lyt.get_length())
Tot_length = lyt.get_length()

# 测试布局中测线参数计算及保存
# for line in lyt.lines:
#     print(line)

# # 测试get_adjacent_line_indices
# print(lyt.get_adjacent_line_indices(1, 3.9))

# 测试覆盖面积
print(f"覆盖面积比例 = {lyt.get_coverage_area(20)/(20*1852*1852)} %")
#Area_ratio = lyt.get_coverage_area(50)/(20*1852*1852)

# 测试重叠长度
threshold_ratio = 0.2
step_length = 10
over_length = lyt.get_overlap_length(threshold_ratio, step_length)/1000
print(f"重叠长度 = {over_length} km")
'''

def output_cal(output):
    for lyt in output:
```

```
        length_all = lyt.length_all
        coverage_ratio = lyt.coverage_ratio
        n = lyt.n
        lst = lyt.lst
        alpha = lyt.alpha
        print("n ={:3}  alpha ={:6.2f}° length_all ={:11.2f}m area_ratio =
            {:6.4f}".format(lyt.n, degrees(alpha), length_all, coverage_ratio))
#output_cal(output)

for lyt in output:
    length_all = lyt.length_all
    coverage_ratio = lyt.coverage_ratio
    plt.scatter(coverage_ratio,length_all,marker='o', edgecolors='c', facecolors='none',s=100)
    plt.xlim([0.75,1.01])
    plt.ylim([0,1000000])
    # plt.xlabel("覆盖面积比例")
    # plt.ylabel("测线总长度")
cnt=0
Output=[]
for lyt in output:
    if lyt.coverage_ratio > 0.9:
        Output.append(lyt)
        cnt=cnt+1

Output.sort(key = lambda x: x.length_all)
# for lyt in Output:
#        length_all = lyt.length_all
#        coverage_ratio = lyt.coverage_ratio
#        n = lyt.n
#        lst = lyt.lst
#        alpha = lyt.alpha
#        print("n ={:3}  alpha ={:6.2f}° length_all ={:11.2f}m area_ratio =
    {:6.4f}".format(lyt.n, degrees(alpha), length_all, coverage_ratio))
# #output_cal(output)
Output[0].lst
```