# 多波束测深系统模型的构建与研究


## 摘　要：

多波束测深系统是一种用于海洋、湖泊等水域深度测量的技术，解决了传统的单波束测深的数据分布不均效率低等弊端。本文主要研究了不同情况下测量覆盖宽度和相邻条带重叠率的数学模型，对带约束的具体探测问题做了测线的规划，并基于**机理分析、*k*-近邻算法、主成分分析**等方法辅助进行分析和求解。

**针对问题一：**主要建立了在海底存在固定坡度且航线与坡边线平行时，多波束测深的覆盖宽度及相邻条带之间重叠率的数学模型，本文首先根据题目背景，对测线距中心点处的距离、波束测深的覆盖宽度和相邻条带之间重叠率进行了定义，利用三角学进行分析的方法对图进行简化处理，并在针对给定多波束换能器的开角、坡度、中心点海水深度的特定地理环境，利用该模型求解得出不同距离下覆盖宽度和重叠率结果，并填入 *Result1.xlsx* 中；随后我们对结果进行了定性分析，得出**覆盖宽度和重叠率随测线距中心点处的距离呈现单调下降**的关系。

**针对问题二：**主要解决考虑航线夹角时，建立覆盖宽度及相邻条带之间重叠率的数学模型的问题。考虑到航线夹角的存在，为了与问题一的模型联系，本文首先引入**等效坡度**的概念，利用机理分析得出等效坡度和航线夹角、实际坡度的关系式，得出新的数学模型。并在针对给定多波束换能器的开角、坡度、中心点海水深度的特定地理环境，利用该模型求解得出不同距离与不同航线夹角下覆盖宽度和重叠率结果，并填入 *Result2.xlsx* 中；随后我们对结果进行了定性分析，发现**覆盖宽度和距离呈线性单调变化**的关系。

**针对问题三：**主要解决针对特定矩形海域的最优测线规划问题，本文首先利用问题二的模型对固定深度覆盖宽度和测线方向的定量关系，得出当航线与坡边线**平行**时，覆盖宽度最大，作为简化模型。然后将测线方向夹角作为变量，建立了**双变量非线性最优化模型**，并利用**贪心算法，使用 Python 编程**，比较两个方案的结果，发现当航线与坡面平行时，测量总长度最短，**测量长度最短为68NM（125936m）**，此时相邻条带之间的重叠率满足条件。

**针对问题四：**主要解决针对海底面不规则的特定矩形海域的最优测线规划问题。本文希望将曲面转化成一个个小的固定坡度的平面，然后利用问题三的模型进行求解。首先利用了点云的概念对地理数据进行数学抽象，然后利用 ***k*-近邻算法**选取可组成小平面的一组点，对这些点做 PCA 分析，得到该小平面的坡度，将问题四转换成沿用问题三的子问题，最后将测线总长度、重叠率作为优化对象，覆盖范围作为约束条件，利用模型三做**非线性最优化，**使用优化算法，最后得到当接受**最多相邻条带重叠率不超过**30%**时**，东西方向测距效果最好，测线的**总长度为**296320.00，**漏测**海区占总待测海域面积的**百分比为**82%；在重叠区域中，重叠率**超过 20% 部分的总长度为**26335.44。

最后，我们对模型进行了灵敏性分析，说明了所建立的模型在对系统施加一定**扰动**后具有一定的稳定性和抗干扰性。分析了模型的优缺点，并针对缺点提出相应的改进方案，最后说明该模型的可推广性。



**关键词：多波束测深、路径规划、机理分析、点云、PCA 分析**

# 问题重述

## 1.1 问题背景

多波束测深系统是一种用于海洋、湖泊等水域深度测量的技术，能同时发出并接收多条水底的声波信号，通过测量声波的传播时间和回波的强度，系统可以同时获取多个点的深度信息，可以从而提供更全面、高分辨率的水深数据，有效地避免了传统的单波束测深系统只能获取一个点的深度信息的弊端。在海洋调查、海底地形测绘、航海安全等领域具有广泛的应用[1]。

对特定海域进行多波束测深的覆盖率和重叠率是实际应用时的重要参数，此外，测线总长度还要求尽可能短以节约探测成本。因此建立覆盖宽度和重叠率的模型，并根据给出对特定海域的测深方案具有重要意义。

## 1.2 需要解决的问题

**问题一：**简化多波束测深系统，分析覆盖宽度和相邻条带之间的重叠率和其他参数的数学关系，根据数学关系给出测线相互平行且海底地形存在一定坡度时的覆盖宽度和相邻条带之间的重叠率的模型；并根据模型求解当多波束换能器的开角为120°，坡度为1.5°，海域中心点处的海水深度为70 $m$时不同测量位置的相关指标，填入表中。

**问题二：**在问题一的基础上，考虑测线方向，完善海底地形存在一定坡度时的多波束测深覆盖宽度模型；并根据模型求解当多波束换能器的开角为120°，坡度为1.5°，海域中心点处的海水深度为120 $m$时不同测量位置的相关指标，填入表中。

**问题三：**在航船实际测量时，考虑到覆盖效率和成本问题，测量长度越短越好，覆盖率需要保持在一定范围。考虑如下场景：海域为西深东浅的、有1.5°坡度的矩形坡面，南北长 2 海里、东西宽 4 海里，并且海域中心点处的海水深度为110 $m$。测量船的多波束换能器的开角为120°，需要设计出一条最优搜索路线，要求测量长度最短、可完全覆盖整个待测海域的测线，同时需要满足相邻条带之间的重叠率为10%～20%的条件。

**问题四：**在给定的单波束测量的测深数据的情况下，重新设计测线。要求：

a) 扫描条带尽可能地覆盖整个待测海域
b) 相邻条带之间的重叠率尽量控制在 20% 以下
c) 测线的总长度尽可能短

计算重新设计的测线指标：

✧ 测线的总长度；
✧ 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。

# 二、 问题分析

## 2.1 问题总分析

首先，我们建立了**航线平行坡边线覆盖宽度和重叠率的数学模型**，求解了问题相关的问题；其次在问题二中，引入航线夹角的概念，建立了**考虑航线夹角时覆盖宽度和重叠率的数学模型**，并求解了相关问题；针对问题三，分别定性和定量的设计了两个模型：**沿坡边线测线模型和一般模型。**最后引入**点云**的概念，将曲面化成一个个小平面，利用问题三的模型，进行求解。

本文的总体分析流程图如下:

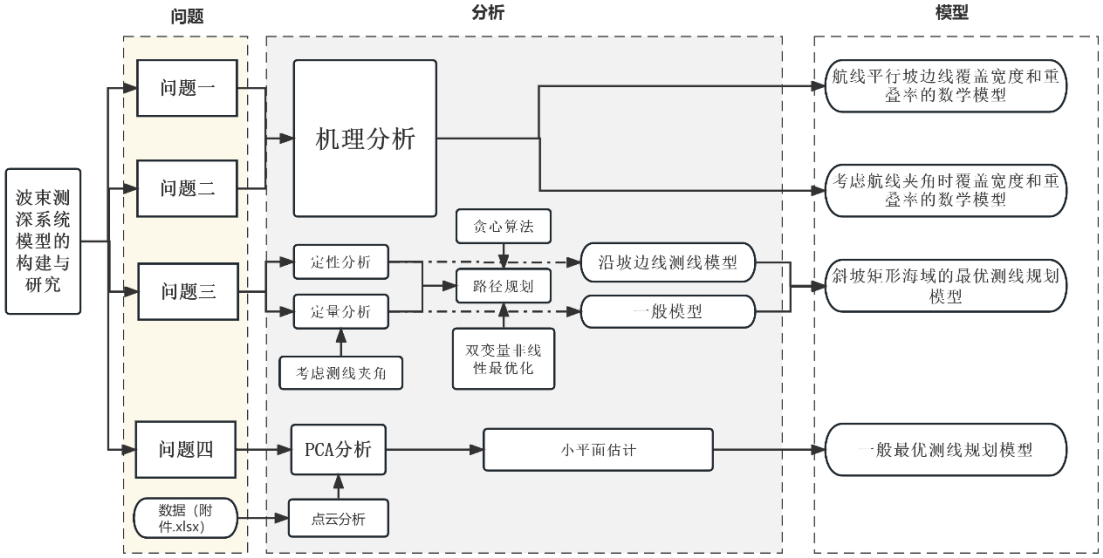


**图 1 问题的总分析**

## 2.2 具体问题分析

### 2.2.1 问题一的分析

根据题目要求，首先要建立建立多波束测深的覆盖宽度及相邻条带之间重叠率的数学模型。而题中给出的相关定义是基于测线相互平行且海底地形平坦时的条件，问题一考虑了海底的坡度，因此需要建立一个在坡度海底地形下的关于多波束测深的覆盖宽度及相邻条带之间重叠率的合理定义，并在此基础上进行求解。

当测线相互平行且海底地形平坦时，如图2所示，覆盖宽度及相邻条带之间重叠率仅仅是关于多波束换能器的开角 $\theta$、探测点海水深度 $D$ 和相邻两条测线的间距 $d$ 的函数。 但是当海底地形有一定坡度 $\alpha$ 时，如图 3，还需将坡度 $\alpha$ 作为变量考虑进模型中。

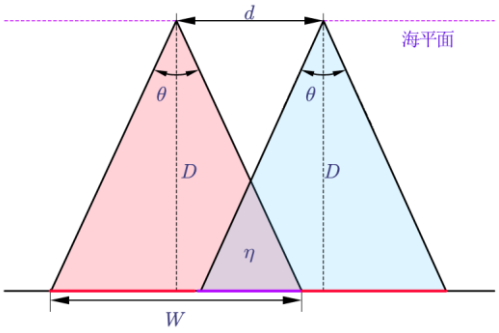


**图 2 测线相互平行且海底地形平坦时**

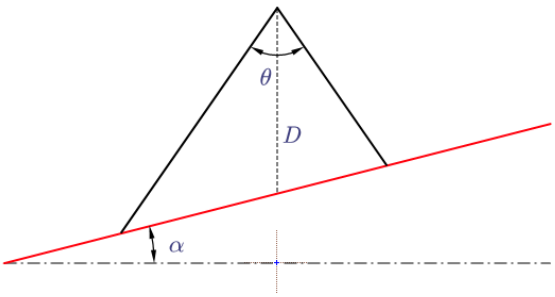


**图 3 测线相互平行且海底地形有坡度时**

因此需要建立以覆盖宽度及相邻条带之间重叠率为因变量，以坡度、多波束换能器的开角、探测点海水深度、和相邻两条测线的间距为变量的函数方程。

然后对这样一种具体情况：多波束换能器的开角为120°，坡度为1.5°，海域中心点处的海水深度为70$m$，对上述模型做实际应用，得出结果。

### 2.2.2 问题二的分析

问题二要求考虑测线方向，完善海底地形存在一定坡度时的多波束测深覆盖宽度模型。该模型与问题一要求不同的是相对探测船测线方向的海底坡度发生了变化，是一个关于测线方向与海底坡面的法向在水平面上投影的夹角 $\beta$ 和地理坡度 $\alpha$ 的函数，

因此首先应该利用几何知识得到等效坡度，然后需要求解测线深度随测点离中心点距离的变化关系，这里需要规定一个距离变化的正方向。利用模型一，根据等效坡度，求出新的覆盖宽度模型。

然后对这样一种具体情况：多波束换能器的开角为120°，坡度为1.5°，海域中心点处的海水深度为120$m$，对上述模型做实际应用，得出结果。

### 2.2.3 问题三的分析

问题三要求在一个给定的固定坡面海域，设计一组测量长度最短、可完全覆盖整个待测海域的测线，且相邻条带之间的重叠率满足10%～20%的要求。这是一个非线性规划最优化问题：

- **优化目标：**测量长度最短；
- **约束条件：**覆盖整个待测海域”和“相邻条带之间的重叠率满足；
- **优化函数：**问题二的模型

待测范围一定时，每次测量范围越大，测量长度越短。首先利用问题二的模型可以得出测量范围与测线方向和测点深度的关系，根据此关系确定航线最优方向。

为了进一步说明该方案的准确性，将测线方向作为变量加入模型，将问题转化成为双变量的非线性规划问题，进而得出最优化的出最佳航线。

最后将简单模型和复杂模型的结果进行对比，说明该方案的准确性。

### 2.2.4 问题四的分析

问题四要求在给定的单波束测量的测深数据的情况下，重新设计测线，要求覆盖足够广、重叠率尽量控制在10%～20%、测线足够短。由于此问题中海底坡面不再固定，不同点对应的坡面是不同的，为了利用问题三中的已经建立的模型，我们拟使用数据点的分布估计一个平面并求出该平面的坡度，将整体分割为一块块固定坡度的小平面，利用问题三的模型加以求解。估计一个平面需要一个点和一个法向量，该问题的难点在于如何求解数据点在曲面的法向量，我们拟采用点云的概念，使用主成分分析法（Principal Component Analysis, PCA）来估计点云法向量，接着利用第三问的模型加以求解。

## 三、 模型假设

1. **海水密度和介质均相同，声波以直线转播且能原路返回，且忽略声波在海底传递时间**[2]；
   若海水密度和介质存在变化，那么多波束测量将不能简化成等腰三角的放射状简化图，给建立模型带来不必要的麻烦。

2. **只考虑测量船前行状态的多波束测深的测量宽度和重叠率，测量船在转弯（或者掉头）处的测量宽度和重叠率不加以考虑；**
   测量船在转弯（或者掉头）处的测量宽度和重叠率难以计算并且占比很小，不予考虑能够降低计算难度并且对最终结果影响不大。

3. **假设待测海域地形在被测点很小的区域范围内能被近似为一片相同坡度的平面，坡度由该点处的法向量所决定。**
   将数据分为多个小平面，既可以继承已有模型，又可以在保证精度的条件下大大简化计算。

## 四、 符号说明

本文所用符号见表 1

**表 1 符号说明**

| 符号 | *rad* 说明 | 单位 |
|---|---|---|
| $d_n$ | 第 $n$ 条测线与前一相邻测线的水平间距 | *m* |
| $l_n$ | 第 $n$ 条测线测线距中心点处的距离 | *m* |
| $r_n$ | 第 $n$ 条测线测线距坡边线处的距离 | *m* |
| $len_{i,j}$ | 代表第 $i$ 个区域内第 $j$ 条行程的测线长度 | *m* |
| $D_n$ | 第 $n$ 条测线距海水深度 | *m* |
| $W_n$ | 第 $n$ 条测线的覆盖宽度 | *m* |
| $\Delta_n$ | 第 $n$ 条测线与前一相邻测线覆盖宽度的重叠宽 $x_n$ 度 | *m* |
| $x_n$ | 第 $n$ 条测线左端点距离船在海底的投影点的距离 | *m* |
| $y_n$ | 第 $n$ 条测线右端点距离船在海底的投影点的距离 | *m* |
| $\gamma_n$ | 第 $n$ 条测线的左覆盖比 | -- |
| $\eta_n$ | 第 $n$ 条测线与前一相邻测线的重叠率 | -- |
| $\alpha$ | 海底坡度 | *rad* |
| $\beta$ | 测线方向与海底坡面的法向在水平面上投影的夹角 | *rad* |
| $\theta$ | 多波束换能器的开角 | *rad* |
| *Length* | 海域南北长度 | *m* |
| *Width* | 海域东西宽度 | *m* |
| $L_{all}$ | 总测量长度 | *m* |
| $l_{slope}$ | **坡边线**：坡面和海底平面的交线 | -- |
| $rate_{missing}$ | 漏测海区占总待测海域面积的百分比 | -- |
| $rate_{cover}$ | 覆盖率 | -- |

# 五、 模型建立与求解

## 5.1 问题一的模型：航线平行坡边线覆盖宽度和重叠率的数学模型

### 5.1.1 模型建立

根据问题描述，建立探测船探测海底为坡度 $\alpha$ 的斜面时，多波束测深相邻条带的示意图，如下图所示，图中各个参数的定义参见表 1。

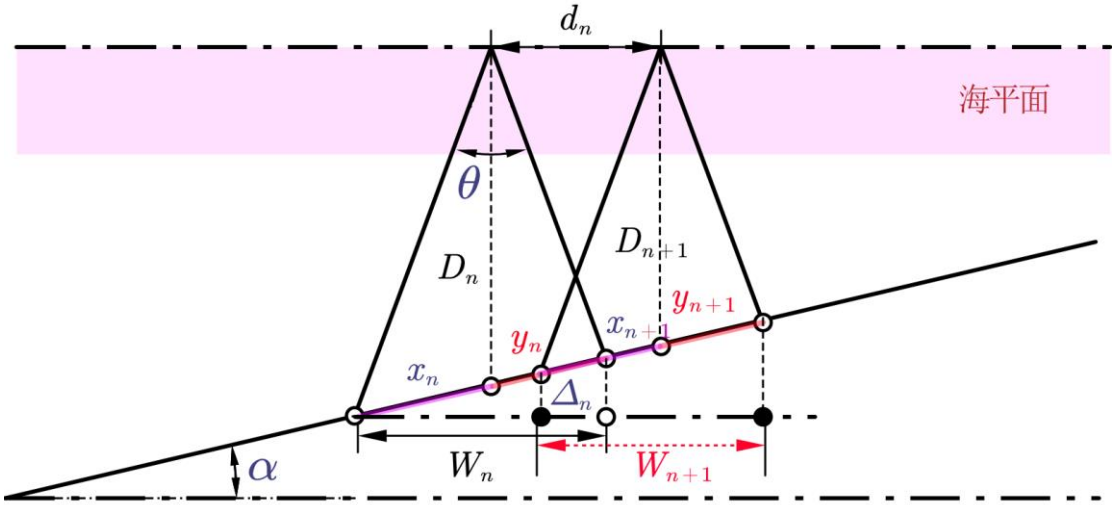


**图 4 相邻条带示意图**

题目要求根据给出的测线相互平行且海底地形平坦时条带的覆盖宽度 $W$ 和相邻条带之间的重叠率 $\eta$ 定义，给出测线相互平行且海底地形存在 $\alpha$ 坡度时的覆盖宽度 $W$ 和相邻条带之间的重叠率 $\eta$ 新的定义，并根据图 4 建立多波束测深的覆盖宽度及相邻条带之间重叠率的数学模型。根据题意，覆盖宽度 $W$ 和相邻条带之间的重叠率 $\eta$ 是关于坡度 $\alpha$ 、多波束换能器的开角 $\theta$ 和探测点海水深度 $D$ 的函数，重叠率 $\eta$ 还和相邻两条测线的间距 $d$ 有关，即：

$$\begin{cases} W_n = f\left(\alpha,\theta,D_n\right) \\ \eta_n = g\left(\alpha,\theta,D_n,d\right) \end{cases}$$

### 5.1.2 模型求解

由图 4 可以得知，海水深度 $D$ 是关于第 $n$ 条测线测线距中心点处的距离，问题一中第 $n$ 条测线与前一相邻测线的水平间距 $d_n$ 恒等于 $200m$，记为 $d$ 。**规定测线测线距中心点处的距离 $l$ 往测线方向左侧为正**，容易得到

$$D_n = 70-(n-5)d\cdot\tan\alpha$$

✧ **覆盖宽度 $W_n$ ：**

根据题意给出覆盖宽度 $W_n$ 的定义：第 $n$ 条测线在海底探测的范围在与测线方向垂直的平面的投影。

那么可以得到，第 $n$ 条测线的覆盖宽度如下所示

$$W_n = \left(x_n + y_n\right)\cos\alpha \tag{5.1}$$

根据正弦定理：

$$\begin{cases}\dfrac{x_n}{\sin\left(\dfrac{\theta}{2}\right)}=\dfrac{D_n}{\sin\left(\dfrac{\pi}{2}-\dfrac{\theta}{2}-\alpha\right)}\\[2ex]\dfrac{y_n}{\sin\left(\dfrac{\theta}{2}\right)}=\dfrac{D_n}{\sin\left(\dfrac{\pi}{2}-\dfrac{\theta}{2}+\alpha\right)}\end{cases}$$

可以得到$x_n$， $y_n$关于坡度$\alpha$、多波束换能器的开角$\theta$和探测点海水深度$D$的函数表达：

$$\begin{cases}x_n=\dfrac{\sin\dfrac{\theta}{2}}{\sin\left(\dfrac{\pi}{2}-\dfrac{\theta}{2}-\alpha\right)}D_n\\[2ex]y_n=\dfrac{\sin\dfrac{\theta}{2}}{\sin\left(\dfrac{\pi}{2}-\dfrac{\theta}{2}+\alpha\right)}D_n\end{cases}\tag{5.2}$$

根据(5.1)和(5.2)整理可以得到覆盖宽度$W_n$的表达式：

$$\begin{aligned}W_n&=\frac{\sin\theta\cos^2\alpha}{\cos\left(\dfrac{\theta}{2}+\alpha\right)\cos\left(\dfrac{\theta}{2}-\alpha\right)}D_n\\&=\frac{\sin\theta\cos^2\alpha}{\cos\left(\dfrac{\theta}{2}+\alpha\right)\cos\left(\dfrac{\theta}{2}-\alpha\right)}\cdot[70-(n-5)d\cdot\tan\alpha]\end{aligned}\tag{5.3}$$

**(1) 相邻条带之间重叠率$\eta_n$：**

首先给出相邻测线重叠宽度$\Delta_n$的**定义**：第$n$条测线与前一相邻测线测量范围的重叠部分在与测线方向垂直的平面的投影。

由图4可以得到，第$n+1$条测线与前一相邻测线覆盖宽度的重叠宽度$\Delta_n=(x_{n+1}+y_n)\cos\alpha-d$。

根据材料定义第$n$条测线与前一相邻测线的重叠率$\eta_n$为第$n$条测线与前一相邻测线覆盖宽度的重叠宽度与第$n$条测线的覆盖宽度的比值，则

$$\begin{aligned}\eta_n&=\frac{\Delta_n}{W_n}\times100\%\\&=\frac{(x_{n+1}+y_n)\cos\alpha-d}{W_n}\times100\%\end{aligned}\tag{5.4}$$

**当计算得到$\eta<0$时，强制令$\eta=0$。**

利用上述模型多波束换能器的开角为120°，坡度为1.5°，海域中心点处的海水深度为70 $m$时的相关指标值，并记录于下表：

**表2 问题一结果**

| 测线距中心点处的距离/m | −800 | −600 | −400 | −200 | 0 | 200 | 400 | 600 | 800 |
|---|---|---|---|---|---|---|---|---|---|
| 海水深度 | 90.95 | 85.71 | 80.47 | 75.24 | 70 | 64.76 | 59.53 | 54.29 | 49.05 |

| /m | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 覆盖宽度/m | 315.71 | 297.53 | 279.35 | 261.17 | 242.98 | 224.81 | 206.63 | 188.45 | 170.27 |
| 与前一条测线的重叠率/% | — | 33.64 | 29.59 | 25.00 | 19.78 | 13.78 | 6.81 | 0 | 0 |

**注：均保留两位小数**

并将表格保存到 result1.xlsx 文件中。

### 5.1.3 模型分析

将测线距中心点处的距离步长取小，应用问题一的模型，使用程序绘图得到如下的数据图：

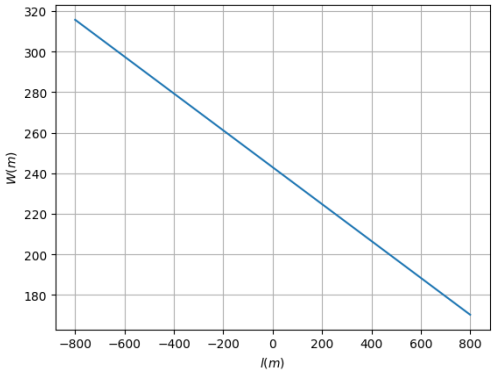


**图 5 覆盖宽度随测线距中心点处的距离变化**

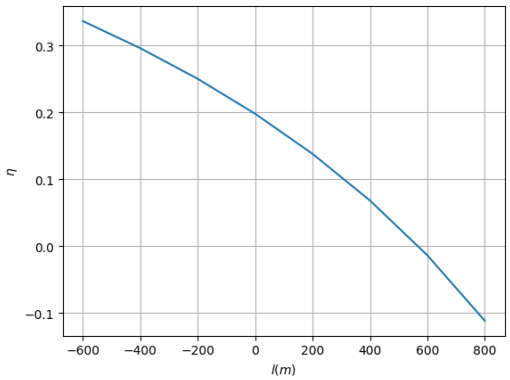


**图 6 重叠率随测线距中心点处的距离变化**

图 5 描述了覆盖宽度 $W$ 随测线距中心点处的距离 $l$ 变化的趋势，由图可以知道，覆盖宽度随测线距中心点处的距离呈现**线性单调下降的关系**。

分析原因：测线距中心点处的距离越大，测绘点海水深度线性单调下降，又因为当其他参数不变时，覆盖宽度 $W$ 和海水深度 $D$ 是正比例关系的，所以覆盖宽度 $W$ 随测线距中心点处的距离 $l$ 呈现线性单调下降。

图 6 描述了与前一条测线的重叠率 $\eta$ 随测线距中心点处的距离 $l$ 变化的趋势，由图可以知道，覆盖宽度随测线距中心点处的距离呈现**单调加速下降的关系**。

分析原因：测线距中心点处的距离越大，测绘点海水深度线性单调下降，又因为当其他参数不变时，重叠率 $\eta$ 和海水深度 $D$ 呈非线性正比关系，所以重叠率 $\eta$ 随测线距中心点处的距离 $l$ 呈现线性单调下降。

## 5.2 问题二的模型：考虑航线夹角时覆盖宽度和重叠率的数学模型

### 5.2.1 模型建立

问题二与问题一的不同在于增加了测线方向与海底坡面的法向在水平面上投影的夹角 $\beta$ 这一个变量。

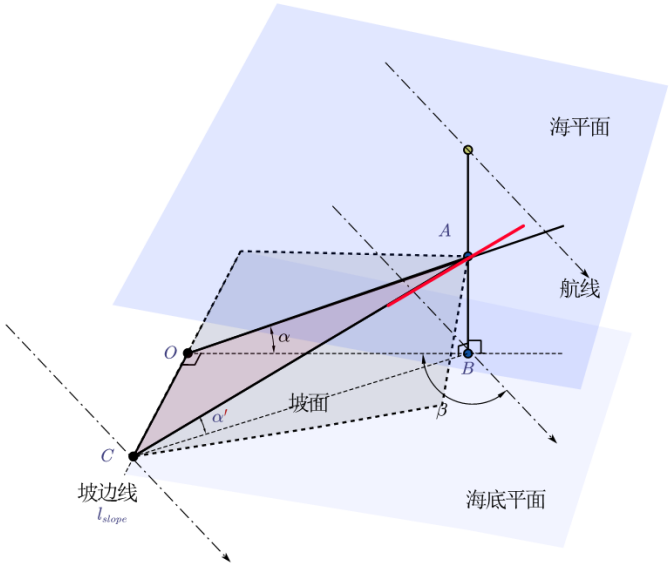


**图 7 问题 2 的示意图**

第一问已经得出多波束测深覆盖宽度与坡度 $\alpha$ 等变量的模型，为了利用这一模型，本文将 $\beta$ 和 $\alpha$ 综合考虑，获得船航行的海底等效坡度 $\alpha'$，再利用第一问得出的模型求解多波束测深覆盖宽度 $W_n$

首先根据三角关系求解海底等效坡度 $\alpha'$，求解简化图如下

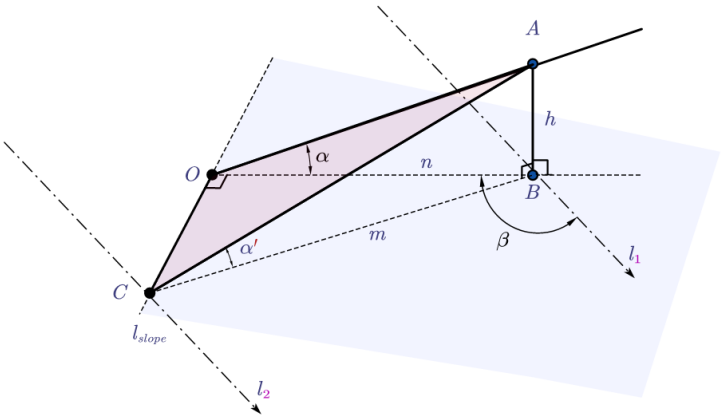


**图 8 求解等效坡度的简化图**

其中 A 点为铅垂测线与坡面的交点、AOB 代表坡面的一个平行截面，则 $\angle AOB=\alpha$，AOC 代表坡面的一部分，$AB\perp$ 水平面（因此 $AB\perp OB$），$l_1$ 为航线在水平面的投影，如图所示，$l_1$ 和 OB 夹角为 $\beta$，过 A 点做测线的延长，坡边线 $l$ 于 C 点，则 $\angle ACB=\alpha'$，作 $l_2$ 平行于 $l_1$ 并经过 C 点，设 $AB=h,CB=m,OB=n$。

### 5.2.2 模型求解

分析图 8，在 $\Delta ABC$ 中：

$$\tan\alpha'=\frac{h}{m} \tag{5.5}$$

在 $\Delta AOB$ 中：

$$\tan\alpha=\frac{h}{n} \tag{5.6}$$

在 $\Delta BOC$ 中，当 $0<\beta<\pi$ 时，$\cos\left(\beta-\frac{\pi}{2}\right)=\frac{n}{m}$；当 $\pi<\beta<2\pi$ 时 $\cos\left(\frac{3\pi}{2}-\beta\right)=\frac{n}{m}$，即：

$$\frac{n}{m}=\begin{cases}\sin\beta, 0<\beta<\pi\\ -\sin\beta, \pi<\beta<2\pi\end{cases}=|\sin\beta| \tag{5.7}$$

联立上式得：

$$\alpha'=\arctan\left(\tan\alpha|\sin\beta|\right) \tag{5.8}$$

另外，在问题二中第 $n$ 条测线距海水深度 $D_n$ 也要相应变化，**规定测线测线距中心点处的距离 $l$ 往测线方向左侧为正**：

$$D_n=\begin{cases}120-(n-1)d\cdot\tan\alpha', 0<\beta<\pi\\ 120+(n-1)d\cdot\tan\alpha', \pi<\beta<2\pi\end{cases}=120-(n-1)d\cdot\tan\alpha\sin\beta \tag{5.9}$$

将 $\alpha'$ 代替 $\alpha$ 带入(5.3)得到多波束测深覆盖宽度的数学模型：

$$W_n=\frac{\sin\theta\cos^2\alpha'}{\cos\left(\frac{\theta}{2}+\alpha'\right)\cos\left(\frac{\theta}{2}-\alpha'\right)}\cdot D_n \tag{5.10}$$

接下来利用模型解决实际问题，令 $\theta=\frac{2}{3}\pi, \alpha=1.5^\circ$。

根据模型测算，记录于下表：

**表 3 问题二结果**

| 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 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 |
| | 45 | 416.12 | 380.45 | 344.77 | 309.10 | 273.42 | 237.75 | 202.08 | 166.40 |
| | 90 | 416.55 | 366.05 | 315.54 | 265.04 | 214.54 | 164.04 | 113.53 | 63.03 |
| | 135 | 416.12 | 380.45 | 344.77 | 309.10 | 273.42 | 237.75 | 202.08 | 166.40 |
| | 180 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 | 415.69 |
| | 225 | 416.12 | 451.79 | 487.47 | 523.14 | 558.82 | 594.49 | 630.16 | 665.84 |
| | 270 | 416.55 | 467.05 | 517.55 | 568.06 | 618.56 | 669.06 | 719.57 | 770.07 |
| | 315 | 416.12 | 451.79 | 487.47 | 523.14 | 558.82 | 594.49 | 630.16 | 665.84 |

**注：均保留两位小数**

并将表格保存到 result2.xlsx 文件中。

### 5.2.3 模型分析

将 $\beta$ 步长取小，应用问题二的模型，以测线方向夹角 $\beta$ 为 $x$ 坐标，测量船距海域中心点处的距离 $l$ 为 $y$ 坐标，覆盖宽度 $W$ 为 $z$ 坐标，得出距离 $l$、夹角 $\beta$ 和覆盖宽度 $W$ 的三维数据图：

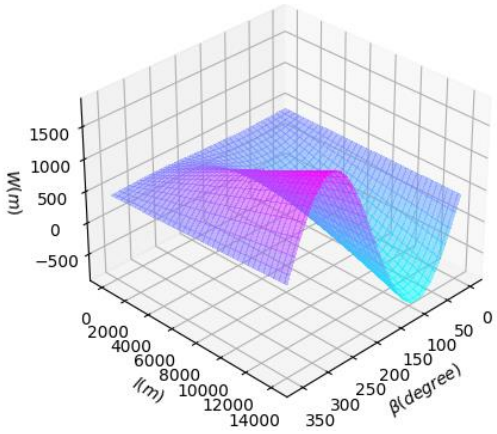


**图 9 覆盖宽度与航线夹角、中心点距离的关系**

由图 9 可知，对于固定的夹角 $\beta$，当 $0<\beta<\pi$ 时，覆盖宽度 $W$ 和距离 $l$ 呈线性单调下降的关系，见图 10，当 $\pi<\beta<2\pi$ 时，覆盖宽度 $W$ 和距离 $l$ 呈线性单调上升的关系，见图 11；对于固定的距离 $l$，覆盖宽度 $W$ 和夹角 $\beta$ 呈正弦函数的关系。

分析原因：当夹角 $\beta$ 固定时，根据(5.8)得知 $\alpha'$ 不变，代入(5.10)得知 $W_n$ 和 $D_n$ 成正比，因为上述规定测线测线距中心点处的距离 $l$ 往测线方向左侧为正，因此当 $0<\beta<\pi$ 时，$D_n$ 和距离 $l$ 成线性单调下降关系；当 $\pi<\beta<2\pi$ 时，$D_n$ 和距离 $l$ 成线性单调上升关系，因此覆盖宽度 $W$ 如图 9 变化；当距离 $l$ 固定时，通过代入化简，得到 $W_n$ 和 $\sin\beta$ 成比例关系，因此覆盖宽度 $W$ 和夹角 $\beta$ 呈正弦函数的关系。

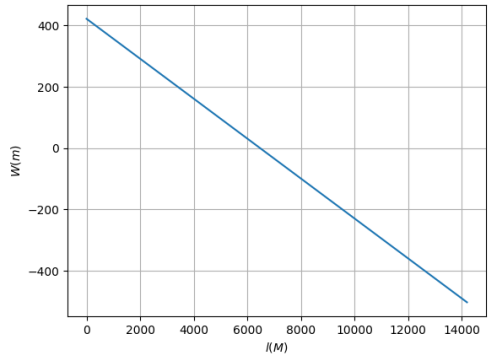


**图 10 $0<\beta<\pi$ 时覆盖宽度随距离线性下降**

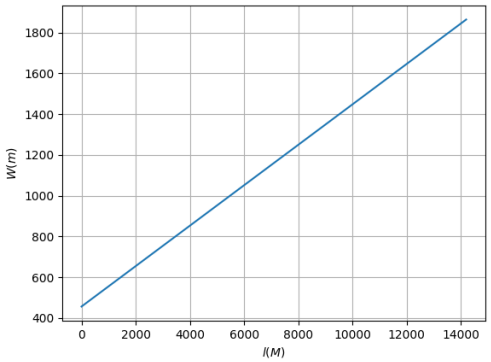


**图 11 $\pi<\beta<2\pi$ 时覆盖宽度随距离线性上升**

## 5.3 问题三的模型：斜坡矩形海域的最优测线规划模型

### 5.3.1 模型准备

受到问题二启发，覆盖面积和测线方向角 $\beta$ 有关，因此利用模型二绘制出覆盖宽度和测线距海水深度、测线方向角之间的图像，如下图所示：

由图 12 可以得知，对固定的海水深度，测线方向角 $\beta=\frac{\pi}{2},\frac{3\pi}{2}$ 时，覆盖宽度最大，参考图 8 可以得知，当测量船沿着海底坡边线方向行驶时，在问题三中即沿着南北方向时，可以达到最大的覆盖宽度（见图 13）。称一次单向测量过程为**一个行程，记作 $R$**，则问题三的思路在于船只沿着南北方向测量时在满足题目约束的条件下获得最小的行程数目的测量方案。

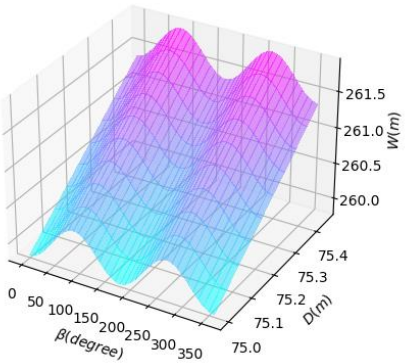


**图 12 覆盖宽度随海水深度、测线方向角的变化**

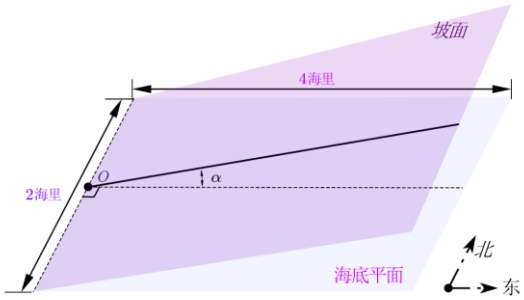


**图 13 问题三简化图**

另外，仅仅通过上述的定性分析说服力度还不够，为了进一步说明一般情况，我们进一步完善模型，将测线方向夹角 $\beta$ 作为另外一个变量加入到优化函数中，将模型变成双变量的非线性优化模型。利用几何知识得到行程总长度 $L_{all}$ 关于测线间距 $d$ 和测线方向夹角 $\beta$ 的表达式 $L_{all}=f(d,\beta)$ 并将其视作优化函数。

为了求解该模型，需要充分利用问题三中给出的各个约束：

A. 矩形海域尺寸约束：要求测线不得超过南北长 2 海里、东西宽 4 海里的矩形海域。
B. 完全覆盖约束：相邻测线必须重叠完全覆盖海域
C. 重叠率约束：重叠率 $\eta$ 满足 $10\%\sim 20\%$

- 另外，为了简化计算作出合理假设：**测量船在给定测量方向只走直线**。

### 5.3.2 模型建立

- **沿坡边线测线模型**

该模型为简化模型，只考虑 $\beta=\frac{\pi}{2},\frac{3\pi}{2}$ 的情况。

由于每次行程为南北走向，根据问题二可以得知，当一个行程的覆盖宽度均相等，容易得到：

*每个行程的总覆盖宽度 = 海域南北范围 × 该行程每个测量点的覆盖宽度*

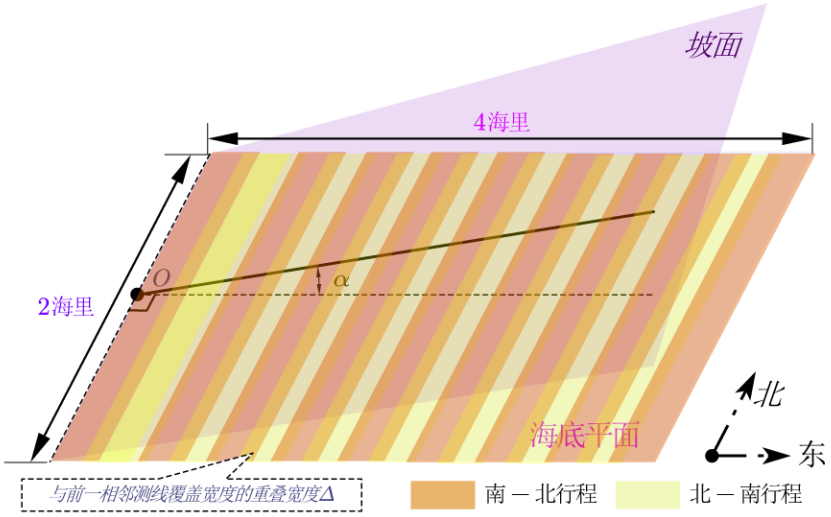


**图 14 测量船航行投影示意图**

图 14 表示了问题三中测量船航行投影示意图，由图可知，只需取坡边线的垂面截

取海底的一个截面求解问题即可，如图 15 所示：

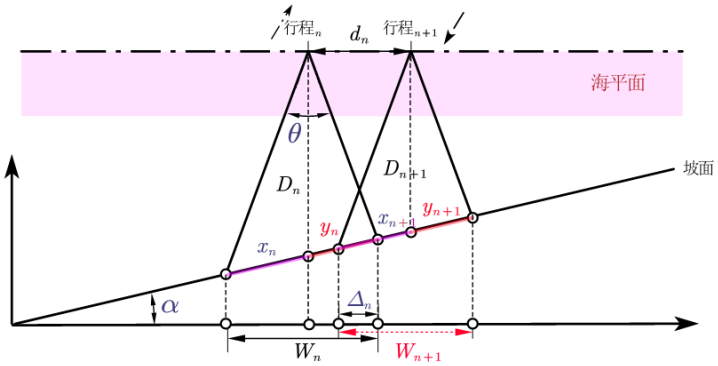


**图 15 海底截面示意图**

设行程的总数为 $N$，定义 $\gamma=\dfrac{x_n}{W_n}=\dfrac{\cos\left(\dfrac{\theta}{2}+\alpha'\right)}{\cos\left(\dfrac{\theta}{2}+\alpha'\right)+\cos\left(\dfrac{\theta}{2}-\alpha'\right)}$ 为第 $n$ 条测线的**左覆盖比**。

**模型二：考虑航线夹角时覆盖宽度和重叠率的数学模型**

$$\begin{cases} W_n=\dfrac{\sin\theta\cos^2\alpha'}{\cos\left(\dfrac{\theta}{2}+\alpha'\right)\cos\left(\dfrac{\theta}{2}-\alpha'\right)}\cdot D_n \\ \eta=\dfrac{\Delta_n}{W_n} \end{cases}$$

其中：
$$\begin{cases} \alpha'=\arctan\left(\tan\alpha\left|\sin\beta\right|\right) \\ D_n=207-r_n\cdot\tan\left(1.5^\circ\right)=207-0.0262r_n \\ \Delta_n=y_i\tan\left(1.5^\circ\right)-3700\tan\left(1.5^\circ\right)+110 \\ \quad\ =0.0262y_i+13.06 \end{cases}$$

通过 Python 求解简化模型可以得出行程数 **N = 34**，总测线长度

$$L_{all}=68NM=125936m \tag{5.11}$$

- **任意方向测线模型**

该模型将测线方向夹角 $\beta$ 作为变量加到上述简单模型中，将模型变成双变量的非线性优化模型。

考虑问题中的各个约束条件，问题三需要解决下列最优化问题：

**模型三：基于模型二的非线性最优化模型**

$$\min\ L_{all}=f(d,\beta) \tag{5.12}$$

$$\begin{cases} \sum_{i=1}^{n}W_i-\sum_{i=1}^{n}\Delta_i\geq L \\ r_1-W_1\times\gamma\geq 0 \qquad ,i=1\ldots N \\ r_N+W_N\times(1-\gamma)\geq L \\ \eta_i\in[10\%,20\%] \end{cases} \tag{5.13}$$

首先求解 $L_{all}=f(d,\beta)$ 的表达式，由于假设了船在一个方向上直行，将问题简化

得出图 16，由于测线长度在不同区域内计算公式不同，故其中将矩形区域分为了三个区域（用双点划线作为分界线隔开并不同颜色表示），方便求解每条测线的长度；另外，图中带颜色的箭头直线表示测线在平面上的投影，蓝色代表往程，红色代表返程。

那么问题转化为求解 $L_{all}=\sum_{i=1}^{N}L_i$ 。

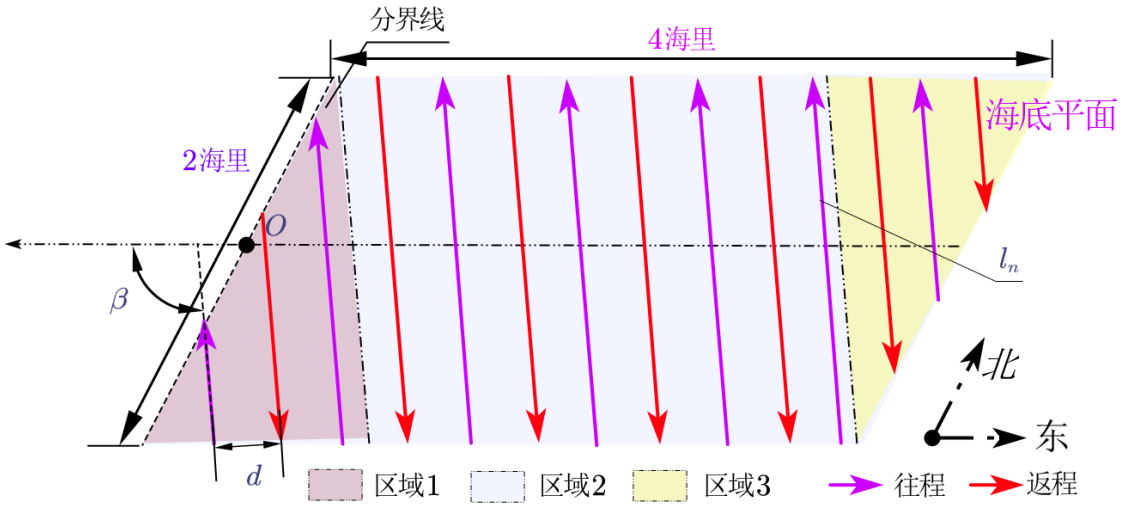


**图 16 行程简化图**

由图 16 容易得到：

$$L_{all}=\sum_{i=1}^{N_1}len_{1,i}+\sum_{i=1}^{N_2}len_{2,i}+\sum_{i=1}^{N_3}len_{3,i} \tag{5.14}$$

其中， $N_i$ 代表每个区域内行程的个数， $len_{i,j}$ 代表第 $i$ 个区域内第 $j$ 条行程的测线长度。

其中：

$$N_i=\begin{cases}\left\lfloor\dfrac{Length\times\cos\beta}{d}\right\rfloor & ,i=1\\ \left\lfloor\dfrac{Width\times\sin\beta}{d}\right\rfloor-\left\lfloor\dfrac{Length\times\cos\beta}{d}\right\rfloor & ,i=2\\ \left\lfloor\dfrac{Width/\sin\beta}{d}\right\rfloor-\left(\left\lfloor\dfrac{Width\times\sin\beta}{d}\right\rfloor-\left\lfloor\dfrac{Length\times\cos\beta}{d}\right\rfloor\right) & ,i=3\end{cases} \tag{5.15}$$

$$len_{i,j}=\begin{cases}j\times d/(\sin\beta\cos\beta) & ,i=1\\ Length/(\sin\beta) & ,i=2\\ \left(Length\times\cos\beta+Width\times\sin\beta-(N_1+N_2+j)d\right)/\sin\beta & ,i=3\end{cases} \tag{5.16}$$

“⌊ ⌋”代表向下取整， $Width$ 代表海域东西长度， $Length$ 代表海域南北长度。在第三问中， $Width=4(NM),Length=2(NM)$ 。

然后将式(5.14)带入到**非线性最优化模型的** Python 程序中做优化函数，式(5.13)作约束条件，得到结果。

### 5. 3. 3 模型结果与分析

✧ 求解沿坡边线测线模型得到结果如下

将模型三代入程序求解，设测量船从坡底扫向坡顶，得到最佳的行程数

**N = 34**

因此总测量长度：

$$L_{all}=68NM=125936m \tag{5.17}$$

相邻条带之间的重叠率$\eta=10\%$，且完全覆盖整个海域。

程序设计路线，设测量船从坡顶扫向坡底，同样得到最佳的行程数$N=34$，说明该方案可靠。

结果如图 17 和图 18 所示：横坐标代表行程数$N$，纵坐标代表测线离坡边线的距离$r$。

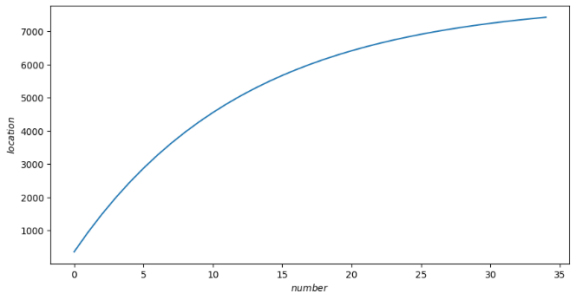


**图 17 测量船从坡底扫向坡顶时行程数的变化**

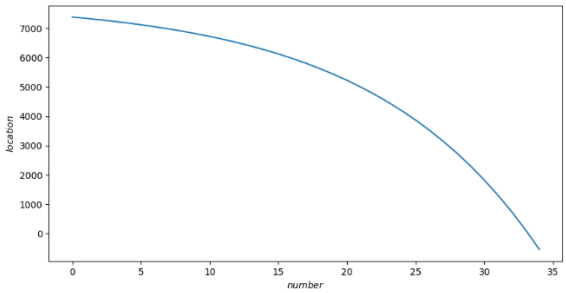


**图 18 测量船从坡顶扫向坡底时行程数的变化**

✧ 求解一般模型如下

给定一系列固定的侧线方向夹角$\beta$，在保证完全覆盖和保证重叠率的条件下，利用 Python 求解最优路线的总长度如下表所示：

**表 4 问题三一般模型的解答**

| 测线方向$\beta$/° | 侧线总长度$L_{all}$/$m$ |
|---|---|
| 0 | -- |
| 30 | 172606 |
| 60 | 146308 |
| 90 | 125936 |
| 120 | 157425 |
| 150 | 178255 |

**注：所有结果均保留两位小数；“--”代表达不到约束要求**

通过分析一般模型的求解答案可以得到，当$\beta=\frac{\pi}{2}$时，在满足约束条件的情况下，规划的路径总长度最短，且$\min\left(L_{all}\right)=125936$：

由此可以得到，无论是**沿坡边线测线模型**还是**一般模型**还是，最终得到当$\beta=\frac{\pi}{2}$时，即为最优路径：

$$\min\left(L_{all}\right)=125936 \tag{5.18}$$

## 5.4 问题四的模型：一般最优测线规划模型

### 5.4.1 模型准备

为了求解问题四，我们需要利用模型三，而模型三仅仅适用于海底为坡度一定的平面斜坡，而问题四给出的数据是一个光滑的曲面坡，首先应该“化曲为直”，将曲面分割成一个个坡度一定的小平面[3]。为了达到这个目的，首先我们需要利用点云的概念，对我们的数据加以处理，然后利用主成分分析法估计点云的法向量，采用当前点的邻近点，拟合出一个局部平面，然后得到法向量。

**(1) 坐标系建立与数据处理**

首先建立坐标系，如图 19 所示，取海平面上纵向坐标和横向坐标均为$0NM$的点为原点，海平面西-东方向和南-北方向分别为坐标系的$x$轴和$y$轴，$z$轴与$x$轴和$y$轴满足右手螺旋法则，指向天空。

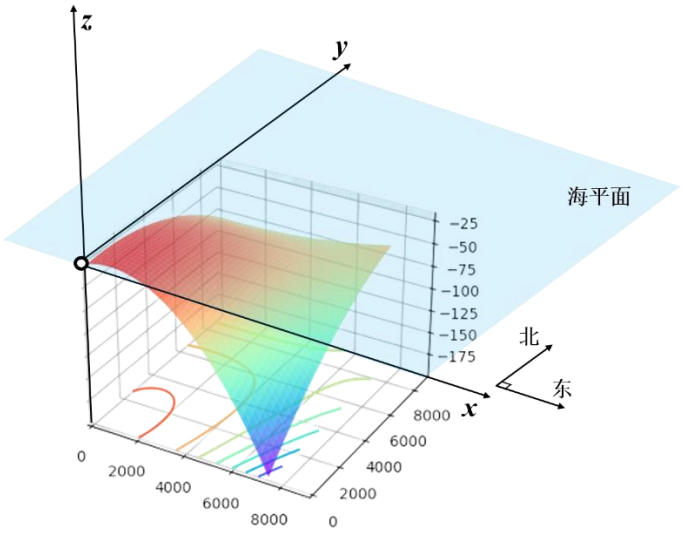


**图 19 海底地形坐标系**

点云是指目标表面特性的点集合，在问题四中，点云数据包括南北、东西平面坐标和对应的海底深度。

将*附件 1.xlsx* 中的给定数据提取出来，将横向坐标（由西向东）设为$x$轴坐标，将纵向坐标（由南向北）设为$y$轴坐标，将海水深度的相反数作为$z$轴坐标。列成行由$x, y, z$数据组成的点云数据。

**(2) PCA 主成分分析估计点云法向量**

为了利用点云数据求解法向量，目前最简单且最著名的法线估计方法是经典的主成分分析方法[4]，该方法的基础是分析一个点周围邻域的方差，并将最小方差的方向设置为法线。

**PCA 点云法向量步骤：**

***Step1*：寻找平面的候选点——候选点到这个平面的距离最小**

1) 使用 **k-近邻算法**，选取当前点的 $k$ 个临近点（$k=8$）；

2) 找到一个平面，使得以上选出得到点到这个平面的距离最小。

***Step2*：建立最小二乘法优化函数**

$$\min_{\mathbf{c},\mathbf{n},\|\mathbf{n}\|=1}\sum_{i=1}^{m}\left(\left(\mathbf{x}_i-\mathbf{c}\right)^T\mathbf{n}\right)^2 \tag{5.19}$$

其中 $\mathbf{x}_i$ 为数据点云，$\mathbf{n}$ 为平面的法向量，$\mathbf{c}=\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ 为平均点。

***Step3*：去中心化**

令 $\tilde{\mathbf{x}}_i=\mathbf{x}_i-\mathbf{c}$，该问题变成

$$\min_{n\in R^n}\sum_{i=1,\|\mathbf{n}\|_2=1}^{m}\left(\tilde{\mathbf{x}}_i^T\mathbf{n}\right)^2 \tag{5.20}$$

括号展开，化简为

$$\max_{z\in R^n,\|z\|_2=1} z^T\left(\tilde{X}\tilde{X}^T\right)z \tag{5.21}$$

***Step4*：PCA 分析**

寻求一个方向 $\mathbf{n}$ 使得所有邻域点在方向 $\mathbf{n}$ 上的投影点的分布最为集中，即点在该方向上的投影的方差最小，因此，PCA 中最不重要的方向向量就是 $\mathbf{n}$。根据 PCA 分析，我们要求的法向量 $\mathbf{n}$ 就是 (5.21)中 $\tilde{X}\tilde{X}^T$ 最小特征值对应的特征向量。

## 5.4.2 模型建立

依然采用问题三的策略：测量船从最西侧开始由南向北沿直线测量，然后测线不断向东平移到最东侧（方案一），此外，还考虑了测量船从最南侧开始由西向东沿直线测量然后测线不断向北平移到最东侧（方案二），如图 20 所示：

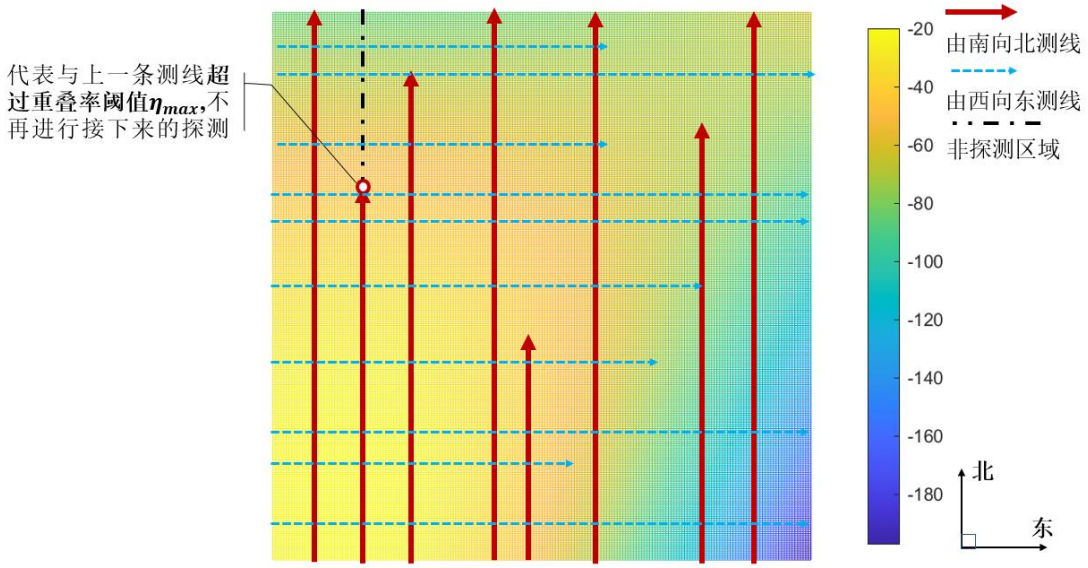


**图 20 问题四的策略**

具体求解步骤如下所示：

(1) 首先利用方向向量 $\mathbf{n}$ 求解等效坡度 $\alpha'$

通过向量的知识[5]容易得到坡度 $\alpha$：

$$\begin{aligned}\alpha &= \langle \mathbf{n}, (0,0,1) \rangle \\ &= \arccos\left( \frac{\mathbf{n} \cdot (0,0,1)}{\|\mathbf{n}\|_2} \right)\end{aligned} \tag{5.22}$$

由于航线始终向北，计算的到测线方向夹角 $\beta$：

$$\begin{aligned}\beta &= \langle \mathbf{n} \cdot A, h_{vec} \rangle \\ &= \arccos\left( \frac{(\mathbf{n} \cdot A) \cdot h_{vec}}{\|(\mathbf{n} \cdot A) \cdot h_{vec}\|_2} \right)\end{aligned} \tag{5.23}$$

其中 $3\times3$ 矩阵 $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$，其中 $h_{vec}$ 代表航行方向，当测量船由南向北沿直线测量时 $h_{vec} = (0,1,0)$，当测量船由西向东沿直线测量时 $h_{vec} = (1,0,0)$。

利用(5.8)可以得到等效坡度 $\alpha'$，由此可知，$\alpha'$ 是关于 $\mathbf{n}$ 的一个函数，记作：

$$\alpha' = \mathrm{A}(\mathbf{n}) \tag{5.24}$$

**算法 1：航线决策算法**

**输入：** 数据点云 $\mathbf{x}$，法向量 $\mathbf{n}$

**输出：** 测线的总长度 $L_{all}$

给定步长 $h \leftarrow 0.2$

$\theta \leftarrow 120$

$l_x \leftarrow 0$

$l_y \leftarrow 0$

初始化总的测线长度 $L_{all} \leftarrow 0$

**for** $i \leftarrow 0$ *to* $4/0.02$

**do**

计算沿着西—东北方向的测线距离 $l_y = i * h$

**for** $j \leftarrow 0$ *to* $5/0.02$

**do**

计算沿着南—北方向的测线距离 $l_x = j * h$

利用模型二计算各个参数

$$\begin{cases} \alpha_{i,j} = \arccos\left( \dfrac{\mathbf{n}_{i,j} \cdot (0,0,1)}{\|\mathbf{n}_{i,j}\|_2} \right) \\ \beta_{i,j} = \arccos\left( \dfrac{(\mathbf{n}_{i,j} \cdot A) \cdot h_{vec}}{\|(\mathbf{n}_{i,j} \cdot A) \cdot h_{vec}\|_2} \right) \\ \alpha'_{i,j} = \arctan\left( \tan \alpha_{i,j} \left| \sin \beta_{i,j} \right| \right) \\ D_{i,j} = \mathbf{x}_{i,j} \cdot (0,0,1) \end{cases}$$

利用模型二计算测线的覆盖宽度 $W_{i,j} = \dfrac{\sin\theta \cos^2 \alpha'_{i,j}}{\cos\left(\dfrac{\theta}{2} + \alpha'_{i,j}\right)\cos\left(\dfrac{\theta}{2} - \alpha'_{i,j}\right)} \cdot D_{i,j}$

**if** $i=1$
**do**
$L_{all}=L_{all}+l_x$,**break**
**else**
**do**
$\Delta_{i,j}=\left(x_{i,j}+y_{i-1,j}\right)\cos\alpha'-h$
利用模型二计算相邻条带之间的重叠率$\eta_{i,j}=\dfrac{\Delta_{i,j}}{W_{i,j}}\times100\%$
if $\eta_{i,j}\geq20\%$
**do**
判断是否满足$\eta<20\%$的要求，如果不满足，则跳出循环
$L_{all}=L_{all}+l_x$
**break**
**end if**
**end if**
**end for**
**end for**
**return** $L_{all}$

### 5.4.3 模型求解

使用矩阵图的方式对结果进行分析，见：

使用 Pyhton 实现**算法 1**，得到如下结果

**表 5 东-西方向重叠率阈值分析表**

| $\eta_{\max}$ | 0.21 | 0.25 | 0.30 | 0.35 | 0.40 |
|---|---|---|---|---|---|
| $L_{all}/m$ | 268540.00 | 277800.00 | 296320.00 | 314840.00 | 342620.00 |
| $rate_{cover}$ | 0.77 | 0.79 | 0.82 | 0.84 | 0.88 |
| 重叠长度/$m$ | 814.88 | 10000.80 | 26335.44 | 46522.24 | 78006.24 |

**表 6 南-北方向重叠率阈值分析表**

| $\eta_{\max}$ | 0.21 | 0.25 | 0.30 | 0.35 | 0.40 |
|---|---|---|---|---|---|
| $L_{all}/m$ | 744504.00 | 744504.00 | 744504.00 | 744504.00 | 744504.00 |
| $rate_{cover}$ | 0.70 | 0.72 | 0.74 | 0.77 | 0.79 |
| 重叠长度/$m$ | 9000.72 | 38336.40 | 67672.08 | 96859.60 | 126454.56 |

对表 5 和表 6 的数据分析发现**东-西测线**效果较好，可以得到当$\eta_{\max}=0.3$时，结果最优。利用 Python 作出东-西测线航线设计图：

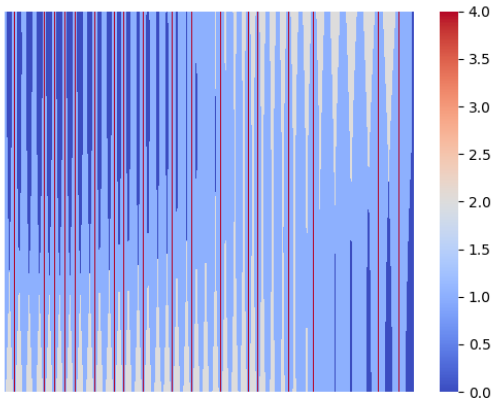


**图 21 东-西航线设计图**

模型所用颜色和数据如表所示：

**表 7 颜色和数据对应关系**

| 数值（测线矩阵） | 颜色（热力图） | 解释 |
|---|---|---|
| 3 或者 4 | 橙红色 | 测线路径 |
| 2 | 白色 | 重叠区 |
| 1 | 浅蓝色 | 覆盖区 |
| 0 | 蓝色 | 未覆盖区 |

# 六、 模型分析

## 6.1 灵敏性分析

### 6.1.1 波浪潮汐的影响—影响深度（问题三）

考虑到现实的航洋测深工程中常常遇到海浪潮汐的的影响，假设探测船能在波浪中保持平稳，即，探测波形依然是对称的等腰射线，因此，波浪潮汐主要影响探测深度 $D$，假设带来的扰动为

$$D_{disturbance}=5\times\sin\left(r_n\right) \tag{5.25}$$

即幅度为 $5m$ 的正弦波，其中 $r_n$ 为第 $n$ 条测线测线距坡边线处的距离。

通过分析可以得到扰动对规划结果影响不大，验证了模型对微小的深度扰动的不是特别敏感。

### 6.1.2 $\eta$ 的阈值变化（问题四）

题目要求覆盖宽度的重叠率尽量控制在 20% 以下，为了求解出最优方案，我们在问题四种给定了几个 $\eta$ 的阈值 $\eta_{\max}$，最后根据规划路径后的结果得出，当 $\eta_{\max}=24\%$ 时，能规划出最优路径。

由于在问题四中，分别分析了 $\eta_{\max}=20\%,\eta_{\max}=25\%,\eta_{\max}=30\%,\eta_{\max}=35\%,\eta_{\max}=40\%$ 的情况，由于步长太大，精确度低，现在分析 $\eta_{\max}$ 在 $30\%$ 附近取值时的优化结果，得到如下结果：

分析表格可以得到如下结论：

✧ **当 $\eta_{\max}=23.3\%$ 时，结果最优；**

✧ **当 $23\%<\eta_{\max}<25\%$ 时，$\eta_{\max}$ 对优化结果的影响较小。**

**表 8 重叠率的灵敏性分析**

| 上限 | 0.25 | 0.26 | 0.27 | 0.28 | 0.29 | 0.30 | 0.31 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\eta_{\max}$ | 277800.00 | 277800.00 | 287060.00 | 287060.00 | 296320.00 | 296320.00 | 305580.00 |
| $L_{all}/m$ | 0.79 | 0.78 | 0.80 | 0.80 | 0.82 | 0.82 | 0.83 |
| $rate_{cover}$ | 10000.80 | 11926.88 | 15112.32 | 18520.00 | 22816.64 | 26335.44 | 30743.00 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\eta_{\max}$ | 0.30 | 0.31 | 0.32 | 0.33 | 0.34 | 0.35 | 0.36 |
| $L_{all}/m$ | 296320.00 | 305580.00 | 305580.00 | 305580.00 | 314840.00 | 314840.00 | 314840.00 |
| $rate_{cover}$ | 0.82 | 0.83 | 0.84 | 0.83 | 0.85 | 0.84 | 0.86 |
| 重叠长度/m | 26335.44 | 30743.00 | 33373.04 | 37262.24 | 44522.08 | 46522.24 | 51930.00 |

#### 6.1.3 k-近邻算法 k 变化（问题四）

对于问题四求解过程中用到的 *k*-近邻算法，研究发现当近邻个数$k=8$时，发现法向量求解效果最优。

### 6.2 误差分析

#### 6.2.1 模型求解误差

问题四采用离散化的思想，将曲面化成一个个小平面，可能存在离散化所带来的误差；除此之外，理论值与测量值也会存在误差，两者综合产生第一种误差，将**搜索步长减小**，可以有效减小该误差。

#### 6.2.2 局部最优解误差

求解问题四时，为了简化计算，同时为了不陷入局部最优，需要人为干预来给定重叠率的上限，可能造成冗余解或缺解。

## 七、 模型评价、改进与推广

### 7.1 模型评价

#### 7.1.1 模型的优点

1. 在问题一中，使用机理分析的方法，得出覆盖宽度$W$和相邻条带之间的重叠率$\eta$是关于坡度$\alpha$、多波束换能器的开角$\theta$和探测点海水深度$D$的函数，并得出了函数的显式，具有一般性意义，为之后的问题奠定了理论基础；
2. 在问题二中，引入了等效坡度$\alpha'$的概念，一方面，将坡度$\alpha$和测线方向与海底坡面的法向在水平面上投影的夹角$\beta$结合分析，简化了分析过程；另一方面，充分利用问题一的模型，具有良好的继承性；
3. 在问题四的求解过程中，首先将题目所给的数据整理成点云数据，然后根据点云数据的性质，使用 *k*-近邻算法获得一组组点阵，接着使用 PCA 主成分分析估计点云法向量，确定平面的坡度，进而利用第三问的模型对问题四进行求解，该方法继承性强，便于推广。
4. 在问题四的求解中，使用 0-1 矩阵图来代表所测点是否被覆盖过，覆盖多少次，便

于后续分析，便于推广

### 7.1.2 模型的缺点

1. 在问题二中，引入了等效坡度$\alpha'$的概念，想法新颖，虽然能使问题求解更加简单，但是较难理解。
2. 在求解问题三、四的过程中，考虑到求解时间和效率，没有更加精细地处理点云数据，步长还可以取的更短，若这样考虑可能会得到更优策略。

## 7.2 模型的改进

1、改进之处主要包括以下几点：

(1) 在求解问题三、四过程中，本质上使用了贪心算法，为了提高求解的精确性，我们可以在原有的模型上，采用粒子群算法等算法进行最优化。

(2) 问题四中，可以进一步缩短仿真步长，得出更加精确的优化结果

## 7.3 模型的推广

我们所建立的多波束测深模型可以推广到其他探测领域，在求解问题四中，我们引入了点云的概念，采用 PCA 方法处理点云所代表的地理环境，可以应用在近地勘测卫星的轨道优化和潜艇深水探测的航线规划等更多实际问题，具有重要意义。

# 八、 参考文献


[1] 余启义. 基于多波束测深技术的海底地形测量[J]. 测绘与空间地理信息, 2022, 45(9): 262–264.

[2] 李凡, 金绍华, 边刚, 等. 多波束测深误差改进模型构建与验证[J]. 测绘学报, 2022, 51(5): 762–771.

[3] 吴波. 基于三维点云数据的海底地形配准技术研究[D]. 哈尔滨工程大学, 2021.

[4] 舒琪, 赵锐, 万旺根, 等. 基于局部点云拓展的法向量估计算法[J]. 工业控制计算机, 2023, 36(4): 90–92.

[5] 姜启源，谢金星，叶俊.数学模型 (第四版) [M.北京: 高等教育出版社,2016.

# 九、 附录

附录 1
支撑材料的文件列表

| 文件夹 | 文件名 | 主要用途/功能 |
|---|---|---|
| 数据 | A4.txt | 利用问题四的路径规划结果 |
| | PointCloud.txt | 问题四的点云数据，供 PCA 分析用 |
| | result1.xlsx | 该表格是我们利用**航线与坡边线平行时覆盖宽度和重叠率的数学模型**求解问题一第二小问的结果 |
| | result2.xlsx | 该表格是我们利用**考虑航线夹角时覆盖宽度和重叠率的数学模型**求解问题二第二小问的结果 |

附录 2
**问题一：应用模型一求解所给第二小问的 Python 源代码**

```
import math
import matplotlib.pyplot as plt
theta = math.radians(120)
alpha = math.radians(1.5)
d = 200
D = []
W = []
l = []
x = []
y = []
eta = []

for i in range(1, 10):
    l.append((i-5) * d)
    D.append(70 - (i-5) * d * math.tan(alpha))
    x.append(math.sin(theta/2) * D[i-1] / math.sin(math.pi/2-theta/2-alpha))
    y.append(math.sin(theta/2) * D[i-1] / math.sin(math.pi/2-theta/2+alpha))
    W.append(math.sin(theta) * math.cos(alpha)**2 * (70 - (i-5) * d * math.tan(alpha)) / (math.cos(theta/2-alpha) * math.cos(theta/2+alpha)))

for i in range(1, 9):
    eta.append(float(((x[i]+y[i-1]) * math.cos(alpha) - d) / W[i-1]))

plt.figure(1)
plt.plot(l, W)
plt.xlabel('$l(m)$')
plt.ylabel('$W(m)$')
plt.grid()
plt.show()
plt.figure(2)
plt.plot(l[1:], eta)
```

```
plt.xlabel('$l(m)$')
plt.ylabel('$\\eta$')
plt.grid()
plt.show()
```

附录 3

问题二：应用模型二求解所给第二小问的 Python 源代码

```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.ndimage import gaussian_filter1d

theta = 120*np.pi/180
alpha = 1.5*np.pi/180
beta = np.arange(0, math.radians(360), math.radians(5))
alpha_ = np.arctan(np.tan(alpha)*(np.sin(beta)))


d = 200
D = np.zeros((72, len(alpha_)))
W = np.zeros((len(alpha_), 72))
l = np.zeros(72)
for i in range(72):
    l[i] = i*d
    for j in range(len(alpha_)):
        D[i, j] = 120 - i*d*np.tan(alpha_[j])

W = np.sin(theta)*np.cos(alpha_)**2/(np.cos(theta/2-alpha_)*np.cos(theta/2+alpha))*D


fig2 = plt.figure(1)
# y_smoothed = gaussian_filter1d(W[:,4], sigma=5)
plt.grid()
# plt.plot(beta*180/math.pi, W[:,0])
plt.plot(l, W[:,55])
plt.xlabel('$l(M)$')
# plt.xlabel('$\\beta (degree)$')
plt.ylabel('$W(m)$')


beta, l = np.meshgrid(beta, l)
fig = plt.figure(2)
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X=beta*180/math.pi,Y=l,Z=W,cmap='cool',linewidth=0, antialiased=False,shade =True, alpha = 0.5)
ax.set_xlabel('$\\beta(degree)$')
ax.set_ylabel('$l(m)$')
ax.set_zlabel('$W(m)$')
```

```
41. plt.show()
```

附录 4

问题三：求解问题三沿坡边线测线模型的 Python 源代码

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  from Q3_BasicMath import calculate_Wn_1
4.  from math import *
5.  from scipy.optimize import fsolve
6.
7.  import random
8.  # 用于记录每一个y_i 的位置
9.  line_positions = [0]
10. L = 1852 * 4
11. gamma = sin(np.radians(31.5)) / (sin(np.radians(31.5)) + sin(np.radian
s(28.5)))
12. step = 0.02*1852
13. simulate_times = 100
14.
15. # 设置宽度左右两边的比例关系
16. west_width_percent = sin(radians(61.5))/(sin(radians(61.5))+sin(radian
s(58.5)))
17.
18. # 设置计算出来的宽度
19. def detect_width(y_position):
20.     return calculate_Wn_1(y_position)
21.
22.
23. # 从前向后算
24. def Solvevalue_1(unsolvedvalue,edge):
25.     y = unsolvedvalue[0]
26.     edge = edge
27.     return [
28.         detect_width(y)*gamma - y + edge,
29.     ]
30. def Calculate_lines_1(y):
31.     line_positions[0]=y
32.     while line_positions[-1] <= L:
33.         width = detect_width(line_positions[-1])
34.         edge = line_positions[-1]+width*(1-gamma-0.1)
35.         # print(detect_width(edge)*gamma - edge)
36.         Solvevalue_1([0],edge)
37.         y_next = fsolve(Solvevalue_1,[0],args=edge)
38.         line_positions.append(y_next[0])
39.     temp1 = line_positions[len(line_positions)-2]
```

```
    temp2 = line_positions[-1]
    flag_last = temp1 + detect_width(temp1)*(1-gamma)
    flag_next = temp2 - detect_width(temp2)*(gamma)
    ita = (flag_last-flag_next)/ detect_width(temp1)
    return ita,temp1,temp2,flag_last,flag_next

# 从后向前算
def Solvevalue_2(unsolvedvalue,edge):
    y = unsolvedvalue[0]
    edge = edge
    return [
        detect_width(y)*(1-gamma) + y - edge,
    ]
def Calculate_lines_2(y):
    line_positions[0]=y
    while line_positions[-1] >= 0:
        width = detect_width(line_positions[-1])
        edge = line_positions[-1]-width*(gamma-0.1)
        print(edge)
        Solvevalue_2([0],edge)
        y_next = fsolve(Solvevalue_2,[0],args=edge)
        line_positions.append(y_next[0])
    temp1 = line_positions[len(line_positions)-2]
    temp2 = line_positions[-1]
    flag_last = temp1 - detect_width(temp1)*gamma
    flag_next = temp2 + detect_width(temp2)*(1-gamma)
    ita = (flag_next-flag_last)/ detect_width(temp1)
    return ita,temp1,temp2,flag_last,flag_next


Solvevalue_1([0],0)
solved = fsolve(Solvevalue_1,[0],args=0)
print(solved)
print(Calculate_lines_1(solved[0]))
plt.figure(figsize=(10,5))
plt.plot(line_positions)

line_positions=[0]
Solvevalue_2([0],L)
solved = fsolve(Solvevalue_2,[0],args=L)
print(solved)
print(Calculate_lines_2(solved[0]))
plt.figure(figsize=(10,5))
plt.plot(line_positions)
import numpy as np
import math

# 问题1 计算出Wn
def calculate_Wn_1(y_position):
    theta = math.radians(120)
    alpha = math.radians(1.5)
```

```
        Depth = 2 * 1852 * math.tan(math.radians(1.5)) + 110 - y_position
* math.tan(math.radians(1.5))
        # print(Depth)
        W = math.sin(theta) * math.cos(alpha)**2 * Depth / (math.cos(theta
/2-alpha) * math.cos(theta/2+alpha))
        return W

# print(calculate_Wn_1(2*1852))


#记录每单位的移动对应的测量数量
# def detect_array():
#     detect_array = []
#     for i in range(0,L,step):
#         detect_array.append((i,detect_width(i)))
#     return detect_array
```

附录 5

**问题三：求解问题三任意方向测线模型的 Python 源代码**

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import math
width = np.linspace(0, 4 * 1852, 100)
height = np.linspace(0, 2 * 1852, 100)
x, y = np.meshgrid(width, height)
z = y * math.tan(math.radians(1.5)) - 2*1850*math.tan(math.radians(1.5))
+110
# 生成颜色映射
cmap = cm.get_cmap('jet')
# 根据 z 数据生成颜色值
z_min = np.min(z)
z_max = np.max(z)
normalized_z = (z - z_min) / (z_max - z_min)  # 将 z 数据归一化到[0, 1]范围
color = cmap(normalized_z)  # 根据归一化后的 z 值获取颜色
# 绘制三维图形
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z,linewidth=0, antialiased=False,shade =True,
alpha = 0.5,cmap='cool')
# 设置坐标轴标签
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_zlabel('Depth')
plt.show()
```

附录 6

问题三：定性分析 Python 源代码

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

theta = 120 * np.pi / 180
alpha = 1.5 * np.pi / 180
beta = np.arange(0, 361, 5) * np.pi / 180
alpha_ = np.arctan(np.tan(alpha) * np.sin(beta))
D = np.arange(75, 75.5,0.05)
W = np.zeros((len(beta), len(D)))

for i in range(len(beta)):
    for j in range(len(D)):
        W[i, j] = np.sin(theta) * np.cos(alpha_[i]) ** 2 / (np.cos(theta / 2 - alpha_[i]) * np.cos(theta / 2 + alpha_[i])) * D[j]

beta = beta * 180 / np.pi
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
beta, D = np.meshgrid(beta, D)
ax.plot_surface(beta, D, W.T, linewidth=0, antialiased=False,shade =True, alpha = 0.5,cmap='cool')
ax.set_xlabel('$\\beta(degree)$')
ax.set_ylabel('$D(m)$')
ax.set_zlabel('$W(m)$')
    plt.show()
```

附录 7

问题四：求解点云法向量的 Python 源代码

```
# 利用 PCA 分析进行法向量计算，并加载数据集中的文件进行验证
import open3d as o3d
import os
import numpy as np
from pyntcloud import PyntCloud
from pandas import DataFrame
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# matplotlib 显示点云函数
#def Point_Cloud_Show(points):
#    fig = plt.figure(dpi=150)
#    ax = fig.add_subplot(111, projection='3d')
#    ax.scatter(points[:, 0], points[:, 1], points[:, 2], cmap='spectral', s=2, linewidths=0, alpha=1, marker=".")
#    plt.title('Point Cloud')
#    ax.set_xlabel('x')
#    ax.set_ylabel('y')
```

```
#     ax.set_zlabel('z')
#     plt.show()

# 二维点云显示函数
#def Point_Show(pca_point_cloud):
#     x = []
#     y = []
#     pca_point_cloud = np.asarray(pca_point_cloud)
#     for i in range(10000):
#         x.append(pca_point_cloud[i][0])
#         y.append(pca_point_cloud[i][1])
#     plt.scatter(x, y)
#     plt.show()

# 功能：计算PCA 的函数
# 输入：
#     data：点云，NX3 的矩阵
#     correlation：区分np 的cov 和corrcoef，不输入时默认为False
#     sort: 特征值排序，排序是为了其他功能方便使用，不输入时默认为True
# 输出：
#     eigenvalues：特征值
#     eigenvectors：特征向量
def PCA(data, correlation=False, sort=True):
    # data => (10000, 3)  data_mean => (1, 3)
    data_mean = np.mean(data, axis=0)  # 对列求均值
    # normalize_data => (10000, 3)
    normalize_data = data - data_mean  # 数据归一化操作
    # H => (3, 3)
    H = np.dot(normalize_data.transpose(), normalize_data)
    # eigenvectors => (3,3)  eigenvalues => (3,)  eigenvectors_transpose => (3,3)
    eigenvectors, eigenvalues, eigenvectors_transpose = np.linalg.svd(H)  # SVD 分解
    # 将特征值从大到小进行排序，便于提取主成分向量
    if sort:
        sort = eigenvalues.argsort()[::-1]
        eigenvalues = eigenvalues[sort]
        eigenvectors = eigenvectors[:, sort]

    return eigenvalues, eigenvectors, normalize_data

# 主函数
def main():
    # 指定点云路径
    # cat_index = 10 # 物体编号，范围是0-39，即对应数据集中40 个物体
    # root_dir = '/Users/renqian/cloud_lesson/ModelNet40/ply_data_points' # 数据集路径
    # cat = os.listdir(root_dir)
    # filename = os.path.join(root_dir, cat[cat_index],'train', cat[cat_index]+'_0001.ply') # 默认使用第一个点云

```

```
    # # 原始点云显示、可视化点云PCA之后的结果、PCA降维之后依据成分向量还原点云、用PCA分析点云主方向
    # ************************************************************************************************
    # 1、加载原始点云(text)
    # raw_point_cloud_matrix = np.genfromtxt(r"E:\\data.txt", delimiter=" ")
    raw_point_cloud_matrix = np.loadtxt('E:\\data.txt')
    raw_point_cloud_matrix_part = raw_point_cloud_matrix[:,0:3]
    #raw_point_cloud = pd.DataFrame(raw_point_cloud_matrix_part)
    #raw_point_cloud_matrix = np.genfromtxt(r"E:\\data.txt", delimiter=" ")
    # raw_point_cloud_matrix_part = > (10000, 3)
    #raw_point_cloud_matrix_part = raw_point_cloud_matrix[:, 0:3]
    raw_point_cloud = DataFrame(raw_point_cloud_matrix_part)  # 选取每一列的第0至第2个元素
    raw_point_cloud.columns = ['x', 'y', 'z']
    point_cloud_pynt = PyntCloud(raw_point_cloud)
    point_cloud_o3d = point_cloud_pynt.to_instance("open3d", mesh=False)
    #o3d.visualization.draw_geometries([point_cloud_o3d])
    #Point_Cloud_Show(raw_point_cloud_matrix_part)
    # 2、可视化点云PCA之后的结果
    eigenvalues, eigenvectors, normalize_data = PCA(raw_point_cloud_matrix_part)
    # vector => (3,2)
    vector = np.mat(eigenvectors[:, 0:2])
    # vector_transpose => (2,3)
    vector_transpose = vector.transpose()
    # pca_point_cloud_1 => (10000, 2)
    #pca_point_cloud_1 = np.dot(normalize_data, vector)
    #print(pca_point_cloud_1)
    # 3、PCA降维之后成分还原显示
    #Point_Show(pca_point_cloud_1)
    # pca_point_cloud_1 => (10000, 3)
    #pca_point_cloud_2 = np.dot(pca_point_cloud_1, vector_transpose)
    #Point_Cloud_Show(pca_point_cloud_2)
    # 4、用PCA分析点云主方向
    primary_orientation_ = eigenvectors[:, 0]
    second_orientation = eigenvectors[:, 1]
    #print('the main orientation of this pointcloud is: ', primary_orientation_)
    #print('the second orientation of this pointcloud is: ', second_orientation)
    point = [[0, 0, 0], primary_orientation_, second_orientation]
    lines = [[0, 1], [0, 2]]
    colors = [[1, 0, 0], [0, 1, 0]]
    # 构造open3d中的LineSet对象，用于主成分和次主成分显示
    line_set = o3d.geometry.LineSet(points=o3d.utility.Vector3dVector(point), lines=o3d.utility.Vector2iVector(lines))
    line_set.colors = o3d.utility.Vector3dVector(colors)
```

```
    #o3d.visualization.draw_geometries([point_cloud_o3d, line_set])

    # 循环计算每个点的法向量
    # ************************************************************************
    # 从点云中获取点，只对点进行处理
    points = point_cloud_pynt.points
    print('total points number is:', points.shape[0])
    normals = []
    # 由于最近邻搜索，此处允许直接调用open3d中的函数
    pcd_tree = o3d.geometry.KDTreeFlann(point_cloud_o3d)
    # 每一点的法向量计算，通过PCA降维，对应最小特征值的成分向量近似为法向量
    for i in range(points.shape[0]):
        [_, idx, _] = pcd_tree.search_knn_vector_3d(point_cloud_o3d.points[i], 15)
        k_nearest_point = np.asarray(point_cloud_o3d.points)[idx, :]
        w, v, _ = PCA(k_nearest_point)
        normals.append(v[:, 2])
        #if i%100==0:
        #    print(normals[i])

    normals = np.array(normals, dtype=np.float64)
    #print(normals.shape)
    # TODO: 此处把法向量存放在了normals中
    point_cloud_o3d.normals = o3d.utility.Vector3dVector(normals)
    # 法向量可视化，根据open3d文档，需要在显示窗口按住键‘n’才可以看到法向量
    #o3d.visualization.draw_geometries([point_cloud_o3d])

if __name__ == '__main__':
    main()
```

附录 8

**问题四：利用法向量数据求解对应方向覆盖宽度的 Python 源代码**

```
from Q4_Map_Modeling import *
from math import *
from Q4_SetNormalVector import *
import numpy as np
# 基本参数初始化
theta = radians(120)                # 表示测量张角
alpha = np.zeros((251,201))         # 表示海底平面斜坡坡度
beta = np.zeros((251,201))          # 表示法向量的投影和测线方向的夹角
alpha_ = np.zeros((251,201))        # 表示等效的坡度大小
gamma = np.zeros((251,201))         # 表示宽度在当前行进方向的左右分布系数（表示deep的部分）
Height = 5 * 1852                   # 表示的是地图的长度（单位：m）x
Width = 4 * 1852                    # 表示的是地图的宽度（单位：m）y
Step = 0.02 * 1852                  # 表示的是各单位的移动步长（单位：m）
```

```
Map = Map()                                  # 初始化扫描地图
Ita = []                                     # 记录出现过的重复率
W = np.zeros((251,201))                      # 表示存储宽度的矩阵
# 记录从什么位置进行寻路

# 函数：定位参数设置 注意这里输入的是对应的模块坐标
def Get_width(theta,x_position, y_position, gamma):
    raw = x_position
    col = y_position
    raw = int(raw)
    col = int(col)
    width = Calculate_W(theta,raw,col)  # 这里需要加上对应的函数
    Map.set_width_map(raw, col, width*gamma[raw,col], width*(1-gamma[raw,col]))
    return raw, col, width*gamma[raw,col], width*(1-gamma[raw,col])

# 函数：寻路算法
def find_path(Direction):
    if (Direction == 'Verticle'):
        left_edge = 0
        right_edge = Width
        road_map = np.zeros((251,201))
        for col in range(0, 201):
            road_map = np.copy(Map.road_map)     # 得到路径地图
            for raw in range(0, 251):
                i = col * Step                   # 表示y 的数值
                j = raw * Step                   # 表示x 的数值
                raw = int(raw)
                col = int(col)
                # 检查当前区域是否被检测到过
                if Map.valid_map[raw][col] == 0:
                    road_map[raw][col] = 1       # 设置当前的节点为路径节点
                    width = Map.get_width_by_raw_and_col(raw, col)  # 得到宽度
                    left_area = width[0] // Step
                    right_area = width[1] // Step
                    left_edge_math = i - width[0]   # 表示左侧的具体值
                    right_edge_math = i + width[1]  # 表示右侧的具体值
                    left_edge = col - left_area if col > left_area else 0  # 左位置
                    right_edge = col + right_area if col + right_area < Width // Step else (Width // Step)-1  # 右位置
                    left_edge = int(left_edge)
                    right_edge = int(right_edge)
                    # 找到并修改成已经检查的区域块
                    change_node = np.arange(left_edge, right_edge+1)
                    for c in change_node:
                        Map.set_valid_map(raw, c)

                    # 检查重叠率
```

```
60.                     for t in range(left_edge, col):
61.                         if (road_map[raw][t] == 1):
62.                             left_node_width = Map.get_width_by_raw_and_
   col(raw, t)
63.                             ita = (t*Step+left_node_width[1]-(i-
   left_edge_math)) / (left_node_width[0]+left_node_width[1])
64.                             if ita >= 0.2:
65.                                 break
66.                             Ita.append(ita)
67.                 else:
68.                     break
69.             if (raw == 251):
70.                 Map.road_map = np.copy(road_map)
71.         return
72.     elif (Direction == "Horizen"):
73.         for j in np.arange(0, Height+1, Step):
74.             road_map = Map.road_map                    # 得到路径地图
75.             for i in np.arange(0, Width+1, Step):
76.                 col = i // Step                        # 设置对应的列
77.                 raw = j // Step                        # 设置对应的行
78.                 # 检查当前区域是否被检测到过
79.                 if (Map.valid_map[raw][col] == 0):
80.                     road_map[raw][col] = 1                              #
   设置当前的节点为路径节点
81.                     width = Map.get_width_by_raw_and_col(raw, col)  #
   得到宽度
82.                     up_area = width[0] // Step                          #
   得到左侧宽度
83.                     down_area = width[1] // Step                        #
   得到右侧宽度
84.                     up_edge = col - up_edge if col > up_area else 0 #
   左位置
85.                     down_edge = col + down_edge if col + \
86.                         down_area < Height//Step else Height//Step-
   1# 右位置
87.
88.                     # 找到并修改成已经检查的区域块
89.                     change_node = np.arange(down_edge, up_edge)
90.                     Map.set_valid_map(change_node, col)
91.
92.                     # 检查重叠率
93.                     for t in range(down_edge, raw):
94.                         if (road_map[t][col] == 1):
95.                             down_node_width = Map.get_width_by_raw_and_
   col(
96.                                 t, col)
97.                             ita = (t + down_node_width[1]-
   down_edge) / \
98.                                 (down_node_width[0] + down_node_width[1
   ])
99.                             if ita >= 0.2:
```

```
                                    break
                                else:
                                    if (t == Width):
                                        Map.road_map = road_map
                                    Ita.append(ita)
                            else:
                                break
                    return

def ReadVectors():
    normals = set_normal_vector()
    for i in range(len(normals)):
        raw = i // 201
        col = i % 201
        Map.vector_map[raw][col]=normals[i]
    return normals

# 函数：计算坡度和测线张角
def Calculate_alpha_beta(Normal_vector,Direction):
    line_direction = np.zeros(3)
    if(Direction == "Verticle"):
        line_direction = np.array((1,0,0))
    elif(Direction == "Horizen"):
        line_direction = np.array((0,1,0))
    z = np.array((0,0,1))
    b = np.array((Normal_vector[0],Normal_vector[1],0))
    n = np.array((Normal_vector[0],Normal_vector[1],Normal_vector[2]))
    cal_beta = Calculate_angle(line_direction,b)
    cal_alpha = Calculate_angle(n,z)
    return cal_beta,cal_alpha

# 函数：计算夹角
def Calculate_angle(vector1,vector2):
    vector1 = np.array(vector1)
    vector2 = np.array(vector2)
    l_x = sqrt(vector1.dot(vector1))
    l_y = sqrt(vector2.dot(vector2))
    dian = vector1.dot(vector2)
    cos_ = dian / (l_x * l_y)
    angle = acos(cos_)
    return angle

# 函数：计算覆盖宽度
def Calculate_W(theta,raw,col):
    D = Map.depth
    w = sin(theta)*cos(alpha_[raw,col])**2/(cos(theta/2-
  alpha_[raw,col])*cos(theta/2+alpha_[raw,col])) * D[raw,col]
    return w
# 函数：保存文件
def text_save(filename, data):#filename 为写入 CSV 文件的路径，data 为要写
  入数据列表.
```

```
    file = open(filename,'a')
    for i in range(len(data)):
        s = str(data[i]).replace('[','').replace(']','')#去除[],这两行按数据不同，可以选择
        s = s.replace("'",'').replace(',','') +'\n'   #去除单引号，逗号，每行末尾追加换行符
        file.write(s)
    file.close()
    print("保存文件成功")

def Calculate_Width():
    ReadVectors()
    H = len(Map.vector_map)
    W_ = len(Map.vector_map[0])
    for i in range(H):
        for j in range(W_):
            beta[i,j],alpha[i,j] =Calculate_alpha_beta(Map.vector_map[i,j],"Verticle")
            alpha_[i,j] = atan(tan(alpha[i,j])*(sin(beta[i,j])))
            gamma[i,j] = cos(theta/2+alpha_[i,j])/(cos(theta/2+alpha_[i,j]) + cos(theta/2-alpha_[i,j]))
            Get_width(theta,i,j,gamma)
    return Map.width_map

def ShowArray(array):
    array = np.array(array)
    for i in range(len(array)):
            print(array[i,:])

Calculate_Width()
find_path("Verticle")
map = Map.road_map
print(map)
print(len(map),len(map[0]))
```

附录 9

**绘图：问题四寻路算法 Python 源代码**

```
from Q4_Map_WidthGet import *

percent = 0.25 # 用于记录\ita 限高

# 寻路算法
def find_path(Direction):
    if (Direction == 'Verticle'):
        # 各种变量的初始化
        left_edge = 0
        right_edge = Width
        road_map = np.zeros((Raw_Number,Col_Number))
        ita = -999
```

```
13.         for col in range(0, Col_Number):
14.             valid_map = np.copy(Map.valid_map)  # 将valid_map进行复制
15.             road_map = np.copy(Map.road_map)  # 得到路径地图
16.             Ita_temp = []
17.             for raw in range(0, Raw_Number):
18.                 i = col * Step  # 表示y的数值
19.                 j = raw * Step  # 表示x的数值
20.                 raw = int(raw)
21.                 col = int(col)
22.
23.                 # 检查当前区域是否被检测到过
24.                 if Map.valid_map[raw][col] == 0:
25.                     road_map[raw][col] = 1  # 设置当前的节点为路径节点
26.                     width = Map.get_width_by_raw_and_col(raw, col)  #
得到宽度
27.                     left_edge_math = i - width[0] # 表示左侧的具体值
28.                     right_edge_math = i + width[1] # 表示右侧的具体值
29.                     left_edge = int(left_edge_math//Step) if left_edge
_math >= 0 else 0  # 左位置
30.                     right_edge = int(right_edge_math//Step) if right_e
dge_math <= Width else int(Width//Step-1)  # 右位置
31.                     left_edge = left_edge + 1 if left_edge*Step<left_e
dge else left_edge
32.                     right_edge = right_edge + 1 if right_edge*Step<rig
ht_edge else right_edge
33.
34.                     # 找到并修改成已经检查的区域块
35.                     change_node = range(left_edge, right_edge+1)
36.                     for c in change_node:
37.                         valid_map[raw][c] += 1
38.
39.                     # 检查重叠率
40.                     for t in range(1, col+1):
41.                         if ( int(road_map[raw][col-t]) == 1 ):
42.                             left_node_width = Map.get_width_by_raw_and
_col(raw, col-t)
43.                             delta = (col-t)*Step+left_node_width[1]-
left_edge_math
44.                             # delta =
45.                             ita = delta / sum(left_node_width)
46.                             break
47.
48.                     # 判断选择是否合法
49.                     Ita_temp.append(ita)
50.                     if ita >= percent:
51.                         break
52.                 else:
53.                     break
54.             # 判断当前路径是否能够到
55.             if (raw == Raw_Number-1):
56.                 Map.road_map = np.copy(road_map)
```

```
57.                 Map.valid_map = np.copy(valid_map)
58.                 Ita.append(Ita_temp)
59.         return
60.
61.     elif (Direction == "Horizen"):
62.         # 各种变量的初始化
63.         up_edge = 0
64.         down_edge = Width
65.         road_map = np.zeros((Raw_Number,Col_Number))
66.         ita = -999
67.         for raw in range(0, Raw_Number):
68.             valid_map = np.copy(Map.valid_map)  # 将 valid_map 进行复制
69.             road_map = np.copy(Map.road_map)  # 得到路径地图
70.             Ita_temp = []
71.             for col in range(0, Col_Number):
72.                 i = col * Step  # 表示 y 的数值
73.                 j = raw * Step  # 表示 x 的数值
74.                 raw = int(raw)
75.                 col = int(col)
76.
77.                 # 检查当前区域是否被检测到过
78.                 if Map.valid_map[raw][col] == 0:
79.                     road_map[raw][col] = 1  # 设置当前的节点为路径节点
80.                     width = Map.get_width_by_raw_and_col(raw, col)  #
得到宽度
81.                     up_edge_math = j - width[0] # 表示左侧的具体值
82.                     down_edge_math = j + width[1] # 表示右侧的具体值
83.                     up_edge = int(up_edge_math//Step) if up_edge_math
>= 0 else 0  # 左位置
84.                     down_edge = int(down_edge_math//Step) if down_edge
_math <= Height else int(Height//Step-1)  # 右位置
85.                     up_edge = up_edge + 1 if up_edge*Step<up_edge else
 up_edge
86.                     down_edge = down_edge + 1 if down_edge*Step<down_e
dge else down_edge
87.
88.                     # 找到并修改成已经检查的区域块
89.                     change_node = range(up_edge, down_edge+1)
90.                     for c in change_node:
91.                         valid_map[c][col] += 1
92.
93.                     # 检查重叠率
94.                     for t in range(1, raw+1):
95.                         if ( int(road_map[raw-t][col]) == 1 ):
96.                             up_node_width = Map.get_width_by_raw_and_c
ol(raw-t, col)
97.                             delta = (raw-
t) * Step + up_node_width[1] - up_edge_math
98.                             # delta =
99.                             ita = delta / sum(up_node_width)
100.                            break
```

```
101.
102.                     # 判断选择是否合法
103.                     Ita_temp.append(ita)
104.                     if ita >= percent:
105.                         break
106.                 else:
107.                     break
108.             # 判断当前路径是否能够到
109.             if (col == Col_Number-1):
110.                 Map.road_map = np.copy(road_map)
111.                 Map.valid_map = np.copy(valid_map)
112.                 Ita.append(Ita_temp)
113.         return
114.
115. def Calculate_Cover():
116.     count = 0
117.     for i in range(len(Map.valid_map)):
118.         for j in range(len(Map.valid_map[0])):
119.             if(Map.valid_map[i][j] != 0):
120.                 count+=1
121.     return count*Step**2/(Width*Height)
122.
123. def Calculate_Ita():
124.     count = 0
125.     count_ita = 0
126.     for item in Ita:
127.         for i in item:
128.             count_ita+=1
129.             if i > 0.2:
130.                 count+=1
131.     return count / count_ita
132.
133. def text_save(filename, data):  # filename 为写入 CSV 文件的路径，data 为
要写入数据列表.
134.     file = open(filename, 'a')
135.     for i in range(len(data)):
136.         s = str(data[i]).replace(
137.             '[', '').replace(']', '')  # 去除[],这两行按数据不同，可以选
择
138.         s = s.replace("'", '').replace(',', '') + '\n'  # 去除单引号，
逗号，每行末尾追加换行符
139.         file.write(s)
140.     file.close()
141.     print("保存文件成功")
142.
143.
144. Calculate_Width()
145. find_path("Horizen")
146. text_save("map_horizen.txt", Map.valid_map + Map.road_map*3)
147. i_ta = Calculate_Ita()
148. cover =Calculate_Cover()
```

```
149.text_save("map_horizen_ita.txt",[f"超过 20%的部分:{i_ta},\n 覆盖
率:{cover},\n 超过 20%的长度:{i_ta*Step}"])
150.# print(Calculate_Cover())
151.# print(Calculate_Ita())
```

附录 10

**问题四：求解问题四构建 Map 巡航地图模型的 Python 源代码**

```
1.  import numpy as np
2.  from math import *
3.  import pandas as pd
4.  import networkx
5.
6.  Raw_Number = int(251) * int(4)
7.  Col_Number = int(201) * int(4)
8.  raw_divide = col_divide = int(4)
9.
10. raw_block = Raw_Number // raw_divide
11. col_block = Col_Number // col_divide
12.
13.
14. class Map:
15.
16.     def __init__(self) -> None:
17.         self.width_map = np.zeros((Raw_Number, Col_Number), dtype=tupl
e)
18.         self.road_map = np.zeros((Raw_Number, Col_Number))
19.         self.valid_map = np.zeros((Raw_Number, Col_Number),dtype=int)
20.         self.depth = self.get_depth()
21.         self.vector_map = np.zeros((Raw_Number,Col_Number),dtype=list)
22.
23.     # 获得每个点的深度矩阵
24.     def get_depth(self):
25.         excel_file = "C:\\Users\\Slater\\Desktop\\Math\\B 题\\附件
1.xlsx"
26.         # 使用 pandas 的 read_excel 函数读取数据
27.         df = pd.read_excel(excel_file, header=1, index_col=1)
28.         z = df.values[0:, 1:].astype('float')
29.         z_ = np.zeros((Raw_Number,Col_Number))
30.         for i in range(0,raw_block):
31.             for j in range(0,col_block):
32.                 for t in range (0,raw_divide):
33.                     for k in range (0,col_divide):
34.                         z_[i*raw_divide+t,j*col_divide+k] += z[i,j]
35.         return z_
36.
37.     # 获得点在当前方向的宽度
38.     def get_width_by_raw_and_col(self, x, y):
39.         return self.width_map[x, y]
40.
```

```
    # 表示是否被标记
    def set_valid_map(self, x, y) -> int:
        x = int(x)
        y = int(y)
        self.valid_map[x, y] += 1
        return self.valid_map[x, y]

    # 标记是合法的路径
    def set_road_map(self, x, y) -> bool:
        if self.road_map[x, y] == 1:
            return False
        self.road_map[x, y] = 1
        return True

    # 设置深度的数据
    def set_width_map(self, x, y, value_left, value_right) -> bool:
        x = int(x)
        y = int(y)
        # print(x,y)
        self.width_map[x, y] = (value_left, value_right)
        # print(self.width_map)
        return True
```

附录 11

绘图：绘制问题四海底地形图的 Python 源代码

```
import numpy as np
from matplotlib import pyplot as plt

#    定义三维数据,注意复制表格中的数据到dataa.txt 文件中
data = np.loadtxt('E:\\dataa.txt')
yy = np.arange(0, 5.02 * 1852, 0.02 * 1852)
xx = np.arange(0, 4.02 * 1852, 0.02 * 1852)
X, Y = np.meshgrid(xx, yy)
Z = data[:, :]

# 定义坐标图
fig = plt.figure()
ax3 = plt.axes(projection='3d')

# 作图
ax3.set_xlim(0, 9000)
ax3.set_ylim(0, 9000)
ax3.plot_surface(X, Y, Z, alpha=0.7, cmap='rainbow')
ax3.contour(X, Y, Z, zdim='z', offset=-200, cmap='rainbow')  # 等高线图，要设置offset，为Z 的最小值
plt.show()
```