# 基于贪心算法的多波束海水深度测量测线规划模型

## 摘要

多波束测深是一种用来测量海水深度的技术：可一次同时发出多条探测波，覆盖某条带内多个点的海底深度值，实现“线—面”测量，具有突出的技术进步意义。本文主要基于多波束测深技术，建立了不同海域位置下波束条带覆盖宽度、重叠率的数学模型模型，并利用贪心算法、二分法、插值法等方法解决带有约束的路径规划问题，设计测线布设方式。

**针对问题一**，将实际问题转化为平面几何模型，依据海底坡面倾角、测量船换能器开角、海水深度、两次测量位置间距等物理量的几何关系，计算得到波束带覆盖宽度以及相邻波束带重叠率的数学模型 (14)(8)，并借助 C++ 语言计算得到结果 result1.xlsx。

**针对问题二**，将实际问题转化为立体几何模型，依据测线方向与海底坡面的法向在水平面上投影的夹角和其他物理量的关系，作一垂直于测线方向的截面，将问题简化为问题一中平面几何模型，计算得到波束带覆盖宽度的数学模型 (15)，并借助 C++ 语言计算得到结果 result2.xlsx。

**针对问题三**，建立带有约束的优化模型。首先，根据引文 [2]，及在问题二中观察到的规律，对于该坡度均匀的海底面，规划测线使得其平行于坡面的等高线（在本题中意味着测线为南北走向），可以更好地利用探测范围，减小重叠率，于是原问题转化为一维规划问题。然后利用二分法确定第一条测线的位置，利用贪心算法确定其余测线起点在东西走向上的坐标，发现相邻两条测线重叠率为 10% 时测线总长度最短。最终借助 C++ 语言求得最合理的测线布设应为 34 条间距不一的、平行于南北走向的直线，总长度为 68nmile。

**针对问题四**，在问题三的基础上建立带有约束的多目标优化模型，仍然采用贪心算法。首先将附件中的测深数据建立为格点模型，进行离散化处理，并利用线性插值法，将每个点的距离缩小为 4.63m。其次，为了良好定义重叠率、简便计算格点模型、避免侧漏，我们仍然考虑将测线规划为东西或南北走向的直线。接着，进行探测点标记，对于每一测线，在垂直其方向上标记可以被探测到的点。然后进行有效性检验，遍历两测线间的所有格点，分析该测线与前一条测线的重叠率和漏测情况。最后进行间距优化，通过枚举法试探测线间距，尽可能使间距变大，直至恰好不漏测。重复上述过程指导测量范围覆盖整个海域。最终借助 C++ 语言求得换能器张角不同时的最合理测线布设方案。

**关键字**：多波束测深　贪心算法　线性插值法　优化模型

# 一、 问题重述

## 1.1 问题背景

单波束测深是一种测量水深的技术，利用声波在水中作匀速直线传播的原理，可以测量一个点的水深值。在不同位置连续发出单波束，点动成线，可以测量测线上各处的水深值，但无法得到该海域全部位置的水深值。而多波束测深可以发射出一条含多条波束的波束带，瞬间测量具有一定宽度的全覆盖水深条带。随着船体的移动，线动成面，便可以得到该海域各处的水深值。本题主要利用多波束测深法测量不同地形的水深值，并对测线布设进行合理规划。

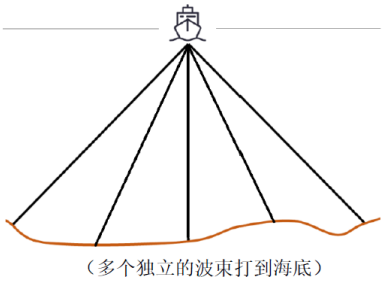


**图 1　多波束探深工作原理图 1**

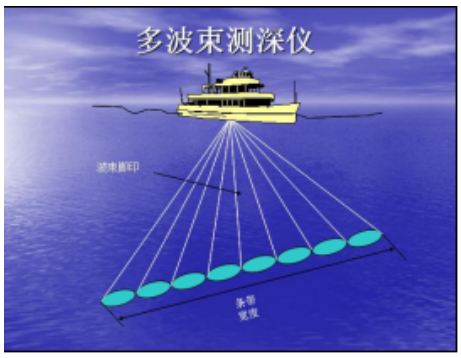


**图 2　多波束探深工作原理图 2[1]**

## 1.2 问题要求

基于上述背景，要求建立数学模型解决如下问题：

多波束测深中，换能器开角和水深均会影响条带的覆盖宽度。在海底地形平坦、测线平行时，可定义相邻条带之间的重叠率。一般要求相邻测线的测幅有 10%-20% 的相互重叠以避免产生探测盲区、保证测量便利。若重叠率小于 0，则表示漏测。现实中对测线进行布设时，由于真实海底地形的起伏变化较大，在浅水区会出现漏测，在深水区会出现重叠率过大的情况。

问题一：若测深区域与水平面有一 $\alpha$ 夹角，建立多波束测深的覆盖宽度及相邻条带之间重叠率的数学模型，并求解多波束换能器的开角为 120° 坡度为 1.5°，海域中心点处的海水深度为 70m 时各处位置的海水宽度、覆盖宽度、重叠率。

问题二：在给出的矩形待测海域上，若测线方向与坡面法向在水平面上的投影呈一夹角，建立多波束测深的覆盖宽度数学模型，并利用模型计算船距海域中心点不同距离、不同夹角下的覆盖宽度。

问题三：某矩形海域南北长 2nmile、东西宽 4nmile，海域中心点处水深 110m，西深东浅。给该海域布设测线，要求该测线可测量全部海域水深值，并且测线总和最短，相邻条带覆盖率满足要求。

问题四：利用某海域单波束测深得到的水深值，给该海域规划多波束测深路线，要求该测线尽量可测量全部海域水深值，并且测线总和尽量短，相邻条带覆盖率在 20% 以下。设计完成后计算测线总长度、漏测比例、条带覆盖率超标比例。

# 二、 问题分析

## 2.1 问题一分析

问题一可依据题目图示建立平面几何模型，根据几何关系进行推导，从而建立多波束带覆盖宽度及相邻波束带重叠率的数学模型。问题一即为问题二 $\beta = 90^\circ$ 的特殊情形。

## 2.2 问题二分析

在问题一的基础上，对测线方向不垂直于海底坡面的法向在水平面上投影时进行建模。由于多波束全覆盖水深条带与测线垂直，故考虑作测线的垂直平面，找出该平面与海底坡面的交线，并求出该交线与海平面的夹角，转化为问题一的平面几何模型，利用问题一结论进行解决。

## 2.3 问题三分析

问题三是带约束的规划问题。由题目要求可知，该海域西深东浅，自西向东坡度不变，地形平坦。又由多波束测深的测线布设相关规定 [1][2]，我们考虑沿等高线布设的测线，并希望在保证相邻两条测线重叠率在 $10\% - 20\%$ 的情况下总航程尽可能短。故考虑平行于南北走向的测线。则问题回归至问题一的模型。通过寻找覆盖范围与测线之间的关系，可以进一步利用算法优化求得的解。

## 2.4 问题四分析

问题四在问题三的基础上增大了海底坡面的复杂性。首先对所给不同位置的海深数据 (附件.xlsx) 进行可视化处理，得到海底地形轮廓图。然后对网格数据进行插值处理，再用贪心算法设计最优测线。

# 三、 模型假设

- 不考虑海水密度变化引起的探测波偏折，即假设探测波在海水中严格沿直线传播。

- 不考虑测量船转弯和速度增加、减小造成的误差，即假设任何时刻探测波所形成平面均与海平面垂直、与测线垂直。亦不考虑测量船因海上风浪引起的倾斜，即假设任何时刻探测波以测量船的铅垂线为轴向两边对称发出。
- 出于结果呈现的统一性，所有涉及覆盖长度/覆盖面积/覆盖率的问题结果都以投影到海平面长度/面积等来表示。
- 问题三中计算总航程时，只计算测量时的测线长度，不考虑调头、转弯等非测量时的路程。

# 四、 符号说明

| 符号 | 说明 | 单位 |
|---|---|---|
| $W$ | 多波束测深条带的覆盖宽度 | m |
| $\theta$ | 换能器开角 | ° |
| $D$ | 测量船所处位置的海水深度 | m |
| $\eta$ | 相邻多波束带重叠率 | |
| $\alpha$ | 海底坡面角度 | ° |
| $v$ | 两波束带重叠宽度 | m |
| $x_1$ | 波束带轴线左侧覆盖宽度 | m |
| $x_2$ | 波束带轴线右侧覆盖宽度 | m |

# 五、 问题一模型

## 5.1 模型的建立

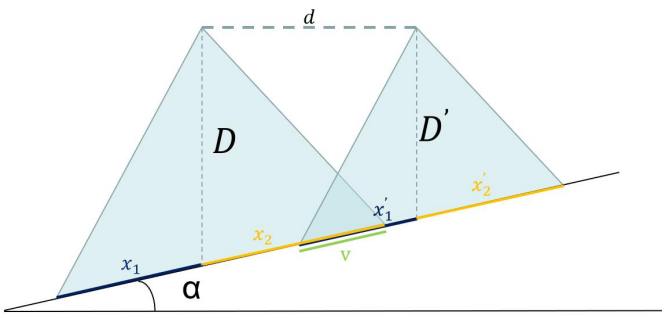


**图 3　问题一图示**

如图3所示，测量船于水深 $D$ 处发出探测波，形成横向扇面角为 $\theta$ 的全覆盖水深条带。作 $\theta$ 的角平分线，设多波束带左右区域在坡面上的长度分别为 $x_1, x_2$。根据三角形几何关系可得

$$\begin{aligned} x_1 &= \frac{D\tan\frac{\theta}{2}}{\cos\alpha - \sin\alpha\tan\frac{\theta}{2}} \\ x_2 &= \frac{D\tan\frac{\theta}{2}}{\cos\alpha + \sin\alpha\tan\frac{\theta}{2}} \end{aligned} \tag{1}$$

而多波束带的覆盖宽度应为 $x_1, x_2$ 之和在水平面的投影长度

$$W = (x_1 + x_2)\cos\alpha \tag{2}$$

联立 (1)(2) 式即得

$$W = 2D\frac{\cos^2\alpha\sin\theta}{\cos 2\alpha + \cos\theta} \tag{3}$$

设测量船在下一位置发出多波束带时与原位置距离为 $d$，海水深度为 $D'$。则根据几何关系可得

$$D^{'} = D - d\tan\alpha \tag{4}$$

同理，设右侧测线的多波束带左右区域在坡面上的长度分别为 $x_1', x_2'$，则可得

$$\begin{aligned} x_1^{'} &= \frac{D^{'}\tan\frac{\theta}{2}}{\cos\alpha - \sin\alpha\tan\frac{\theta}{2}} \\ x_2^{'} &= \frac{D^{'}\tan\frac{\theta}{2}}{\cos\alpha + \sin\alpha\tan\frac{\theta}{2}} \end{aligned} \tag{5}$$

设两三角形区域在坡面上的重叠区域长度为 $v$，易知

$$v = x_2 + x_1^{'} - \frac{d}{\cos\alpha} \tag{6}$$

考虑到重叠率应为重叠区域长度 $v$ 在水平面的投影长度与两测线间距 $d$ 之比，故

$$\eta = \frac{v\cos\alpha}{d} \tag{7}$$

联立 (1)(4)(5)(6)(7) 式即得相邻条带覆盖率为

$$\eta = \frac{\cos^2\alpha(1 + \cos\theta + \sin\theta(\tan\alpha - 2D/d))}{\cos 2\alpha + \cos\theta} \tag{8}$$

### 5.2 模型的求解

根据上述推导，在测线距中心点距离为 $d$ 时，海水深度为

$$D' = D - d\tan\alpha \tag{9}$$

多波束带覆盖宽度数学模型为

$$W = 2D\frac{\cos^2\alpha\sin\theta}{cos2\alpha+\cos\theta} \tag{10}$$

相邻波束带重叠率数学模型为

$$\eta = \frac{\cos^2\alpha(1+\cos\theta+\sin\theta(\tan\alpha-2D/d))}{\cos 2\alpha+\cos\theta} \tag{11}$$

代入 $\theta = 120°, \alpha = 1.5°, D = 70\text{m}$ 可得表1结论。对表格中数据绘制折线图4，可知深水区覆盖宽度 $W$ 远大于浅水区，重叠率 $\eta$ 略大于浅水区。由浅入深，覆盖宽度增长速率相比于重叠率较快。

**表 1　问题一求解结果**

| 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 | 90.948 | 85.711 | 80.474 | 75.237 | 70 | 64.763 | 59.526 | 54.288 | 49.051 |
| 覆盖宽度/m | 315.705 | 297.526 | 279.346 | 261.166 | 242.987 | 224.807 | 206.628 | 188.338 | 170.269 |
| 与前一条测线的重叠率/% | —— | 34.7 | 30.6 | 25.9 | 20.5 | 14.3 | 7.1 | −1.5 | −11.8 |

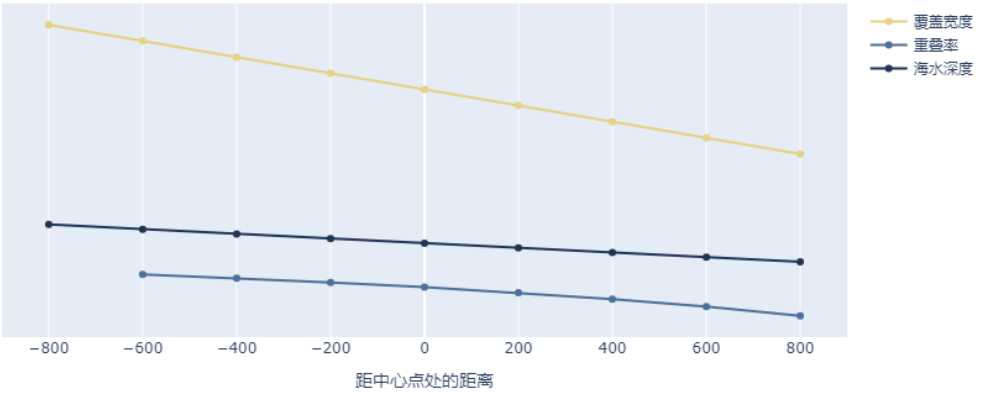


**图 4　问题一结果可视化：距中心点不同位置处的海水深度、覆盖宽度及重叠率**

# 六、 问题二模型

## 6.1 模型的建立

由于波束带位于与测线方向垂直的平面，故过测量点作垂直于测线的平面，如图 2 虚线所示。该平面与海底坡面的交线在海平面上的投影即为波束带的覆盖宽度。从几何

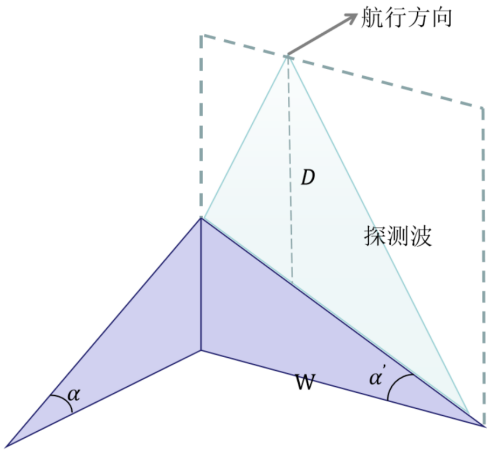


**图 5　问题二图示**

上看，这条交线在局部可以看作一个等效坡角为 $\alpha'$ 的新的斜坡。基于问题一的数学模型，此时问题转化为求该交线。

由测线方向与海底坡面的法向在水平面上投影的夹角为 $\beta$ 可知

$$\alpha' = \arctan\left|\tan\alpha \sin\beta\right| \tag{12}$$

设探测波形成的波束带于轴线两侧在海底坡面的覆盖长度分别为 $x_1''$，$x_2''$。由问题一 (1) 式可知

$$\begin{aligned} x_1'' &= \frac{D\tan\frac{\theta}{2}}{\cos\alpha\prime - \sin\alpha\prime\tan\frac{\theta}{2}} \\ x_2'' &= \frac{D\tan\frac{\theta}{2}}{\cos\alpha\prime + \sin\alpha\prime\tan\frac{\theta}{2}} \end{aligned} \tag{13}$$

则覆盖宽度

$$W = (x_1'' + x_2'')\cos\alpha\prime \tag{14}$$

联立(12)(13)(14)可得

$$W = \frac{2D\sin\theta}{(\cos\theta - 1)\left|\sin\beta\tan\alpha\right|^2 + \cos\theta + 1} \tag{15}$$

### 6.2 模型的求解

设测量船所在位置海水深度为 $D'$，根据几何关系可得

$$D' = D - d\sin(\beta - \frac{\pi}{2})\tan\alpha \tag{16}$$

由上述推导知此时覆盖宽度的数学模型为

$$W = \frac{2D'\sin\theta}{(\cos\theta - 1)\left|\sin\beta\tan\alpha\right|^2 + \cos\theta + 1} \tag{17}$$

代入 $\theta = 120°, \alpha = 1.5°, D = 70\text{m}$（注意数据中 $d$ 的单位为 nmile，需进行单位换算 1 nmile=1852 m），计算结果见表2。

**表 2　问题二求解结果**

| 覆盖宽度 (m) / 船距海域中心的距离 (nmile) / 测线方向夹角 (°) | 0 | 0.3 | 0.6 | 0.9 | 1.2 | 1.5 | 1.8 | 2.1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 415.692 | 466.091 | 516.490 | 566.889 | 617.288 | 667.686 | 718.085 | 768.484 |
| 45 | 416.120 | 451.794 | 487.468 | 523.142 | 558.816 | 594.491 | 630.165 | 665.839 |
| 90 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 |
| 135 | 416.120 | 380.446 | 344.772 | 309.098 | 273.424 | 237.750 | 202.076 | 166.402 |
| 180 | 415.692 | 365.293 | 314.894 | 264.496 | 214.097 | 163.698 | 113.299 | 62.900 |
| 225 | 416.120 | 380.446 | 344.772 | 309.098 | 273.424 | 237.750 | 202.076 | 166.402 |
| 270 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 | 416.549 |
| 315 | 416.120 | 451.794 | 487.468 | 523.142 | 558.816 | 594.491 | 630.165 | 665.839 |

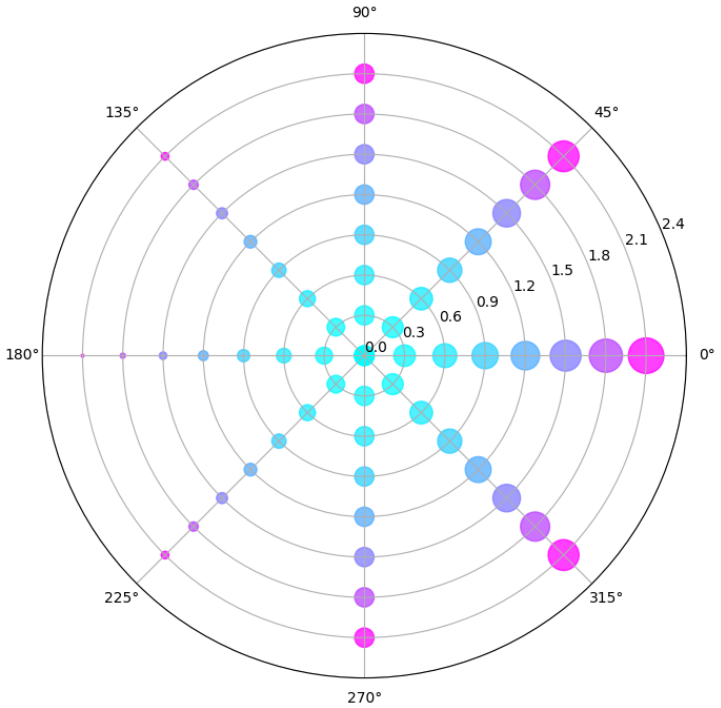


**图 6　问题二结果可视化**

对结果进行可视化处理，绘制极坐标图6。该图中，小圆圈的半径完全正比于在该坐标处计算得到的覆盖宽度。于是从图像中我们可以观察出，如果沿着垂直于海底等深线方向规划测线，在深度较浅的区域可覆盖的宽度会急剧降低。在规划下一条测线时，为了在浅水区保持覆盖率，势必要在深水区有很高的重叠率，并且相邻测线的距离很短，从而对同一区域需要布置更多条测线。从以上分析不难得出，沿着 $\beta = 0°$ 布置测线比较劣的规划策略；而沿着 $\beta = 90°$ 布置测线会尽可能减少，这为第三小问的求解策略奠定了基础。

# 七、 问题三模型

## 7.1 模型的建立

本问题是一个带约束的规划问题,要求完全覆盖海域,且将重叠率限制在 $10\%-20\%$ 的范围内，在此基础上求解最小测线长度。

## 7.2 模型的求解

问题二的结论启发我们，对于这样坡度均匀的海底面，规划测线使得其平行于坡面的等高线（在本题中意味着测线为南北走向），能更好地利用探测范围，减小重叠率。由此，我们只需要确定各个测线起点在东西走向上的坐标，并将所有测线设计为南北走向即可[1]。于是原问题转化为一个一维规划问题。规划、贪心算法如下：

### 7.2.1 首条测线的确定

为尽可能减少测量范围因超出待测海域带来的“浪费”，我们期望第一条测线应尽可能覆盖到待测海域边缘。采用二分法直接试探该测线所处的位置，即可得到第一条测线的位置。

### 7.2.2 其余测线的确定

在第一条测线的基础上，我们假设第 $n$ 条测线已经确定，考虑第 $n+1$ 条测线的规划。贪心算法指出，使得总测线条数第 $n+1$ 条测线的最优方案一定是在其与第 $n$ 条测线重叠率恰为 $10\%$ 时出现。

该贪心算法的正确性可通过如下方式证明：注意到测线覆盖的范围

$$s = x_2 + x_1' - v$$

[1]此外，参考文献 [2] 作为具有权威性的技术标准，也指出了在进行多波束测深系统测量时测线应当平行于海底等高线的若干条平行直线，本方案及接下来问题四的方案设计都满足本标准。

重叠率越高，意味着在第 $n$ 条测线已经确定的情况下，第 $n+1$ 条测线应该更靠近第 $n$ 条，也即 $d$ 更小。下面探究 $d$ 和第二条波束带有效覆盖宽度的变化关系。若固定第 $n$ 条测线，另一条测线与之距离 $d$ 变化时：$d$ 减小，导致 $D'$ 变大，则第二条测线覆盖宽度变大，可能导致有效覆盖长度变大；而 $d$ 增大，导致两波束带区域的重叠长度减小，也可能导致有效覆盖长度变大。设相邻测线的有效覆盖宽度为 $W_0$，则

$$W_0 = d + x_2' \cos\alpha - x_2 \cos\alpha \tag{18}$$

联立式 (1)(4)(5)(18) 可得

$$W_0 = d \cos\alpha \cos\left(\frac{\theta}{2}\right) \sec\left(\alpha - \frac{\theta}{2}\right) \tag{19}$$

对 $W_0$ 关于 $d$ 求偏导可得

$$\frac{\partial W_0}{\partial d} = \cos\alpha \cos\left(\frac{\theta}{2}\right) \sec\left(\alpha - \frac{\theta}{2}\right) \tag{20}$$

式 (20) 在 $\alpha = 1.5°$，$\theta = 120°$ 处的取值为 0.181389。$W_0$ 关于 $d$ 单调递增，故 $d$ 更小时一定会导致第 $n+1$ 条线有效覆盖的边界更小。因而贪心地使每条测线与上一条的重合率尽可能小是正确的选择。

由于证明过程中同时体现了测线边界关于测线起点位置的单调性，我们可以连续地使用二分法，依次求出各个测线的起点，直到完全覆盖整个待测海域。此时给出的即为最优的测线规划方案，也即原规划问题的解。

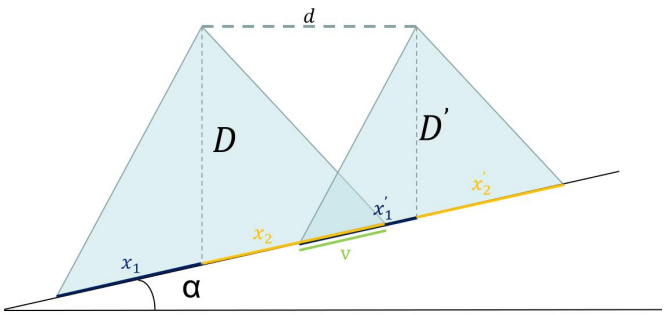


**图 7　问题三求解图示**

### 7.3 模型求解结果展示

本问题利用 C++ 语言编写程序（程序源代码见附录和支撑材料），利用以上连续进行二分法的思路给出了最优测线规划，共 34 趟航程，总路程为 68nmile。结果见附件 A 表4和图9（结果给出的是南北走向的测线，并以海面中心为坐标原点，西边界的坐标记为 $-3704.0$m，东边界的坐标记为 3704.0m）


**图 8　问题三算法流程图**

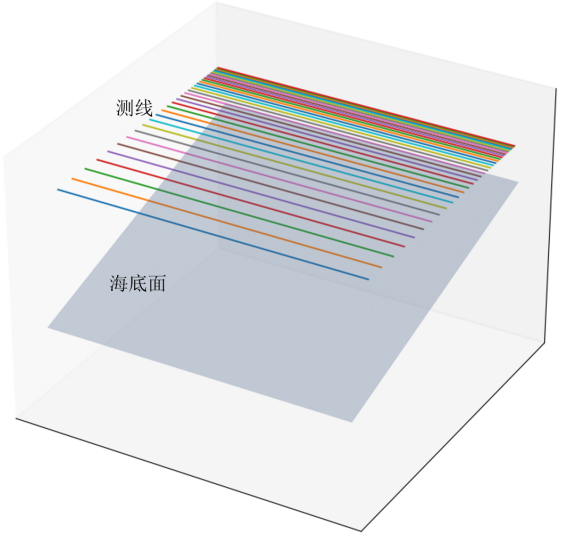


**图 9　问题三测线布设图**

# 八、 问题四模型

## 8.1 模型的建立

首先依据附件中各位置处的海底深度绘制海水深度等值线图海底平面图10与海底地形图11、12。可知该海域整体呈现“东浅西深”，“南浅北深”，西北角最深的地形。

对于本问题，我们可以将附件中的测深数据图建立为格点模型，相当于对原本连续的海底深度数据进行离散化处理。

出于实际测绘中对不漏测的强烈要求，我们首先指定不漏测作为强约束条件，并以

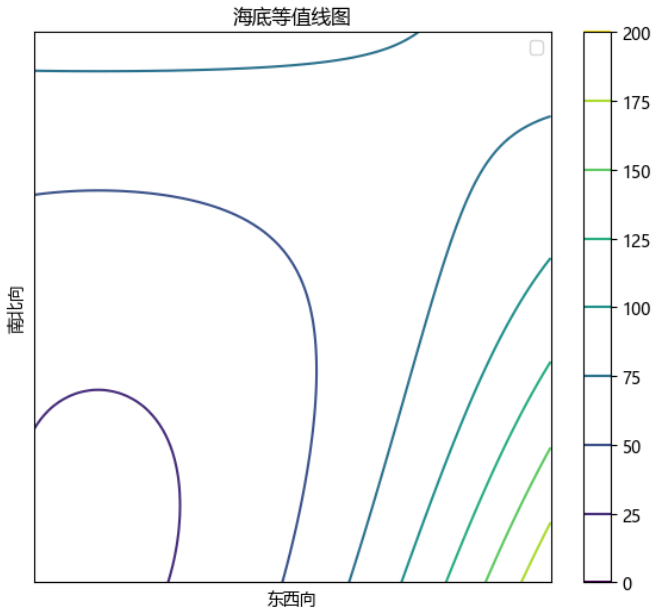


**图 10　海底等值线图**

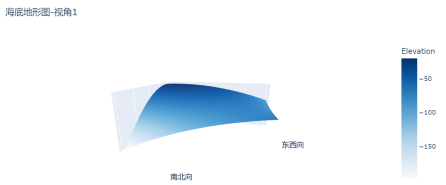


**图 11　海底地形图-视角 1**

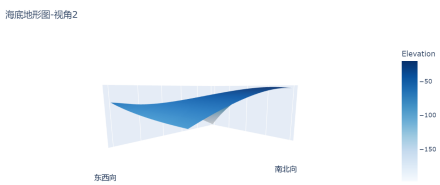


**图 12　海底地形图-视角 2**

重合率尽可能低作为另一较弱的约束条件，在以上约束条件下规划使得测线总长度最短。

## 8.2 模型的求解

对于上面建立的格点模型，由于其不再具有东西或南北走向的等高线，问题三中的单调性条件自然不再成立，但本问题中我们仍然考虑将测线规划为若干条东西或南北走向的直线，原因是这样的处理对于格点模型计算简便，容易良好地定义、分析其重叠率，同时也避免了曲线或斜线导致边界处漏测。

由于单调性条件不成立，将二分方法修改为朴素的枚举法。仍然考虑问题三中的贪心法：

### 8.2.1 探测点标记

对于每一测线，在垂直其方向上标记所有能够被探测的点。分析可知，对同一点发出的探测波，各个格点处能够被探测的充分必要条件是：距离波源由近到远，被探测点

关于波源形成张角的正切值构成一不下降子序列（否则，距离远但正切较小的点会被遮挡），且不超过探测角半值的正切 $\tan\frac{\theta}{2}$，因此我们可以将波源沿着测线方向移动，对于每一个格点都标记出其能够探测到的格点，从而实现对整条测线的标记。

### 8.2.2 有效性检验

分析该测线与前一条测线的重叠率和漏测情况。只需遍历两测线间的所有格点，检测其标记次数是否为 0（代表漏测）和是否大于 1（代表重复测量）。

### 8.2.3 间距优化

在确保不漏测的情况下，通过枚举试探，尽可能使两条测线的间距变大，直到试探到恰好不漏测情况为止。重复上述过程直到测量范围完全覆盖整个海域。

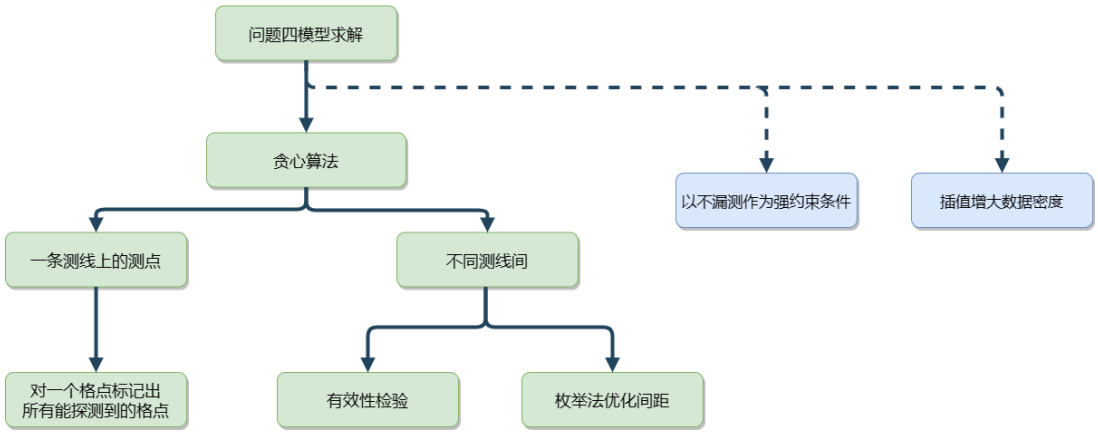


**图 13　问题四算法流程图**

另外，考虑到原始数据中，每两格点之间的距离为 0.02nmile = 37.04m，在浅水区（深度约 20m）很容易因为格点距离过大使得计算产生严重误差。为此，我们利用线性插值法，将每格点间的距离扩大为 0.0025nmile = 4.63m，以便接下来的精确计算。值得一提的是，在现实问题中，如果考虑船本身宽度和导航精确程度，4.63m 的精度已经可以很好地满足需要了，因此我们只需要将测线固定在格点上即可。

## 8.3 模型求解结果展示

本问题利用 C++ 语言编写程序（程序源代码见附录和支撑材料），分别按照测线沿东西走向和沿南北走向进行试探，并计算出一组符合题目要求的解。

为了更真实地考虑测量情况，我们选取了张角 $\theta$ 从 80° 到 160° 的多种情况进行测量，最终规划的路径相关指标如下表：

**表 3　不同换能器开角下的测线布设指标**

| 换能器开角 $\theta$/° | 总航程/nmile | 覆盖率 $\eta > 20\%$ 路程/nmile | 漏测区域占比/% |
|---|---|---|---|
| 80 | 625 | 343.035 | 0 |
| 100 | 430 | 213.360 | 0 |
| 120 | 300 | 153.555 | 0 |
| 140 | 190 | 94.665 | 0 |
| 160 | 90 | 42.460 | 0 |

# 九、 模型的评价

## 9.1 模型的优点

- 在问题一、二中，使用没有误差的精确几何模型进行推导，没有使用 $\alpha$ 的小角度近似，这使得本模型在海底倾角 $\alpha$ 较大时也适用，拓宽了本模型的适用范围。
- 在问题三、四中，以避免漏测作为强约束条件，使得测线规划能够满足完整覆盖整个待测海域。
- 在问题三中，换能器覆盖范围关于测线位置的单调性很好地诠释了本模型中贪心法和二分法的本质，即通过减小重叠率达到局部最优的方法，实现测线数量尽可能少的全局最优。
- 问题四采用了插值法增大数据密度，使得格点划分间距更小，从而使计算更加精确。另一方面，我们适当控制了其精确程度能够与实际行船时的控制尺度相匹配，也避免了因为格点划分过密导致计算时间过长，浪费计算资源。

## 9.2 模型的缺点

- 问题四中求解测线时，仍采用“平行线”式布局，原因是“非平行线”式布局难以定义覆盖率。加之我们要求尽可能避免漏测，使得深水区的测线布设受到浅水区的影响，可能造成一定程度上资源的浪费。
- 问题四中模型只能接受地理位置分布均匀的数据点，而难以处理分布不规律的数据集，如果需要处理这样的数据必须使用更高级的插值方法。
- 实际海域中往往会出现非平滑海底的情况，如粗糙地形图14（冲沟）、蚀余台地图15和斑状海底16，这是线性插值方法所难以模拟的。模型需要进一步的优化才能更好地解决这些情况下的约束问题。

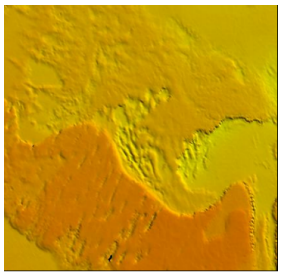

**图 14　粗糙海底**

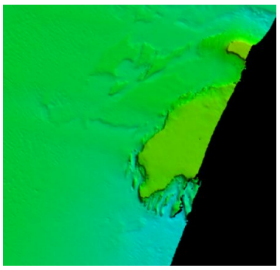

**图 15　蚀余海底**

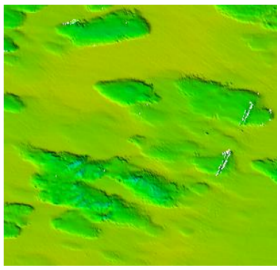

**图 16　斑状海底**

### 9.3 模型的推广

考虑到本问题作为一个带有约束的连续路线规划问题，可以在当前模型的基础上引入更好的算法。一方面，可以引入三次插值等更高次插值方式，更真实地还原海洋深度情况，从而使得测线设置更精准；另一方面，如果可以不局限于平行直线式的测线布置，也可以尝试利用启发式算法（如 A* 算法）或其他优化算法（如遗传算法、机器学习方法等）来进行连续寻线，从而减少因浅水区制约导致的深水区重叠率过高问题。

## 参考文献


[1] 高慎明, 王晓云. 广西某海底管道项目的测线布设方法与优化 [A]. 中国石油学会石油工程专业委员会、中国石油学会石油工程建设专业标准委员会. 中国石油石化工程建设创新发展大会–石油天然气勘察技术第二十六次技术交流研讨会论文集 [C]. 中国石油学会石油工程专业委员会、中国石油学会石油工程建设专业标准委员会: 中国建筑学会工程勘察分会,2018:4.

[2] 中华人民共和国交通运输行业标准 JT / T 790—2010: 多波束测深系统测量技术要求.

[3] 丁建棣. 海底石油管道的声学检测方法及其三维可视化系统研究 [D]. 天津大学,2018.

# 附录 A　问题三结果

## 表 4　问题三测线布设

| 测线序号 | 测线坐标（以海域中心点为坐标原点） | 左覆盖宽度 $x_1$/m | 右覆盖宽度 $x_2$/m |
|---|---|---|---|
| 1 | -3345.4782 | 358.6447 | 327.5233 |
| 2 | -2753.8453 | 330.5265 | 301.8450 |
| 3 | -2208.6250 | 304.6141 | 278.1812 |
| 4 | -1706.3130 | 280.7410 | 256.3797 |
| 5 | -1243.6591 | 258.7526 | 236.2994 |
| 6 | -817.0736 | 238.4785 | 217.7846 |
| 7 | -423.9091 | 219.7928 | 200.7203 |
| 8 | -61.2199 | 202.5555 | 184.9787 |
| 9 | 272.7291 | 186.6841 | 170.4846 |
| 10 | 580.8813 | 172.0387 | 157.1100 |
| 11 | 864.6614 | 158.5516 | 144.7933 |
| 12 | 1125.8463 | 146.1384 | 133.4572 |
| 13 | 1366.6928 | 134.6918 | 123.0039 |
| 14 | 1588.5865 | 124.1460 | 113.3732 |
| 15 | 1793.3244 | 114.4155 | 104.4871 |
| 16 | 1982.0015 | 105.4484 | 96.2981 |
| 17 | 2156.0605 | 97.1760 | 88.7435 |
| 18 | 2316.3811 | 89.5565 | 81.7852 |
| 19 | 2464.1328 | 82.5344 | 75.3724 |
| 20 | 2600.3211 | 76.0618 | 69.4616 |
| 21 | 2725.8139 | 70.0976 | 64.0149 |
| 22 | 2841.3641 | 64.6059 | 58.9997 |
| 23 | 2947.8252 | 59.5462 | 54.3791 |
| 24 | 3046.0491 | 54.8779 | 50.1159 |
| 25 | 3136.5130 | 50.5785 | 46.1896 |
| 26 | 3219.8506 | 46.6178 | 42.5725 |
| 27 | 3296.6923 | 42.9658 | 39.2374 |
| 28 | 3367.5484 | 39.5982 | 36.1621 |
| 29 | 3432.8778 | 36.4933 | 33.3266 |
| 30 | 3493.0160 | 33.6352 | 30.7165 |
| 31 | 3548.4574 | 31.0002 | 28.3102 |
| 32 | 3599.6217 | 28.5686 | 26.0895 |
| 33 | 3646.7295 | 26.3297 | 24.0450 |
| 34 | 3690.1093 | 24.2680 | 22.1622 |

# 附录 B　文件列表

**表 5　支撑材料文件列表**

| 文件名 | 文件描述 |
|---|---|
| B1.cpp | 解决问题 1 所需程序的 C++ 语言源代码 |
| B2.cpp | 解决问题 2 所需程序的 C++ 语言源代码 |
| B3.cpp | 解决问题 3 所需程序的 C++ 语言源代码 |
| B4.cpp | 解决问题 4 所需程序的 C++ 语言源代码 |
| interp.py | 对问题 4 原始数据进行线性插值所需程序的 Python 语言源代码 |
| result1.xlsx | 问题 1 求解结果 |
| result2.xlsx | 问题 2 求解结果 |
| result4.txt | 问题 4 求解结果，包括各测线具体坐标（以待测海域最西侧为坐标原点） |

# 附录 C　代码

B1.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
double s[10]={0,-800,-600,-400,-200,0,200,400,600,800};//distance from central of the sea
double depth[10];//depth at certain distance
double w[10];//cover distance. should be calculated downto the ground with cos(alpha).
const double Pi = 3.1415926535897932;//Pi
inline double left_width(double D,double alpha,double theta)//all angles are in degrees, and will be trans into rad when calculating.
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) - sin(alpha*Pi/180.0)*tan((theta/2.0)*Pi/180.0));
}
inline double right_width(double D,double alpha,double theta)
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) + sin(alpha*Pi/180.0)*tan((theta/2.0)*Pi/180.0));
}
int main()
{
    int n = 9;
    double alpha = 1.5;
```

```
    double theta = 120.0;
    depth[5] = 70.0;
    for(register int i=1;i<=n;i++)
    {
        depth[i] = depth[5] - s[i] * tan(alpha*Pi/180.0);
        printf("%lf    ",depth[i]);
    }
    printf("\n");
    for(register int i=1;i<=n;i++)
    {
        w[i] = left_width(depth[i],alpha,theta) + right_width(depth[i],alpha,theta)
    ;
        printf("%lf    ",w[i]*cos(alpha*Pi/180.0));
    }
    printf("\n-             ");
    for(register int i=2;i<=n;i++)
    {
        double cover_width = left_width(depth[i],alpha,theta) + right_width(depth[i
    -1],alpha,theta) - (s[i] - s[i-1])/cos(alpha*Pi/180.0);
        double cover_ratio = cover_width / (left_width(depth[i],alpha,theta) +
    right_width(depth[i-1],alpha,theta));
        printf("%lf    ",cover_ratio);
    }
    return 0;
}
```

B2.cpp

```
#include<bits/stdc++.h>
using namespace std;
double s[9]={0,0,0.3,0.6,0.9,1.2,1.5,1.8,2.1};//distance from central of the sea
double beta[10]={0,0,45.0,90.0,135.0,180.0,225.0,270.0,315.0};
double w[10];//cover distance. should go with cos(alpha_prime)
const double Pi = 3.1415926535897932;//Pi
inline double left_width(double D,double alpha,double theta)//all angles are in
    degrees, and will be trans into rad when calculating.
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) - sin(alpha*Pi/180.0)
    *tan((theta/2.0)*Pi/180.0));
}
inline double right_width(double D,double alpha,double theta)
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) + sin(alpha*Pi/180.0)
    *tan((theta/2.0)*Pi/180.0));
}
int main()
{
    freopen("B2.out","w",stdout);
```

```
    int n = 8;
    for(register int i=1;i<=n;i++)
        s[i] = s[i]*1852.0; //nmile to metre
    double alpha = 1.5;
    double theta = 120.0;
    double depth_zero = 120.0;
    for(register int i=1;i<=n;i++) // angle
    {
    for(register int j=1;j<=n;j++) // distance
        {
            double alpha_prime = 180.0 / Pi * atan(fabs(tan(alpha * Pi/180.0) * sin(beta[i] * Pi/180.0)));
            double depth = depth_zero - s[j] * sin((beta[i] - 90)*Pi/180.0) *tan(alpha * Pi/180.0);
            double width = left_width(depth,alpha_prime,theta) + right_width(depth,alpha_prime,theta);
            printf("%.3lf,",width * cos(alpha_prime*Pi/180.0));
        }
        printf("\n");
    }
    return 0;
}
```

B3.cpp

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
const double lrange = 0.100, rrange = 0.102; //left and right range for eta
const double Pi = 3.1415926535897932;//Pi
const double eps = 1e-7;

inline double left_width(double D,double alpha,double theta)//all angles are in degrees, and will be trans into rad when calculating.
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) - sin(alpha*Pi/180.0)*tan((theta/2.0)*Pi/180.0));
}
inline double right_width(double D,double alpha,double theta)
{
    return D * tan((theta/2.0)*Pi/180.0)/(cos(alpha*Pi/180.0) + sin(alpha*Pi/180.0)*tan((theta/2.0)*Pi/180.0));
}
inline double depth(double s,double alpha)//calculate depth at distance s(metre), which ranges from -2.0*1852 to +2.0*1852
{
    return 110.0 + s * tan(alpha * Pi /180.0);
}
```

```
double position[200];
int main()
{
    double left = -2.0*1852.0;
    double right = +2.0*1852.0;
    double ll = left;
    double rr = right;
    double mid = (left + right)/2.0;
    double alpha = 1.5,theta=120.0;
    //find the first line
    while(true)
    {
        mid = (ll + rr)/2.0;
        double left_bound = mid - left_width(depth(mid, alpha),alpha,theta)  * cos(alpha * Pi/180.0);
        if(fabs(left_bound -left) <= eps)
        {
            position[1] = mid;
            break;
        }
        else
        {
            if(left_bound -left< eps) ll = mid;
            if(left_bound -left> eps) rr = mid;
        }
    }
    //then
    printf("%9.4lf\n",position[1]);
    int total = 1;
    double last = position[total];
    double right_bound = last + right_width(depth(last, alpha),alpha,theta) * cos(alpha * Pi/180.0);//find the first line's right boundary as the new left boundary
    while(right_bound - right <=eps)
    {
        total ++;
        double ll = right_bound;
        double rr = right;
        while(rr - ll >eps)
        {
            mid = (ll + rr)/2.0;
            double left_bound = (mid - left_width(depth(mid, alpha),alpha,theta)) * cos(alpha * Pi/180.0);
            double cover_ratio = 1.0 - (mid - last)/((left_width(depth(mid, alpha),alpha,theta) + right_width(depth(last, alpha),alpha,theta)) * cos(alpha * Pi/180.0) );
            if(lrange <= cover_ratio && rrange >= cover_ratio)
```

```
            {
                position[total] = mid;
                last = mid;
                right_bound = last + right_width(depth(last, alpha),alpha,theta) *
    cos(alpha * Pi/180.0);
                break;
            }
            else
            {
                if (cover_ratio < lrange) rr = mid;
                if (cover_ratio > rrange) ll = mid;
            }
        }
    }
    freopen("B3.out","w",stdout);
    printf("total = %d\n",total);
    for(register int i=1;i<=total;i++)
    {
        printf("%d line is at %9.4lf\n",i,position[i]);
        printf("left width = %9.4lf, right width = %9.4lf\n",left_width(depth(
    position[i],alpha),alpha,theta),right_width(depth(position[i],alpha),alpha,theta
    ));
    }
    return 0;
}
```

B4.cpp

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
const double lrange = 0.000, rrange = 0.00100; //left and right range for eta
const double Pi = 3.1415926535897932;//Pi
const double eps = 1e-7;
int n=2000,m=1600;
const double deltaX = 1852.0 * 4.0 /m;
const double deltaY = 1852.0 * 5.0 /n;

double depth[3000][2100];
int lines=0;
int position[3000];
int detected[3000][2100];
int tempo[3000][2100];

int empty(int map1[3000][2100])
{
    for(register int i=n;i>=0;i--)
    for(register int j=m;j>=0;j--)
```

```
    {
        if(map1[i][j] ==0) return -1;
    }
    return 0;
}
int map1[3000][2100];
int detect(int fp[3000][2100],int last,int pos,double theta,int mode)
{
    int repeat = 0;     //number of repeatedly detected blocks in this round
    int total = 0;      //number of detected blocks in this round
    for(register int i=0;i<=n;i++)
    for(register int j=0;j<=m;j++)
        map1[i][j] = fp[i][j];
    for(register int i=0;i<=n;i++)
    {
        double maxTan = 0;
        for(register int j=pos;j>=0;j--)
        {
            double slope = deltaX * (pos-j) / depth[i][j];
            if (slope < maxTan) continue;                           // a hill.
            if (slope > tan((theta/2.0) * Pi /180.0)) break;        // beyond the
    detector.
            maxTan = slope;
            if(map1[i][j] != 0)  repeat++;
            map1[i][j]++;
            total++;
        }
        maxTan = 0;
        for(register int j=pos+1;j<=m;j++)
        {
            double slope = deltaX * (j-pos) / depth[i][j];
            if (slope < maxTan) continue;                           // a hill.
            if (slope > tan((theta/2.0) * Pi /180.0)) break;        // beyond the
    detector.
            maxTan = slope;
            if(map1[i][j] != 0)  repeat++;
            map1[i][j]++;
            total++;
        }
    }
    for(register int i=0; i<=n;i++)
    for(register int j=last+1;j<=pos;j++)
        if (map1[i][j]==0) return -1;                               // no blocks
    should be left!

    double cover_ratio = 1.0 * repeat / (1.0* repeat + 1.0 * (pos - last)*n);

```

```
    if ((mode ==0)||((lrange <= cover_ratio) && (cover_ratio <= rrange)))
    {
        lines++;
        position[lines] = pos;
        for(register int i=0;i<=n;i++)
        for(register int j=0;j<=m;j++)
            detected[i][j] = map1[i][j];
        return 0;
    }
    if(rrange < cover_ratio) return 1;  //repeatedly cover too much.
}


int main()
{
    freopen("input.txt","r",stdin);
    for(register int i=0;i<=n;i++)
    for(register int j=0;j<=m;j++)
    {
        scanf("%lf",&depth[i][j]);
    }
    memset(detected,0,sizeof(detected));
    memset(position,0,sizeof(position));
    memset(tempo,0,sizeof(tempo));
    int last = n * m;
    double theta = 160;
    for(register int j = 1; j<=m ; j++)                 // here find the first line
    {
        memset(tempo , 0 , sizeof(tempo));
        int flag = detect(tempo,0,j,theta,1);
        if(flag == -1)
        {
            lines = 0;
            memset(position , 0 , sizeof(position));
            memset(detected , 0 , sizeof(detected));
            detect(detected , 0 , j-1 , theta,1);
            break;
        }
    }
    int now = position[lines]+1;
    while((empty(detected)!=0) && (now<=m))
    {
        int flag =detect(detected,position[lines],now,theta,1);
        if (flag ==0)
        {
            now = position[lines]+1;
            printf("%d in\n",position[lines]);
```

```
        }
        if (flag ==1) now ++;
        if (flag ==-1)
        {
            detect(detected,position[lines],now-1,theta,0);
            now = position[lines]+1;
            printf("%d in\n",position[lines]);
        }
    }
    printf("all have %d\n",lines);
    freopen("B4.out","w",stdout);
    for(register int i=1;i<=lines;i++)
    {
        printf("%.3lf  ",4.0*position[i]/m);
    }
    int allrepeat=0;
    int lost = 0;
    for(register int i=2;i<=lines;i++)
    {
        int pre = position[i-1],aft = position[i];

        for(register int s=0;s<=n;s++)
        {
            int repeat = 0;
            for(register int j=pre+1;j<=aft;j++)
            {
                if(detected[s][j] == 0) lost++;
                if(detected[s][j] >1) repeat++;
            }
            double ratio = 1.0 * repeat / (1.0* repeat + 1.0 * (aft - pre));
            if (ratio >=0.200) allrepeat++;
        }
    }
    printf("\nlength is %.3lf nmiles:\n",5.0* lines);
    printf("lost is %d, repeat ratio length is %.3lf nmiles\n ",lost,deltaY*
    allrepeat/1852.0);
    return 0;
}
```

interp.py

```
from scipy.interpolate import interp2d
import numpy as np
import matplotlib.pyplot as plt
import csv

x = np.arange(0,4.02,0.02)
y = np.arange(0,5.02,0.02)
```

```
8
9  X, Y =np.meshgrid(x,y)
10 with open('table.csv','r') as file:
11     csv_reader = csv.reader(file)
12     data_array = []
13     for row in csv_reader:
14         data_array.append(row)
15
16 Z = np.array(data_array,dtype='float64')
17 print(Z)
18
19 x1 = np.arange(0,4.02,0.0025)
20 y1 = np.arange(0,5.02,0.0025)
21
22 f1 = interp2d(X , Y , Z , kind='linear')
23 Z1 = f1(x1,y1)
24 b = np.savetxt('newtable.txt', Z1, fmt='%.2f', delimiter=' ')
```