空洞探测建模实验报告

江鹏飞　PB20030896　ID 号：075

杨哲骏　PB20611836　ID 号：107

2023 年 5 月 28 日

摘要

山体，大坝，隧洞等的内部结构可以用弹性波来测量，由于数据量一般很大，就必须要有符合实际的数学模型和算法。基于上述背景，本次实验便是一个简化的二维空间探测问题，针对该问题，我们提出了若干模型并进行优化。

对于问题一，首先我们分析原始数据发现可以由**几何模型**确定空洞位置和数量，虽然该方法并不具有普适性，但其解出的空洞位置较精确，可以与后续模型的结果进行对比；然后我们基于最直观的想法建立了**代数模型**，即通过求解出直线 P_iQ_j 或 R_iS_j 在正方形网格中各个小单元内的距离，结合已给的数据，建立**线性方程组**，并通过**最小二乘法**求解该超定方程组；有了代数模型的启发，我们还通过**贪心算法**来确定空洞位置，贪心策略为选择的空洞位置及数目使得理论时间与实际时间的误差最小，算法复杂度为多项式级；然后我们又对代数模型进行改进，使用**染色法**更为精细地确定了空洞位置，且染色法的退化情况与我们使用几何模型分析得到的解完全一致；最后我们建立**概率模型**，假设每个正方形以一定概率为空洞，通过极大似然估计来估计这个概率，即用经过正方形的波线上空洞的比例来估计正方形为空洞的概率。

对于问题二，我们先基于几何模型直接从理论上分析求解，同时通过代数模型得到更为详细的结果，也使用概率模型得到了较好的结果。

最后，我们对解决问题采用的模型进行了评价与总结。

关键字：线性方程组　最小二乘法　贪心算法　染色方法　极大似然估计

目录

一、问题重述 ····· **3**

1.1 背景资料 ····· 3

1.2 需要解决的问题 ····· 3

二、问题分析 ····· **3**

2.1 问题一分析 ····· 3

2.2 问题二分析 ····· 5

三、模型的假设 ····· **5**

四、符号说明 ····· **6**

五、模型的建立与求解 ····· **6**

5.1 问题一模型的建立及求解 ····· 6

5.1.1 几何模型 ····· 6

5.1.2 代数模型 ····· 8

5.1.3 贪心算法 ····· 11

5.1.4 代数模型的改进-染色法 ····· 13

5.1.5 概率模型 ····· 19

5.2 问题二模型的建立及求解 ····· 23

5.2.1 几何模型分析求解 ····· 23

5.2.2 代数模型 ····· 25

5.2.3 概率模型 ····· 27

六、模型的评价 ····· **27**

参考文献 ····· **28**

附录 A 代数模型算法 ····· **29**

附录 B 贪心算法 ····· **33**

附录 C 概率模型算法 ····· **37**

一、 问题重述

1.1 背景资料

山体、隧洞、坝体等的某些内部结构可用弹性波测量来确定. 一个简化问题可描述为: 一块由均匀介质构成的矩形平板内有一些充满空气的空洞在木板的两个邻边分别等距地设置若干波源, 在它们的对边对等地安放同样多的接收器, 记录弹性波由每个波源到达对边上每个接收器的时间, 根据弹性波在介质中和在空气中不同的传播速度, 来确定板内空洞的位置. 现考察如下的具体问题:

一块 240 m × 240 m 的平板, 在 AB 边等距地设置 7 个波源 $P_i(i = 1, 2, \cdots, 7)$, 在 CD 边对等地安放 7 个接收器 $Q_j(j = 1, 2, \cdots, 7)$, 记录由 P_i 发出的弹性波到达 Q_j 的时间 t_{ij} s ; 在 AD 边等距地设置 7 个波源 $R_i(i = 1, 2, \cdots, 7)$, 在 BC 边对等地安放 7 个接收器 $S_j(j = 1, 2, \cdots, 7)$, 记录由 R_i 发出的弹性波到达 S_j 的时间 τ_{ij} s . 已知弹性波在介质和空气中的传播速度分别为 2880 m/s 和 320 m/s, 且弹性波沿板边缘的传播速度与在介质中的传播速度相同。

1.2 需要解决的问题

需要解决的问题如下:

(1) 确定题中空洞的位置。

(2) 只根据由 P_i 发出的弹性波到达 Q_j 的时间 t_{ij}, 能确定空洞的位置吗? 讨论在同样能够确定空洞位置的前提下, 减少波源和接收器的方法。

二、 问题分析

2.1 问题一分析

问题一需要确定空洞的位置, 我们的思路图如下所示:

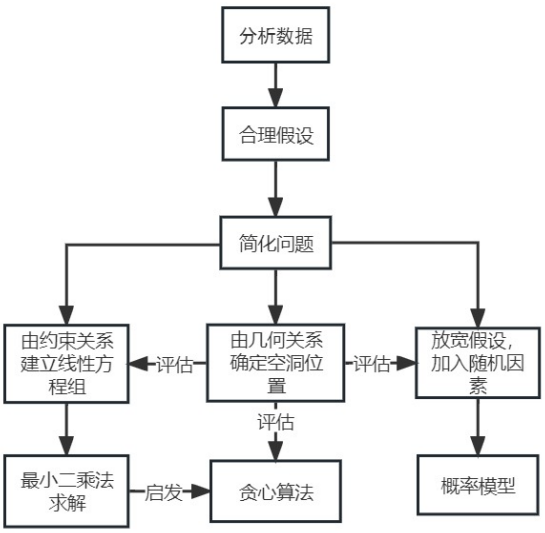


图 1　问题一流程图

对于问题一，确定空洞位置的方法多样，采用不同的假设方法，会有完全不同的难度，也有不同的解题思路。我们整体的模型主要有五种：

1. 几何模型：经过简单计算发现，存在一部分波线的传播时间小于相应区域全部是介质情况下需要的传播时间。这只能是因为测量误差。可以认为这部分波线经过的区域都是介质。从而可以通过作图的方法确定空洞的位置。

2. 代数模型：我们以图中 D 点为原点，建立坐标系，以给出的波沿直线传播时间，列出约束条件并建立线性方程组。我们发现该方程组为超定方程组，因此用最小二乘法解出近似解。此外，我们还可以将平板分的更细，如分成 12×12 或 18×18，经实验，划分的更细一定程度可以提高模型精度，但我们不能通过将平板划的足够细来不断提高精度。这是因为题中给出的波线只有 49×2 条，当网格被划分的过小时，会有很多小正方形没有一条波线经过它，也就没有一点关于小正方形的信息，要判断他们是介质或是空洞是不可能的，这也是该模型的局限之处。

3. 贪心算法：由代数模型我们得到启发，还是在 6×6 的方格上，若我们假设出空洞的数目及位置，那么我们便可以算出全部波线的传播时间，记该时间矩阵为 $T\prime$，我们计算其与题中已给的测量时间矩阵 T 的误差，并以此为贪心选择策略，计算出使误差最小的空洞数目及位置，当误差上升时算法终止。

4. 代数模型的改进-染色法：代数模型求得的解整体较粗糙，但我们可以利用其求出的中间结果，在 98 条波线中标记出一定不经过空洞的（标为红色）。然后我们计算出

12×12 划分中每个小方格内的红线长度和（红点个数），并按红点个数分成四个不同层级，将网格染成四种颜色，空洞所在位置即为其中一种颜色的方格。

5. 概率模型：假设空洞的分布在整个区域中服从均匀分布，将波线看作对于空洞的抽样。计算波线经过空洞的长度和总长度的比值的均值，得到 0.2641 ，则约有 **9.51** 个小正方形是空洞。为了便于计算，假设 6×6 的方格中每个方格以一定概率服从均匀分布，则可以估计第 n 个小正方形为空洞的概率。

2.2 问题二分析

对于问题二，如果移去了 R_i 发到 S_j 的波（$i, j = 1, 2, ...7$），我们猜测应该是不能确定平板的位置的：因为题中原本给出的数据就十分有限，此时如果还去掉了时间矩阵 τ，那么必然会导致平板上有许多区域我们获取不到其任何信息，对其是介质还是空气便无从分析；若我们要减少波源和接收器，同时还要确定空洞的位置，那么我们可以考虑设置一个误差标准，逐渐移去不同的波源和接收器，计算各个情况下的误差，在精度容许范围内，选择合适的若干种情况作为最终结果。

三、 模型的假设

- 所有的探测均在同一个平面内进行所谓的空洞即平面上的一个区域。
- 弹性波在波源与接收器之间沿直线传播；在空洞与板的交界处不考虑折射与反射现象的影响。
- 对弹性波传播时间的测量存在测量误差。
- 弹性波在介质和空气中的传播速度均为匀速。

  以上的假设是对该问题全局的假设，由于解决问题的方法不同，每个模型采取的不同假设将在各个部分分别介绍。

四、 符号说明

符号	意义
T	波源 P_i 发出的弹性波到达接收器 Q_j 的时间矩阵
τ	波源 R_i 发出的弹性波到达接收器 S_j 的时间矩阵
P_iQ_j	波源 P_i 出发的波到 Q_j 的总长度
R_iS_j	波源 R_i 出发的波到 S_j 的总长度
$d_{ij}^{(k)}$	波线 P_iQ_j 经过第 k 个小正方形的长度
D	所有波线在每个小正方形内长度拼接成的矩阵
$x[i]$	第 i 个方格是否为空洞
p_{ij}	波线 P_iQ_j 为空洞的概率

五、 模型的建立与求解

5.1 问题一模型的建立及求解

5.1.1 几何模型

如果我们对数据进行分析, 就会发现一些矛盾。波线即使与平板边平行也有 240 m, 如果不与平板的边平行, 长度大于 240 m, 波在介质中传播速度为 2880 m/s, 即使波线丝毫没有经过空洞, 完全在介质中传播最少也要 $240/2880 = 0.0833$(s), 但是传播时间表格中有十六个数据小于 0.0833 , 其中最小的一个为 0.0583 。造成这个现象的原因肯定是由于测量误差, 而且这些误差是将波运行时间缩短了。假设每一个小正方形或者全是介质或者全是空气, 若波线经过一个小正方形空洞一般要经过 40 m 以上的空气, 至少要多花 0.111 s 的运行时间。如果这批波的运行时间小于 0.0833 s 的波线经过空洞则测量误差将大于 0.14 s, 将是最小运行时间的 2.3 倍。在测量中这样大的误差使测量失去了意义, 因此可以认为这种情况是不可能发生的, 也就是这十六条波线都没有经过小正方形空洞, 经过的都是介质。

如果上述猜想是正确的, 则每条这样的波线至少可以排除 6 个小正方形是定洞, 它们的全体可能将大多数介质小正方形排除了, 甚至全排除了。下一步的工作就是逐个验证剩下的小正方形是否是空洞了。

为此可以先计算每条波线 (长度不全相同) 波全是在介质中运行的时间, 可以列表如下:

$t_{ij}(\tau_{ij})$	$Q_1\,(S_1)$	$Q_2\,(S_2)$	$Q_3\,(S_3)$	$Q_4\,(S_4)$	$Q_5\,(S_5)$	$Q_6\,(S_6)$	$Q_7\,(S_7)$
$P_1\,(R_1)$	0.0833	0.0844	0.0878	0.0931	0.1001	0.1084	0.1178
$P_2\,(R_2)$	0.0844	0.0833	0.0844	0.0878	0.0931	0.1001	0.1084
$P_3\,(R_3)$	0.0878	0.0844	0.0833	0.0844	0.0878	0.0931	0.1001
$P_4\,(R_4)$	0.0931	0.0878	0.0844	0.0833	0.0844	0.0878	0.0931
$P_5\,(R_5)$	0.1001	0.0931	0.0878	0.0844	0.0833	0.0844	0.0878
$P_6\,(R_6)$	0.1084	0.1001	0.0931	0.0878	0.0844	0.0833	0.0844
$P_7\,(R_7)$	0.1178	0.1084	0.1001	0.0931	0.0878	0.0844	0.0833

将上述表格中数据与题目所给数据逐一对比, 去掉 P_iQ_i, R_jS_j, 这些波线 (因为它们末穿过任一个小正方形, 对判断无用), 发现 $P_1Q_2, P_2Q_1, P_4Q_6, P_6Q_7$ 等十四条波线的运行时间或小于上表所列数据或仅比上表所列数据大了不到 0.02 s 。如果经过空洞, 误差将大于 0.09 s, 根据上面分析这些波线都没有经过空洞。这样就可以用几何作图的方法, 在平板图上作出上述十四条波线, 以决定可以排除多少小正方形, 肯定它们不是空洞, 同时也决定还有多少小正方形可能是空洞。如下图所示:

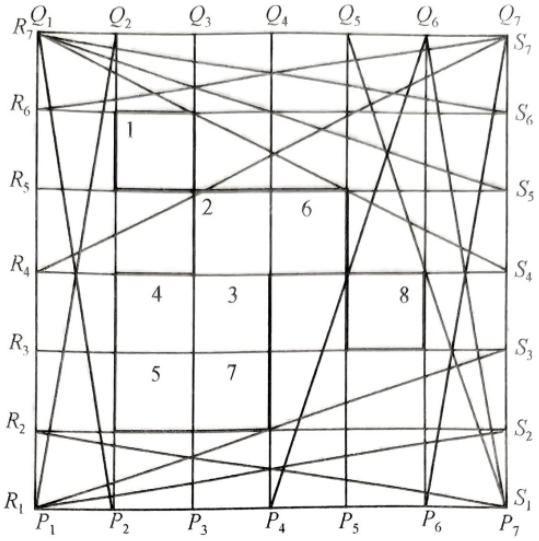


图 2　几何模型波线图

由图可知十四条波线共排除了二十八个小正方形, 仅有图中标出数字的八个小正方形可能存在空洞。

下一步的工作是确定这八个小正方形中哪几个是空洞以及这八个小正方形中空洞是否是造成 98 条波线传播时间延长的惟一原因或主要原因。因为排除掉的小正方形已

很多, 不妨认为剩下的八个小正方形全是空洞 (如果不是空洞, 则还应该有一些波线的传播时间在 0.1 s 左右), 计算经过这八个小正方形的波线的传播时间。例如 R_4S_1, 有

$$p_{41} + q_{41} = 268.3282 \text{ m}$$

其中 $p_{41} = 178.8891$ m, $q_{41} = 89.4427$ m 。理论传播时间为 0.3416 s, 测量值为 0.3655 s, 仅相差 0.0239 s 。再如 R_3S_1,

$$p_{31} + q_{31} = 252.98229 \text{ m}$$

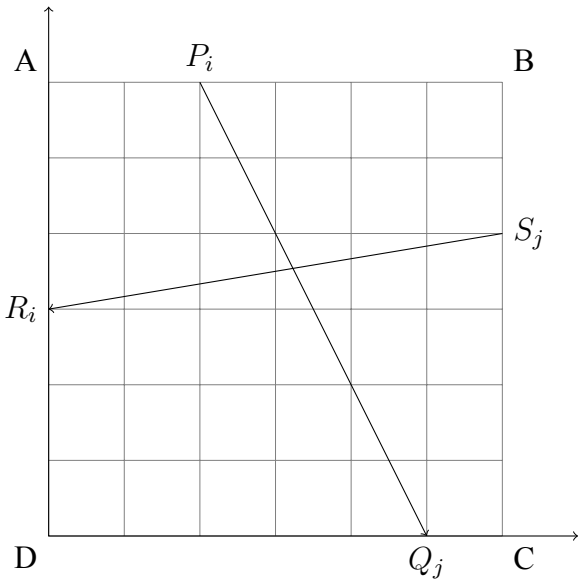
其中 $p_{31} = 168.6548$ m, $q_{31} = 84.3274$ m 。理论传播时间为 0.3221 s, 测量值为 0.34565 s, 仅相差 0.0234 s

这样逐一进行验证，发现误差均不小于不作为空洞处理的误差，由此可以确定找到的八个小正方形确是空洞，

5.1.2 代数模型

分析题目可知，这是一个随机型问题，为了简化，我们假设空洞是 $40m \times 40m$ 的正方形，平板被划分为 6×6 的小正方形网格，每个小正方内全是介质或者全是空气。

我们很自然地想到在平板上建立坐标系, 以 D 为原点, DC 为 x 轴正向, DA 为 y 轴正向。设小正方形边长为 1，每个小正方形用其左上角顶点坐标表示，则 C 点坐标为 $(6, 0)$ 。从而每个波源 $P_i(R_i)$ 及每个接收器 $Q_j(S_j)$ 的坐标也就确定了, 每个波线的方程也就可以确定。



设波源 P_i 发出的弹性波到达接收器 Q_j 的时间为 t_{ij}, 记波线 P_iQ_j 经过的介质长度为 p_{ij}, 经过的空气的总长度为 q_{ij}, 则对波线 P_iQ_j 可列出方程组:

$$\begin{cases} \frac{p_{ij}}{2880} + \frac{q_{ij}}{320} = t_{ij} \\ p_{ij} + q_{ij} = \sqrt{(i-j)^2 + 6^2} \end{cases}$$

可以计算波线 P_iQ_j 经过每个小正方形的长度。因 P_i 的坐标 $(i,6)$, Q_j 的坐标是 $(j,0)$ 波线 P_iQ_j 的方程可以表示为:

$$y=\frac{6}{i-j}\cdot(x-j)\Leftrightarrow x=\frac{6j+(i-j)y}{6}$$

将直线方程与某个小正方形的四条边的方程 $x=k, y=l, x=k+1, y=l+1$, 联立求解, 可以确定波线 P_iQ_j 不经过某个小正方形或者与小正方形的两个交点, 从而计算出波线 P_iQ_j 经过某个小正方形的长度 $d_{ij}^{(k)}$ (不经过则为零)。

将 36 个小正方形按从上到下，从左到右的顺序进行编号, 即 D 为顶点的小正方形为 0 号, C 为顶点的小正方形为 5 号, A 点为顶点的小正方形为 30 号, B 点为点的小正方形为 35 号。第 i 个小正方形的特征量记为

$$x_i=\begin{cases}0, & \text{第}i\text{ 个小正方形全是介质}\\ 1, & \text{第}i\text{ 个小正方形全是空气}\end{cases}$$

36 个方格的特征量记为 $x=(x_1,\cdots,x_{36})'$, 记 $d'_{ij}=\left(d_{ij}^{(1)},\cdots,d_{ij}^{(36)}\right)$, 则波线 P_iQ_j 过空气的总长度为: $d'_{ij}x=\sum_{k=1}^{36}d_{ij}^{(k)}\cdot x_k=q_{ij}$ 。波线 P_iQ_j 经过介质的总长度为 $d'_{ij}\left(1_{36}-x\right)=\sum_{k=1}^{36}d_{ij}^{(k)}\left(1-x_k\right)=p_{ij}$ 。其中 1_{36} 代表每个分量为 1 的 36 维的向量。因波线 P_iQ_i 不经过任一小正方形内部, 无法诀算 p_{ii} 及 q_{ii}, 暂不考虑。再将波线 P_iQ_j 按先 i 后 j 先小后大的顺序排成一列即

$$(P_0Q_1,P_0Q_2,\cdots,P_0Q_6,P_1Q_0,P_1Q_2,\cdots,P_1Q_6,P_2Q_0,\cdots,P_6Q_5)'$$

将 d'_{ij} 按这一顺序上下相接排成一个 42×36 的矩阵, 对 R_iS_j 同样操作合成一个矩阵 D 。再将 t_{ij} 按同一顺序上下相接成一个 84 维向量 t 。根据前面分析就有

$$Dx/2880+D\left(1_{36}-x\right)/320=t$$

上述方程组有 84 个方程, 36 个末知的 0-1 变量, 方程数明显多于未知数个数，为超定方程组。且由于测量误差等式两边不再相等, 如果 x 不是方程组的解, 代人方程组, 则等号两边的差别会更大, 因此我们猜测若 x 确是原方程组的解, 等号两边的差所构成的向量应是最小的。又由于差是向量, 最小的含义应是向量的长度最小。

在这个想法之下, 我们改写方程组为 $Ax=t'$, 误差向量为 $Ax-t$, 向量的平方为 $(Ax-t')'(Ax-t')$, 其中 A 、t' 是已知的, 因此误差向量长度的平方是 x 的函数, 要求最小值即: $\min_x(Ax-t')'(Ax-t')$ 。这是一个无约束的最小值问题，可以对函数求导, 令导数为零, 求出极小值点。

$$2A'Ax-2A't'=0, x=\left(A'A\right)^{-1}A't'$$

上式解出的其实是最小二乘解，在 python 中我们可以直接调用库函数解决，得到的解为:

$$\begin{array}{cccccc} -0.1190 & -0.0764 & 0.23951 & -0.0900 & -0.4140 & -0.0414 \\ -0.3322 & -0.8698 & -0.8457 & -0.4665 & 0.17427 & -0.2266 \\ -0.3698 & -0.8196 & -0.8859 & -0.6052 & -0.3987 & -0.4894 \\ 0.01385 & -0.3153 & -1.2875 & -0.6560 & -0.7402 & 0.32291 \\ 0.14386 & -1.3269 & -0.4212 & 0.27654 & -0.3042 & 0.04964 \\ 0.03792 & -0.1729 & -0.3557 & 0.01573 & 0.14044 & -0.1810 \end{array}$$

分析结果可知，所得到的解的每个分量绝对值都接近 0 或 1 , 我们用其相近值替换, 可得 0-1 解为:

$$x = \begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

其在平板上的空洞位置标记如图:

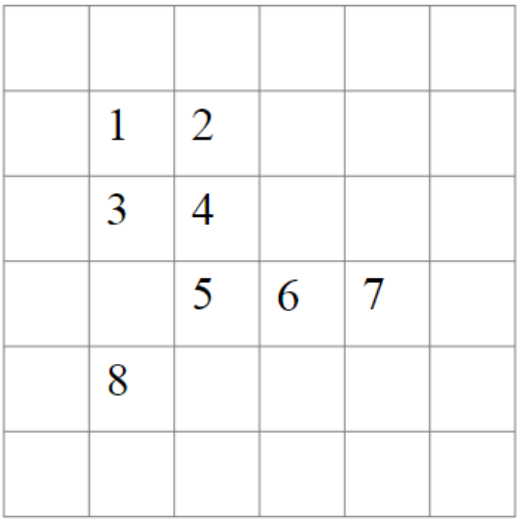


图 3　6×6 划分空洞位置

至此，我们在附加假设的前提下大致确定了空洞的位置。但可以很明显看到，我们的结果比较粗糙，精度也较差。而要提高模型精度，我们很自然想到将平板划分的更细一些，如将其划成 12×12 的，让波源和接收器仍等间距放置，在这种情况下我们可得

的新的最小二乘解 x' 如下：

$$x' = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

计算两种划分的误差向量长度平方可知，12×12 划分确实可以降低误差。对比几何模型中得到的空洞位置可知，12×12 更接近几何模型的结果。那么我们是否可以将平板划分的足够细，来得到足够高的精度呢？但仔细分析后我们发现上述想法并不可行，当划分的越细，小正方形越多，而波线数并未增加，从而会有很多小正方形没有一条波线经过，平板上会有许多没有一条波线经过的死区，尤其是在平板的四周。我们不应奢望该模型有更高的精度。

5.1.3 贪心算法

上述几何模型是直接根据约束列出线性方程组并求出近似解，而题中给出的数据是测量时间矩阵，在几何模型建立的坐标系的基础上，我们可以做如下考虑：若我们假设出空洞的位置及数量，那么就可以计算出理论传播时间矩阵 T^{ab}，其中上标 ab 表示空洞位置，我们可以计算出 $T+\tau$ 与 $T^{ab}+\tau^{ab}$ 间的误差，并将其作为贪心选择标准。每次我们遍历所有空洞可能的位置，并计算出对应的误差，然后选择误差最小的空洞位置作为当前的解，一步一步进行下去，直至误差上升，此时算法终止，输出空洞位置的近似解。该算法的具体描述如下：

首先我们有必要介绍一些符号和假设条件：借由代数模型，在网格中建立坐标系，假设空洞的坐标为 (a,b)，则我们可以计算出 49×2 条波线的传播时间，误差为：

$$\Delta t = \sum_{\mathrm{i}=1}^{7}\sum_{\mathrm{j}=1}^{7}\left(\mathrm{T}_{\mathrm{ij}}^{\mathrm{ab}} - \mathrm{T}_{\mathrm{ij}}\right)^2 + \sum_{\mathrm{i}=1}^{7}\sum_{\mathrm{j}=1}^{7}\left(\tau_{\mathrm{ij}}^{\mathrm{ab}} - \tau_{\mathrm{ij}}\right)^2$$

取误差最小值作为当前贪心选择，即：

$$\Delta t_{min} = \min_{ab}(\Delta t)$$

将上述思想用算法伪代码表示为：

Algorithm 1 Greedy algorithm

Initialization:$\{T\}, \{\tau\}, \Delta t_{min} \leftarrow inf, loc \leftarrow [\ ]$
repeat
  $num \leftarrow num + 1$
  temp_t $\leftarrow \min_{a,b} get\_delta\_t(a, b)$
  if $temp\_t \leq \Delta t_{min}$ **then**
    $delta\_t\_min \leftarrow temp\_t$
    loc.append(index(temp_t))
  end if
until $temp\_t > \Delta t_{min}$
Update $loc, \Delta t_{min}$

Output: The position of each hole: loc

运行上述算法，初始我们假设方格没有空洞，即：

$$x = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

遍历上述 36 个位置后，可得第一个空洞位置为: $[3, 2]$（该坐标表示行号和列号，从 0 开始，且左上角为坐标远点），可标注为：

$$x = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

不断运行算法，我们发现当计算到第 8 个空洞时，继续增加空洞数目会导致误差值的增大，故算法结束，终止时的结果为：

	4	7			
	6	2	3		
		1	8		
	5				

5.1.4 代数模型的改进-染色法

在代数模型中，我们用最小二乘法确定出 8 个空洞，每个都是 $40m \times 40m$ 的正方形，这里给出的假设是，$40m \times 40m$ 的方格要么全是空洞，要么全是介质. 但仔细思考，恐怕很难让人信服，哪有那么规范、整齐、一致的空洞块呢? 于是，我们设法在其的基础上进行改进，提出了一种更为精细的判别空洞位置的方法，且该染色方法的退化情况比代数模型更为精确，与我们最开始在几何模型中理论分析的解完全吻合。该方法的具体过程如下：

1. 在题目给出的图中，作出所有从 P_i 到 Q_j ，从 R_i 到 S_j 的弹性波线 $(i, j = 1, ..., 7)$，如下图，这里用的是红色虚线，表示初始全部为介质，后续我们将用蓝色实线标记出空洞。由图可知，这九十八条线几乎均匀地经过图中的每一部分。

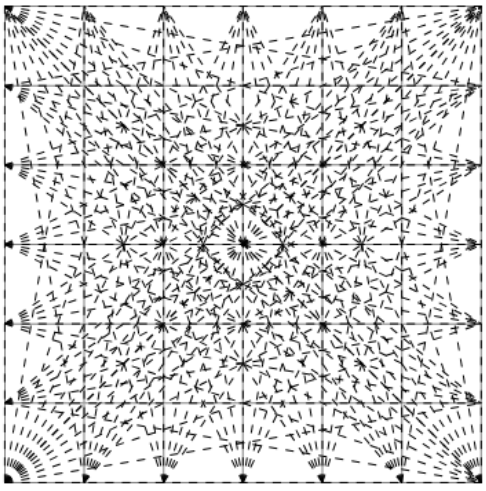


图 4　弹性波波线图

2. 用勾股定理分别求出 P_i 到 Q_j, 从 $R_i(i = 1, \cdots, 7)$ 到 $S_j(j = 1, \cdots, 7)$ 的距离 $|P_iQ, |, |R_iS, |$, 如下表：

距离 (米)	Q_1, S_1	Q_2, S_2	Q_3, S_3	Q_4, S_4	Q_5, S_5	Q_6, S_6	Q_7, S_7
P_1, R_1	240	243.3105	252.9822	268.3282	288.4441	312.4100	339.4114
P_2, R_2	243.3105	240	243.3105	252.9822	268.3282	288.4441	312.4100
P_3, R_3	252.9822	243.3105	240	243.3105	252.9822	268.3282	288.4441
P_4, R_4	268.3282	252.9822	243.3105	240	243.3105	252.9822	268.3282
P_5, R_5	288.4441	268.3282	252.9822	243.3105	240	243.3105	252.9822
P_6, R_6	312.4100	288.4441	268.3282	252.9822	243.3105	240	243.3105
P_7, R_7	339.4114	312.4100	288.4441	268.3282	252.9822	243.3105	240

3. 求 $t_{b_{ij}} = \frac{|P_iQ_j|}{2880} = \frac{|R_rS_j|}{2880}$. 弹性波从 P_i, R_i 经均匀介质分别到达 Q_{ij}, $S_j(i,j = 1,\cdots,7)$ 的时间称为标准时间, 如下:

时间 (秒)	Q_1, S_1	Q_2, S_2	Q_3, S_3	Q_4, S_4	Q_5, S_5	Q_6, S_6	Q_7, S_7
P_1, R_1	0.083333	0.084484	0.087842	0.093172	0.10016	0.10848	0.11785
P_2, R_2	0.084484	0.083333	0.084484	0.087842	0.093172	0.10016	0.10848
P_3, R_3	0.087842	0.084484	0.083333	0.084484	0.087842	0.093172	0.10016
P_4, R_4	0.093172	0.087842	0.084484	0.083333	0.084484	0.087842	0.093172
P_5, R_5	0.10016	0.093172	0.087842	0.084484	0.083333	0.084484	0.087842
P_6, R_6	0.10848	0.10016	0.093172	0.087842	0.084484	0.083333	0.084484
P_7, R_7	0.11785	0.10848	0.10016	0.093172	0.087842	0.084484	0.083333

4. 求

$$d_{t_{ij}} = \frac{t_{ij} - t_{b_{ij}}}{t_{b_{ij}}} \cdot 100\% \quad (i,j = 1,\cdots,7),$$

$$d_{\tau_{ij}} = \frac{\tau_{ij} - \tau_{b_{ij}}}{\tau_{b_{ij}}} \cdot 100\% \quad (i,j = 1,\cdots,7).$$

列出时间相对误差表

$d_{t_{ij}}\%$	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
P_1	-26.68	5.942	127.3	118.1	317.4	353.8	379
P_2	17.07	-28.96	422.3	391.6	411.9	423.3	250.7
P_3	247.5	389	-28.24	391.6	373.2	282.4	91.56
P_4	245.7	407	378.2	-11.44	111.8	-15.76	127.7
P_5	248.4	386.1	157.7	126.9	0.684	109.3	106.1
P_6	250.9	217.2	153.7	248.9	162.4	12.68	22.04
P_7	265.8	213.1	256	109.7	-13.48	-18.56	25.04

$d_{\tau_{ij}}\%$	S_1	S_2	S_3	S_4	S_5	S_6	S_7
R_1	-22.6	-28.74	-7.445	277.3	286.1	297.7	385.4
R_2	-10.87	-16	237.6	394.2	274.7	379.2	359.1
R_3	293.5	279.3	16.88	384.5	382.7	387.3	210.7
R_4	292.3	274.5	402.7	20.84	284.6	143	9.155
R_5	216	158.5	265.9	285.4	8.484	121.8	142.5
R_6	153.4	288.5	532.7	243.4	143.6	0.924	-16.43
R_7	276.2	353.4	289.8	-15.64	-19.28	8.191	-30.04

5. 由于测量误差一般都服从正态分布, 且同一类仪器的测量精度大致相等, 因此由上面两表综合分析得出: 当 $d_{t_{ij}}, d_{\tau_{ij}} \in [-31\%, 31\%]$ 时, 弹性波几乎未穿过空洞, 在图 4 弹性波波线图的基础上用红线 (实线) 表示, 如下所示:

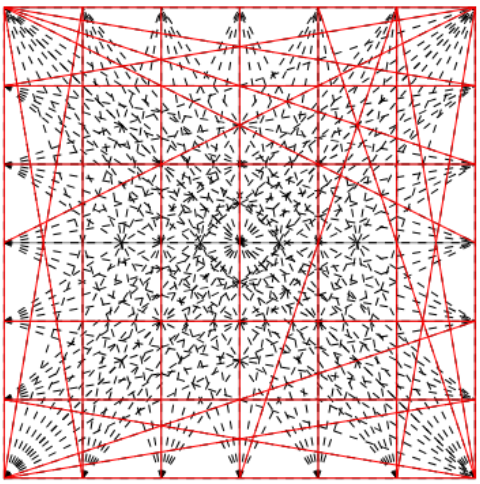


图 5　标记后的弹性波波线图

6. 统计出 36 个方格中的红点数（这里的红点数可以近似理解为红线在每个小正方内的长度和）, 见下表:

387	324	283	245	353	352
244	164	244	281	284	281
281	205	164	164	281	281
244	164	164	203	164	282
244	164	164	246	206	324
347	288	284	282	244	347

若仍有假设, 40 m × 40 m 的方格要么全是介质, 要么全是空洞, 则红点数最少 (皆为 164 点) 的方格为空洞，与几何模型的结果完全相同。

7. 由于在实际问题中, 40 m × 40 m 的正方形要么是空洞, 要么是介质的情形极少, 所以我们认为用二值逻辑来判断哪些正方形是介质, 哪些是正方形是空洞, 不如先对方格加细, 用多值逻辑来处理更符合实际。于是, 将方格加细, 把每个 40 m × 40 m 的方格等分为 4 个小正方格，即将图 5 的大正方形等分成 144 份, 重新统计出每个 20 m × 20 m 的小方格中的红点数, 得到权值表, 如下:

113	116	81	63	61	61	61	61	79	79	97	97
77	81	81	99	80	81	62	61	98	97	81	77
61	61	41	41	61	61	80	79	80	62	61	61
61	61	41	41	60	62	62	60	62	80	80	79
61	61	61	61	41	41	41	41	80	80	80	79
79	80	42	41	41	41	41	41	60	61	80	70
61	61	41	41	41	41	41	60	41	41	80	61
61	61	41	41	41	41	41	61	41	41	80	61
61	61	41	41	41	41	43	61	43	61	99	81
61	61	41	41	41	41	79	63	61	43	63	81
78	80	63	81	81	80	80	61	61	61	80	97
93	96	81	63	61	62	80	61	61	61	77	93

8. 分析权值表, 当权值出现大的断层时, 则将该处作为一个分段点, 分成几个区域进行染色. 即将权值在 41-43 间的小方格染色白色, 60-63 间的小方格染成蓝色, 77 – 81 间的小方格染成灰色, 93 以上的小方格染成黄色. 如下图中用 1 表示白色, 用 2 表示蓝色, 用 3 表示灰色, 用 4 表示黄色。

4	4	3	2	2	2	2	2	3	3	4	4
3	3	3	4	3	3	2	2	4	4	3	3
2	2	1	1	2	2	3	3	3	2	2	2
2	2	1	1	2	2	2	2	2	3	3	3
2	2	2	2	1	1	1	1	3	3	2	2
3	3	1	1	1	1	1	1	2	2	3	3
2	2	1	1	1	1	1	2	1	1	3	2
2	2	1	1	1	1	1	2	1	1	3	2
2	2	1	1	1	1	1	2	1	2	4	3
2	2	1	1	1	1	3	2	2	1	2	3
3	3	2	3	3	3	3	2	2	2	3	4
4	4	3	2	2	2	3	2	2	2	3	4

9. 分析上图, 可得: 空洞 (群) 主要集中在白色区域, 蓝色区域次之, 灰色区域空洞小且稀少, 黄色区域几乎没有空洞。将上述结果可视化，标记出空洞位置如下:

4	4	3	2	2	2	2	2	3	3	4	4
3	3	3	4	3	3	2	2	4	4	3	3
2	2	1	1	2	2	3	3	3	2	2	2
2	2	1	1	2	2	2	2	2	3	3	3
2	2	2	2	1	1	1	1	3	3	2	2
3	3	1	1	1	1	1	1	2	2	3	3
2	2	1	1	1	1	1	2	1	1	3	2
2	2	1	1	1	1	1	2	1	1	3	2
2	2	1	1	1	1	1	2	1	2	4	3
2	2	1	1	1	1	3	2	2	1	2	3
3	3	2	3	3	3	3	2	2	2	3	4
4	4	3	2	2	2	3	2	2	2	3	4

5.1.5 概率模型

在该部分我们将提出一种概率模型，是对问题更进一步深入。回顾前面的模型可知，我们为了解决问题，做出了太多太强的假设，正如我们在染色法中提到的，空洞的位置，大小，形状都应是随机的，因此本节我们就放松条件，得到更具一般性的模型。另外，注意到在代数模型中，平行于平板边的 14 条波线并没有被考虑，因此如何充分利用已知的信息成了我们改进的方向之一。

因为对于空洞分布没有什么先验知识, 我们可以设想空洞服从均匀分布, 即每点或每个小块是空洞的概率是相等的。根据误差理论可以假设波线的传播时间误差服从正态分布。我们可以对测量数据先进行修正, 正如在几何模型中讨论过的有十六条波线传播时间小于全在介质中传播时间, 这肯定是由于测量误差引起的, 而且知道是由于误差为负而缩短了传播时间, 因此将这十六条波线上测量传播时间修改为全在介质中传播时间, 显然是缩小了测量误差, 对于空洞的估计是有好处的。

根据上述假设我们也可以估计出空洞面积的大小。已假设空洞的分布服从均匀分布, 则 98 条波线的试验就是对均匀分布的抽样。均匀分布只有一个参数就是概率密度, 就可以根据抽样的样品对这个参数进行估计, 根据熟知的极大似然估计方法, 可以用样本频率代替概率。

但是如果假设空洞分布在整个平块上是均匀分布, 则空洞就像细砂一样散布于平板

上, 这也是不符合实际情况的。实际上这么大平板上空洞不会全是细如针的洞, 会有大有小。因此, 可以将均匀分布的假设用于每个小正方形, 只是不同的小正方形中空洞的概率是不同的。我们便可以得到以下的概率模型:

假设每条波线上空洞的频率代表波线上空洞均匀分布的概率，以一个小正方形 x 为总体，从而经过 x 的各条波线与 x 的公共部分即为对于 x 的抽样。由于不同波线经过同一个小正方形内部的长度不同，即抽样的数量不同，所以若波线长度大，则权重应该大。由此估计第 n 个小正方形是空洞的概率为:

$$P_n = \frac{\sum_{i,j} p_{ij} d_{ij}^{(n)}}{\sum_{i,j} d_{ij}^{(n)}}$$

具体地，需要综合使用从两个角度得到的观测数据。若第 n 个小正方形是空洞的概率矩阵为 (P_{ij})，使用 P_iQ_j 计算得到概率矩阵 $(P_{ij}^{(1)})$，R_iS_j 计算得到概率矩阵 $(P_{ij}^{(2)})$，则

$$(P_{ij}) = \frac{(P_{ij}^{(1)}) + (P_{ij}^{(2)})}{2}$$

由此得到结果:

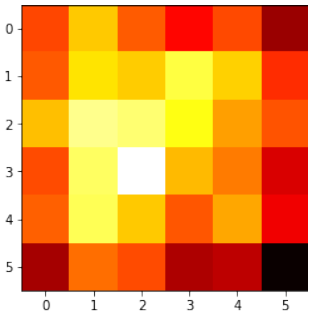


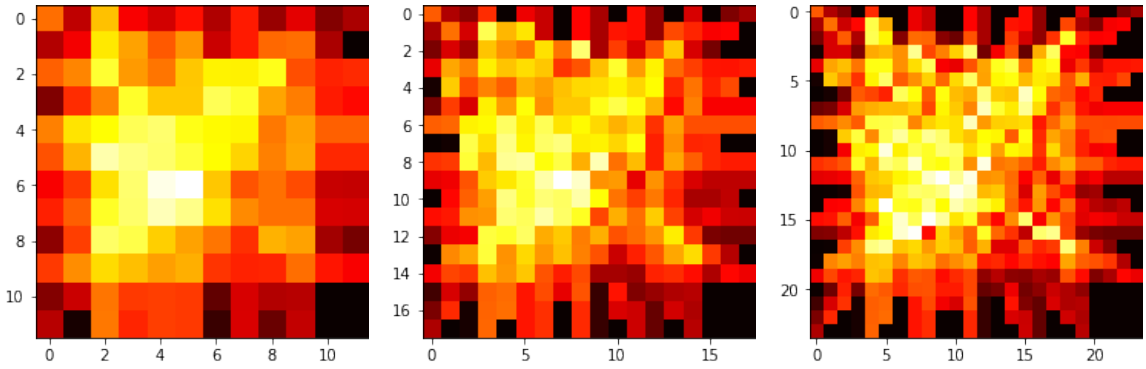
图 6　小正方形空洞概率热力图

考虑更进一步地划分平板为 $6k \times 6k$ 个小正方形, 则第 n 个小正方形是空洞的概率为:

$$P_n = \frac{\sum_{i|k,j|k} p_{ij} d_{ij}^{(n)}}{\sum_{i|k,j|k} d_{ij}^{(n)}}$$

观察图 6 可以看出：空洞和介质之间的区分不够明显，在几何模型得到的八个小正方形之外存在较亮的小正方形，即存在介质以较大的概率是空洞。原因是在计算小正方形空洞概率时，对于不同的小正方形使用相同的 p_{ij}，即用波线上空洞出现的概率来对波线经过的所有空洞进行计算。这使得空洞均匀化。

考虑将波线上空洞比例 p_{ij} 按照已经计算得到的空洞概率和波线在空洞内的长度进行加权，得到新的波线空洞比例 p'_{ij}，从而得到新的 P'_n，如此迭代直到收敛。



(a) 12×12 (b) 18×18 (c) 24×24

图 7　不同划分对应的概率热力图

将上述思想表示为如下算法伪代码。

得到图 8 的结果，可见空洞和介质之间的区分度得以提高。

Algorithm 2 概率迭代算法

1: Initialization: $\boldsymbol{p}$ $(6 \times 6 \times m^2)$ 为波线上空洞比例，$\boldsymbol{d}$ $(m \times m \times m^2)$ 为波线在小正方形内的长度。

2: **repeat**

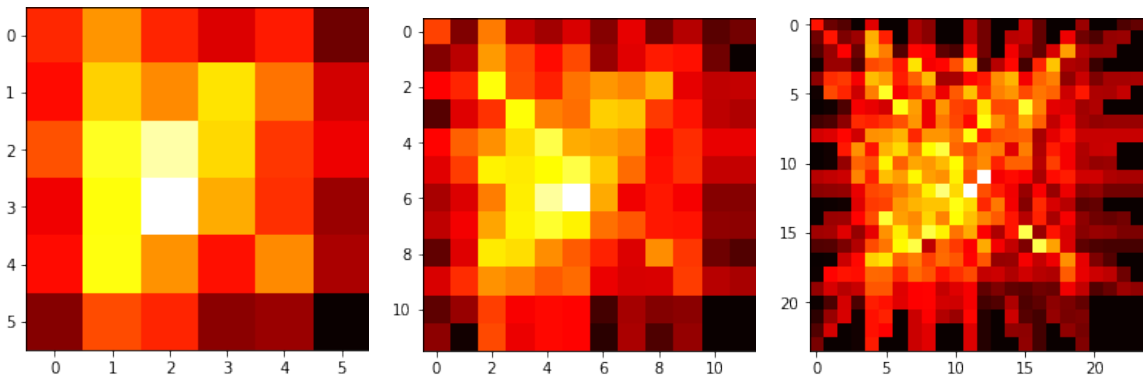
3: $\quad \boldsymbol{P}[n] \leftarrow \dfrac{\sum_{i,j} p_{ij} d_{ij}^{(n)}}{\sum_{i,j} d_{ij}^{(n)}}, 1 \leq n \leq m^2$

4: $\quad \boldsymbol{\alpha} \leftarrow \text{softmax}(\boldsymbol{P} \odot \boldsymbol{d}, \text{axis} = 2)$

5: $\quad \boldsymbol{p} \leftarrow \boldsymbol{\alpha} \odot \boldsymbol{p}$

6: **until** Convergent

Output: 空洞概率矩阵 $\boldsymbol{P}$



(a) 6×6 (b) 12×12 (c) 18×18

图 8　迭代算法不同划分对应的概率热力图

我们可以对误差的标准差进行估计：

1. 取十六个测量传播时间小于波线在全是介质中的传播时间的波线作为样本, 这些都是发生负偏差造成了, 其中负偏差最大的为 $0.0833-0.0583=0.025$, 按正态分布 $\frac{1}{32}=0.031$ 的分位点应是 2.25σ, 故标准差 σ 可取为 0.011 。

2. 由于 P_1Q_7 与 R_1S_7 是同一个波线, P_7Q_1 与 R_7S_1 是同一根波线, 在 98 根波线中只有这两对作了重复试验, 因此可以据此估计标准差。估计的原理是依据顺序统计量的分布。

设 ξ_1,ξ_2 是相互独立同分布的随机变量, 设 $\xi_2^*\geqslant\xi_1^*$, 设 ξ_i 的分布函数、密度函数分别为 $F(x)$ 、$f(x)$, 则 $P\left(\xi_2^*<x\right)=F^2(x), P\left(\xi_1^*<x\right)=1-((1-F(x))^2=2F(x)-F^2(x)$, 记 $G(x,y)=P\left(\xi_1^*<x,\xi_2^*<y\right)$。

若 $x\geqslant y$, 则 $G(x,y)=P\left(\xi_2^*<y\right)=F^2(y)$。

故 ξ_1^* 、ξ_2^* 的联合密度函数为 $g(x,y)=\begin{cases}u & x\geqslant y\\ 2f(x)f(y) & x<y\end{cases}$。

设 $R=\xi_2^*-\xi_1^*$ 的分布密度函数 $f_R(r)$, 显然若 $r\leqslant 0$, 则 $f_R(r)=0$ 。若 $r>0$ 则

$$P(R<r)=\iint_{y-x<r}g(x,y)\mathrm{d}x\,\mathrm{d}y=\int_{-\infty}^{+\infty}\left[\int_{-\infty}^{x+r}g(x,y)\mathrm{d}y\right]\mathrm{d}x$$

$$=\int_{-\infty}^{+\infty}\left[\int_{-\infty}^{r}g(x,x+z)\mathrm{d}z\right]\mathrm{d}x=\int_{-\infty}^{r}\left[\int_{-\infty}^{+\infty}g(x,x+z)\mathrm{d}x\right]\mathrm{d}z$$

上式两边对 r 求导, 因此

$$f_R(r)=\int_{-\infty}^{+\infty}g(x,x+r)\mathrm{d}x=2\int_{-\infty}^{+\infty}f(x)f(x+r)\mathrm{d}x$$

由误差理论设 ξ 服从正态分布 $N\left(0,\sigma^2\right)$, 即 $f(x)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{x^2}{2\sigma^2}}$, 代入上式, 并且对 r 求数学期望, 即

$$\begin{aligned}
&2\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}rf(x)f(x+r)\mathrm{d}x\,\mathrm{d}r\\
=&2\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\exp\left(-\frac{x^2}{2\sigma^2}\right)\exp\left(-\frac{(x+r)^2}{2\sigma^2}\right)\cdot r\cdot\mathrm{d}r\,\mathrm{d}x/\left(2\pi\sigma^2\right)\\
=&\frac{2}{2\pi\sigma^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\exp\left(-\frac{\left(\sqrt{2}x+\frac{\sqrt{2}}{2}r\right)^2+r^2/2}{2\sigma^2}\right)r\,\mathrm{d}x\,\mathrm{d}r\\
=&2\int_{-\infty}^{+\infty}r\cdot\exp\left(-\frac{r^2}{2\sigma^2}\right)\cdot\sqrt{\pi}\sigma\mathrm{d}r/\left(2\pi\sigma^2\right)\\
=&\frac{2}{2\pi\sigma^2}\int_{-\infty}^{+\infty}\exp\left(-\frac{r^2}{2\sigma^2}\right)\mathrm{d}\left(\frac{r^2}{2\sigma^2}\right)\cdot\sqrt{\pi}\sigma^3\\
=&\frac{\sigma}{\sqrt{\pi}}
\end{aligned}$$

因此可以用 $(\xi_2-\xi_1)\sqrt{\pi}$ 作为 σ 的估计。而本题两对波线两次测量之差分别为 $0.5721-0.5646=0.0075$, $0.4434-0.4311=0.0123$, 取平均为 0.0099 , 因此可用 0.0175 作为 σ 的估计。

3. 也可以用前面已用过的 16 个负偏差的数据, 以全在介质中传播时间为真实均值, 计算样本标准差。即

$$\sigma = \sqrt{\sum_{i=1}^{16} (t_i - \tau_i)^2 / 15} = 0.0178$$

分析测量传播时间数据, 在 $0.1042 \sim 0.1768$ 之间没有任何数据, 这一异常现象也说明 $0.1042 - 0.0833 = 0.0209$ 是测量误差的一个最大值（正偏差）。因而上面对于 σ 的估计是可信的。

5.2 问题二模型的建立及求解

5.2.1 几何模型分析求解

1. 问题二第一个问题的求解

问题二中要求的是通过理解和分析是否在只有一组对波的情况下确定空洞的所在位置如题意可知只知道 P_iQ_j 这一组对波时判断空洞所在的位置。由假设可知在网格经过的最短距离都是 240 m 则通过对问题一的解决可知, 波传播的时间理论测量最短是 0.0833 s 在 P_iQ_j 这一组对波中最短时间是 0.0592s , 依题意可知在空气中波传播的速度要小于在介质中传播的速度。测得的实际值应当大于 0.0833 s 。假设中测得的实际时间数据全部正确, 由此可知, 测量存在误差:

$$\mathrm{t}_{\text{误差}} = 0.0833 - 0.0592 = 0.0241$$

根据题目中的实际测量数据和理论测量的差值与测量误差的比较, 差值不超过测量误差的说明在这条波线所经过的空格中没有空洞存在, 依题意可知把实测时间与理论时间的差值不大于测量误差的波线划到图中。空格即为空洞所在位置

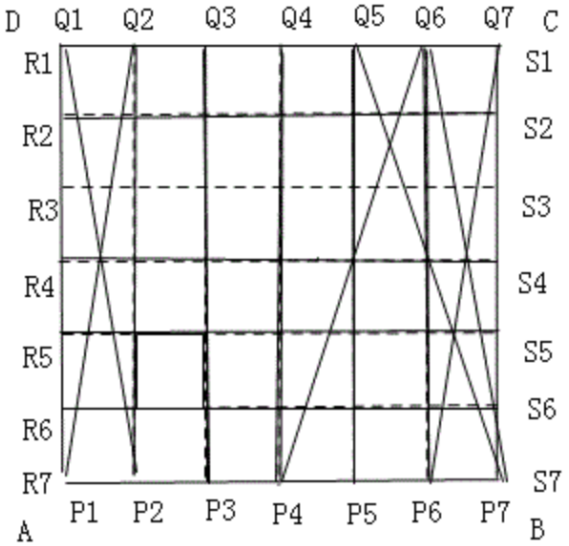


图 9　只用一对边时所得空洞分布

从图中可以看出在只选择一组对边的波来探测未知地方的空洞时，根据图中表示的网格中没有线经过的地方即是空洞存在的区域，和问题一中探测到的空洞比较可以看出，选择一组对边测得的空洞数要远多于选两组对边的空洞数。所以只用一组对边是难以在实际中探测到空洞的。

另外，假如有些空洞和 AB 边近似平行，依题意我们只有 AB 边的波的发射点，因此可以确定空洞是否存在也可同时确定在 AB 方向的位置，但是无法确定在 AD 方向的位置，最终还是无法确定空洞具体位置。

综上所述只有一组对边的波无法确定空洞具体位置。

2. 问题二第二个问题的求解

在这个问题中要求在不影响探测空洞准确位值的前提下尽量减少波的发射和接收点的数量讨论在同样能够确定空洞位置的的前提下, 减少波源和接收点器的方法。由上分析可以得出, 一个波源或接受源是否能预测出空洞所在位置取决于与它相关的波线的交点分布情况。由上可分为两个方面：与其它波线的交点个数及交点位置。去掉一个波源, 在其他波源和接受源不变的情况下, 求出不重复的交点个数和坐标方差。比较去掉波源前后不重复的交点个数, 如果两组数据相差不大表明信息量少预测空洞位置不准确。而不重复交点的坐标方差表明交点的分散程度, 方差小说明离散程度小, 交点集中在一个较小的面积内, 使得空洞更为集中并且可靠。综上所述, 应该选择交点减少量小的并且方差较大的波源去掉。在去掉这一波源的情况下, 再用同样的方法去掉其它波源。对于本题:

$$X = 1/n\left[(x_1-\bar{x})^2+(x_2-\bar{x})^2+\ldots\ldots+(x_n-\bar{x})^2\right], n = 1\cdots 7$$

$$Y = 1/n\left[(y_1-\bar{y})^2+(y_2-\bar{y})^2+\ldots\ldots+(y_n-\bar{y})^2\right], n = 1\cdots 7$$

	不重复交点数	$X-$ 方差	Y-方差
原始情况	2041	2.1242	2.1200
去掉第 1 个波源	1820	2.1283	1.9877
去掉第 2 个波源	1737	2.0937	2.0785
去掉第 3 个波源	1774	2.1223	2.2010
去掉第 4 个波源	1791	2.1146	2.2444
去掉第 5 个波源	1774	2.1223	2.2010
去掉第 6 个波源	1737	2.0937	2.0785
去掉第 7 个波源	1820	2.1283	1.9877

根据上述分析得：这里应去掉第 4 个波源。虽然第 1 个波源交点数较多, 但是 Y-方差很小, 如果去除第 1 个波源后交点分布很不均匀, 所以不去除。

5.2.2 代数模型

对于题目中减少波源的方式，在代数模型下，我们做出如下简单的假设：移去的波源或接收器是在原来平板上等间距放置的基础上随机移去，而其它波源或接收器不受影响。对应到数据矩阵上，即我们做出的改变是**删去传播时间矩阵 T 或 τ 的若干行或若干列**。在原有的坐标系下（这里我们采用 6×6 的方格划分），直线方程并未改变，可用的数据点的坐标也为变化，唯一变化的是矩阵 D 的行数。因此，我们仍然按照代数模型的思路，列出约束满足的线性方程组，采用最小二乘法解出该方程组，记该解为 x_{del}。

接下来我们很自然想到如何衡量我们得到的解的好坏呢, 因为最终我们肯定是要选出若干个最符合“标准”的结果的。若我们采用和代数模型一样的指标即 $\|\boldsymbol{Ax} - \boldsymbol{t'}\|_2$，很显然，由于使用数据量的减少，上述二范数的计算结果总体会减小。极端情况如考虑去掉所有波源和接收器，此时 $Ax = t' = \mathbf{0}$，得二范数为 0，显然不符合我们的要求。若考虑将上述二范数除以 t' 的长度，直观上理解即将误差平均到每个数据上，虽较为合理，但经实验发现，这样算出的结果差距较小，难以作出合理选择。

进一步分析题目可知，我们的目标是依据减少后的数据列出约束来确定空洞的位置，于是我们可以以完整数据得到的解为标准，计算新解与标准解的误差，即：

$$\|\boldsymbol{x_{del}} - \boldsymbol{x}\|_2$$

同时，因 x 得到的空洞位置为 8 个，因此我们规定：若新解与原解有至少 4 个空洞位置不同，那么我们认为该种接收器减少策略误差过大，即不能确认空洞位置。

有了上述的准备，我们可以先回答问题二的第一个问题：若只根据 P_i 发射到 Q_j 的时间，我们计算得 $\|\boldsymbol{x_{del}} - \boldsymbol{x}\|_2 = 3.74$，且得到的解 x_{del} 与原来的 8 个空洞位置几乎没有重合，显然这种方法是不能确定空洞位置的。这也与我们之前对于问题二的定性分析结果相一致。

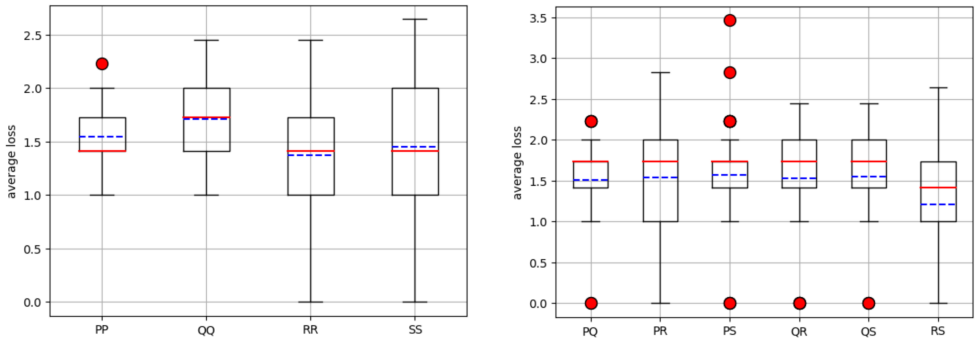
对于问题二的第二问，若采用直接枚举的办法，对于平板四条边上的接收器，共有 $(7!)^4$ 种结果。而经实验测试，当移去的仪器数目总和超过 3 时，计算出的误差已不合题意，因此我们可对仪器数目为 1，2，3 进行列举，所得实验结果如下：

若去掉一个仪器：

	P	Q	R	S
去掉第 0 个仪器	1.732	1	1	1
去掉第 1 个仪器	1.414	1.414	1	1
去掉第 2 个仪器	1	2	0	1
去掉第 3 个仪器	0	1.414	1	0
去掉第 4 个仪器	1	0	0	2
去掉第 5 个仪器	0	1.414	1.732	1
去掉第 6 个仪器	1.414	1.414	1	0

由上述结果可知，对应到题目种仪器的编号，可知移去 **P4，P6，Q5，R3，R5，S4，S7** 中的任意一个，对结果的影响都较小，均可以确定空洞的位置。

若去掉两个仪器，因两个仪器可以从平板的四条边任意选择，可知选的这两个仪器来源的边的情况有 10 种。而且由于 $\|\boldsymbol{x}_{\boldsymbol{del}} - \boldsymbol{x}\|_2$ 的值与仪器的选择并没有增长或减小的趋势，因此直接绘制折线图是不太合适的，我们在这里绘制箱线图，可描述仪器来自不同边的误差值的分布情况，具体如下：



(a) (b)

图 10　去掉两个仪器实验结果

我们对上图稍作解释：首先坐标横轴表示要移去的仪器的来源（如 PP 表示移去 AB 边的两个波源，其它同理），纵轴则是计算所得的二范数的值。箱线图的红线表示中位数，蓝线表示平均数，红点表示离群点。由箱线图可知，去除两个仪器后仍能确定空洞位置的出现在 **RR，SS，PQ，PR，RS，PS，QR，QS，RS**，然后查看程序的结果可知，这些仪器具体为：$R_3R_4, S_2S_5, P_4Q_5, P_6R_3, P_5S_4, Q_3R_5, Q_5S_4, R_5S_3$

若去掉三个仪器，采用同样的方法，但由于情况数较多，我们仅绘制来自同一条边上的数据：

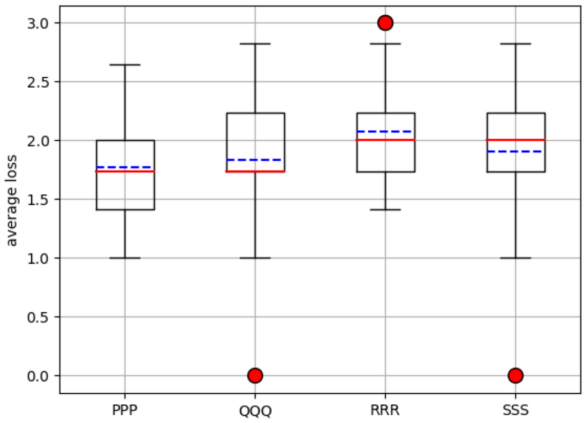
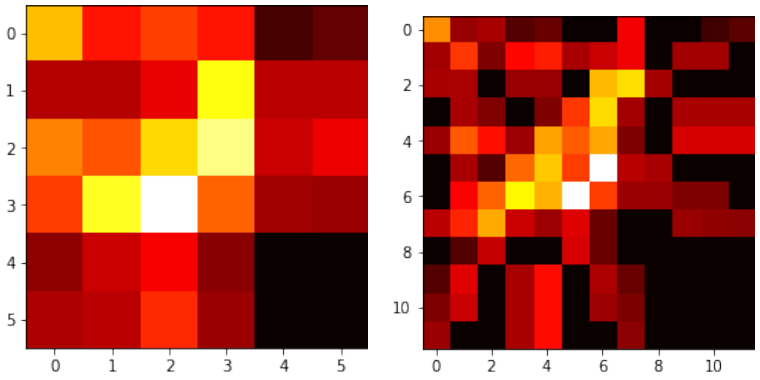


图 11　去除三个仪器实验结果

可知，此时仍能确定空洞位置的有：$Q_3Q_4Q_6, S_2S_5S_7$ 。

5.2.3 概率模型

考虑使用概率迭代算法，每个方向使用**四**对波源，即以 2 为步长选取序号为 1,3,5,7 的波源，得到图 12 的结果。可知此时仍能大致确定空洞位置。



(a) 6×6　　(b) 12×12

图 12　四对波源不同划分对应的概率热力图

六、 模型的评价

几何模型比较简单初等，，得出的解也较为准确，这是其优点；但是无法进一步细化，而且普适性较差，题目数据修改后就不一定适用了。

代数模型是一种较直观的方法，在本次实验中其最大的优点便是可以得到丰富的结果。但其缺点也较明显，那就是对于问题的假设太多太强，且得到的解也不够精确。

贪心法是从代数模型上得到的启发，其在 6×6 网格划分中得到的结果比代数模型更精确；染色法是对代数模型的改进，同时也有几何模型的思想，利用了比代数模型更多的信息，得到的结果也更让人信服。

概率模型基于极大似然估计，由比较宽松的条件得到了合理有效的结果。通过热力图观察可知，概率迭代算法较好地确定了空洞的位置，结果和几何模型一致。得到有效结果所需迭代次数小于 10 次，所以算法的复杂度也较低。模型的缺点是没有进一步讨论空洞的其他可能分布，比如正态分布。

本次实验对于该问题的解决过程也给了我们重要的启示：在刚开始上手这道题目时，我们觉得无从下手，所以尝试先对已给的信息和数据进行分析。而我们也较为幸运地通过几何关系先大致确定了空洞的位置；之后，我们尝试将问题简化，强加上若干假设以便于对问题建模，这便得到了代数模型；最后，我们从已有的模型得到启发，或是对已有模型进行改进，一步步放松假设，逐渐得到更精细和普适性更强的模型。这种面对一个难题时的解决过程在今后的科研或是实验是尤为重要的。

参考文献

[1] 朱道元. 数学建模案例精选 [M] . 科学出版社，2003.

[2] 杜秀丽，朱群生，王建锋. 探测空洞的方法. 数学的实践与认识，2001，31（4）.

[3] 张莉，杨礼，陈华韦. 空洞探测的数学模型与算法 [J]. 华北水利水电学院学报，2009，30（5）：106-108.

[4] 徐茂良，郑小勇，何彦. 陈忠明染色方法在空洞探测中的应用 [J]. 大学数学，2003，19（2）：38-45.

附录 A　代数模型算法

```
import numpy as np
# 将网络划分为n*n个小正方形
# 暴力枚举出直线在每个小正方形的内的长度


def compute_dis(x_1, y_1, x_2, y_2):
    if x_1 is None or x_2 is None or y_1 is None or y_2 is None:
        return 0
    else:
        return ((x_2-x_1)**2+(y_2-y_1)**2)**0.5


def get_length_PQ(i, j, n):
    # y = 6/(i-j)*(x-j) ==> (i-j)y=6(x-j) ==> 6x-(i-j)y = 6j
    # 直线方程:  x = col_id ; y = row_id
    # x = np.zeros(36)
    # d_ij = np.zeros(36)
    x = np.zeros(n*n)
    d_ij = np.zeros(n*n)
    if i == j:
        if i != 6:
            for number in range(6):
                d_ij[i+number*6] = 1
        if i == 6:
            for number in range(6):
                d_ij[i-1+number*6] = 1
    else:
        for num in range(n*n):
            # 每个小正方形用左上角的点表示
            row_id = num // n
            col_id = num % n
            x_1, y_1, x_2, y_2 = None, None, None, None
            x_3, y_3, x_4, y_4 = None, None, None, None
            y_temp_1 = n / (i-j) * (col_id-j)
            x_temp_1 = (n*j+(i-j)*row_id)/n
            y_temp_2 = n / (i - j) * (col_id+1-j)
            x_temp_2 = (n * j + (i - j) * (row_id+1)) / n
            if row_id <= y_temp_1 <= row_id+1:
                x_1 = col_id
                y_1 = y_temp_1
            if col_id <= x_temp_1 <= col_id+1:
                x_2 = x_temp_1
                y_2 = row_id
            if row_id <= y_temp_2 <= row_id+1:
```

```
            x_3 = col_id+1
            y_3 = y_temp_2
        if col_id <= x_temp_2 <= col_id+1:
            x_4 = x_temp_2
            y_4 = row_id+1
        dis = max(compute_dis(x_1, y_1, x_2, y_2), compute_dis(x_1, y_1, x_3, y_3),
                  compute_dis(x_1, y_1, x_4, y_4), compute_dis(x_2, y_2, x_3, y_3),
                  compute_dis(x_2, y_2, x_4, y_4), compute_dis(x_3, y_3, x_4, y_4))
        if dis != 0:
            d_ij[num] = dis
            x[num] = 1
    # 计算P_iQ_j的长度和通过介质的长度
    q_ij, p_ij = 0, 0
    for k in range(n*n):
        q_ij += 40 * d_ij[k] * x[k] # 通过空气的总长度
        p_ij += 40 * d_ij[k] * (1-x[k]) # 通过介质的总长度
    return d_ij*40


def get_length_RS(i, j, n):
    # y = (j-i)/6*x+i ==> x=(6y-6i)/(j-i)
    # 直线方程： x = col_id ; y = row_id
    x = np.zeros(n*n)
    d_ij = np.zeros(n*n)
    if i == j:
        if i != 6:
            for number in range(6):
                d_ij[i+number*6] = 1
        if i == 6:
            for number in range(6):
                d_ij[i-1+number*6] = 1
    else:
        for num in range(n*n):
            row_id = num // n
            col_id = num % n
            x_1, y_1, x_2, y_2 = None, None, None, None
            x_3, y_3, x_4, y_4 = None, None, None, None
            y_temp_1 = (j-i)/n*col_id + i
            x_temp_1 = (n*row_id-n*i)/(j-i)
            y_temp_2 = (j-i)/n*(col_id+1) + i
            x_temp_2 = (n*(row_id+1)-n*i)/(j-i)
            if row_id <= y_temp_1 <= row_id+1:
                x_1 = col_id
                y_1 = y_temp_1
            if col_id <= x_temp_1 <= col_id+1:
                x_2 = x_temp_1
                y_2 = row_id
```

```
            if row_id <= y_temp_2 <= row_id+1:
                x_3 = col_id+1
                y_3 = y_temp_2
            if col_id <= x_temp_2 <= col_id+1:
                x_4 = x_temp_2
                y_4 = row_id+1
            dis = max(compute_dis(x_1, y_1, x_2, y_2), compute_dis(x_1, y_1, x_3, y_3),
                      compute_dis(x_1, y_1, x_4, y_4), compute_dis(x_2, y_2, x_3, y_3),
                      compute_dis(x_2, y_2, x_4, y_4), compute_dis(x_3, y_3, x_4, y_4))
            if dis != 0:
                d_ij[num] = dis
                x[num] = 1
    # 计算P_iQ_j的长度和通过介质的长度
    q_ij, p_ij = 0, 0
    for k in range(n*n):
        q_ij += 40 * d_ij[k] * x[k] # 通过空气的总长度
        p_ij += 40 * d_ij[k] * (1-x[k]) # 通过介质的总长度
    return d_ij*40


def get_matrix_D(n, P, Q, R, S):
    D_init = np.array([np.zeros(n*n)])
    for i in range(7):
        # for index_P in range(len(P)):
        #     if i != P[index_P]:
        if i not in P:
            for j in range(7):
                # for index_Q in range(len(Q)):
                #     if index_Q != j:
                if j not in Q:
                    d_ij = get_length_PQ(i, j, n)
                    D_init = np.vstack((D_init, d_ij.reshape(1, n*n)))
    for k in range(7):
        # for index_R in range(len(R)):
        #     if k != index_R:
        if k not in R:
            for l in range(7):
                # for index_S in range(len(S)):
                #     if index_S != l:
                if l not in S:
                    d_ij = get_length_RS(k, l, n)
                    D_init = np.vstack((D_init, d_ij.reshape(1, n*n)))
    D = np.delete(D_init, 0, axis=0)
    return D


def get_vector_t(P, Q, R, S): # 传入参数为要删去的数据
```

```
    T_PQ = np.array([[0.0611, 0.0895, 0.1996, 0.2032, 0.4181, 0.4923, 0.5646],
                     [0.0989, 0.0592, 0.4413, 0.4318, 0.4770, 0.5242, 0.3805],
                     [0.3052, 0.4131, 0.0598, 0.4153, 0.4156, 0.3563, 0.1919],
                     [0.3221, 0.4453, 0.4040, 0.0738, 0.1789, 0.0740, 0.2122],
                     [0.3490, 0.4529, 0.2263, 0.1917, 0.0839, 0.1768, 0.1810],
                     [0.3807, 0.3177, 0.2364, 0.3064, 0.2217, 0.0939, 0.1031],
                     [0.4311, 0.3397, 0.3566, 0.1954, 0.0760, 0.0688, 0.1042]])

    T_RS = np.array([[0.0645, 0.0602, 0.0813, 0.3516, 0.3867, 0.4314, 0.5721],
                     [0.0753, 0.0700, 0.2852, 0.4341, 0.3491, 0.4800, 0.4980],
                     [0.3456, 0.3205, 0.0974, 0.4093, 0.4240, 0.4540, 0.3112],
                     [0.3655, 0.3289, 0.4247, 0.1007, 0.3249, 0.2134, 0.1017],
                     [0.3165, 0.2409, 0.3214, 0.3256, 0.0904, 0.1874, 0.2130],
                     [0.2749, 0.3891, 0.5895, 0.3016, 0.2058, 0.0841, 0.0706],
                     [0.4434, 0.4919, 0.3904, 0.0786, 0.0709, 0.0914, 0.0583]])
    # for i in range(len(P)):
    T_PQ = np.delete(T_PQ, P, axis=0)
    # for i in range(len(Q)):
    T_PQ = np.delete(T_PQ, Q, axis=1)
    # for i in range(len(R)):
    T_RS = np.delete(T_RS, R, axis=0)
    # for i in range(len(S)):
    T_RS = np.delete(T_RS, S, axis=1)
    t1 = T_PQ.flatten()
    t2 = T_RS.flatten()
    t = np.concatenate((t1, t2))
    return t


def get_x(n, P, Q, R, S):
    D = get_matrix_D(n, P, Q, R, S)
    t = get_vector_t(P, Q, R, S)
    vector_one = np.ones(n*n)
    # print(D.shape)
    # print(t.shape)
    t_ = t - np.dot(D, vector_one)/320/40
    A = (1/2880 - 1/320) * D
    # 最小二乘解出x
    x1 = np.linalg.lstsq(A, t_, rcond=None)[0]
    # temp = np.linalg.inv(np.dot(A.T, A))
    # x = np.dot(np.dot(temp, A.T), t_)
    print('最小二乘解为: ')
    print(x1)
    for i in range(n*n):
        # if abs(abs(x1[i])-1) <= 0.26:
        if abs(x1[i]) >= 0.48*(max(abs(x1))-min(abs(x1))):
            x1[i] = 1
```

```
        else:
            x1[i] = 0
    # print('%d*%d划分中删去指定的波源和接收器后所得空洞位置的近似解为(1表示该位置有空洞): ' % (n,
        n))
    # print(x1)
    exect_res = np.array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
        0,
                          0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
    error = np.linalg.norm(exect_res-x1)
    print('与使用全部数据的误差为: %f\n' % error)
    return error




if __name__ == '__main__':
    # 传入的P, Q, R, S参数表示要删去的波源或接收器的编号 (从0开始)
    # P, Q, R, S = [], [], [], [0, 1, 2]
    # err = []
    # for u in range(7):
    #     S[0] = u
    #     for v in range(u+1, 7):
    #         S[1] = v
    #         for w in range(v+1, 7):
    #             S[2] = w
    #             erroring = get_x(6, P, Q, R, S)
    #             err.append(erroring)
    # print(err)
    P, Q, R, S = [5], [], [], []
    get_x(6, P, Q, R, S)
```

附录 B　贪心算法

```
import numpy as np
# 利用贪心法对空洞探测问题的数据进行分析后设定合理的误差
# 以理论时间与实际时间的误差作为贪心选择标准 确定空洞所处的区域以及空洞的个数. 算法复杂度为多项式级


def compute_dis(x_1, y_1, x_2, y_2):
    if x_1 is None or x_2 is None or y_1 is None or y_2 is None:
        return 0
    else:
        return ((x_2-x_1)**2+(y_2-y_1)**2)**0.5
```

```python
# 计算波沿直线P_iQ_j传播的时间
def cal_t_PQ(i, j, loc): # loc表示空洞的坐标
    d_ij = np.zeros(36)
    for num in range(36):
        # 每个小正方形用左上角的点表示
        row_id = num // 6
        col_id = num % 6
        x_1, y_1, x_2, y_2 = None, None, None, None
        x_3, y_3, x_4, y_4 = None, None, None, None
        y_temp_1 = 6 / (i - j) * (col_id - j)
        x_temp_1 = (6 * j + (i - j) * row_id) / 6
        y_temp_2 = 6 / (i - j) * (col_id + 1 - j)
        x_temp_2 = (6 * j + (i - j) * (row_id + 1)) / 6
        if row_id <= y_temp_1 <= row_id + 1:
            x_1 = col_id
            y_1 = y_temp_1
        if col_id <= x_temp_1 <= col_id + 1:
            x_2 = x_temp_1
            y_2 = row_id
        if row_id <= y_temp_2 <= row_id + 1:
            x_3 = col_id + 1
            y_3 = y_temp_2
        if col_id <= x_temp_2 <= col_id + 1:
            x_4 = x_temp_2
            y_4 = row_id + 1
        dis = max(compute_dis(x_1, y_1, x_2, y_2), compute_dis(x_1, y_1, x_3, y_3),
                  compute_dis(x_1, y_1, x_4, y_4), compute_dis(x_2, y_2, x_3, y_3),
                  compute_dis(x_2, y_2, x_4, y_4), compute_dis(x_3, y_3, x_4, y_4))
        if dis != 0:
            d_ij[num] = dis * 40
    t_ij = 0.0

    for k in range(36):
        modify = False
        for l in range(len(loc)):
            if k == loc[l][0]*6+loc[l][1]:
                modify = True
                t_ij += d_ij[k]/320
        if not modify:
            t_ij += d_ij[k]/2880
    return t_ij


# 计算波沿直线R_iS_j传播的时间
def cal_t_RS(i, j, loc): # a, b表示空洞的坐标
    d_ij = np.zeros(36)
    for num in range(36):
```

```
            # 每个小正方形用左上角的点表示
            row_id = num // 6
            col_id = num % 6
            x_1, y_1, x_2, y_2 = None, None, None, None
            x_3, y_3, x_4, y_4 = None, None, None, None
            y_temp_1 = (j - i) / 6 * col_id + i
            x_temp_1 = (6 * row_id - 6 * i) / (j - i)
            y_temp_2 = (j - i) / 6 * (col_id + 1) + i
            x_temp_2 = (6 * (row_id + 1) - 6 * i) / (j - i)
            if row_id <= y_temp_1 <= row_id + 1:
                x_1 = col_id
                y_1 = y_temp_1
            if col_id <= x_temp_1 <= col_id + 1:
                x_2 = x_temp_1
                y_2 = row_id
            if row_id <= y_temp_2 <= row_id + 1:
                x_3 = col_id + 1
                y_3 = y_temp_2
            if col_id <= x_temp_2 <= col_id + 1:
                x_4 = x_temp_2
                y_4 = row_id + 1
            dis = max(compute_dis(x_1, y_1, x_2, y_2), compute_dis(x_1, y_1, x_3, y_3),
                      compute_dis(x_1, y_1, x_4, y_4), compute_dis(x_2, y_2, x_3, y_3),
                      compute_dis(x_2, y_2, x_4, y_4), compute_dis(x_3, y_3, x_4, y_4))
            if dis != 0:
                d_ij[num] = dis * 40
        tao_ij = 0.0
        for k in range(36):
            modify = False
            for l in range(len(loc)):
                if k == loc[l][0] * 6 + loc[l][1]:
                    modify = True
                    tao_ij += d_ij[k] / 320
            if not modify:
                tao_ij += d_ij[k] / 2880
        return tao_ij


    def get_delta_t(loc):
        t = np.array([0.0611, 0.0895, 0.1996, 0.2032, 0.4181, 0.4923, 0.5646,
                      0.0989, 0.0592, 0.4413, 0.4318, 0.4770, 0.5242, 0.3805,
                      0.3052, 0.4131, 0.0598, 0.4153, 0.4156, 0.3563, 0.1919,
                      0.3221, 0.4453, 0.4040, 0.0738, 0.1789, 0.0740, 0.2122,
                      0.3490, 0.4529, 0.2263, 0.1917, 0.0839, 0.1768, 0.1810,
                      0.3807, 0.3177, 0.2364, 0.3064, 0.2217, 0.0939, 0.1031,
                      0.4311, 0.3397, 0.3566, 0.1954, 0.0760, 0.0688, 0.1042,
```

```
                0.0645, 0.0602, 0.0813, 0.3516, 0.3867, 0.4314, 0.5721,
                0.0753, 0.0700, 0.2852, 0.4341, 0.3491, 0.4800, 0.4980,
                0.3456, 0.3205, 0.0974, 0.4093, 0.4240, 0.4540, 0.3112,
                0.3655, 0.3289, 0.4247, 0.1007, 0.3249, 0.2134, 0.1017,
                0.3165, 0.2409, 0.3214, 0.3256, 0.0904, 0.1874, 0.2130,
                0.2749, 0.3891, 0.5895, 0.3016, 0.2058, 0.0841, 0.0706,
                0.4434, 0.4919, 0.3904, 0.0786, 0.0709, 0.0914, 0.0583])
    T = np.zeros((7, 7))
    Tao = np.zeros((7, 7))
    for i in range(7): # 7个探测器，7个接收器
        for j in range(7):
            if i != j:
                T[i][j] = cal_t_PQ(i, j, loc)
                Tao[i][j] = cal_t_RS(i, j, loc)
            else:
                T[i][j] = t[i*7+j]
                Tao[i][j] = t[49+i*7+j]
    delta_t1 = np.linalg.norm(T.reshape(-1)-t[:49])
    delta_t2 = np.linalg.norm(Tao.reshape(-1)-t[49:])
    delta_t = delta_t1 + delta_t2
    # print(delta_t1, delta_t2)
    return delta_t


if __name__ == '__main__':
    delta_t_min = 2.3266+2.7728 # 没有空洞时计算的理论传播时间与实际测量时间的误差
    # 假设有一个空洞（即先确认出第一个空洞的位置）
    for m in range(6):
        for n in range(6):
            temp_t = get_delta_t([[m, n]])
            if temp_t <= delta_t_min:
                delta_t_min = temp_t
                loc_1 = [m, n]
    print('贪心法得到第1个空洞位置：\t%s' % loc_1)
    print('此时理论时间与实际时间的误差为：\t%f' % delta_t_min)
    loc = np.array(loc_1)
    for num in range(1, 36):
        t_record = []
        for m in range(6):
            for n in range(6):
                temp_t = get_delta_t(np.vstack((loc, [m, n])))
                t_record.append(temp_t)
        if min(t_record) > delta_t_min:
            print('此时算得的误差增大，空洞数目不能再增加，贪心法结束')
            print('贪心法所得所有空洞位置为：')
            for k in range(loc.shape[0]):
                print('%d:\t%s' % (k+1, loc[k]))
```

```
            print('算法终止时的误差: \t%f' % delta_t_min)
            break
        else:
            delta_t_min = min(t_record)
            min_index = np.argmin(t_record)
            loc = np.vstack((loc, [min_index//6, min_index % 6]))
            print('贪心法得到第 %d 个空洞位置: \t%s' % (num+1, [min_index//6, min_index % 6]))
            print('理论时间与实际时间的误差为: \t%f' % delta_t_min)
```

附录 C　概率模型算法

```
import numpy as np
from scipy.special import softmax

def get_length_PQ(i, j, m=6):
    # y = 6/(i-j)*(x-j) ==> (i-j)y=6(x-j) ==> 6x-(i-j)y = 6j
    # 直线方程:  x = col_id ; y = row_id
    x = np.zeros(m**2)
    d_ij = np.zeros(m**2)
    for num in range(m**2):
        # 每个小正方形用左上角的点表示
        row_id = num // m
        col_id = num % m
        x_1, y_1, x_2, y_2 = None, None, None, None
        x_3, y_3, x_4, y_4 = None, None, None, None
        y_temp_1 = m / (i-j) * (col_id-j)
        x_temp_1 = (m * j + (i - j) * row_id)/m
        y_temp_2 = m / (i - j) * (col_id+1-j)
        x_temp_2 = (m * j + (i - j) * (row_id+1)) / m
        if row_id <= y_temp_1 <= row_id+1:
            x_1 = col_id
            y_1 = y_temp_1
        if col_id <= x_temp_1 <= col_id+1:
            x_2 = x_temp_1
            y_2 = row_id
        if row_id <= y_temp_2 <= row_id+1:
            x_3 = col_id+1
            y_3 = y_temp_2
        if col_id <= x_temp_2 <= col_id+1:
            x_4 = x_temp_2
            y_4 = row_id+1
        dis = max(compute_dis(x_1, y_1, x_2, y_2), compute_dis(x_1, y_1, x_3, y_3),
                  compute_dis(x_1, y_1, x_4, y_4), compute_dis(x_2, y_2, x_3, y_3),
                  compute_dis(x_2, y_2, x_4, y_4), compute_dis(x_3, y_3, x_4, y_4))
        if dis != 0:
```

```
            d_ij[num] = dis
            x[num] = 1
    return d_ij


def get_line_red_all():
    """计算直线上空洞的比例"""
    m=6
    t1 = np.array([[0.0611, 0.0895, 0.1996, 0.2032, 0.4181, 0.4923, 0.5646],
       [0.0989, 0.0592, 0.4413, 0.4318, 0.477 , 0.5242, 0.3805],
       [0.3052, 0.4131, 0.0598, 0.4153, 0.4156, 0.3563, 0.1919],
       [0.3221, 0.4453, 0.404 , 0.0738, 0.1789, 0.074 , 0.2122],
       [0.349 , 0.4529, 0.2263, 0.1917, 0.0839, 0.1768, 0.181 ],
       [0.3807, 0.3177, 0.2364, 0.3064, 0.2217, 0.0939, 0.1031],
       [0.4311, 0.3397, 0.3566, 0.1954, 0.076 , 0.0688, 0.1042]])
    t2 = np.array([[0.0645, 0.0602, 0.0813, 0.3516, 0.3867, 0.4314, 0.5721],
       [0.0753, 0.07 , 0.2852, 0.4341, 0.3491, 0.48 , 0.498 ],
       [0.3456, 0.3205, 0.0974, 0.4093, 0.424 , 0.454 , 0.3112],
       [0.3655, 0.3289, 0.4247, 0.1007, 0.3249, 0.2134, 0.1017],
       [0.3165, 0.2409, 0.3214, 0.3256, 0.0904, 0.1874, 0.213 ],
       [0.2749, 0.3891, 0.5895, 0.3016, 0.2058, 0.0841, 0.0706],
       [0.4434, 0.4919, 0.3904, 0.0786, 0.0709, 0.0914, 0.0583]])
    p_line = np.zeros((m, m, 2))
    for i in range(m):
        for j in range(m):
            d_ij= (m**2 + (i-j)**2)**0.5*40
            p_line[i, j, 0] = max(
                (2880 * 320 * t1[i, j] - d_ij * 320) / (d_ij * (2880 - 320)), 0)
            p_line[i, j, 1] = max(
                (2880 * 320 * t2[i, j] - d_ij * 320) / (d_ij * (2880 - 320)), 0)

    return p_line


def get_line_red():
    """计算直线上空洞的比例"""
    m = 6
    t1 = np.array([[0.0611, 0.0895, 0.1996, 0.2032, 0.4181, 0.4923, 0.5646],
       [0.0989, 0.0592, 0.4413, 0.4318, 0.477 , 0.5242, 0.3805],
       [0.3052, 0.4131, 0.0598, 0.4153, 0.4156, 0.3563, 0.1919],
       [0.3221, 0.4453, 0.404 , 0.0738, 0.1789, 0.074 , 0.2122],
       [0.349 , 0.4529, 0.2263, 0.1917, 0.0839, 0.1768, 0.181 ],
       [0.3807, 0.3177, 0.2364, 0.3064, 0.2217, 0.0939, 0.1031],
       [0.4311, 0.3397, 0.3566, 0.1954, 0.076 , 0.0688, 0.1042]])
    p_line = np.zeros((m, m))
    for i in range(m):
        for j in range(m):
```

```
            d_ij= (m**2 + (i-j)**2)**0.5*40
            p_line[i, j] = max(
                (2880 * 320 * t1[i, j] - d_ij * 320) / (d_ij * (2880 - 320)), 0)

    return p_line

def get_dis(m):
    """计算接收器之间的距离"""
    dis = np.zeros((m,m))
    for i in range(m):
        for j in range(m):
            dis[i, j] = (m**2 + (i-j)**2)**0.5 * 40

    return dis

def get_length(m = 6):
    """计算直线 (i, j) 在第 n 个小正方形内的长度"""
    d = np.zeros((m, m, m**2)) # 直线 (i, j) 在第 n 个小正方形内的长度
    for i in range(m):
        for j in range(m):
            if i != j:
                d[i, j] = get_length_PQ(i, j, m)

    return d

def get_P(m=6, b=6):
    """计算小正方形 n 为空洞的概率
    + m: 小正方形个数
    + b: 需要的发射器个数 - 1
    """
    P = np.zeros(m**2) # 小正方形 n 为空洞的概率
    d = get_length(m) # 直线 (i, j) 在第 n 个小正方形内的长度
    P_2 = np.zeros(m**2)
    p_line = get_line_red_all() # 直线 (i, j) 上空洞的比例
    if m > 6 or b < 6: # 假设 m % b == 0, 且 6 // b == 0
        stride = m // b
        d = d[::stride, ::stride] # [b, b, m**2]
        stride = 6 // b
        p_line = p_line[::stride, ::stride] # [b, b, 2]
    # 计算小正方形 n 为空洞的概率
    for n in range(m**2):
        P[n] = (p_line[...,0] * d[...,n]).sum() / (d[...,n].sum() + 1e-10)
        P_2[n] = (p_line[...,1] * d[...,n]).sum() / (d[...,n].sum() + 1e-10)
    P_2 = P_2.reshape((m, m)).T.flatten()
    P = (P + P_2) / 2
    return P
```

```python
def get_P_iter(m=6, b=6, epsilon=1e-3, iter_num=100):
    """迭代算法计算小正方形 n 为空洞的概率
    + m: 小正方形个数
    + b: 需要的发射器个数 - 1
    """
    d = get_length(m) # 直线 (i, j) 在第 n 个小正方形内的长度
    P = get_P(m, b)
    P_2 = np.zeros(m**2)
    P_new = np.zeros(m**2)
    p_line = get_line_red_all().reshape(6, 6, 1, 2).repeat(m**2, axis=2)
    p_line_new = np.zeros_like(p_line)
    if m > 6 or b < 6: # 假设 m % b == 0, 且 6 // b == 0
        stride = m // b
        d = d[::stride, ::stride] # [b, b, m**2]
        stride = 6 // b
        p_line = p_line[::stride, ::stride] # [b, b, 2]
        p_line_new = p_line_new[::stride, ::stride] # [b, b, 2]
    for i in range(iter_num):
        if abs(P_new - P).sum() < epsilon:
            return i, get_P(m, b) , P
        P_new = P.copy()
        p_line_new = p_line.copy()
        alpha = softmax(P.reshape(1,1,-1) * d, 2) # 加权和 shape==(b, b, m**2)
        p_line[...,0] = alpha * p_line_new[...,0] # \sum alpha_i * x_i
        alpha = np.transpose(alpha.reshape(b,b,m,m), (0,1,3,2)).reshape(b,b,m**2)
        p_line[...,1] = alpha * p_line_new[...,1]
        for n in range(m**2):
            P[n] = (p_line[...,n, 0] * d[...,n]).sum() / (d[...,n].sum() + 1e-10)
            P_2[n] = (p_line[...,n, 1] * d[...,n]).sum() / (d[...,n].sum() + 1e-10)
        P_2 = P_2.reshape((m, m)).T.flatten()
        P = (P + P_2) / 2

    return iter_num, get_P(m, b), P

if __name__ == "__main__":
    import numpy as np
    import matplotlib.pyplot as plt

    print(get_P_iter(6, 1e-3))
```