基于迭代曲面拟合与离散化的主动反射面形状调节方案设计

摘要

本文对天眼——“FAST”射电望远镜的主动反射面形状调节机制进行建模分析，对其工作原理与过程进行深入探究。首先根据所给数据，计算出基准球面的平均半径约为 300.400m，接着在目标天体位于球心正上方的状态下，求解出世界坐标系下的最优理想抛物面方程。根据天体位置的变化，建立相应的**天体坐标系**，通过旋转矩阵进行两坐标系之间的相互变换。

为尽量贴合理想抛物面，首先尝试主索点位于目标抛物面之上的方法，再进行拟合方案的修改优化，直至达到理想程度。本题主要采用**旋转矩阵**实现坐标转换，利用**迭代算法**进行拟合方案优化，并结合解析几何对问题进行求解。

针对问题一：将三维立体问题简化为二维平面图形，即将对理想抛物面方程的求解转化为对理想抛物线方程的求解。通过解析几何分析，建立可能的抛物线方程。利用**非线性规划**，以促动器伸缩范围、照明区域口径大小等现实情况为约束条件，寻找可以使促动器伸缩长度尽可能小的焦距值，从而确定最终的理想抛物线方程。将该理想抛物线绕着 z 轴旋转，得到理想抛物面方程约为 $z = 0.0017816(x^2 + y^2) - 300.74$。

针对问题二：当目标天体位置移动，以天体与球心所在直线为 z 轴，建立天体坐标系，从而使目标天体在该坐标系下位于球心正上方，将问题二与问题一建立联系，并通过**旋转矩阵**对坐标点进行变换与反变换处理。

首先引入拟合偏差均方根 (RMS) 的概念，选取一系列标记点，将球面离散化处理，来描述拟合效果。方案一选择先让 300 米口径范围内的所有主索节点移至理想抛物面上，发现 RMS 值不太理想。方案二要求计算出 300 米口径内主索节点周围标记点偏离距离均值的均方根值，并将每一个主索节点沿着径向调整，利用**迭代算法**，直至偏差均方根值达到理想状态。最终 300 米口径中所包含的主索节点有 692 个，理想抛物面的顶点精确到小数点后三位为 (-49.375 , -36.931, -294.347)

针对问题三：对电磁波的传播路线进行数学分析。将在 300m 口径内的所有球面形反射面板进行**离散化处理**，根据天文观测的精度要求，每个散点间距不大于 1.5 米，于是在所有反射面板上选取出 45 个标记点，对所有的标记点进行反射电磁波的路径计算，并找出反射电磁波在馈源舱平面的击中位置。对于每一块反射板统计出落在接收有效面积内光点所占比例，并以该板在天体坐标系下水平面投影面积占比为权重，最终将所有 300 米口径内反射板对应的带权值相加，作为该状态下的接收率。最终工作抛物面状态下接收率约为 61.323%，基准球面下接收率约为 5.724%。

关键字：迭代算法　曲面拟合　离散化　旋转矩阵　非线性规划

一、 问题重述

1.1 问题背景

中国天眼——500 米口径球面射电望远镜，简称 FAST，目前是世界上口径最大的射电望远镜，其灵敏度、观测范围都使其他望远镜设备望尘莫及，预计将在未来二三十年内保持世界顶尖设计水平。[1][2] 具有中国独立自主知识产权的中国天眼在体现了我国突飞猛进的技术创新能力的同时，也将对众多基础科学领域提供新的机遇与可能性，并在西部发展、国防建设和国家安全等方面发挥不可替代的作用。[3] 对 FAST 结构的分析和数学模型的创建，将有助于国家大型工程的科普进程，并为管理者对 FAST 实施进一步应用与维护提供便利。[4]

1.2 问题的提出

作为世界上第一个采用变位工作方式索网结构的射电望远镜，FAST 由主动反射面、信号接收系统 (馈源舱)，以及相关的控制、测量和支撑系统组成。其中主动反射面系统由主索网、反射面板、下拉索、促动器和支撑结构组成。

主动反射面分为基准态与工作态。基准态为以一个半径约 300 米，口径 500 米的球面，工作态时反射面则会被调节成一个口径 300 米的近似旋转抛物面。馈源舱只能在焦面上移动。当 FAST 观测天体目标 S 时，馈源舱移至球心 C 和 S 的连线与焦面的交点 P 处。调节反射板排布，使来自目标天体的平行电磁波反射汇聚到馈源舱的有效区域。工作剖面如图 1所示。

而技术的关键在于如何在满足反射面板调节的约束下，确定理想抛物面，并且通过下拉索与促动器的配合来将反射面调节为工作抛物面，使该抛物面贴近理想抛物面，从而可以获得最佳接收效果。

在已知所有主索节点和促动器的坐标、编号及对应关系，以及他们与 4300 块反射板的对应方式后，需解决以下问题：

1. 确立当目标天体位于基准球面正上方时的理想抛物面。
2. 当目标天体位于 α=36.795°，β=78.169° 时确立理想抛物面，并建立反射面板调节模型，使其尽量贴近理想抛物面。最终标注出顶点坐标、300 米口径内主索节点编号、位置坐标和各促动器的伸缩量。
3. 基于第二问的答案，计算调节后馈源舱的接收比，并与基准反射球面的接收比进行比较。

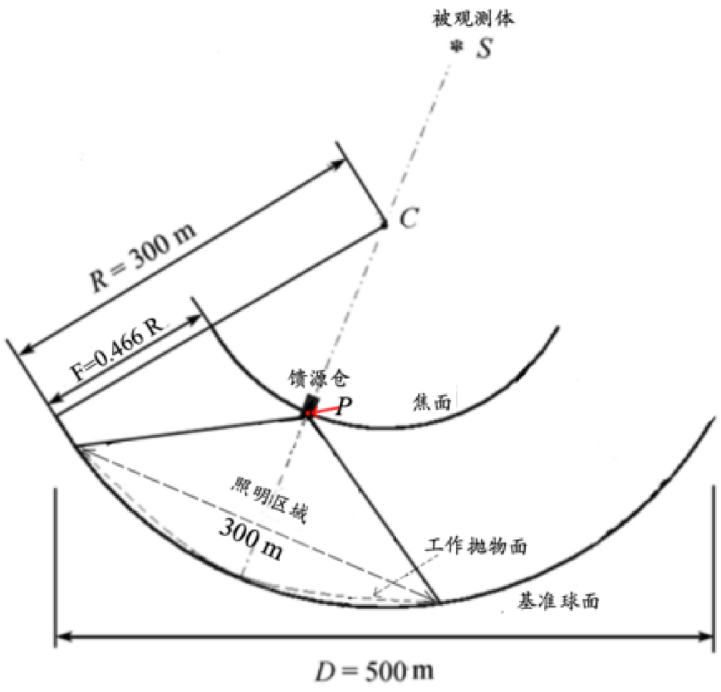


图 **1**　**FAST** 工作剖面图

二、 模型假设

1. **假设**：望远镜外部环境理想，对面板与电磁波反射影响忽略；

   解释：忽略热胀冷缩、风力雨水等对索网结构以及电磁波传播路径的影响。
2. **假设**：反射面板在调整过程中不会发生形变；

   解释：为便于计算模拟反射面板与理想抛物面的拟合贴近过程，认定反射面板在整个过程中一直维持半径为 R 的球面状态。
3. **假设**：不考虑主索、下拉索与主索节点的重力，以及索网内力影响；

   解释：由于结构自重与索网内力的存在，实际促动器在伸缩过程中的实际移动距离会比理想状态下稍小一点，为方便拟合与贴近，我们忽略扰动，认为其可以直接达到理想状态。
4. **假设**：馈源舱不会阻挡电磁波传播路径；

   解释：300 米口径内的所有反射板都可以均匀接收到电磁波。
5. **假设**：下拉索方向一直保持不变；

   解释：当下拉索牵动主索节点进行伸缩时，只会沿着下拉索的初始方向进行，不考虑由于伸缩调节带来的方向上的轻微改动。
6. **假设**：反射面板间有一定的缝隙存在，但缝隙非常小；

   解释：反射面板间缝隙的存在能确保反射面板在变位时不会被羁押、拉扯而变形，但在计算反射信号比时，我们认为各个主索节点间反射面板成填充状态，不会有信号的遗漏。

三、 符号说明

表 **1**　论文符号说明

符号	意义	单位
A	抛物线顶点	/
R	基准球面半径	m
$\frac{p}{2}$	抛物线焦距 (即 A 与 P 间的距离)	m
r	抛物线上任一点与圆心 C 间距离	m
t	抛物线上任一点的横坐标	/
$\mathbf{R_x}, \mathbf{R_y}, \mathbf{R_z}$	分别绕 x, y, z 轴旋转所对应的旋转矩阵	/
$\mathbf{R}$	从世界坐标系转换至天体坐标系的旋转矩阵	/
num	300 米口径内主索节点个数	个
X_0, Y_0, Z_0	天体坐标系下主索基准态时节点坐标	/
X, Y, Z	天体坐标系下主索节点坐标	/
$x_{below}, y_{below}, z_{below}$	天体坐标系下地锚点坐标	/
$x_{above}, y_{above}, z_{above}$	天体坐标系下基准态时顶端坐标	/
RMS_i	第 i 次移动后拟合偏差均方根	m
$\overline{s_{ij}}$	第 i 次移动后标号 j 的主索节点附近标记点偏离距离的均值	m

四、 问题的分析

4.1 问题一的分析

当观测天体 S 位于基准球面正上方，即 α=0°，β=90° 时，该理想抛物面关于直线 CS 对称。当目标天体向望远镜发射出一组竖直向下的平行电磁波信号，信号会经过理想抛物面的反射汇聚到馈源舱 P 处。做截面图，从二维的图像上观察，则可知理想抛物面是由一个以 P 为焦点，最低点 A 为顶点的抛物线，以 CS 为轴旋转所得。

由于基准状态下，促动器顶端径向伸缩量为 0，其径向伸缩范围为 −0.6∼0.6 米，所以照明区域内的理想抛物面应该在促动器可以伸缩的范围之内。为了减少能量损耗并方

便对反射面板进行调节，我们应尽量缩短促动器的伸缩调节距离，所以我们找出所有可能性情况中所需调节的伸缩量的最大值尽可能小的情况，即为最佳的理想抛物面。

4.2 问题二的分析

由于目标天体的方位发生变化，首先建立世界坐标系与天体坐标系，在天体坐标系中，CS 仍沿着 z 轴方向，通过旋转矩阵对世界坐标系旋转变换得到天体坐标系，则变换后的坐标系中理想曲面方程与问题一结果相同。

同理将所有节点坐标通过旋转矩阵变换到天体坐标系中，可以得到天体坐标系下的节点分布，便于量化表达口径约束与伸缩长度限制。为将球形面板与理想抛物面尽可能贴近，首先考虑所有主索点位于理想抛物面上的情况，但最终计算结果并不理想。方案二在每一块球形面板上确定一系列标记点，计算标记点与抛物面间距离的均值，并以该均值为距离大小，沿促动器伸缩方向移动主索点，直至偏差均方根值达到理想范围，确立最优拟合方式。最终通过反变换，将天体坐标系下各点还原到世界坐标系中。

4.3 问题三的分析

在完成反射板的拟合过程后，由于每一个板的位置与朝向各不相同，想要精确计算出球面子块的反射结果非常困难。为解决这一问题，可以将口径内的各个反射面板进行离散化处理，选取较为均匀分布的标记点，并对所有标记点进行反射路径与最终落点计算。由于我们对连续性问题进行了离散化处理，每个反射板的倾斜程度会对其贡献率产生影响，所以将每块板的投影面积占比为权重，最终将每块板的有效反射率带上权值相加，作为接收率。

五、 模型的建立与求解

5.1 建模准备工作

题中将基准球面理想化为一个半径为 300m，口径为 500m 的球体，但根据实际数据可观测得知，该球体半径并不是为确切的 300m，而由于促动器的调节范围为 $-0.6 \sim 0.6$m 这样一个很小的范围内，所以零点几的偏差也会带来很大的影响，从而导致建模结果不理想。所以首先根据各个主索节点的初始坐标，计算出实际的基准球面半径，再开始建模工作。

$$R_i = \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{1}$$

$$R = \overline{R_i} = \frac{\sum_{i=1}^{n} R_i}{n} \tag{2}$$

其中 x_i, y_i, z_i 为各个主索节点的坐标，R_i 表示主索节点初始时与原点的距离，n 表示主索节点个数，R 为最终基准球面半径。最终得到半径 R 约为 300.4m。

5.2 问题一

5.2.1 二维平面坐标系的确立

如图 2所示，做截面图，从二维几何角度分析，以 C 为原点，CS 方向为 y 轴正方向建立平面直角坐标系。基准球面所对应圆的方程式为:

$$x^2 + y^2 = R^2 \quad (y < 0, -250 \le x \le 250) \tag{3}$$

由于促动器的伸缩范围为 −0.6~0.6 米, 则促动器最大伸缩的上下边界为:

$$x^2 + y^2 = (R + 0.6)^2 \quad (y < 0, -250 \le x \le 250) \tag{4}$$

$$x^2 + y^2 = (R - 0.6)^2 \quad (y < 0, -250 \le x \le 250) \tag{5}$$

设顶点 A 与焦点 P 之间的距离 (焦距) 为 $\frac{p}{2}$，则 A 点坐标为 $(0,-\frac{p}{2}-0.534R)$，P 点坐标为 $(0,-0.534R)$。以 A 为顶点，P 为焦点的抛物线方程为：

$$y = -\frac{p}{2} - 0.534R + \frac{x^2}{2p} \tag{6}$$

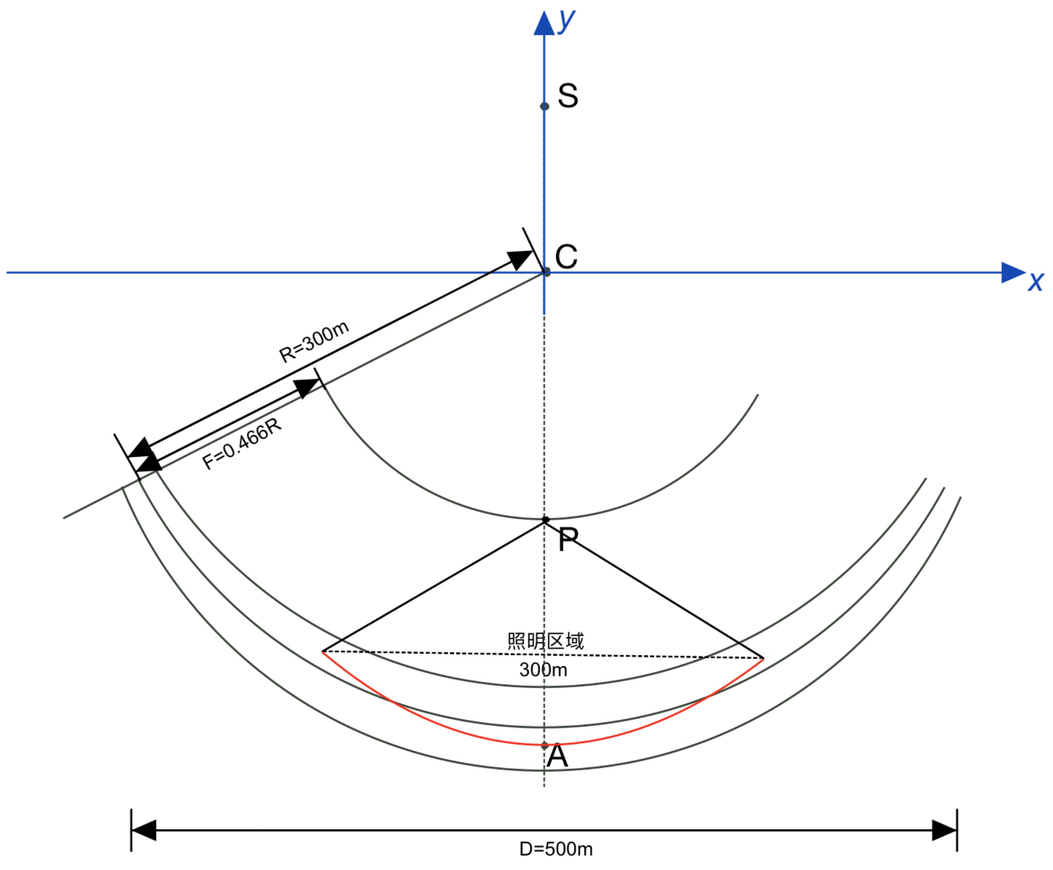


图 2　二维平面坐标系

5.2.2 顶点范围确立与最优方案选取

考虑到促动器的伸缩范围以及照明区域的口径大小限制，理想抛物线服从以下约束条件：

$$s.t.\begin{cases} x^2+y^2 \geq (R-0.6)^2 \\ x^2+y^2 \leq (R+0.6)^2 \\ -150 \leq x \leq 150 \\ y<0 \end{cases} \tag{7}$$

顶点 A 的取值范围为 $(-\frac{p}{2}-R, \frac{p}{2}-R)$，计算理想抛物线上的点与圆心 C 的距离：

$$r=\sqrt{t^2+(-\frac{p}{2}-0.534R+\frac{t}{2p})^2} \tag{8}$$

其中 r 表示抛物线上任意一点距离圆心 C 的距离，t 表示该点的横坐标，p 为两倍的焦距。若需满足(7)的约束条件，则要求 r 值恒大于 $R-0.6$ 小于 $R+0.6$。

在选择最终的最优理想抛物线时，我们决定尽可能缩短促动器所需调节的距离，从而减小调节难度，在节省能量损耗的同时方便操作，减少反射板调节时间。于是最优抛物线的目标函数为：

$$min\{max|r-300|\} \tag{9}$$

5.2.3 问题一最终解答

将式 $s(t)=|r-300|$ 对 t 进行求导运算，找出三个极值点，则 $max|r-300|$ 只可能在三个极值点与两个端点处取得。将五种情况下 t 的取值依次代入到(16)中，则最优状态下促动器的伸缩量 $s=min\{f(p)\}$(已保留 5 位有效数字):

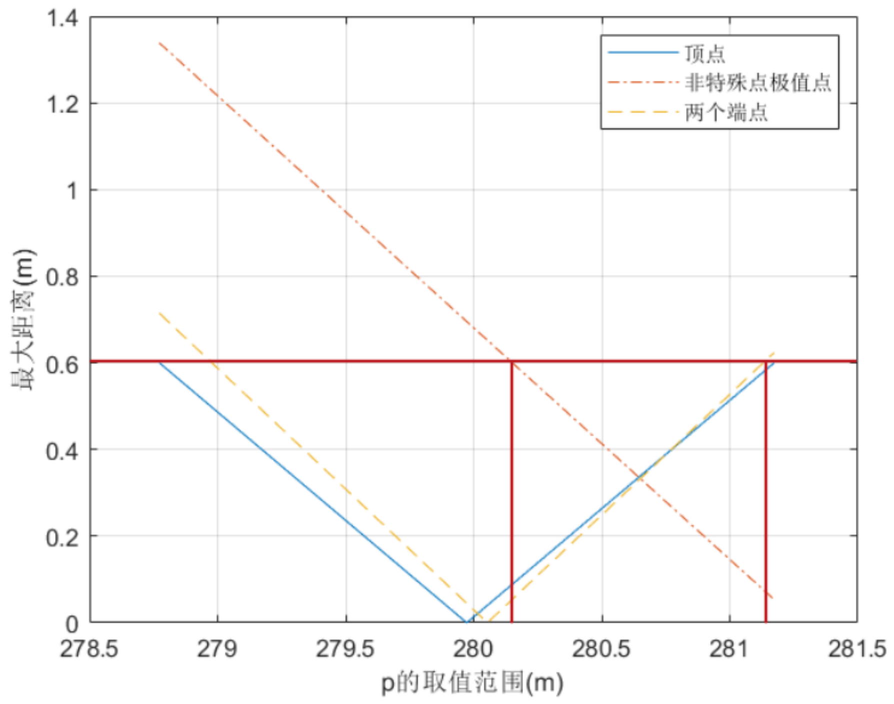
$$f(p)=max\begin{cases} |160.41\sqrt{(0.0031169p+1)^2}-300.40| \\ |17.912\sqrt{p+2.5309\cdot 10^{-28}}-300.40| \\ |11250\sqrt{0.00017778+\frac{(4.4444\cdot 10^{-5}p^2+0.014259p-1)^2}{p^2}}-300.40| \end{cases} \tag{10}$$

其中由于对称性，所以两个端点与两个非顶点极值点所对应的解析式分别相同，最终得到三条关于 p 的曲线。要求满足(9)，则最大距离 $f(p)$ 不得超过 0.6，可能情况为图 3a中红色方框内部区域。

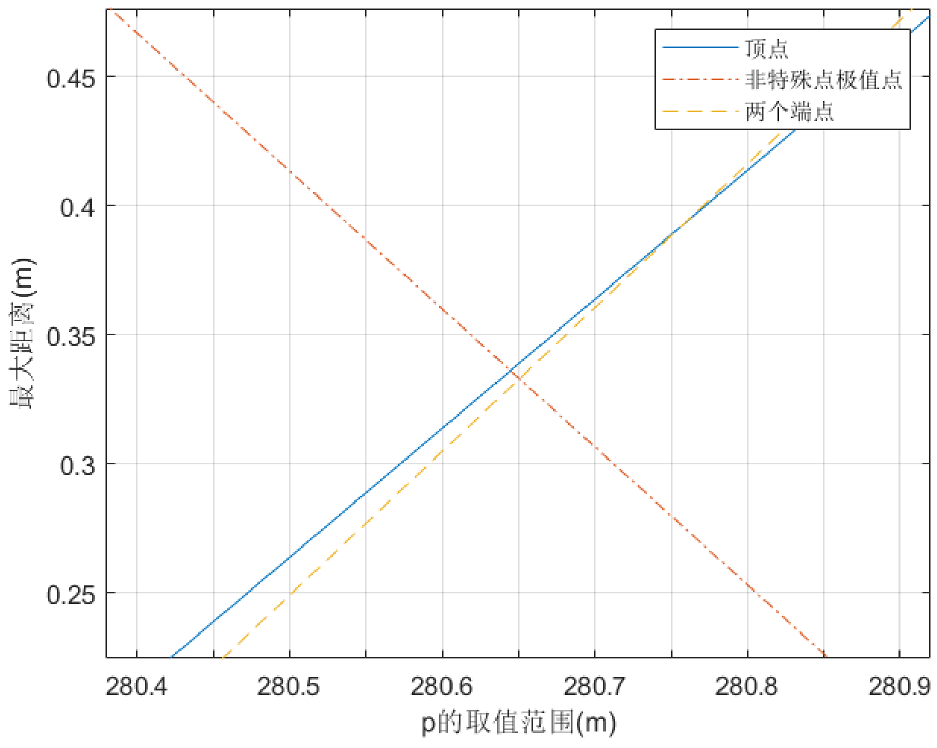
三条曲线分别有三个交点，如图 3b所示，选择 $min\{max|r-300|\}$ 对应的点，即蓝色实线与红色点划线的交点为最佳 p 取值点。最终取值为：$p=280.6446794774304$

由该抛物线旋转所得理想抛物曲面如图 4所示，其表达式为 (保留 5 位有效数字)：

$$z=0.0017816(x^2+y^2)-300.74 \tag{11}$$



(a) $s_{max}(p)$



(b) 交点处放大图

图 **3**　促动器最大移动距离与 **p** 关系图

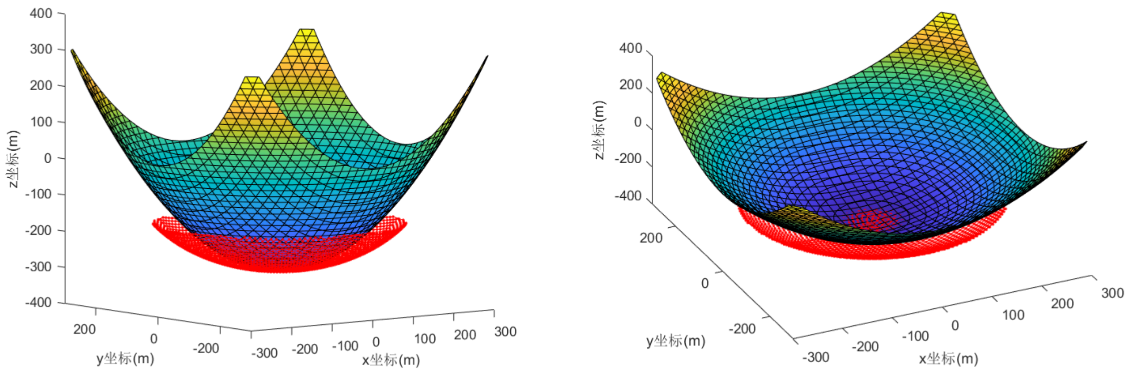
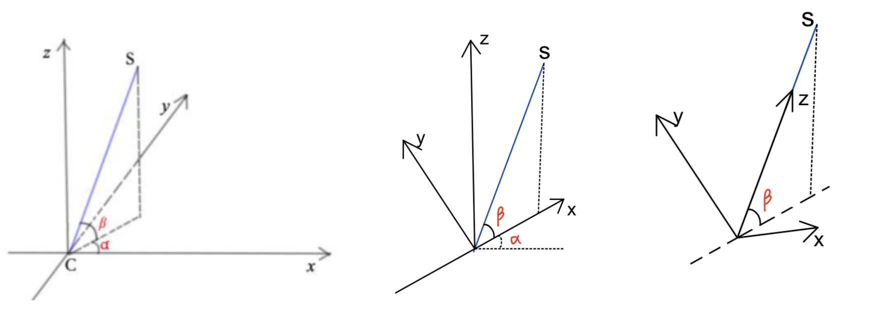


图 **4**　理想曲面

5.3 问题二

5.3.1 旋转矩阵的确立

问题二中目标天体的位置处于 α=36.795°，β=78.169° 的位置，以图 5a所示方向建立世界坐标系。为了将 S 位置转换到坐标系的正上方以方便计算分析，将世界坐标系先后绕 z 轴旋转 α，绕 y 轴旋转 $90° - \beta$, 如图 5b,图 5c所示，得到天体坐标系。



(a) 世界坐标系与 S 方位角　**(b)** 绕 z 轴旋转结果　**(c)** 绕 y 轴旋转结果 **(**天体坐标系**)**

图 **5**　坐标系转换过程

x, y, z 轴分别对应旋转矩阵为：

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_y} = \begin{bmatrix} cos(90° - \beta) & 0 & sin(90° - \beta) \\ 0 & 1 & 0 \\ -sin(90° - \beta) & 0 & cos(90° - \beta) \end{bmatrix}$$

$$\mathbf{R_z} = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

最终将世界坐标系转换为天体坐标系的矩阵为：

$$\begin{aligned}\mathbf{R} &= \mathbf{R_x R_y R_z} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(90^\circ - \beta) & 0 & sin(90^\circ - \beta) \\ 0 & 1 & 0 \\ -sin(90^\circ - \beta) & 0 & cos(90^\circ - \beta) \end{bmatrix} \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} sin\beta cos\alpha & -sin\beta sin\alpha & cos\beta \\ sin\alpha & cos\alpha & 0 \\ -cos\beta cos\alpha & cos\beta sin\alpha & sin\beta \end{bmatrix}\end{aligned}$$

理想抛物面在天体坐标系中的表达式与问题一结果相同。

5.3.2 理想抛物面的确定与顶点坐标计算

设世界坐标系中，理想抛物面的顶点为 $A^T = (x, y, z)$，而在天体坐标系中理想抛物面与问题一中所求相同，顶点为 $A'^T = (0, 0, -300.74)$，则有：

$$\mathbf{A'} = \mathbf{RA}$$

$$\begin{bmatrix} 0 \\ 0 \\ -300.74 \end{bmatrix} = \begin{bmatrix} sin\beta cos\alpha & -sin\beta sin\alpha & cos\beta \\ sin\alpha & cos\alpha & 0 \\ -cos\beta cos\alpha & cos\beta sin\alpha & sin\beta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{12}$$

由此可得世界坐标系下，理想抛物面的顶点坐标为：

$$\mathbf{A} = \mathbf{R^{-1}A'}$$

最终顶点计算结果为：$(-49.3751821, -36.93063567, -294.34729243)$

5.3.3 拟合程度量化分析

由于每一块反射面板均为基准球面的一部分，所以最终的工作抛物面是一个近似旋转抛物面，并不能与理想抛物面完全重合。为了对拟合程度进行量化分析，我们引入拟合偏差均方根 [5] 的概念。

在每一块球面上选取 45 个标记点，对每一块曲面进行离散化分析，如图 6a[5] 所示。以一个主索点为中心点，选取周围六块反射板上与之接近的标记点（即图 6b[5] 中六边形虚线框中的标记点）为研究对象，根据式 (16) 计算出虚线框内所有点与理想抛物

面的距离，并求出均值 $\bar{s}$。计算所有主索节点对应的 $\bar{s}$ 的均方根 RMS。（若点位于理想抛物面以上，距离为正，否则为负）其中 num 表示 300 口径内主索节点个数。

$$RMS = \sqrt{\frac{\sum_{i=1}^{num} \overline{s_i}^2}{num}} \tag{13}$$

在工程中为满足天文观测的需要，要求反射面与工作抛物面的拟合偏差均方根在理想情况（不考虑加工、制作及主索网调节误差）下小于 2.5mm。[6]

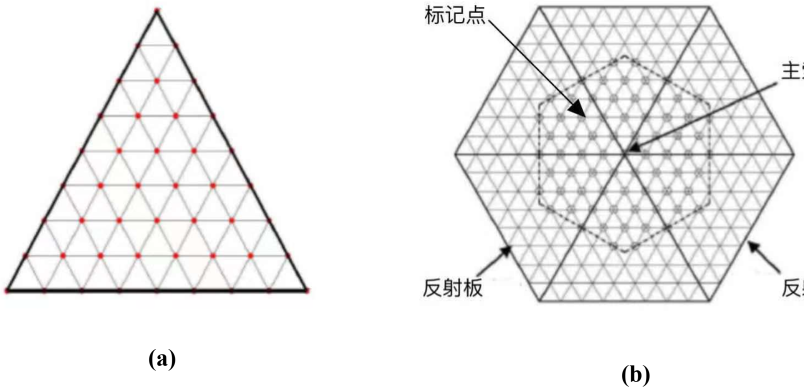


图 **6**　反射板标记点选取与计算

5.3.4 拟合方案一

首先根据题中所提供的的附件 1 与附件 2 中世界坐标系下各个主索节点、下拉索上下端点的坐标通过旋转变换矩阵，变换为天体坐标系下的坐标点。为使反射板与理想抛物面尽量贴合，我们首先将主索节点调整至理想抛物面上，贴近方向即为下拉索的方向。

设天体坐标系下，基准态时主索节点坐标为 (X_0, Y_0, Z_0) 调节后的主索节点坐标为 (X, Y, Z)，调节的方向向量为 $(x_{above} - x_{below}, y_{above} - y_{below}, z_{above} - z_{below})$，可知该主索节点运动轨迹为：

$$\frac{X - x_{below}}{x_{above} - x_{below}} = \frac{Y - y_{below}}{y_{above} - y_{below}} = \frac{Z - z_{below}}{z_{above} - z_{below}} \tag{14}$$

由于促动器伸缩限制以及照明区域口径大小的要求，主索节点坐标应服从以下约束：

$$s.t. \begin{cases} \sqrt{(X - X_0)^2 + (Y - Y_0)^2 + (Z - Z_0)^2} \leq 0.6 \\ \dfrac{X - x_{below}}{x_{above} - x_{below}} = \dfrac{Y - y_{below}}{y_{above} - y_{below}} = \dfrac{Z - z_{below}}{z_{above} - z_{below}} \\ -150 \leq \sqrt{X^2 + Y^2} \leq 150 \\ Z < 0 \end{cases} \tag{15}$$

促动器伸缩长度为：

$$s=\sqrt{(X-x_{above})^2+(Y-y_{above})^2+(Z-z_{above})^2} \tag{16}$$

在确立理想抛物面后，为尽可能将反射面板与抛物面贴近，首先尝试将所有主索节点移动至理想抛物面上的状态。将直线方程式 (14) 与理想抛物面曲面方程式 (11) 联立，得到主索节点运动到抛物面上的各个点。

计算此时的拟合偏差均方根约为 0.0242m，与工程中的理想范围有很大差距，需要寻找更加优化的方案。

5.3.5 拟合方案二

当主索节点都移动至理想抛物面上，如图 7a[6] 所示，由于每一块反射面板都是一个半径为 R 的球面，导致除主索节点以外的其他部分都处于理想抛物面下方，从而未能实现最优化的贴近方式，需要寻找优化方案。

理想中最佳贴近状态下，主索节点会稍微偏离理想抛物面，使得球面反射板较为均匀地分布在理想抛物面的上下两侧，从而减小反射面与抛物面的偏差均方值，如图 7b[6] 所示。

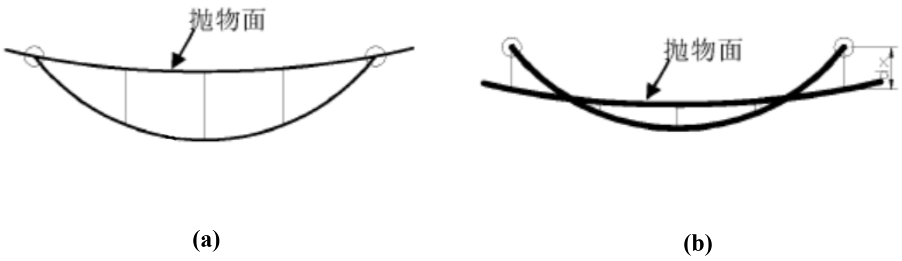


图 7　反射板与抛物面的拟合

反射板初始状态为一个理想球面，计算初始状态下每一个主索节点对应的 $\overline{s_1}$，将主索节点沿径向移动 $\overline{s_1}$ 米的距离，计算此时的拟合偏差均方根 RMS_1。

更新此时主索节点坐标，通过迭代，对上述步骤进行循环操作，第 i 次操作中将主索节点向球心方向移动 $\overline{s_i}$ 距离，并计算此时拟合偏差值的均方根 RMS_i。

$$RMS_i=\sqrt{\frac{\sum_{j=1}^{num}\overline{s_{ij}}^2}{num}} \tag{17}$$

其中 RMS_i 为第 i 次移动后的偏差均方根，num 表示参与运算的主索节点个数，$\overline{s_{ij}}$ 表示参与运算的主索节点附近所有标记点偏离距离的平均值。循环操作，直至 $|RMS_i-RMS_{i-1}|$ 小于 0.1mm 时结束，认为所得结果为优化后的拟合贴近结果。最终主索节点与理想抛物面的偏差距离为：

$$s_j=\sum_{i=1}^{m}\overline{s_{ij}} \tag{18}$$

其中 s_j 为标号为 j 的主索节点与抛物面偏差距离，m 为循环次数，$\overline{s_{ij}}$ 为第 i 次循环时主索节点的移动距离。

算法伪代码如下

Algorithm 1 “FAST”反射板拟合方案二

```
Require: 主索节点初始坐标，理想抛物面表达式，标记点坐标，$i$ =1，$RMS_0$ =0
Ensure: 拟合结果
 1: 计算基准球面上各主索节点对应 $\overline{s_1}$
 2: 将主索节点移动该均值距离
 3: 更新标记点
 4: 计算偏差均方根 $RMS_1$
 5: while $RMS_i - RMS_{i-1} > 0.0001$ do
 6:     i++
 7:     计算各主索节点周围的标记点与理想抛物面距离的均值 $\overline{s_i}$
 8:     将主索节点移动该均值距离
 9:     更新标记点
10:     计算偏差均方根 $RMS_i$
11: end while
```

经过 10 次迭代循环后满足循环终止条件，均方根变化趋势如图 8所示，截止时 RMS 约为 0.108mm。

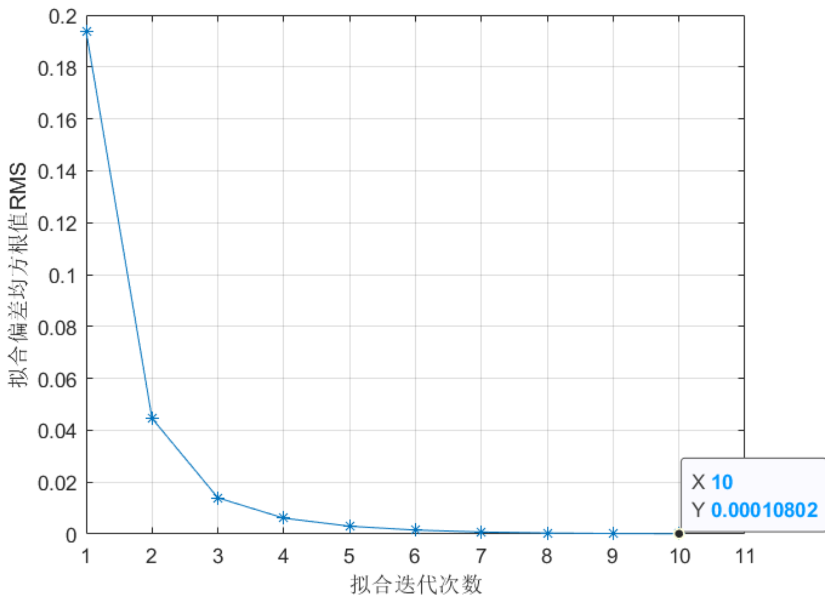


图 8　**RMS** 随迭代次数变化趋势图

确定天体坐标系下各点坐标后，再利用旋转矩阵的逆矩阵，反变换求出世界坐标系

下坐标结果。最终结果数据，包括顶点坐标，主索节点编号、坐标，促动器伸缩量，详情请参考**附录 A** 或支撑材料中“result.xlsx”文件。

5.4 问题三

5.4.1 电磁波传播路径

如图 9所示，对于每一块反射面板上一点 P，入射电磁波与反射电磁波关于径向直线 OP 对称，设 P 点坐标为 (x_p, y_p, z_p)，球心坐标为 (x_o, y_o, z_o)，由于在天体坐标系中，入射电磁波始终沿着竖直方向，所以照射到 P 点处的电磁波直线方程式为：$\begin{cases} x = x_p \\ y = y_p \end{cases}$

设入射电磁波上一点坐标为 (x_p, y_p, z')，反射电磁波上与之关于 OP 对称的一点坐标为 (x, y, z)，两点的中点坐标为 $(\frac{x+x_p}{2}, \frac{y+y_p}{2}, \frac{z+z'}{2})$，且中点在 OP 上，则有以下关系式：

$$\frac{x-x_p}{2(x_p-x_o)} = \frac{y-y_p}{2(y_p-y_o)} = \frac{z+z'-2z_p}{2(z_p-z_o)} \tag{19}$$

$$(x-x_p)(x_p-x_o)+(y-y_p)(y_p-y_o)+(z-z')(z_p-z_o)=0 \tag{20}$$

馈源舱所在平面为：

$$z = -0.534R \tag{21}$$

联立式 (19)、式 (20)、式 (21)，便可以得到反射信号在馈源舱平面的击中坐标，从而判断是否在有效的接收面内。

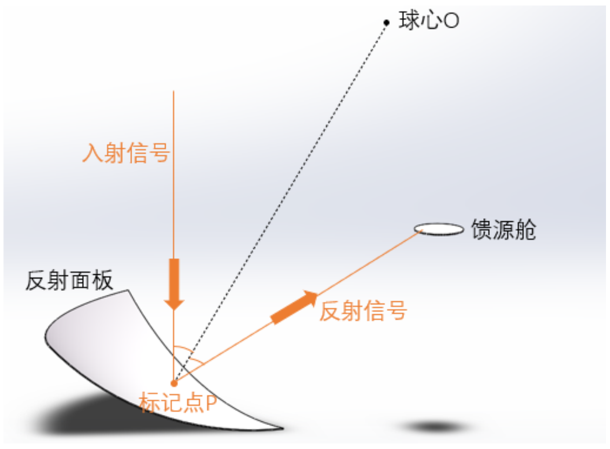


图 **9**　电磁波传播路线

5.4.2 馈源舱接收比计算

由于每一块面板都是一个半径约为 300 米的球面，为方便对该主动反射面的反射接收效率进行研究，将连续性的球面进行离散化处理。根据天文观测的需要，要求在曲面

上所选择的标记点间距不大于 1.5~2m 较为合适。[5] 根据第二问中的标记点选择方式，每个标记点间距离符合要求。我们对口径 300m 范围内的所有标记点进行电磁波入射试验，计算出每一处反射电磁波在馈源舱平面内的击中位置。

对于一块独立的反射板，设该面板的接收率为 w_i，天体坐标系中，300 米口径范围投影至 xoy 平面的面积为 S_{total}，该反射板投影至 xoy 平面的面积为 S_{plate}，300 米口径内反射面板个数为 n。由于我们对连续曲面进行了离散化处理，反射板的倾斜程度会对结果有影响。每块板以 $\frac{S_{plate}}{S_{total}}$ 为权重，最终接受比 w 为：

$$w = \sum_{i=1}^{n} w_i \frac{S_{plate}}{S_{total}} \tag{22}$$

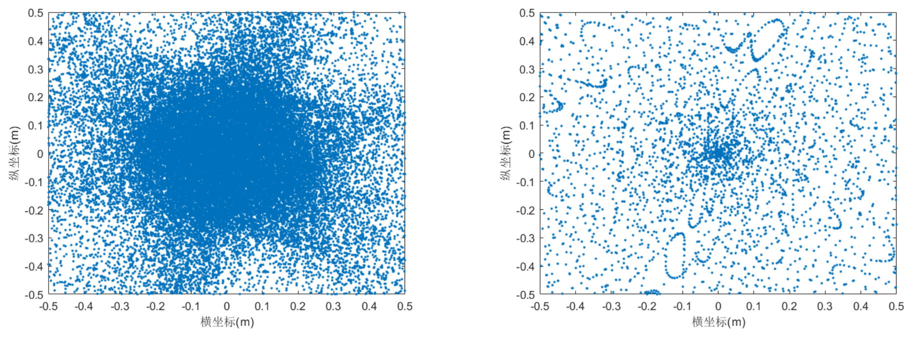
最终结果如表 2所示：

反射面状态	馈源舱接收比
拟合抛物面	61.32298629255489%
基准反射球面	5.723596105664632%

表 **2**　不同状态下馈源舱接收比

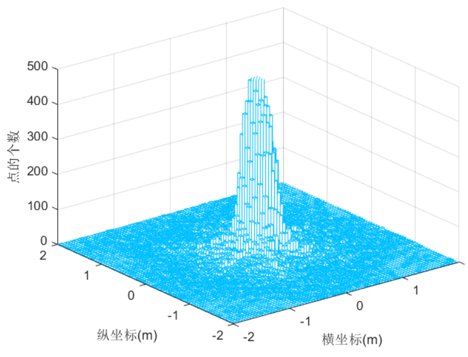
由此可见拟合抛物面的馈源舱接收比远远高于基准反射球面，本文提出的拟合优化方案能够较好解决射电望远镜主动反射面调节功能。

为直观看清散点位置分布，馈源舱圆盘有效接收范围内的散点位置分布图如图 10所示。为更直观看清散点集中程度，馈源舱圆盘有效接收范围内的散点数量分布图如图 11所示。

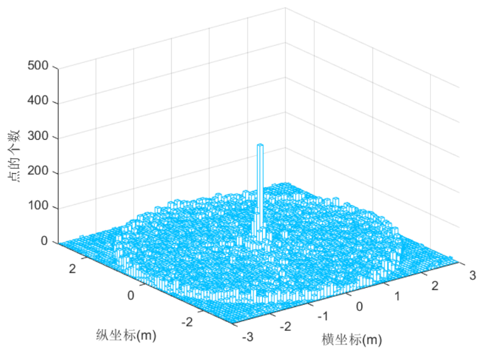


(a) 拟合抛物面结果　　**(b)** 基准反射球面结果

图 **10**　反射电磁波在馈源舱平面散点位置分布图



(a) 拟合抛物面结果



(b) 基准反射球面结果

图 **11**　反射电磁波在馈源舱平面散点数量分布图

六、 模型的总结与评价

6.1 灵敏度分析

最终的最优方案为迭代 10 次后的结果，为验证该方案的合理性与实际效果，我们将迭代次数为 4 至 9 次的拟合方案进行问题三中的电磁波反射试验，计算出不同状况下的馈源舱接收率，如图 12 所示，比较后本文最优方案效果为最佳。

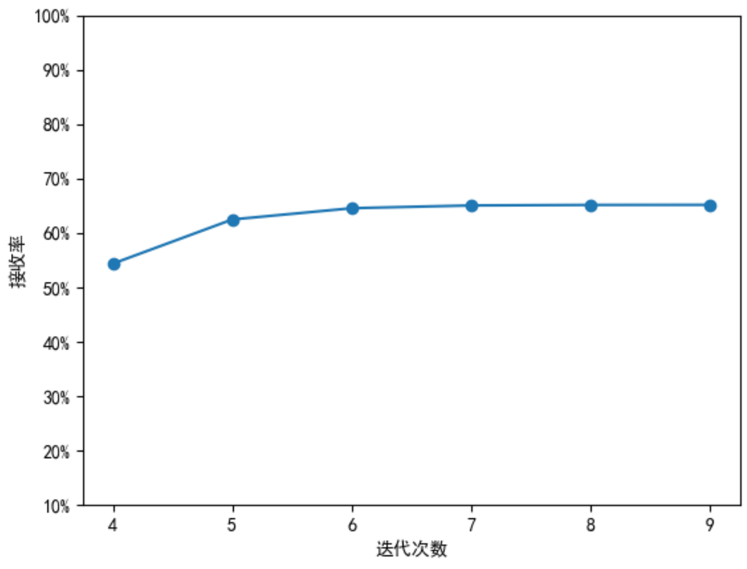


图 **12**　接收率与迭代次数关系图

6.2 模型优点

1. 在计算表达式时，使用 SymPy 进行符号运算，不同于数值运算，符号运算的结果能够清晰体现各变量之间的关系，同时提高模型运用的普适性。
2. 结合工程设计方面的知识与要求，合理设置标记点的选取，将连续性球面离散化处理。同时引入误差均方根，通过迭代优化，将拟合曲面的偏差均方根缩小到工程学要求范围内，符合实际需求。
3. 建立世界坐标系与天体坐标系，使得无论天体如何运动，在天体坐标系中，理想抛物面方程一直不变，且电磁波沿竖直向下入射，从而使不同的天体位置情况下的建模呈现统一化，提高模型的普适能力。
4. 利用 Python 编程实现过程中，将 Numba 库装饰器添加到函数定义中，通过即时编译（JIT）大大提高运算速度与效率。

6.3 模型缺点

1. 建模过程中，理想化反射板，认为由于缝隙的存在，使得反射面板在拉伸移动不会变形。从而未能考虑主索 0.07% 的变化幅度限制。
2. 标记点选取方式为：首先在三个主索节点形成的平面三角区域均匀选取 45 个点，在由球心与之连线投影到球面。所以最终球面上的标记点未能实现完全的均匀分布，而是有极小的偏差。

参考文献

[1] Five hundred meter aperture spherical radio telescope (FAST) [J]. Science in China(Series G:Physics,Mechanics & Astronomy),2006(02):129-148.

[2] 李会贤, 南仁东. [J]. FAST 工程进展及展望 & 自然杂志,2015,37(06):424-434.

[3] Kang Jiao et al. Toward a direct measurement of the cosmic acceleration: roadmap and forecast on FAST[J]. Journal of Cosmology and Astroparticle Physics, 2020, 2020(1)

[4] 赵正旭, 刘曼云, 宋立强, 温晋杰, 赵卫华, 彭育贵. 射电望远镜馈源舱数字化模型的创建[J]. 现代计算机,2019(26):3-7+12.

[5] 钱宏亮. FAST 主动反射面支承结构理论与试验研究[D]. 哈尔滨工业大学,2007.

[6] 沈世钊, 范峰, 钱宏亮. FAST 主动反射面支承结构总体方案研究[J]. 现代计算机, 建筑结构学报,2010,31(12):1-8.

[7] 钱宏亮, 范峰, 沈世钊, 王启明. FAST 反射面支承结构整体索网分析[J]. 哈尔滨工业大学学报,2005(06):750-752.

[8] 庆伟, 姜鹏, 南仁东. FAST 反射面面板单元初始间隙[J]. 机械工程学报,2017,53(17):4-9.

附录 A　问题二最终结果

“result.xlsx”详见支撑文件，以下为表格中具体内容。

理想抛物面的顶点坐标：

X 坐标（米）	Y 坐标（米）	Z 坐标（米）
-49.375182	-36.930636	-294.34729

调节后主索节点编号以及相应坐标:

节点编号	X 坐标（米）	Y 坐标（米）	Z 坐标（米）
A0	8.3089E-15	6.2148E-15	-300.36111
A1	-6.106817	8.40564867	-300.17193
A10	12.1771018	33.5766281	-297.96881
A11	-30.371727	41.8034443	-295.70721
A12	-18.279954	42.0201051	-296.63953
A13	-6.1025489	42.0700736	-297.09656
A14	6.10184907	42.0652684	-297.06252
A15	18.2742843	42.0070743	-296.54754
A16	-36.35972	50.0449818	-293.69232
A17	-24.29734	50.3028748	-294.87388
A18	-12.196521	50.4777938	-295.56823
A19	-3.553E-15	50.5160968	-295.7913
A2	-12.20542	16.7998986	-299.60798
A20	12.195134	50.472055	-295.53462
A21	24.2932885	50.2944869	-294.82472
A22	-42.303312	58.2264114	-291.33487
A23	-30.284614	58.5525436	-292.755
A24	-18.197674	58.7689819	-293.69879
A25	-6.1013497	58.8769622	-294.16725
A26	6.10132745	58.8767473	-294.16618
A27	18.1980651	58.7702451	-293.7051
A28	30.2884862	58.5600305	-292.79244
A29	-48.197266	66.3381771	-288.64507
A3	1.5987E-14	16.8098881	-299.78417

A30	-36.216446	66.719289	-290.30226
A31	-24.176858	66.9914897	-291.48719
A32	-12.098284	67.1553972	-292.20052
A33	-7.105E-15	67.2125869	-292.44817
A34	12.0997606	67.1635942	-292.23619
A35	24.1839913	67.011263	-291.57321
A36	36.2383973	66.7597368	-290.4782
A37	-54.036798	74.3755911	-285.63349
A38	-42.308513	75.2351044	-287.38291
A39	-30.291702	75.5646981	-288.81481
A4	-18.287494	25.1706313	-298.67216
A40	-18.204011	75.7868486	-289.77964
A41	-6.1075742	75.9018764	-290.27758
A42	6.10831693	75.9111057	-290.31288
A43	18.2114667	75.817893	-289.89833
A44	30.3151782	75.6232527	-289.03861
A45	42.3567718	75.3209055	-287.71068
A46	-59.818349	82.3333926	-282.3123
A47	-48.131975	83.255075	-284.27194
A48	-36.37052	84.0475652	-285.79138
A49	-24.310084	84.3415502	-286.99864
A5	-6.1044644	25.2218229	-299.10774
A50	-12.209169	84.5011579	-287.74935
A51	-7.105E-15	84.5852819	-288.03596
A52	12.2142415	84.5362813	-287.86896
A53	24.3288959	84.4068042	-287.2207
A54	36.4238822	84.1706221	-286.20984
A55	48.2624997	83.4808213	-285.04275
A56	-65.53889	90.2067558	-278.69196
A57	-53.906406	91.21345	-280.84678
A58	-42.171335	92.0691867	-282.58137
A59	-30.390052	92.7707844	-283.88598
A6	6.10319115	25.2165687	-299.04541

A60	-18.299178	93.0045855	-284.88217
A61	-6.1114909	93.1893204	-285.40065
A62	6.11260055	93.2062354	-285.45246
A63	18.3149757	93.0848554	-285.12806
A64	30.4514566	92.9582778	-284.45971
A67	-71.197257	97.994732	-274.78861
A68	-59.610251	99.0738459	-277.14436
A69	-47.931243	99.9996633	-279.07147
A7	-24.34536	33.5091932	-297.36989
A70	-36.178413	100.772558	-280.57076
A71	-24.37028	101.396366	-281.65617
A72	-12.197033	101.663277	-282.39638
A73	2.1316E-14	101.792422	-282.75488
A74	12.2133143	101.798972	-282.77331
A79	-76.794757	105.699063	-270.62066
A8	-12.180996	33.5873711	-298.06416
A80	-65.291775	106.926552	-273.06921
A81	-53.643602	107.925681	-275.20143
A82	-41.901021	108.794919	-276.96499
A83	-30.112251	109.519854	-278.32356
A84	-18.325368	109.970008	-279.24084
A85	-6.1225669	110.20478	-279.87754
A9	3.5527E-15	33.6098304	-298.26081
A93	-70.913265	114.692508	-268.99105
A94	-59.343417	115.820456	-271.20994
A95	-47.631799	116.783	-273.18633
B1	6.10510373	8.40329347	-300.08781
B10	35.6965893	-1.2048423	-297.97691
B11	30.3592135	41.7862198	-295.58536
B12	34.3028946	30.3607566	-296.54426
B13	38.1177497	18.8010533	-297.04057
B14	41.8893734	7.19510741	-297.04573
B15	45.5988802	-4.3992274	-296.55677

B16	36.3562119	50.0401688	-293.66399
B17	40.3273513	38.6478429	-294.83611
B18	44.2339694	27.1958493	-295.54054
B19	48.0414764	15.6104466	-295.78019
B2	12.1994929	16.7917403	-299.46254
B20	51.7701314	3.99848908	-295.53489
B21	55.3385259	-7.5614774	-294.81784
B22	42.3196837	58.2489479	-291.44762
B23	46.3379207	46.9069538	-292.81853
B24	50.2761628	35.4732529	-293.73922
B25	54.115424	23.9995629	-294.19774
B26	57.8846715	12.3929496	-294.18752
B27	61.5173207	0.85409386	-293.70513
B28	65.0439491	-10.708609	-292.75109
B29	48.245963	66.4052046	-288.93673
B3	15.9842705	5.19376866	-299.74266
B30	52.3003603	55.1021019	-290.51711
B31	56.27134	43.7192387	-291.64383
B32	60.1524389	32.2709838	-292.312
B33	63.9386433	20.7756373	-292.52304
B34	67.6233399	9.24886939	-292.27251
B35	71.2003304	-2.2919692	-291.55702
B36	74.6636986	-13.830037	-290.37603
B37	54.1560312	74.539696	-286.26369
B38	58.5679047	63.5842746	-287.82234
B39	62.5675837	52.2125613	-289.10309
B4	18.276739	25.1558282	-298.49654
B40	66.504938	40.7648667	-290.01071
B41	70.3419362	29.2813575	-290.45323
B42	74.1127175	17.655501	-290.42886
B43	77.7447384	6.10977242	-289.93677
B44	81.2739013	-5.4611725	-288.98352
B45	84.6830086	-17.000373	-287.57438

B48	68.8603988	60.7091139	-286.48277
B49	72.8256694	49.2679518	-287.49288
B5	22.0936662	13.5951668	-299.01568
B50	76.6805502	37.7681422	-288.0832
B51	80.4977098	26.1557108	-288.22591
B52	84.2032605	14.5122	-287.97473
B53	87.7997215	2.94576945	-287.2431
B54	91.2596799	-8.62679	-286.06474
B55	94.1496243	-20.069346	-284.56341
B6	25.8668965	1.98840255	-299.03536
B61	86.9022132	34.6745962	-285.93752
B62	90.6503123	23.0187609	-285.82357
B63	94.2563081	11.3548971	-285.33541
B64	97.789963	-0.2352429	-284.37649
B65	100.782625	-11.678204	-283.10954
B66	103.643582	-23.133939	-281.48901
B7	24.3314918	33.4901095	-297.20054
B8	28.1681719	21.9559064	-297.95168
B9	31.9588217	10.3846135	-298.21247
C1	9.87939338	-3.2099232	-300.11969
C10	9.89331661	-34.353321	-298.25042
C100	59.5339468	-102.71268	-275.68991
C101	49.7979613	-109.3571	-275.01807
C102	39.9900666	-115.83772	-273.93723
C103	30.0269418	-122.14602	-272.45173
C104	20.0181266	-128.26607	-270.54538
C105	10.0167669	-134.10331	-268.25919
C11	49.1236433	-15.960599	-295.594
C112	88.6981321	-91.595246	-272.27465
C113	79.2044657	-98.614728	-272.45584
C114	69.2956136	-105.7742	-272.38595
C115	59.6226042	-112.50851	-271.90665
C116	49.8403172	-119.07552	-271.02632

C117	39.9952206	-125.43895	-269.74659
C118	30.0009067	-131.65924	-268.05132
C119	20.0385483	-137.6131	-265.97259
C12	39.4825534	-23.246538	-296.60502
C120	10.0222811	-143.35981	-263.49348
C128	88.5888204	-101.83408	-268.91474
C129	79.0251778	-108.7681	-268.89134
C13	29.6728172	-30.455794	-297.17427
C130	69.3741255	-115.55303	-268.52302
C131	59.6381066	-122.17226	-267.78813
C132	49.8326532	-128.63477	-266.71728
C133	39.9546481	-134.90164	-265.24555
C134	30.0193457	-140.9705	-263.38588
C135	20.0385644	-146.82477	-261.12476
C136	10.0272622	-152.45637	-258.46829
C14	19.8021461	-37.643404	-297.26404
C147	69.8547967	-126.81403	-263.66056
C148	60.0170995	-133.33675	-262.6678
C149	50.1485752	-139.6923	-261.3565
C15	9.9168884	-44.77224	-296.8605
C150	40.0840538	-145.89349	-259.58151
C151	30.1020973	-151.82094	-257.44562
C152	20.0665844	-157.57902	-254.96968
C153	10.0333486	-163.09905	-252.10516
C16	58.8166398	-19.110439	-293.61931
C168	40.1771599	-156.77547	-253.66887
C169	30.1465661	-162.60946	-251.29716
C17	49.217386	-26.410596	-294.83307
C170	20.0796655	-168.23919	-248.61199
C171	10.0456708	-171.96249	-246.63793
C18	39.5416436	-33.671646	-295.60159
C19	29.7051802	-40.885127	-295.91833
C2	19.7425929	-6.414424	-299.51411

C20	19.8152252	-48.035403	-295.75359
C21	9.91821173	-55.021855	-295.11178
C22	68.4468819	-22.2398	-291.32901
C23	58.9120722	-29.565558	-292.7312
C24	49.2667671	-36.849643	-293.70496
C25	39.5515023	-44.055016	-294.23012
C26	29.6837486	-51.240696	-294.29445
C27	19.8338036	-58.277274	-293.88393
C28	9.92313873	-65.221817	-292.98961
C29	78.0085392	-25.346543	-288.73327
C3	9.883086	-13.602367	-299.87087
C30	68.5202536	-32.690435	-290.32034
C31	58.9355282	-39.985758	-291.48596
C32	49.2629832	-47.220281	-292.21566
C33	39.5123587	-54.384158	-292.49416
C34	29.695999	-61.462643	-292.30902
C35	19.8278481	-68.445169	-291.65024
C36	9.92356846	-75.317387	-290.50921
C37	87.499315	-28.429572	-285.84897
C38	78.474626	-36.009064	-287.47236
C39	68.9256632	-43.329527	-288.83009
C4	29.5778345	-9.6095371	-298.55149
C40	59.2700811	-50.608851	-289.76275
C41	49.5503768	-57.809623	-290.25054
C42	39.673246	-64.995706	-290.28154
C43	29.8254524	-72.027795	-289.84364
C44	19.9168119	-78.967755	-288.92743
C45	9.99881361	-85.778611	-287.53114
C46	96.9191972	-31.490657	-282.69466
C47	87.929331	-39.08668	-284.447
C48	78.8427803	-46.627266	-285.85265
C49	69.2350104	-53.938769	-286.97177
C5	19.7654575	-16.817389	-299.14052

C50	59.5286451	-61.16841	-287.66801
C51	49.6953925	-68.399603	-287.9075
C52	39.7822057	-75.521662	-287.68838
C53	29.9061867	-82.519049	-286.99101
C54	19.9794268	-89.383716	-285.82458
C55	9.99833548	-95.665799	-284.335
C57	97.3568705	-42.164505	-281.22742
C58	88.2683243	-49.71661	-282.69062
C59	79.1120261	-57.187827	-283.87388
C6	9.89066864	-24.00374	-299.24772
C60	69.4406164	-64.457163	-284.75902
C61	59.6753177	-71.745282	-285.19089
C62	49.7922861	-78.922748	-285.18131
C63	39.8394269	-85.949146	-284.72237
C64	29.9314646	-92.879152	-283.77433
C65	19.9944443	-99.246932	-282.5075
C66	10.0008505	-105.45932	-280.79595
C69	97.6565146	-52.782497	-279.36045
C7	39.3746801	-12.793681	-297.24166
C70	88.5082082	-60.265911	-280.5927
C71	79.2837026	-67.679049	-281.54452
C72	69.5646507	-75.014191	-282.15822
C73	59.7455667	-82.232201	-282.34801
C74	49.8350451	-89.320593	-282.09556
C75	39.8458658	-96.265192	-281.39263
C76	29.9394271	-102.70854	-280.34587
C77	19.9861188	-108.99269	-278.86204
C78	10.0010074	-115.1058	-276.94109
C8	29.5955729	-20.011916	-298.05525
C82	97.9005145	-63.4227	-277.11872
C83	88.6639527	-70.846132	-278.09992
C84	79.356424	-78.089389	-278.86329
C85	69.571634	-85.364375	-279.24261

C86	59.6760284	-92.527135	-279.18887
C87	49.7153502	-99.542373	-278.69397
C88	39.9408969	-106.11207	-277.82467
C89	30.0050031	-112.47969	-276.56043
C9	19.7638301	-27.202578	-298.3965
C90	20.0134351	-118.69622	-274.85929
C91	10.0149935	-124.72623	-272.72678
C96	98.0470948	-74.006164	-274.59871
C97	88.7245404	-81.252666	-275.352
C98	79.3308621	-88.406556	-275.83548
C99	69.4857794	-95.638354	-275.98496
D1	5.3291E-15	-10.391524	-300.23534
D10	-29.64199	-20.043305	-298.52272
D100	-79.339761	-88.416474	-275.86642
D101	-88.686557	-81.217879	-275.23411
D102	-97.889454	-73.88717	-274.15717
D103	-106.96648	-66.350218	-272.6477
D104	-115.86808	-58.707555	-270.69841
D105	-124.49114	-50.985384	-268.35914
D106	0	-148.86946	-260.62583
D107	-10.020461	-143.33378	-263.44561
D108	-20.033691	-137.57975	-265.90812
D109	-29.994096	-131.62936	-267.99047
D11	-3.553E-15	-51.718395	-295.97579
D110	-39.98808	-125.41656	-269.69843
D111	-49.833239	-119.05861	-270.98783
D112	-59.614012	-112.49229	-271.86747
D113	-69.280125	-105.75056	-272.32506
D114	-79.173711	-98.576439	-272.35005
D115	-88.588116	-91.481633	-271.93692
D116	-97.869771	-84.217494	-271.09198
D117	-106.97025	-76.82161	-269.82124
D118	-115.97027	-69.231925	-268.11021

D119	-124.6983	-61.588445	-265.99852
D12	-9.9227728	-44.798792	-297.03657
D120	-133.23884	-53.829315	-263.47729
D121	0	-157.86349	-255.4295
D122	-10.023883	-152.40498	-258.38116
D123	-20.028068	-146.74783	-260.98794
D124	-30.001162	-140.88511	-263.22636
D125	-39.93032	-134.81951	-265.08405
D126	-49.801303	-128.55385	-266.54949
D127	-59.59877	-122.09168	-267.61151
D128	-69.30637	-115.44018	-268.26077
D129	-78.9076	-108.60627	-268.49128
D13	-19.825005	-37.686847	-297.60709
D130	-88.385449	-101.60032	-268.29744
D131	-97.723498	-94.43352	-267.67775
D132	-106.90633	-87.116013	-266.63377
D133	-115.91888	-79.661901	-265.16796
D134	-124.74812	-72.085287	-263.28687
D135	-133.38264	-64.398174	-260.99938
D136	-141.81246	-56.613938	-258.31505
D137	0	-166.68592	-249.93748
D138	-10.029438	-163.03547	-252.00687
D139	-20.051751	-157.46257	-254.78127
D14	-29.720971	-30.505228	-297.65655
D140	-30.073553	-151.67697	-257.20149
D141	-40.032328	-145.70526	-259.2466
D142	-50.065287	-139.46031	-260.92249
D143	-59.906072	-133.09009	-262.1819
D144	-69.689739	-126.51435	-263.03752
D145	-79.351007	-119.76867	-263.47848
D146	-89.390311	-112.48293	-263.49042
D147	-98.799204	-105.38661	-263.0702
D148	-108.08298	-98.117027	-262.22549

D149	-117.18212	-90.724202	-260.9621
D15	-39.559188	-23.291665	-297.18068
D150	-126.21159	-83.10381	-259.2637
D151	-134.94554	-75.463374	-257.1726
D152	-143.50118	-67.701329	-254.67636
D153	-151.83142	-59.869442	-251.79891
D154	2.8422E-14	-175.51973	-244.4233
D155	-10.035755	-171.79266	-246.39549
D156	-20.045858	-167.95583	-248.19325
D157	-30.086715	-162.28661	-250.7982
D158	-40.076518	-156.3828	-253.0335
D159	-50.138561	-150.24555	-254.88639
D16	0	-61.918118	-293.9749
D160	-60.123886	-143.88856	-256.36215
D161	-69.9513	-137.41961	-257.40814
D162	-79.672073	-130.74192	-258.06247
D163	-89.77612	-123.56548	-258.28806
D164	-99.726569	-116.17809	-258.07089
D165	-109.08245	-108.99695	-257.41802
D166	-118.26796	-101.64581	-256.36324
D167	-127.38731	-94.104439	-254.86254
D168	-136.30589	-86.415109	-252.96083
D169	-144.96393	-78.71822	-250.65527
D17	-9.9239088	-55.05345	-295.28124
D170	-153.40604	-70.903072	-247.97483
D171	-160.05844	-62.542859	-246.04742
D174	-20.081203	-176.72516	-242.54718
D175	-30.042432	-172.66771	-244.06567
D176	-40.061843	-166.83797	-246.44285
D177	-50.158213	-160.84774	-248.47053
D178	-60.189781	-154.58741	-250.12834
D179	-70.09357	-148.10235	-251.41736
D18	-19.837785	-48.090067	-296.09018

D180	-79.875337	-141.5205	-252.26507
D181	-90.044024	-134.48554	-252.67699
D182	-100.07692	-127.19305	-252.67382
D183	-109.90654	-119.69222	-252.25285
D184	-119.18002	-112.41562	-251.38879
D185	-128.3901	-104.98706	-250.06597
D186	-137.414	-97.364292	-248.35883
D187	-146.19715	-89.597339	-246.28105
D188	-154.7557	-81.835333	-243.78586
D189	-161.54659	-73.561809	-242.06149
D19	-29.7536	-40.951788	-296.40073
D190	-168.1494	-65.172108	-240.03991
D195	-40.018318	-177.19457	-239.6627
D196	-50.127236	-171.21822	-241.72453
D197	-60.166535	-165.06425	-243.5197
D198	-70.122834	-158.70693	-245.00628
D199	-79.965955	-152.10399	-246.10985
D2	-1.599E-14	-20.773749	-299.73519
D20	-39.620355	-33.738652	-296.18991
D200	-90.197146	-145.18104	-246.70736
D201	-100.27848	-138.02119	-246.90033
D202	-110.19249	-130.63257	-246.6836
D203	-119.92183	-123.02672	-246.05423
D204	-129.22146	-115.68952	-244.91339
D205	-138.32334	-108.17389	-243.39593
D206	-147.21528	-100.49171	-241.50585
D207	-155.88824	-92.656367	-239.25209
D208	-162.81102	-84.45676	-237.72622
D209	-169.53055	-76.14792	-235.8997
D21	-49.323829	-26.467726	-295.47071
D210	-176.0353	-67.738457	-233.77007
D217	-60.036123	-175.36171	-236.80443
D218	-70.04175	-168.99438	-238.33662

D219	-79.93725	-162.46791	-239.58241
D22	-1.421E-14	-72.037756	-291.60773
D220	-90.238616	-155.80613	-240.29529
D221	-100.40668	-148.75493	-240.69122
D222	-110.43981	-141.4503	-240.67489
D223	-120.27052	-133.94104	-240.24552
D224	-129.77233	-126.18868	-239.50427
D225	-138.9787	-118.74911	-238.16383
D226	-147.99954	-111.11641	-236.44038
D227	-156.79071	-103.34687	-234.36135
D228	-163.85288	-95.318558	-232.9792
D229	-170.74936	-87.092584	-231.39229
D23	-9.92851	-65.257133	-293.14825
D230	-177.47633	-78.748574	-229.52107
D231	-184.00512	-70.329086	-227.39013
D24	-19.855467	-58.340922	-294.20489
D240	-79.865028	-172.76554	-232.91136
D241	-90.206055	-166.15511	-233.70781
D242	-100.39865	-159.29941	-234.14762
D243	-110.49617	-152.08441	-234.26998
D244	-120.43459	-144.65853	-234.0626
D245	-130.03968	-137.02064	-233.51261
D246	-139.43012	-129.15786	-232.57733
D247	-148.59002	-121.64563	-231.07479
D248	-157.54179	-113.95061	-229.23645
D249	-164.74664	-106.01226	-228.05044
D25	-29.730958	-51.322191	-294.76247
D250	-171.79653	-97.965893	-226.64228
D26	-39.629836	-44.142266	-294.81292
D265	-110.4847	-162.64923	-227.76472
D266	-120.49881	-155.27558	-227.67293
D267	-130.21415	-147.78009	-227.29033
D268	-139.73617	-140.01671	-226.55207

D269	-149.04185	-132.07134	-225.52361
D27	-49.375136	-36.930678	-294.35094
D28	-59.041938	-29.630741	-293.37655
D29	-7.105E-15	-82.064897	-288.88146
D3	-9.8886053	-13.60996	-300.03821
D30	-9.9284641	-75.354561	-290.65261
D31	-19.848004	-68.514728	-291.94663
D32	-29.740794	-61.555344	-292.74991
D33	-39.587719	-54.487876	-293.05198
D34	-49.369561	-47.322433	-292.84775
D35	-59.066371	-40.074545	-292.13307
D36	-68.659515	-32.756878	-290.9104
D37	-1.421E-14	-91.988113	-285.80501
D38	-10.003134	-85.815702	-287.65547
D39	-19.934982	-79.039786	-289.19095
D4	-1.776E-15	-31.131838	-298.85856
D40	-29.866593	-72.127135	-290.24334
D41	-39.7438	-65.111265	-290.79767
D42	-49.652084	-57.928282	-290.84622
D43	-59.396961	-50.717207	-290.38303
D44	-69.063863	-43.416403	-289.40918
D45	-78.599618	-36.066405	-287.9302
D46	-1.421E-14	-101.79637	-282.38921
D47	-10.00191	-95.699989	-284.4366
D48	-19.994893	-89.452916	-286.04585
D49	-29.942161	-82.618311	-287.33624
D5	-9.896444	-24.017753	-299.42245
D50	-39.845188	-75.641221	-288.14382
D51	-49.787677	-68.526617	-288.44218
D52	-59.645915	-61.288904	-288.23471
D53	-69.364602	-54.039736	-287.50892
D54	-78.961398	-46.69742	-286.28269
D55	-88.011645	-39.123265	-284.71327

D56	0	-111.48236	-278.64436
D57	-10.003543	-105.48771	-280.87152
D58	-20.006759	-99.308067	-282.68152
D59	-29.960838	-92.970292	-284.0528
D6	-19.78709	-16.835801	-299.46801
D60	-39.892294	-86.063186	-285.10016
D61	-49.871377	-79.0481	-285.63424
D62	-59.777175	-71.867728	-285.67762
D63	-69.554244	-64.562654	-285.22502
D64	-79.215649	-57.262734	-284.24572
D65	-88.340657	-49.75735	-282.92227
D66	-97.324596	-42.150525	-281.13419
D67	-2.842E-14	-121.03761	-274.58328
D68	-10.002713	-115.12543	-276.98831
D69	-19.9946	-109.03892	-278.98298
D7	3.5527E-15	-41.45074	-297.60502
D70	-29.961276	-102.78348	-280.55041
D71	-39.886349	-96.363005	-281.67856
D72	-49.897046	-89.431733	-282.44656
D73	-59.826495	-82.343595	-282.73048
D74	-69.654818	-75.111432	-282.52396
D75	-79.36277	-67.746549	-281.82529
D76	-88.557492	-60.299468	-280.74895
D77	-97.604415	-52.754333	-279.2114
D78	-106.48887	-45.125768	-277.22005
D79	7.1054E-15	-130.45656	-270.21779
D8	-9.8992074	-34.373777	-298.42804
D80	-10.015605	-124.73384	-272.74342
D81	-20.017932	-118.72288	-274.92104
D82	-30.018328	-112.52963	-276.68323
D83	-39.967361	-106.18239	-278.00878
D84	-49.75708	-99.625934	-278.92792
D85	-59.731563	-92.613237	-279.44868

D86	-69.631697	-85.438077	-279.48369
D87	-79.403543	-78.135752	-279.02887
D88	-88.683875	-70.86205	-278.1624
D89	-97.813226	-63.366147	-276.87164
D9	-19.786317	-27.233538	-298.73604
D90	-106.80428	-55.766227	-275.12088
D91	-115.61108	-48.101767	-272.92548
D92	1.4211E-14	-139.73484	-265.56192
D93	-10.016182	-134.09548	-268.24353
D94	-20.018063	-128.26567	-270.54452
D95	-30.030653	-122.16112	-272.48541
D96	-40.000829	-115.86891	-274.01097
D97	-49.817266	-109.3995	-275.12469
D98	-59.559263	-102.75636	-275.80715
D99	-69.508271	-95.669313	-276.0743
E1	-9.8844758	-3.2115783	-300.27418
E10	-28.19344	21.9756016	-298.21889
E100	-108.50028	48.0879712	-275.59583
E101	-104.54597	59.1909083	-274.96479
E102	-100.43925	70.2086702	-273.93387
E103	-96.115446	81.1928617	-272.52864
E104	-91.648775	92.0648033	-270.72483
E105	-87.067609	102.770011	-268.69043
E106	-141.55262	-45.993003	-260.56804
E107	-139.42212	-34.763643	-263.45748
E108	-137.05739	-23.46494	-265.94682
E109	-134.47123	-12.150224	-268.02125
E11	-49.224287	-15.993298	-296.19966
E110	-131.63342	-0.7242838	-269.69411
E111	-128.60493	10.6012027	-270.93337
E112	-125.35842	21.9255775	-271.75775
E113	-121.91338	33.1920807	-272.16765
E114	-118.13754	44.8064498	-272.16339

E115	-114.30191	55.9450748	-271.75202
E116	-110.28077	67.0194401	-270.94852
E117	-106.09801	77.9816945	-269.77171
E118	-101.71154	88.9270448	-268.19055
E119	-97.239754	99.6982416	-266.35845
E12	-45.68138	-4.4071785	-297.09327
E121	-150.03033	-48.747866	-255.24684
E122	-148.00195	-37.551815	-258.30803
E123	-145.7465	-26.297882	-260.9731
E124	-143.26544	-15.00353	-263.23424
E125	-140.562	-3.6849574	-265.087
E126	-137.64044	7.63853386	-266.52808
E127	-134.50578	18.9496894	-267.55671
E128	-131.16554	30.2318485	-268.17587
E129	-127.62623	41.4690036	-268.38817
E13	-41.952177	7.20590041	-297.49108
E130	-123.89627	52.6443967	-268.20058
E131	-119.9852	63.746006	-267.62261
E132	-115.90147	74.7619554	-266.66284
E133	-111.64127	85.6715753	-265.30236
E134	-107.27667	96.5204593	-263.70478
E137	-158.3516	-51.451319	-249.65913
E138	-158.03048	-40.809088	-251.80787
E139	-155.89221	-29.576155	-254.68203
E14	-38.161375	18.8225797	-297.38056
E140	-153.52765	-18.266736	-257.16494
E141	-150.9414	-6.9521794	-259.24114
E142	-148.10768	4.51993168	-260.92468
E143	-145.08827	15.8474284	-262.18068
E144	-141.85314	27.1836444	-263.0293
E145	-138.42434	38.4560106	-263.47164
E146	-134.60875	50.2595824	-263.50604
E147	-130.79487	61.4138358	-263.14122

E148	-126.785	72.51315	-262.37052
E149	-122.63597	83.5070092	-261.26133
E15	-34.329386	30.3841939	-296.77323
E150	-118.31708	94.5772569	-259.87674
E154	-166.5144	-54.102923	-243.81489
E155	-166.26729	-43.484827	-246.06978
E156	-165.8087	-32.81165	-248.01159
E157	-163.57369	-21.526393	-250.69477
E158	-161.09121	-10.207951	-252.99816
E159	-158.39331	1.25676105	-254.89737
E16	-58.940209	-19.150588	-294.2362
E160	-155.44913	12.7187971	-256.40016
E161	-152.34608	24.0683965	-257.46813
E162	-149.01423	35.3832747	-258.15051
E163	-145.33121	47.2217298	-258.4132
E164	-141.40883	58.9861991	-258.25296
E165	-137.50259	70.1297628	-257.66526
E166	-133.47325	81.2249962	-256.85491
E17	-55.444486	-7.5759555	-295.38228
E172	-174.51598	-56.702924	-237.72153
E173	-174.36733	-46.115966	-240.07302
E174	-173.9822	-35.451024	-242.13047
E175	-173.36701	-24.766047	-243.87773
E176	-171.01308	-13.450953	-246.38558
E177	-168.4774	-2.0007527	-248.47364
E178	-165.66549	9.47633638	-250.19497
E179	-162.58976	20.9066513	-251.53489
E18	-51.855002	4.00504813	-296.01939
E180	-159.37627	32.2542604	-252.42155
E181	-155.85195	44.1139724	-252.8774
E182	-152.08658	55.9451229	-252.99453
E183	-148.08195	67.6709927	-252.73854
E184	-144.15709	78.8351318	-252.11314

E19	-48.103597	15.6306308	-296.16265
E191	-182.41867	-59.27071	-231.46849
E192	-182.3023	-48.687955	-233.86104
E193	-181.96612	-38.053297	-235.96994
E194	-181.44338	-27.382589	-237.83593
E195	-180.68389	-16.677353	-239.39161
E196	-178.2899	-5.2344129	-241.67169
E197	-175.62932	6.21631733	-243.58996
E198	-172.69415	17.6562669	-245.12744
E199	-169.53587	29.0778092	-246.35002
E2	-19.762704	-6.4209578	-299.81922
E20	-44.273826	27.2203587	-295.80681
E200	-166.20612	40.9827529	-247.09123
E201	-162.61562	52.8373813	-247.45059
E21	-40.347243	38.6669051	-294.98154
E213	-190.03021	-40.64278	-229.71624
E214	-189.52784	-29.93519	-231.63305
E215	-188.85014	-19.221865	-233.30424
E216	-187.88273	-8.5618178	-234.72378
E217	-185.35941	2.90854022	-236.8398
E218	-182.56276	14.4067218	-238.59163
E219	-179.55775	25.8690587	-240.03586
E22	-68.580041	-22.283056	-291.89579
E23	-65.16347	-10.728297	-293.28899
E24	-61.616738	0.85546547	-294.17978
E25	-57.959891	12.409061	-294.56984
E26	-54.164897	24.0215154	-294.46673
E27	-50.300834	35.4906572	-293.88336
E28	-46.340157	46.9092166	-292.83266
E29	-78.130508	-25.386172	-289.18469
E3	-15.996169	5.19764082	-299.96583
E30	-74.77935	-13.851453	-290.82586
E31	-71.299945	-2.2951716	-291.96494

E32	-67.700269	9.25938866	-292.605
E33	-63.988747	20.7919111	-292.75226
E34	-60.173865	32.2824798	-292.41611
E35	-56.264687	43.7140702	-291.60935
E36	-52.26962	55.0697217	-290.34638
E37	-87.578729	-28.455383	-286.10843
E38	-84.764185	-17.016669	-287.85005
E39	-81.348188	-5.4661619	-289.24764
E4	-29.621348	-9.6236773	-298.99071
E40	-77.801048	6.11419857	-290.14676
E41	-74.143343	17.6627956	-290.54888
E42	-70.343801	29.2821337	-290.46093
E43	-66.478509	40.7486658	-289.89547
E44	-62.516017	52.1695221	-288.8648
E45	-58.47956	63.4883754	-287.38822
E46	-96.912823	-31.488586	-282.67607
E47	-94.17489	-20.074733	-284.63978
E48	-91.281285	-8.6288338	-286.13247
E49	-87.81252	2.94619844	-287.28497
E5	-25.89746	1.99074946	-299.38875
E50	-84.195745	14.5109041	-287.94903
E51	-80.466173	26.145464	-288.113
E52	-76.604304	37.7305849	-287.79678
E53	-72.701624	49.1840221	-287.00316
E54	-68.68734	60.556523	-285.76273
E55	-64.298163	71.4952319	-284.23741
E56	-106.12243	-34.481002	-278.89782
E57	-103.48895	-23.099413	-281.06902
E58	-100.66747	-11.664857	-282.78605
E59	-97.674495	-0.2349739	-284.04072
E6	-22.113098	13.6071219	-299.27858
E60	-94.131853	11.3398968	-284.95866
E61	-90.505117	22.9818855	-285.36575

E62	-86.708686	34.597375	-285.30074
E63	-82.765392	46.1263599	-284.77335
E64	-78.806122	57.5468767	-283.76695
E65	-74.502064	68.5315151	-282.47012
E66	-70.072319	79.4332634	-280.77154
E67	-115.19813	-37.430177	-274.78497
E68	-112.6535	-26.078894	-277.16437
E69	-109.92531	-14.684088	-279.09294
E7	-39.446616	-12.817051	-297.78467
E70	-107.01828	-3.2669361	-280.56676
E71	-103.93828	8.15437995	-281.58576
E72	-100.40019	19.8050861	-282.2397
E73	-96.695338	31.4183551	-282.42101
E74	-92.833032	42.9767744	-282.13882
E75	-88.823156	54.4627375	-281.40453
E76	-84.591165	65.4950298	-280.34169
E77	-80.236986	76.4336836	-278.87424
E78	-75.769648	87.2691368	-277.01815
E79	-124.13234	-40.332217	-270.35037
E8	-35.751876	-1.206697	-298.43844
E80	-121.78522	-29.033648	-272.88094
E81	-119.1424	-17.655593	-275.02252
E82	-116.31314	-6.2245331	-276.71669
E83	-113.31426	5.19871692	-277.95533
E84	-110.06794	16.526725	-278.77981
E85	-106.44769	28.1652845	-279.20899
E86	-102.65924	39.7778795	-279.17238
E87	-98.725131	51.3072533	-278.67872
E88	-94.681661	62.3687493	-277.81791
E89	-90.396197	73.3684539	-276.58245
E9	-31.998187	10.3973991	-298.57971
E90	-85.987966	84.2921137	-274.95091
E91	-81.473787	95.0889349	-272.92535

E92	-132.91862	-43.187473	-265.60655
E93	-130.66814	-31.921794	-268.32635
E94	-128.21102	-20.603886	-270.62253
E95	-125.48061	-9.1898186	-272.52416
E96	-122.54868	2.23816144	-273.98767
E97	-119.3976	13.5679214	-275.02694
E98	-116.06043	24.8757035	-275.63624
E99	-112.37117	36.5120368	-275.83962

各促动器伸缩量:

对应主索节点编号	伸缩量（米）
A0	0.03888557
A1	0.048296248
A10	0.298307911
A11	0.212254745
A12	0.242107969
A13	0.277640734
A14	0.312024141
A15	0.335185463
A16	0.263952027
A17	0.281133566
A18	0.304463579
A19	0.326150432
A2	0.073293366
A20	0.33858569
A21	0.331173161
A22	0.307092138
A23	0.314990773
A24	0.326853556
A25	0.336564546
A26	0.337659809
A27	0.320403666
A28	0.276617409
A29	0.333940318

A3	0.144959641
A30	0.335975166
A31	0.338180386
A32	0.337826832
A33	0.327753092
A34	0.301199259
A35	0.249623967
A36	0.154114689
A37	0.336495617
A38	0.334632571
A39	0.330795621
A4	0.111745027
A40	0.321289998
A41	0.300968411
A42	0.26447874
A43	0.198380875
A44	0.098272825
A45	-0.007600544
A46	0.304758778
A47	0.302517762
A48	0.294362862
A49	0.279166131
A5	0.168795941
A50	0.251643825
A51	0.201296183
A52	0.126879124
A53	0.046957796
A54	-0.145055721
A55	-0.511212098
A56	0.230421939
A57	0.232163769
A58	0.221092004
A59	0.198084566

A6	0.231353894
A60	0.16275626
A61	0.108536242
A62	0.054017838
A63	-0.096384054
A64	-0.408615931
A67	0.098886601
A68	0.103525449
A69	0.103116064
A7	0.159538876
A70	0.093700547
A71	0.058152051
A72	0.013956342
A73	-0.119334052
A74	-0.386990112
A79	-0.108222141
A8	0.202277478
A80	-0.038078104
A81	-0.035299795
A82	-0.101979982
A83	-0.2081932
A84	-0.273434797
A85	-0.455287495
A9	0.251575806
A93	-0.497171895
A94	-0.416760695
A95	-0.494687436
B1	0.132470812
B10	0.290149214
B11	0.335953169
B12	0.338503493
B13	0.334195073
B14	0.328990188

B15	0.325854921
B16	0.29289964
B17	0.319573704
B18	0.332577787
B19	0.337427182
B2	0.219085132
B20	0.338313912
B21	0.338174696
B22	0.190949509
B23	0.24987131
B24	0.285546229
B25	0.305464115
B26	0.315890304
B27	0.320380449
B28	0.319004621
B29	0.030739221
B3	0.186532752
B30	0.113896355
B31	0.176922313
B32	0.223352741
B33	0.250935534
B34	0.263907983
B35	0.26628573
B36	0.259717388
B37	-0.325551082
B38	-0.124190207
B39	0.031285405
B4	0.288308726
B40	0.082007151
B41	0.119379492
B42	0.144569222
B43	0.158572493
B44	0.155511719

B45	0.134710321
B48	-0.431653934
B49	-0.237668113
B5	0.261201922
B50	-0.096591636
B51	0.003321771
B52	0.016551648
B53	0.023530285
B54	0.007308193
B55	-0.005183006
B6	0.241447669
B61	-0.45635143
B62	-0.336451604
B63	-0.314918568
B64	-0.320613788
B65	-0.33996677
B66	-0.454246398
B7	0.330517905
B8	0.315561986
B9	0.300216554
C1	0.100569248
C10	0.014679426
C100	0.234725943
C101	0.276894814
C102	0.301229934
C103	0.314424711
C104	0.320352554
C105	0.32150213
C11	0.327181388
C112	-0.250273006
C113	0.016057937
C114	0.093112049
C115	0.156082176

C116	0.202896384
C117	0.237026698
C118	0.257061014
C119	0.265866046
C12	0.277024
C120	0.264174579
C128	-0.487767804
C129	-0.229684834
C13	0.19913766
C130	-0.049480715
C131	0.0780188
C132	0.119589159
C133	0.149925511
C134	0.15653896
C135	0.157912804
C136	0.150907192
C14	0.108456522
C147	-0.396169082
C148	-0.224894936
C149	-0.159856991
C15	0.018516409
C150	-0.056609489
C151	0.010006946
C152	-0.005752225
C153	-0.031408883
C16	0.33855603
C168	-0.49993794
C169	-0.433785257
C17	0.322665657
C170	-0.458140884
C171	-0.43646607
C18	0.270596272
C19	0.197286708

C2	0.167392072
C20	0.116264787
C21	0.039002722
C22	0.31312279
C23	0.339387983
C24	0.32055105
C25	0.272431021
C26	0.206812788
C27	0.137695275
C28	0.074514041
C29	0.242247009
C3	0.058124665
C30	0.317285472
C31	0.339443028
C32	0.322284362
C33	0.280568547
C34	0.226408237
C35	0.170324997
C36	0.122055976
C37	0.110120673
C38	0.241232609
C39	0.314917904
C4	0.233063231
C40	0.338780649
C41	0.328932251
C42	0.296878615
C43	0.255018422
C44	0.213789844
C45	0.179862313
C46	-0.101690224
C47	0.11771153
C48	0.230021465
C49	0.307266689

C5	0.135888214
C50	0.336491415
C51	0.335179198
C52	0.315247605
C53	0.287143397
C54	0.259497672
C55	0.235939131
C57	-0.174663786
C58	0.105039644
C59	0.210879223
C6	0.028283778
C60	0.292536368
C61	0.329239038
C62	0.339322353
C63	0.331171122
C64	0.316153108
C65	0.299558482
C66	0.286494006
C69	-0.207838586
C7	0.289001506
C70	0.070214294
C71	0.177201923
C72	0.267288163
C73	0.313096703
C74	0.333940195
C75	0.339165207
C76	0.334406596
C77	0.32847572
C78	0.323769839
C8	0.211255764
C82	-0.268773164
C83	0.033357477
C84	0.133096658

C85	0.227232155
C86	0.285005531
C87	0.315415113
C88	0.330646832
C89	0.336965824
C9	0.115026765
C90	0.338215503
C91	0.338674629
C96	-0.423413472
C97	-0.087520194
C98	0.076241155
C99	0.162138275
D1	-0.015144201
D10	-0.2595649
D100	0.042550409
D101	0.04112812
D102	0.060280735
D103	0.098581787
D104	0.150609751
D105	0.209690287
D106	0.253098022
D107	0.318691197
D108	0.338615312
D109	0.325198572
D11	-0.060389538
D110	0.290619722
D111	0.245532207
D112	0.199349614
D113	0.16023379
D114	0.132691471
D115	0.122655244
D116	0.130166583
D117	0.153955619

D118	0.191127133
D119	0.236618592
D12	-0.159636527
D120	0.282613785
D121	0.124609994
D122	0.252118918
D123	0.315237624
D124	0.338381377
D125	0.332737294
D126	0.308490061
D127	0.27610215
D128	0.243948578
D129	0.217597368
D13	-0.23809546
D130	0.202928282
D131	0.201815219
D132	0.213598425
D133	0.237755265
D134	0.269408222
D135	0.302079293
D136	0.328918913
D137	-0.021488046
D138	0.085715562
D139	0.216232685
D14	-0.288053877
D140	0.294867134
D141	0.331034733
D142	0.339267625
D143	0.331223749
D144	0.314624152
D145	0.296503618
D146	0.283015427
D147	0.277341574

D148	0.281333073
D149	0.293713381
D15	-0.305459556
D150	0.31124497
D151	0.328568971
D152	0.33983643
D153	0.333550658
D154	-0.515590844
D155	-0.140292516
D156	0.048587703
D157	0.163523326
D158	0.253730271
D159	0.308480969
D16	-0.024833143
D160	0.332481908
D161	0.339289459
D162	0.337575661
D163	0.331496226
D164	0.32779063
D165	0.327769594
D166	0.331199146
D167	0.336569863
D168	0.339917929
D169	0.334634455
D17	-0.133476721
D170	0.312911553
D171	0.283811678
D174	-0.372988097
D175	-0.074319392
D176	0.109848163
D177	0.190865639
D178	0.259379611
D179	0.303230756

D18	-0.225485335
D180	0.323344829
D181	0.333477566
D182	0.337237429
D183	0.33788805
D184	0.337324249
D185	0.334218731
D186	0.325812847
D187	0.306992979
D188	0.270159676
D189	0.229303404
D19	-0.292094805
D190	0.165345126
D195	-0.328509979
D196	-0.031976466
D197	0.119613078
D198	0.177945912
D199	0.232885459
D2	-0.054216151
D20	-0.326744129
D200	0.270728263
D201	0.290953603
D202	0.299625383
D203	0.300726387
D204	0.291781542
D205	0.27222342
D206	0.23981226
D207	0.186728815
D208	0.143069487
D209	0.087071118
D21	-0.326319131
D210	0.024578804
D217	-0.320181925

D218	-0.048684877
D219	0.090734108
D22	0.026031339
D220	0.132720671
D221	0.163628594
D222	0.184006173
D223	0.194906714
D224	0.188693172
D225	0.169138701
D226	0.142130038
D227	0.084635108
D228	0.045504065
D229	-0.071018311
D23	-0.088098286
D230	-0.231272881
D231	-0.44948637
D24	-0.190227784
D240	-0.389180539
D241	-0.206451288
D242	-0.068671751
D243	0.030795814
D244	0.040427689
D245	0.044618538
D246	0.042184181
D247	-0.053426929
D248	-0.188884968
D249	-0.244609103
D25	-0.270585482
D250	-0.395743669
D26	-0.322047354
D265	-0.496225547
D266	-0.374958945
D267	-0.358367942

D268	-0.360250452
D269	-0.460934115
D27	-0.33945208
D28	-0.322122147
D29	0.08819546
D3	-0.109485565
D30	-0.02615971
D31	-0.134783853
D32	-0.22634189
D33	-0.291789701
D34	-0.326816487
D35	-0.326710385
D36	-0.292621168
D37	0.156309854
D38	0.050042361
D39	-0.060003059
D4	-0.075665989
D40	-0.158895446
D41	-0.236719825
D42	-0.28691578
D43	-0.303554175
D44	-0.286733308
D45	-0.236810303
D46	0.223001974
D47	0.128687483
D48	0.027143555
D49	-0.073870317
D5	-0.14710496
D50	-0.159819569
D51	-0.222075792
D52	-0.254629134
D53	-0.254440987
D54	-0.221562159

D55	-0.163390593
D56	0.281698885
D57	0.205721879
D58	0.114701952
D59	0.021674245
D6	-0.192833682
D60	-0.066984059
D61	-0.137244668
D62	-0.182891907
D63	-0.198574999
D64	-0.182324303
D65	-0.141036358
D66	-0.075015776
D67	0.323283831
D68	0.27261215
D69	0.198734121
D7	-0.077863402
D70	0.115473642
D71	0.034271362
D72	-0.039416229
D73	-0.093401627
D74	-0.121749683
D75	-0.122194377
D76	-0.09702293
D77	-0.047452891
D78	0.02151069
D79	0.33913487
D8	-0.164209384
D80	0.320363524
D81	0.270808258
D82	0.20372237
D83	0.131794164
D84	0.063505982

D85	0.005719697
D86	-0.031915049
D87	-0.045190554
D88	-0.034130241
D89	-0.000694906
D9	-0.226663688
D90	0.052637858
D91	0.120053112
D92	0.318179784
D93	0.33902941
D94	0.321301935
D95	0.277329873
D96	0.220447593
D97	0.160534692
D98	0.107081238
D99	0.064943519
E1	-0.054010181
E10	0.046437366
E100	0.337158454
E101	0.33503471
E102	0.304907908
E103	0.229716452
E104	0.121312446
E105	-0.160886025
E106	0.319652456
E107	0.305176968
E108	0.294955958
E109	0.290735428
E11	-0.287654741
E110	0.295423612
E111	0.305845327
E112	0.320494594
E113	0.333792323

E114	0.338479553
E115	0.326821106
E116	0.289076088
E117	0.209072575
E118	0.10116137
E119	-0.169546318
E12	-0.217017387
E121	0.339338585
E122	0.337074665
E123	0.332295529
E124	0.329390059
E125	0.329403226
E126	0.332597908
E127	0.3375613
E128	0.338946284
E129	0.332869771
E13	-0.120899013
E130	0.311300302
E131	0.263653694
E132	0.18087458
E133	0.085626321
E134	-0.206994528
E137	0.313073453
E138	0.322857968
E139	0.333161762
E14	-0.009248871
E140	0.336305505
E141	0.337353554
E142	0.336747622
E143	0.332620592
E144	0.323999238
E145	0.304296751
E146	0.265224294

E147	0.196316298
E148	0.115337598
E149	-0.050404947
E15	0.106820403
E150	-0.398339181
E154	0.233436316
E155	0.256540619
E156	0.268428437
E157	0.287343808
E158	0.295645142
E159	0.295548711
E16	-0.291867717
E160	0.287983573
E161	0.269355174
E162	0.235209137
E163	0.186109658
E164	0.116086893
E165	0.039561323
E166	-0.244289036
E17	-0.236310354
E172	0.095720533
E173	0.123930123
E174	0.143763476
E175	0.157054532
E176	0.179631072
E177	0.18710563
E178	0.179425888
E179	0.162944179
E18	-0.153607644
E180	0.137209341
E181	0.095479061
E182	-0.043621773
E183	-0.239860361

E184	-0.527276241
E19	-0.050571177
E191	-0.211540638
E192	-0.092304233
E193	-0.002342448
E194	0.004508252
E195	0.011659804
E196	0.033693361
E197	0.0329674
E198	0.029479602
E199	-0.060031932
E2	-0.13845254
E20	0.062226736
E200	-0.196263832
E201	-0.377896801
E21	0.171561055
E213	-0.486904035
E214	-0.383647332
E215	-0.373529628
E216	-0.379779944
E217	-0.365094656
E218	-0.370140317
E219	-0.477644132
E22	-0.270693736
E23	-0.232366125
E24	-0.164572046
E25	-0.074093484
E26	0.031076959
E27	0.138282099
E28	0.235390222
E29	-0.227036804
E3	-0.036992187
E30	-0.205234592

E31	-0.153629096
E32	-0.077532515
E33	0.015737608
E34	0.116438456
E35	0.212418433
E36	0.290371403
E37	-0.162441376
E38	-0.153124484
E39	-0.118902544
E4	-0.208530626
E40	-0.05888607
E41	0.02048697
E42	0.111420128
E43	0.201342459
E44	0.278861959
E45	0.329090277
E46	-0.081927188
E47	-0.085804286
E48	-0.063812023
E49	-0.020252688
E5	-0.113277748
E50	0.043362984
E51	0.121006055
E52	0.202178163
E53	0.274437067
E54	0.324442032
E55	0.33896338
E56	0.008697754
E57	-0.00536879
E58	0.003669145
E59	0.034457058
E6	-0.002691125
E60	0.082142439

E61	0.145256403
E62	0.213659057
E63	0.277441634
E64	0.323952857
E65	0.339269417
E66	0.312576235
E67	0.102875599
E68	0.081843802
E69	0.080018941
E7	-0.259249044
E70	0.097978317
E71	0.133228147
E72	0.180623447
E73	0.235509041
E74	0.287922835
E75	0.326487143
E76	0.338878006
E77	0.315346931
E78	0.240275459
E79	0.19192207
E8	-0.174678384
E80	0.169065944
E81	0.160021322
E82	0.16742522
E83	0.189519125
E84	0.22297962
E85	0.263376784
E86	0.302729846
E87	0.331837481
E88	0.337941301
E89	0.31307468
E9	-0.069345854
E90	0.238191045

E91	0.120203534
E92	0.26775016
E93	0.246378637
E94	0.234773102
E95	0.234641429
E96	0.245969849
E97	0.267217438
E98	0.293161975
E99	0.320254425

附录 B　重要代码

运行要求：

Python 3.8.10 64-bit

pandas 1.2.0

numpy 1.20.3+mkl

numba 0.54.0

matplotlib 3.3.3

matplotlib-inline 0.1.2

sympy 1.7.1

scipy 1.6.0

openpyxl 3.0.7

Jupyter notebook（用于运行.ipynb 文件）

所有的“this_dataset.bak”、“this_dataset.dat”、“this_dataset.dir”是 shelve 生成的数据存储文件，不需要主动运行。

“__pycache__”文件夹中的文件不需要主动运行。

所有文件中最长的运行时间在 20 秒左右。

2.1　准备工作代码

说明：读取数据的三个.py 文件分别读取“附件 1.cxv”,“附件 2.cxv”,“附件 3.cxv”。

只需要主动运行“生成 this_dataset.py”文件，生成“this_dataset.py”用于后续问题的计算。其他的.py 文件都是被调用的，不需要主动运行。

“this_dataset.bak”、“this_dataset.dat”、“this_dataset.dir”是 shelve 生成的数据存储文件，不需要主动运行。“__pycache__”文件夹中的文件是 Python 在 import 过程中自动

生成的，不需要主动运行。

2.1.1 读节点数据.py

```
import pandas as pd
import numpy as np
data = pd.read_csv("附件2.csv",encoding="gbk")
a=data["对应主索节点编号"]
b=data['下端点X坐标（米）']
c=data['下端点Y坐标（米）']
d=data['下端点Z坐标（米）']
e=data['基准态时上端点X坐标（米）']
f=data['基准态时上端点Y坐标（米）']
g=data['基准态时上端点Z坐标（米）']
data_info节点={a[i]:([b[i],c[i],d[i]],[e[i],f[i],g[i]]) for i in range(2226)}
```

2.1.2 读结点.py

```
import pandas as pd
import numpy as np
data = pd.read_csv("附件1.csv",encoding="gbk")
a=data['节点编号']
b=data['X坐标（米）']
c=data['Y坐标（米）']
d=data['Z坐标（米）']
data节点={a[i]:[b[i],c[i],d[i]] for i in range(2226)}
```

2.1.3 读反射面板的顶点.py

```
import pandas as pd
data = pd.read_csv("附件3.csv",encoding="gbk")
a=data['主索节点1']
b=data['主索节点2']
c=data['主索节点3']
data面板=[(a[i],b[i],c[i]) for i in range(4300)]
# print(data面板)
```

2.1.4 生成 this_dataset.py

```
from 读结点 import data节点
from 读节点数据 import data_info节点
from 读反射面板的顶点 import data面板
```

```
import shelve
with shelve.open("this_dataset") as d:
    d["data节点"]=data节点
    d["data_info节点"]=data_info节点
    d["data面板"]=data面板
```

2.2 问题一代码

说明：“抛物线.ipynb”是计算理想抛物面的程序过程。

```
[1]: import shelve
     with shelve.open("this_dataset") as dat:
         data 节点=dat["data 节点"]
```

```
[2]: import numpy as np
     from numba import njit
     @njit
     def pjfunc(x:np.ndarray):
         return np.sqrt(np.sum(np.square(x)))
```

```
[3]: R=0
     for i in data 节点:
         R+=pjfunc(np.array(data 节点 [i]))
     R/=len(data 节点)
```

```
[4]: R
```

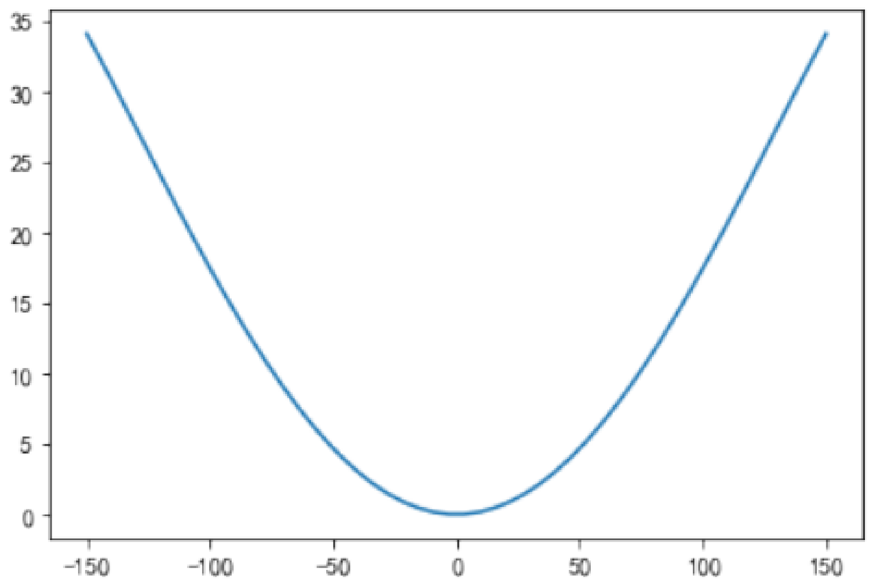
```
[4]: 300.4000111215409
```

```
[5]: a=1/(0.466*2*R)
```

1 尝试顶点在原点，发现失败（伸缩量过大）

```
[6]: #y=a*x^2-R
     import numpy as np
     t=np.linspace(-150,150)
     d=np.sqrt(np.square(t)+np.square(a*t**2-R))
     import matplotlib.pyplot as plt
     plt.plot(t,np.abs(d-R))
```

```
[6]: [<matplotlib.lines.Line2D at 0x1ec695fcfa0>]
```



2 所有点都伸缩

计算所有可能的极值点

```
[7]: import sympy as sy
     t,p=sy.symbols("t,p")
     val=sy.sqrt(t**2+(t**2/(2*p)-sy.Rational(534,1000)*R-p/2)**2)-R
     res=sy.solve(sy.diff(val,t),t)
```

```
[8]: res# 打印极值点
```

```
[8]: [0.0,
      -6.00212753600341e-15*sqrt(-p*(2.77580888431871e+28*p - 8.9055502506161e+30)),
      6.00212753600341e-15*sqrt(-p*(2.77580888431871e+28*p - 8.9055502506161e+30))]
```

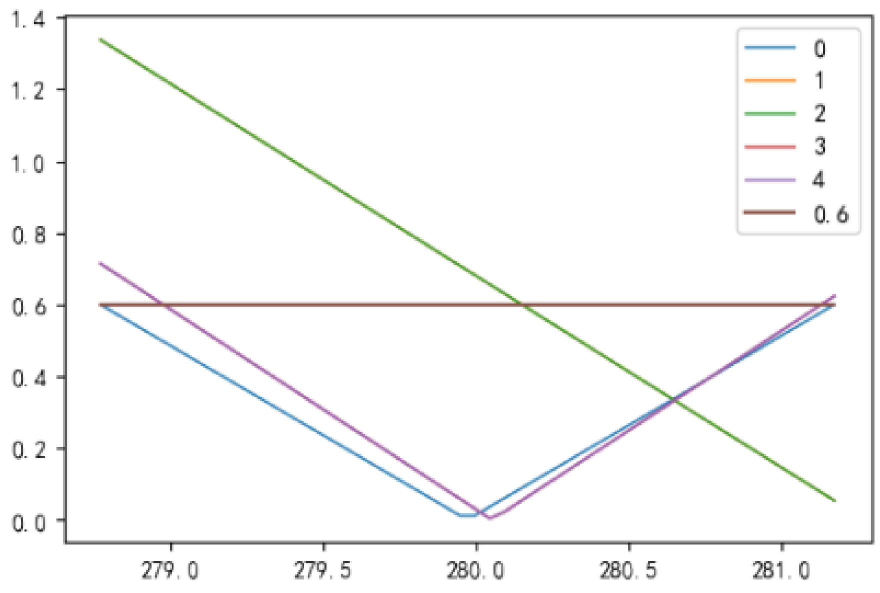
```
[9]: susped=[]
     for i in res+[-150,150]:# 添加端点
         susped.append(sy.simplify(val.subs(t,i)))
```

```
[10]: for i in susped: # 打印所有可能的最值点
          print(i)
```

```
160.413605938903*sqrt((0.00311694258771563*p + 1)**2) - 300.400011121541
17.9116501718241*sqrt(p + 2.53088044781459e-28) - 300.400011121541
17.9116501718241*sqrt(p + 2.53088044781459e-28) - 300.400011121541
11250.0*sqrt(0.000177777777777778 + (4.44444444444444e-5*p**2 +
0.0142589871945691*p - 1)**2/p**2) - 300.400011121541
11250.0*sqrt(0.000177777777777778 + (4.44444444444444e-5*p**2 +
0.0142589871945691*p - 1)**2/p**2) - 300.400011121541
```

画出图（在 0.6 以下的 p 范围为所求范围）

```
[11]: t=np.linspace(0.466*R-0.6,0.466*R+0.6)*2
      fig=plt.figure(dpi=100)
      for ii in range(len(susped)):
          plt.plot(t,[abs(susped[ii].subs(p,i)) for i in t],label=str(ii),linewidth=1)
      plt.plot(t,[0.6 for i in t],label=0.6)
      plt.legend()
      plt.show()
```



求 2 与 4 的交点

```
[12]: resu1=sy.nsolve(-susped[2]-susped[4],p,280)
      print(float(resu1))
      float(-susped[2].subs(p,resu1))
```

```
280.6505555952178
```

```
[12]: 0.332793212668264
```

求 2 与 0 的交点

```
[13]: resu2=sy.solve(-susped[2]-susped[0],p)
      print(float(resu2[0]))
      print(float(-susped[2].subs(p,resu2[0])))
      sy.simplify(-susped[2].subs(p,resu2[0]))
```

```
280.6446794774304
0.335934556077234
```

[13]: 0.335934556077234

求 0 与 4 的交点

```
[14]: resu3=sy.solve(susped[4]-susped[0],p)
      for i in resu3:
          print(i)# 保留第二个结果
```

```
40.0702431376183
280.756968740196
624.939121110421 - 5835031104.25085*I
624.939121110421 + 5835031104.25085*I
```

```
[15]: print(float(resu3[1]))
      print(float(susped[0].subs(p,resu3[1])))
```

```
280.75696874019565
0.3920791874597853
```

求可用范围

```
[16]: p1=sy.solve(-susped[2]-sy.Rational(6,10),p)[0]
      print(float(p1))
```

```
280.15094524684145
```

```
[17]: p2=sy.solve(susped[4]-sy.Rational(6,10),p)[3]
      print(p2)
```

```
281.130255084147
```

3 最优抛物面方程

```
[18]: print("z={}*(x^2+y^2)".format(1/(2*280.6446794774304))+"-{}".format(0.534*R+280.
      ↪6446794774304/2))
```

```
z=0.001781612254082338*(x^2+y^2)-300.735945677618
```

$$z = 0.001781612254082338(x^2 + y^2) - 300.735945677618$$

2.3 问题二代码

说明：运行“main.py”会打印迭代中的均方根误差，并将计算好的坐标数据写入“result.xlsx”。

“点到抛物线的最短距离的公式.ipynb”是用 sympy 进行符号计算得到点到抛物面的最短距离的公式的过程。“点到抛物线的球心径向距离.ipynb”是用 sympy 进行符号计算得到点到抛物面沿球心径向方向的的距离的公式的过程。

“this_dataset.bak”、“this_dataset.dat”、“this_dataset.dir”是 shelve 生成的数据存储文件，不需要主动运行。

2.3.1 点到抛物线的径向距离

```
[1]: import sympy as sy
```

```
[4]: x,y,a,b,x0,y0=sy.symbols("x,y,a,b,x0,y0")
```

```
[6]: res=sy.solve(sy.Eq(y0/x0*x,a*x**2-b),x)
```

```
[7]: res[0]
```

[7]: $$\frac{y_0 - \sqrt{4abx_0^2 + y_0^2}}{2ax_0}$$

[11]:
```
x,y,z=sy.symbols("x,y,z")
sy.simplify(sy.sqrt((res[1]-x0)**2+(y0*res[1]/x0-y0)**2)).subs(x0**2,x**2+y**2).
↪subs(y0,z)
```

[11]:
$$\frac{\sqrt{\frac{\left(x_0^2+y_0^2\right)\left(2ax_0^2-y_0-\sqrt{4abx_0^2+y_0^2}\right)^2}{a^2x_0^4}}}{2}$$

1 点到抛物线的球心径向距离

[12]:
```
x,y,z=sy.symbols("x,y,z")
sy.simplify(sy.sqrt((res[1]-x0)**2+(y0*res[1]/x0-y0)**2)).subs(x0**2,x**2+y**2).
↪subs(y0,z)
```

[12]:
$$\frac{\sqrt{\frac{(x^2+y^2+z^2)\left(2a(x^2+y^2)-z-\sqrt{4ab(x^2+y^2)+z^2}\right)^2}{a^2(x^2+y^2)^2}}}{2}$$

2.3.2 点到抛物线的最短距离

[1]:
```
from sympy import *
a,b,t,x0,y0=symbols("a,b,t,x0,y0")
```

[2]:
```
resu=solve((a*t**2-b-y0)*2*a*t+(t-x0),t)
resu
```

[2]:
```
[-(-1/2 - sqrt(3)*I/2)*(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 +
1)**3/(2*a**6))/2 - 27*x0/(4*a**2))**(1/3)/3 + (-2*a*b - 2*a*y0 +
1)/(2*a**2*(-1/2 - sqrt(3)*I/2)*(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 +
1)**3/(2*a**6))/2 - 27*x0/(4*a**2))**(1/3)),
 -(-1/2 + sqrt(3)*I/2)*(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 +
1)**3/(2*a**6))/2 - 27*x0/(4*a**2))**(1/3)/3 + (-2*a*b - 2*a*y0 +
1)/(2*a**2*(-1/2 + sqrt(3)*I/2)*(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 +
1)**3/(2*a**6))/2 - 27*x0/(4*a**2))**(1/3)),
 -(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 + 1)**3/(2*a**6))/2 -
27*x0/(4*a**2))**(1/3)/3 + (-2*a*b - 2*a*y0 +
1)/(2*a**2*(sqrt(729*x0**2/(4*a**4) + 27*(-2*a*b - 2*a*y0 + 1)**3/(2*a**6))/2 -
27*x0/(4*a**2))**(1/3))]
```

[3]:
```
len(resu)
```

[3]:
```
3
```

[4]:
```
simplify(resu[0])# 复根, 舍去
```

[4]:

$$\frac{\sqrt[3]{6}\left(a^2\left(\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}\right)^{\frac{2}{3}}\left(1+\sqrt{3}i\right)^2+4\sqrt[3]{6}\left(2ab+2ay_0-1\right)\right)}{12a^2\sqrt[3]{\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}}\left(1+\sqrt{3}i\right)}$$

[5]:
```
simplify(resu[1])# 复根，舍去
```

[5]:

$$\frac{\sqrt[3]{6}\left(a^2\left(\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}\right)^{\frac{2}{3}}\left(1-\sqrt{3}i\right)^2+4\sqrt[3]{6}\left(2ab+2ay_0-1\right)\right)}{12a^2\sqrt[3]{\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}}\left(1-\sqrt{3}i\right)}$$

[6]:
```
t=simplify(resu[2])# 实根，代入
t
```

[6]:

$$\frac{\sqrt[3]{6}\left(-a^2\left(\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}\right)^{\frac{2}{3}}+\sqrt[3]{6}\left(-2ab-2ay_0+1\right)\right)}{6a^2\sqrt[3]{\frac{\sqrt{3}a^2\sqrt{\frac{27a^2x_0^2-2(2ab+2ay_0-1)^3}{a^6}}-9x_0}{a^2}}}$$

1 点到抛物面的最短距离公式

[7]:
```
x,y,z=symbols("x,y,z")
simplify(sqrt((t-x0)**2+(a*t**2-b-y0)**2).subs(x0**2,x**2+y**2).
↪subs(x0,-sqrt(x**2+y**2)).subs(y0,z))
```

```
sqrt(((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
1)**3)/a**6) + 9*sqrt(x**2 +
y**2))/a**2)**(2/3)*(36*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2)
- 2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 +
y**2))/a**2)**(2/3)*(6*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2)
- 2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 +
y**2))/a**2)**(1/3)*sqrt(x**2 + y**2) -
6**(1/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b +
2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) +
6**(1/3)*(2*a*b + 2*a*z - 1)))**2 +
(36*a**3*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z
- 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3)*(b + z) -
6**(2/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b +
2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) - 6**(1/3)*(-
2*a*b - 2*a*z + 1))**2)**2)/(a**2*(sqrt(3)*a**2*sqrt((27*a**2*(x**2 +
y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))**2))/36
```

2.3.3 main.py

```python
from numba import njit
from math import pi,sin,cos,radians
import numpy as np
from scipy.optimize import root
import openpyxl,shelve

with shelve.open("this_dataset") as dat:
    data节点=dat["data节点"]
    data_info节点=dat["data_info节点"]
    data面板=dat["data面板"]

@njit#向量的模
def pjfunc(x:np.ndarray):
    return np.sqrt(np.sum(np.square(x)))
R=0
for i in data节点:
    R+=pjfunc(np.array(data节点[i]))
R/=len(data节点)

@njit#单位化
def dwh(x:np.ndarray):
    return x/np.sqrt(np.sum(np.square(x)))
@njit#两点间距离
def distance(x:np.ndarray,y:np.ndarray):
    return np.sqrt(np.sum(np.square(x-y)))
@njit
def _法向量(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    return np.cross(a-b,a-c)
def roll(x:np.ndarray,rol,pit,yaw):
    """x,y,z"""
    Rx=np.array([
        [1,0,0],
        [0,cos(rol),-sin(rol)],
        [0,sin(rol),cos(rol)]
    ])
    Ry=np.array([
        [cos(pit),0,sin(pit)],
        [0,1,0],
        [-sin(pit),0,cos(pit)]
    ])
    Rz=np.array([
        [cos(yaw),-sin(yaw),0],
        [sin(yaw),cos(yaw),0],
        [0,0,1]
    ])
    return Rz@Ry@Rx@x
def func(x:np.ndarray):
    zz=roll(x,0,0,-36.795*pi/180)
    zz=roll(zz,0,-radians(90-78.169),0)
    for k in range(3):
        x[k]=zz[k]
def rfunc(x:np.ndarray):
    zz=roll(x,0,radians(90-78.169),0)
    zz=roll(zz,0,0,36.795*pi/180)
    for k in range(3):
        x[k]=zz[k]
# 变换后的坐标系里面的抛物面方程: z=0.001781612254082338*(x^2+y^2)-300.735945677618
@njit#抛物面
def pwm(x,y):
    return 0.001781612254082338*(x**2+y**2)-300.735945677618
aa=np.array([0,0,-300.735945677618])
@njit#抛物面的法向量
def fpwm(x,y):
    return np.array([-2*0.001781612254082338*x,-2*0.001781612254082338*y,1])
@njit#d是否在abc三角形中
def in_tr(a:np.ndarray,b:np.ndarray,c:np.ndarray,d:np.ndarray):
    fa=lambda x,y:((x-b[0])*(b[1]-c[1])-(y-b[1])*(b[0]-c[0]))
    fb=lambda x,y:((x-a[0])*(a[1]-c[1])-(y-a[1])*(a[0]-c[0]))
    fc=lambda x,y:((x-b[0])*(b[1]-a[1])-(y-b[1])*(b[0]-a[0]))
    return fa(d[0],d[1])*fa(a[0],a[1])>0 and fb(d[0],d[1])*fb(b[0],b[1])>0 and
        fc(d[0],d[1])*fc(c[0],c[1])>0
@njit#_面板.train得到平均法向量
def _train(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    x0=min(a[0],b[0],c[0])
    x1=max(a[0],b[0],c[0])
    y0=min(a[1],b[1],c[1])
    y1=max(a[1],b[1],c[1])
    x=np.linspace(x0,x1)
    y=np.linspace(y0,y1)
    fxl=np.zeros((3,))
    total_n=0
    for i in x:
        for j in y:
            if in_tr(a,b,c,np.array([i,j])):
                fxl+=fpwm(i,j)
                total_n+=1
    return fxl/total_n
@njit#球坐标下点到抛物面的距离
def distance_to_pwm(pos:np.ndarray):
    x,y,z=pos[0],pos[1],pos[2]
    a,b=0.001781612254082338,300.735945677618
    return
```

```
        ((x**2+y**2+z**2)*(2*a*(x**2+y**2)-z-(4*a*b*(x**2+y**2)+z**2)**0.5)**2)**0.5/(2*a*(x**2+y**2))
@njit
def sqrt(x):
    return x**0.5
@njit#点到抛物面的最短距离
def min_distance_topwm(pos:np.ndarray):
    a,b=0.001781612254082338,300.735945677618
    x,y,z=pos[0],pos[1],pos[2]
    return sqrt(((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(36*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) -
        2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 +
        y**2))/a**2)**(2/3)*(6*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b +
        2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(1/3)*sqrt(x**2 + y**2) -
        6**(1/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) + 6**(1/3)*(2*a*b + 2*a*z - 1)))**2 +
        (36*a**3*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(b + z) -
        6**(2/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) - 6**(1/3)*(-2*a*b - 2*a*z +
        1))**2)**2)/(a**2*(sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))**2))/36
@njit#_节点.move_to_pwm
def _dpos_to_pwm(pos:np.ndarray):
    dwpos=pos/np.sum(pos)
    distan=1000000
    keep_i=0
    for i in np.linspace(-0.6,0.6,10000):#pos+dwpos*i
        # b=(pos+dwpos*i)
        # a=abs(b[2]-pwm(b[0],b[1]))
        a=distance_to_pwm(pos+dwpos*i)
        if a<distan:
            distan=a
            keep_i=i
    return keep_i*dwpos,distan
@njit#获取abc组成的面板在a那边的1/3散点
def _getallpos(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    AB=b-a
    AC=c-a
    AB/=8
    AC/=8
    n=0
    reta=np.zeros((17,3),dtype=np.float64)
    for i in range(5):
        for j in range(5-i):
            reta[n]+=AC*i+AB*j
            n+=1
    reta[n]+=AC*3+AB*2
    n+=1
    reta[n]+=AC*2+AB*3
    return reta+a
@njit#将上一个函数的返回值变成在球面上的点
def getallpos2ball(x:np.ndarray,center:np.ndarray):
    for i in range(len(x)):
        x[i]=x[i]-center
        x[i]=x[i]*R/np.sqrt(np.sum(np.square(x[i])))
        x[i]=x[i]+center
    return x
@njit#计算1/3块板上的散点到抛物面的最短距离的平均值
def _mean_distance2pwm(ii:np.ndarray):
    num=0
    total_distance=0
    for i in range(17):
        a=min_distance_topwm(ii[i])
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14:
            f=1/2
        else:
            f=1.
        if ii[i][2]>pwm(ii[i][0],ii[i][1]):
            t=-1
        else:
            t=1
        num+=f
        total_distance+=f*a*t
    return total_distance,num

class _节点:
    def __init__(self,name,pos,info_pos):
        self.name=name
        self.pos=np.array(pos)
        self.xpos,self.spos=np.array(info_pos[0]),np.array(info_pos[1])
        self.board=[]
        self.boarddots=dict()
    def move(self,x:float):# 移动节点（向上为正
        fxxl=self.spos-self.xpos
        fxxl/=pjfunc(fxxl)
        if fxxl[2]<0:
            fxxl*=-1
        dpos=fxxl*x
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
    def move_to_pwm(self):# 将自己贴到抛物面上（在伸缩范围内）
```

```
        dpos,distan=_dpos_to_pwm(self.pos)
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
        return distan
    def mean_distance2pwm(self):# 六边形的散点到抛物面的平均距离(自己在抛物面的下边为正)
        mnum=0
        mtotal_distance=0
        for i in self.boarddots:
            au,bu=_mean_distance2pwm(self.boarddots[i])
            mnum+=bu
            mtotal_distance+=au
        return mtotal_distance/mnum
    def __str__(self):
        return "<节点"+self.name+str(self.pos)+'>'
    def __repr__(self):
        return "<节点"+self.name+str(self.pos)+"下"+str(self.xpos)+"上"+str(self.spos)+'>'

d节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}# 所有的节点
# dr节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}
class _面板:
    def __init__(self,d1:_节点,d2:_节点,d3:_节点,ind:int):
        self.ds=[d1,d2,d3]
        self.bc=np.array([distance(d1.pos,d2.pos),distance(d2.pos,d3.pos),distance(d3.pos,d1.pos)])#
            初始三边长度
        self.ind=ind#索引
        d1.board.append(ind)
        d2.board.append(ind)
        d3.board.append(ind)
        self.expected_fxl=None
    def train(self):
        self.expected_fxl=_train(*(self.ds[i].pos for i in range(3)))
    def getallpos(self):# 将散点给到自己的三个顶点
        centerofball=root(lambda
            x:np.array([distance(self.ds[0].pos,x)-R,distance(self.ds[1].pos,x)-R,distance(self.ds[2].pos,x)-R])
        self.ds[0].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos),
        self.ds[1].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos),
        self.ds[2].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos),
    def 平着的getallpos(self):# 没用
        self.ds[0].boarddots[self.ind]=(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos))
        self.ds[1].boarddots[self.ind]=(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos))
        self.ds[2].boarddots[self.ind]=(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos))
    def refarea(self):
        fxl=_法向量(*(self.ds[i].pos for i in range(3)))
        return fxl
    def __str__(self):
        return "<面板:{}:{}:{}>".format(*(i.name for i in self.ds))
    def __repr__(self):
        return "<面板:{}:{}:{}>".format(*(i for i in self.ds))
d面板=[_面板(*(d节点[j] for j in i),ii) for ii,i in enumerate(data面板)]# 所有的面板

for i in d节点:# 坐标变换
    func(d节点[i].pos)
    func(d节点[i].spos)
    func(d节点[i].xpos)

in_r150节点=set()# 在新坐标系下位于口径300内的节点
@njit#平面上的点到原点的距离
def _f(a,b):
    return (a**2+b**2)**0.5
for i in d节点:
    if _f(d节点[i].pos[0],d节点[i].pos[1])<150:
        in_r150节点.add(i)#统计半径150m以内的主索节点

@njit# 计算带有权重的17个点的平方和
def pingfang17(x:np.ndarray):
    ret=0.
    for i in range(17):
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14 or i==15 or i==16:
            f=1/2
        else:
            f=1.
        ret+=min_distance_topwm(x[i])**2*f
    return ret

for i in d面板:
    i.getallpos()# 更新散点
pre=-100
while True:# 调整位置
    fang=0
    total=0# * 12+1/6
    for i in in_r150节点:
        d=d节点[i].mean_distance2pwm()
        d节点[i].move(d)
        for ooioo in d节点[i].board:
            d面板[ooioo].getallpos()
        fang+=d**2
        total+=1
    h=sqrt(fang/total)#*(12+1/6)
    print("均方根误差:",h)
    if abs(pre-h)<0.0001:
        break
    pre=h


# 求抛物线顶点坐标(反变换回原坐标系)
a=np.array([0,0,-300.735945677618])
rfunc(a)
# 反变换回原坐标系
for i in d节点:# 坐标反变换
    rfunc(d节点[i].pos)
    rfunc(d节点[i].spos)
    rfunc(d节点[i].xpos)
# 写入表格
new_excel = openpyxl.load_workbook("附件4.xlsx")
ws0 = new_excel.worksheets[0]
for i in range(3):
    ws0.cell(2,i+1,a[i])
ws = new_excel.worksheets[1]
cr=2
in_r150节点=list(in_r150节点)
in_r150节点.sort()
for i in in_r150节点:
    ws.cell(cr,1,i)
    for j in range(3):
        ws.cell(cr,j+2,d节点[i].pos[j])
    cr+=1
ws3=new_excel.worksheets[2]
cr=2
for i in in_r150节点:
    ws3.cell(cr,1,i)
    if d节点[i].pos[2]>data节点[i][2]:
        ws3.cell(cr,2,distance(d节点[i].pos,np.array(data节点[i])))
    else:
        ws3.cell(cr,2,-distance(d节点[i].pos,np.array(data节点[i])))
    cr+=1
new_excel.save("result.xlsx")
print("result.xlsx 写入完成")
```

2.4 问题三代码

说明：运行“main.py”会打印迭代中的均方根误差，最后打印出工作抛物面的接收率。运行“计算球面接收率.py”会打印出球面的接收率。

“this_dataset.bak”、“this_dataset.dat”、“this_dataset.dir”是 shelve 生成的数据存储文件，不需要主动运行。

2.4.1 计算球面接收率.py

```python
import shelve
from numba import njit
from math import pi,sin,cos,radians
import numpy as np
from scipy.optimize import root

with shelve.open("this_dataset") as dat:
    data节点=dat["data节点"]
    data_info节点=dat["data_info节点"]
    data面板=dat["data面板"]

@njit#向量的模
def pjfunc(x:np.ndarray):
    return np.sqrt(np.sum(np.square(x)))
R=0
for i in data节点:
    R+=pjfunc(np.array(data节点[i]))
R/=len(data节点)

@njit#单位化
def dwh(x:np.ndarray):
    return x/np.sqrt(np.sum(np.square(x)))
@njit#两点间距离
def distance(x:np.ndarray,y:np.ndarray):
    return np.sqrt(np.sum(np.square(x-y)))
@njit
def _法向量(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    return np.cross(a-b,a-c)
def roll(x:np.ndarray,rol,pit,yaw):
    """x,y,z"""
    Rx=np.array([
        [1,0,0],
        [0,cos(rol),-sin(rol)],
        [0,sin(rol),cos(rol)]
    ])
    Ry=np.array([
        [cos(pit),0,sin(pit)],
        [0,1,0],
        [-sin(pit),0,cos(pit)]
    ])
    Rz=np.array([
        [cos(yaw),-sin(yaw),0],
        [sin(yaw),cos(yaw),0],
        [0,0,1]
    ])
    return Rz@Ry@Rx@x
def func(x:np.ndarray):
    zz=roll(x,0,0,-36.795*pi/180)
    zz=roll(zz,0,-radians(90-78.169),0)
    for k in range(3):
        x[k]=zz[k]
def rfunc(x:np.ndarray):
    zz=roll(x,0,radians(90-78.169),0)
    zz=roll(zz,0,0,36.795*pi/180)
    for k in range(3):
        x[k]=zz[k]
# 变换后的坐标系里面的抛物面方程: z=0.001781612254082338*(x^2+y^2)-300.735945677618
@njit#抛物面
def pwm(x,y):
    return 0.001781612254082338*(x**2+y**2)-300.735945677618
aa=np.array([0,0,-300.735945677618])
@njit#抛物面的法向量
def fpwm(x,y):
    return np.array([-2*0.001781612254082338*x,-2*0.001781612254082338*y,1])
@njit#d是否在abc三角形中
def in_tr(a:np.ndarray,b:np.ndarray,c:np.ndarray,d:np.ndarray):
    fa=lambda x,y:((x-b[0])*(b[1]-c[1])-(y-b[1])*(b[0]-c[0]))
    fb=lambda x,y:((x-a[0])*(a[1]-c[1])-(y-a[1])*(a[0]-c[0]))
    fc=lambda x,y:((x-b[0])*(b[1]-a[1])-(y-b[1])*(b[0]-a[0]))
    return fa(d[0],d[1])*fa(a[0],a[1])>0 and fb(d[0],d[1])*fb(b[0],b[1])>0 and
        fc(d[0],d[1])*fc(c[0],c[1])>0
@njit#_面板.train得到平均法向量
def _train(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    x0=min(a[0],b[0],c[0])
    x1=max(a[0],b[0],c[0])
    y0=min(a[1],b[1],c[1])
    y1=max(a[1],b[1],c[1])
    x=np.linspace(x0,x1)
    y=np.linspace(y0,y1)
    fxl=np.zeros((3,))
    total_n=0
    for i in x:
        for j in y:
            if in_tr(a,b,c,np.array([i,j])):
                fxl+=fpwm(i,j)
                total_n+=1
    return fxl/total_n
@njit#球坐标下点到抛物面的距离
def distance_to_pwm(pos:np.ndarray):
    x,y,z=pos[0],pos[1],pos[2]
    a,b=0.001781612254082338,300.735945677618
    return
```

```
        ((x**2+y**2+z**2)*(2*a*(x**2+y**2)-z-(4*a*b*(x**2+y**2)+z**2)**0.5)**2)**0.5/(2*a*(x**2+y**2))
@njit
def sqrt(x):
    return x**0.5
@njit#点到抛物面的最短距离
def min_distance_topwm(pos:np.ndarray):
    a,b=0.001781612254082338,300.735945677618
    x,y,z=pos[0],pos[1],pos[2]
    return sqrt(((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(36*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) -
        2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 +
        y**2))/a**2)**(2/3)*(6*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b +
        2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(1/3)*sqrt(x**2 + y**2) -
        6**(1/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) + 6**(1/3)*(2*a*b + 2*a*z - 1)))**2 +
        (36*a**3*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(b + z) -
        6**(2/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) - 6**(1/3)*(-2*a*b - 2*a*z +
        1))**2)**2)/(a**2*(sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))**2))/36
@njit#_节点.move_to_pwm
def _dpos_to_pwm(pos:np.ndarray):
    dwpos=pos/np.sum(pos)
    distan=1000000
    keep_i=0
    for i in np.linspace(-0.6,0.6,10000):#pos+dwpos*i
        # b=(pos+dwpos*i)
        # a=abs(b[2]-pwm(b[0],b[1]))
        a=distance_to_pwm(pos+dwpos*i)
        if a<distan:
            distan=a
            keep_i=i
    return keep_i*dwpos,distan
@njit#获取abc组成的面板在a那边的1/3散点
def _getallpos(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    AB=b-a
    AC=c-a
    AB/=8
    AC/=8
    n=0
    reta=np.zeros((17,3),dtype=np.float64)
    for i in range(5):
        for j in range(5-i):
            reta[n]+=AC*i+AB*j
            n+=1
    reta[n]+=AC*3+AB*2
    n+=1
    reta[n]+=AC*2+AB*3
    return reta+a
@njit#将上一个函数的返回值变成在球面上的点
def getallpos2ball(x:np.ndarray,center:np.ndarray):
    for i in range(len(x)):
        x[i]=x[i]-center
        x[i]=x[i]*R/np.sqrt(np.sum(np.square(x[i])))
        x[i]=x[i]+center
    return x
@njit#计算1/3块板上的散点到抛物面的最短距离的平均值
def _mean_distance2pwm(ii:np.ndarray):
    num=0
    total_distance=0
    for i in range(17):
        a=min_distance_topwm(ii[i])
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14:
            f=1/2
        else:
            f=1.
        if ii[i][2]>pwm(ii[i][0],ii[i][1]):
            t=-1
        else:
            t=1
        num+=f
        total_distance+=f*a*t
    return total_distance,num
@njit
def diancheng(a:np.ndarray,b:np.ndarray):
    ret=0
    for i in range(a.shape[0]):
        ret+=a[i]*b[i]
    return ret
@njit
def _duichen(ruse:np.ndarray,fxl:np.ndarray):
    b=fxl/np.sqrt(np.sum(np.square(fxl)))
    b*=diancheng(b,ruse)
    c=(b-ruse)*2
    return ruse+c
@njit
def _refdot(pos1:np.ndarray,fxxl:np.ndarray,z=-160.2):
    if fxxl[2]<0:
        fxxl*=-1
    f1=(z-pos1[2])/fxxl[2]
    p1=pos1+f1*fxxl
    if p1[0]**2+p1[1]**2<0.25:
        return 1.
    else:
        return 0.
    # return p1
@njit
def _refdots(dj1:np.ndarray,dj2:np.ndarray,dj3:np.ndarray,center:np.ndarray):
    all_lights=0
    geted_lights=0
    ret=np.zeros((17*3,3))
    for i in range(17):
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==5 or i==9 or i==12 or i==15 or i==16:
            f=0.5
        elif i==4 or i==14:
            f=0.25
        else:
            f=1.
        # ret[i]+=(_refdot(dj1[i],_duichen(np.array([0,0,1]),center-dj1[i])))
        # ret[17+i]+=(_refdot(dj2[i],_duichen(np.array([0,0,1]),center-dj2[i])))
        # ret[17*2+i]+=(_refdot(dj3[i],_duichen(np.array([0,0,1]),center-dj3[i])))
        all_lights+=3*f
        geted_lights+=(_refdot(dj1[i],_duichen(np.array([0,0,1]),center-dj1[i])))*f
        geted_lights+=(_refdot(dj2[i],_duichen(np.array([0,0,1]),center-dj2[i])))*f
        geted_lights+=(_refdot(dj3[i],_duichen(np.array([0,0,1]),center-dj3[i])))*f
    return geted_lights,all_lights
    # return ret
@njit# 计算三角形面积
def tr_square(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    return abs(a[0]*(b[1]-c[1])+b[0]*(c[1]-a[1])+c[0]*(a[1]-b[1]))/2

class _节点:
    def __init__(self,name,pos,info_pos):
        self.name=name
        self.pos=np.array(pos)
        self.xpos,self.spos=np.array(info_pos[0]),np.array(info_pos[1])
        self.board=[]
        self.boarddots=dict()
    def move(self,x:float):# 移动节点（向上为正
        fxxl=self.spos-self.xpos
        fxxl/=pjfunc(fxxl)
        if fxxl[2]<0:
            fxxl*=-1
        dpos=fxxl*x
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
    def move_to_pwm(self):# 将自己贴到抛物面上（在伸缩范围内）
        dpos,distan=_dpos_to_pwm(self.pos)
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
        return distan
    def mean_distance2pwm(self):# 六边形的散点到抛物面的平均距离（自己在抛物面的下边为正）
        mnum=0
        mtotal_distance=0
        for i in self.boarddots:
            au,bu=_mean_distance2pwm(self.boarddots[i])
            mnum+=bu
            mtotal_distance+=au
        return mtotal_distance/mnum
    def __str__(self):
        return "<节点"+self.name+str(self.pos)+'>'
    def __repr__(self):
        return "<节点"+self.name+str(self.pos)+"下"+str(self.xpos)+"上"+str(self.spos)+'>'

d节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}# 所有的节点
# dr节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}
class _面板:
    def __init__(self,d1:_节点,d2:_节点,d3:_节点,ind:int):
        self.ds=[d1,d2,d3]
        self.bc=np.array([distance(d1.pos,d2.pos),distance(d2.pos,d3.pos),distance(d3.pos,d1.pos)])#
            初始三边长度
        self.ind=ind#在d面板中的索引
        d1.board.append(ind)
        d2.board.append(ind)
        d3.board.append(ind)
        self.expected_fxl=None
    def train(self):
        self.expected_fxl=_train(*(self.ds[i].pos for i in range(3)))
    def getallpos(self):# 将散点给到自己的三个顶点
        centerofball=root(lambda
            x:np.array([distance(self.ds[0].pos,x)-R,distance(self.ds[1].pos,x)-R,distance(self.ds[2].pos,x)-R])
        self.ds[0].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos),
        self.ds[1].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos),
        self.ds[2].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos),
    def refarea(self):
        centerofball=root(lambda
            x:np.array([distance(self.ds[0].pos,x)-R,distance(self.ds[1].pos,x)-R,distance(self.ds[2].pos,x)-R])
        percent = _refdots(
            getallpos2ball(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos),centerofball),
            getallpos2ball(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos),centerofball),
            getallpos2ball(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos),centerofball),
            centerofball
        )
        return percent[0]/percent[1],tr_square(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos)

    def __str__(self):
        return "<面板:{}:{}:{}>".format(*(i.name for i in self.ds))
    def __repr__(self):
        return "<面板:{}:{}:{}>".format(*(i for i in self.ds))
d面板=[_面板(*(d节点[j] for j in i),ii) for ii,i in enumerate(data面板)]# 所有的面板

for i in d节点:# 坐标变换
    func(d节点[i].pos)
    func(d节点[i].spos)
    func(d节点[i].xpos)

in_r150节点=set()# 在新坐标系下位于口径300内的节点
@njit#平面上的点到原点的距离
def _f(a,b):
    return (a**2+b**2)**0.5
for i in d节点:
    if _f(d节点[i].pos[0],d节点[i].pos[1])<150:
        in_r150节点.add(i)# 统计150以内的点

@njit# 计算带有权重的17个点的平方和
def pingfang17(x:np.ndarray):
    ret=0.
    for i in range(17):
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14 or i==15 or i==16:
            f=1/2
        else:
            f=1.
        ret+=min_distance_topwm(x[i])**2*f
    return ret

for i in d面板:
    i.getallpos()# 更新散点

totao_perc,all_area=0,0
for i in d面板:
    if i.ds[0].name in in_r150节点 and i.ds[1].name in in_r150节点 and i.ds[2].name in
         in_r150节点:
        percen,area=i.refarea()
        totao_perc+=percen*area
        all_area+=area
print("球面接收率",totao_perc/all_area)
```

2.4.2 main.py

```python
import shelve
from numba import njit
from math import pi,sin,cos,radians
import numpy as np
from scipy.optimize import root

with shelve.open("this_dataset") as dat:
    data节点=dat["data节点"]
    data_info节点=dat["data_info节点"]
    data面板=dat["data面板"]

@njit#向量的模
def pjfunc(x:np.ndarray):
    return np.sqrt(np.sum(np.square(x)))
R=0
for i in data节点:
    R+=pjfunc(np.array(data节点[i]))
R/=len(data节点)

@njit#单位化
def dwh(x:np.ndarray):
    return x/np.sqrt(np.sum(np.square(x)))
@njit#两点间距离
def distance(x:np.ndarray,y:np.ndarray):
    return np.sqrt(np.sum(np.square(x-y)))
@njit
def _法向量(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    return np.cross(a-b,a-c)
def roll(x:np.ndarray,rol,pit,yaw):
    """x,y,z"""
    Rx=np.array([
        [1,0,0],
        [0,cos(rol),-sin(rol)],
        [0,sin(rol),cos(rol)]
    ])
    Ry=np.array([
        [cos(pit),0,sin(pit)],
        [0,1,0],
        [-sin(pit),0,cos(pit)]
    ])
    Rz=np.array([
        [cos(yaw),-sin(yaw),0],
        [sin(yaw),cos(yaw),0],
        [0,0,1]
    ])
    return Rz@Ry@Rx@x
def func(x:np.ndarray):
    zz=roll(x,0,0,-36.795*pi/180)
    zz=roll(zz,0,-radians(90-78.169),0)
    for k in range(3):
        x[k]=zz[k]
def rfunc(x:np.ndarray):
    zz=roll(x,0,radians(90-78.169),0)
    zz=roll(zz,0,0,36.795*pi/180)
    for k in range(3):
        x[k]=zz[k]
# 变换后的坐标系里面的抛物面方程：z=0.001781612254082338*(x^2+y^2)-300.735945677618
@njit#抛物面
def pwm(x,y):
    return 0.001781612254082338*(x**2+y**2)-300.735945677618
aa=np.array([0,0,-300.735945677618])
@njit#抛物面的法向量
def fpwm(x,y):
    return np.array([-2*0.001781612254082338*x,-2*0.001781612254082338*y,1])
@njit#d是否在abc三角形中
def in_tr(a:np.ndarray,b:np.ndarray,c:np.ndarray,d:np.ndarray):
    fa=lambda x,y:((x-b[0])*(b[1]-c[1])-(y-b[1])*(b[0]-c[0]))
    fb=lambda x,y:((x-a[0])*(a[1]-c[1])-(y-a[1])*(a[0]-c[0]))
    fc=lambda x,y:((x-b[0])*(b[1]-a[1])-(y-b[1])*(b[0]-a[0]))
    return fa(d[0],d[1])*fa(a[0],a[1])>0 and fb(d[0],d[1])*fb(b[0],b[1])>0 and
        fc(d[0],d[1])*fc(c[0],c[1])>0
@njit#_面板.train得到平均法向量
def _train(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    x0=min(a[0],b[0],c[0])
    x1=max(a[0],b[0],c[0])
    y0=min(a[1],b[1],c[1])
    y1=max(a[1],b[1],c[1])
    x=np.linspace(x0,x1)
    y=np.linspace(y0,y1)
    fxl=np.zeros((3,))
    total_n=0
    for i in x:
        for j in y:
            if in_tr(a,b,c,np.array([i,j])):
                fxl+=fpwm(i,j)
                total_n+=1
    return fxl/total_n
@njit#球坐标下点到抛物面的距离
def distance_to_pwm(pos:np.ndarray):
    x,y,z=pos[0],pos[1],pos[2]
    a,b=0.001781612254082338,300.735945677618
    return
        ((x**2+y**2+z**2)*(2*a*(x**2+y**2)-z-(4*a*b*(x**2+y**2)+z**2)**0.5)**2)**0.5/(2*a*(x**2+y**2))
@njit
def sqrt(x):
    return x**0.5
@njit#点到抛物面的最短距离
def min_distance_topwm(pos:np.ndarray):
    a,b=0.001781612254082338,300.735945677618
    x,y,z=pos[0],pos[1],pos[2]
    return sqrt(((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(36*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) -
        2*(2*a*b + 2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 +
        y**2))/a**2)**(2/3)*(6*a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b +
        2*a*z - 1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(1/3)*sqrt(x**2 + y**2) -
        6**(1/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) + 6**(1/3)*(2*a*b + 2*a*z - 1)))**2 +
        (36*a**3*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z - 1)**3)/a**6) +
        9*sqrt(x**2 + y**2))/a**2)**(2/3)*(b + z) -
        6**(2/3)*(a**2*((sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))/a**2)**(2/3) - 6**(1/3)*(-2*a*b - 2*a*z +
        1))**2)**2)/(a**2*(sqrt(3)*a**2*sqrt((27*a**2*(x**2 + y**2) - 2*(2*a*b + 2*a*z -
        1)**3)/a**6) + 9*sqrt(x**2 + y**2))**2))/36
@njit#_节点.move_to_pwm
def _dpos_to_pwm(pos:np.ndarray):
    dwpos=pos/np.sum(pos)
    distan=1000000
    keep_i=0
    for i in np.linspace(-0.6,0.6,10000):#pos+dwpos*i
        # b=(pos+dwpos*i)
        # a=abs(b[2]-pwm(b[0],b[1]))
        a=distance_to_pwm(pos+dwpos*i)
        if a<distan:
            distan=a
            keep_i=i
    return keep_i*dwpos,distan
@njit#获取abc组成的面板在a那边的1/3散点
def _getallpos(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    AB=b-a
    AC=c-a
    AB/=8
    AC/=8
    n=0
    reta=np.zeros((17,3),dtype=np.float64)
    for i in range(5):
        for j in range(5-i):
            reta[n]+=AC*i+AB*j
            n+=1
    reta[n]+=AC*3+AB*2
    n+=1
    reta[n]+=AC*2+AB*3
    return reta+a
@njit#将上一个函数的返回值变成在球面上的点
def getallpos2ball(x:np.ndarray,center:np.ndarray):
    for i in range(len(x)):
        x[i]=x[i]-center
        x[i]=x[i]*R/np.sqrt(np.sum(np.square(x[i])))
        x[i]=x[i]+center
    return x
@njit#计算1/3块板上的散点到抛物面的最短距离的平均值
def _mean_distance2pwm(ii:np.ndarray):
    num=0
    total_distance=0
    for i in range(17):
        a=min_distance_topwm(ii[i])
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14:
            f=1/2
        else:
            f=1.
        if ii[i][2]>pwm(ii[i][0],ii[i][1]):
            t=-1
        else:
            t=1
        num+=f
        total_distance+=f*a*t
    return total_distance,num
@njit
def diancheng(a:np.ndarray,b:np.ndarray):
    ret=0
    for i in range(a.shape[0]):
        ret+=a[i]*b[i]
    return ret
@njit
def _duichen(ruse:np.ndarray,fxl:np.ndarray):
    b=fxl/np.sqrt(np.sum(np.square(fxl)))
    b*=diancheng(b,ruse)
    c=(b-ruse)*2
    return ruse+c
@njit
def _refdot(pos1:np.ndarray,fxxl:np.ndarray,z=-160.2):
    if fxxl[2]<0:
        fxxl*=-1
    f1=(z-pos1[2])/fxxl[2]
    p1=pos1+f1*fxxl
    if p1[0]**2+p1[1]**2<0.25:
        return 1.
    else:
        return 0.
    # return p1
@njit
def _refdots(dj1:np.ndarray,dj2:np.ndarray,dj3:np.ndarray,center:np.ndarray):
    all_lights=0
    geted_lights=0
    ret=np.zeros((17*3,3))
    for i in range(17):
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==5 or i==9 or i==12 or i==15 or i==16:
            f=0.5
        elif i==4 or i==14:
            f=0.25
        else:
            f=1.
        # ret[i]+=(_refdot(dj1[i],_duichen(np.array([0,0,1]),center-dj1[i])))
        # ret[17+i]+=(_refdot(dj2[i],_duichen(np.array([0,0,1]),center-dj2[i])))
        # ret[17*2+i]+=(_refdot(dj3[i],_duichen(np.array([0,0,1]),center-dj3[i])))
        all_lights+=3*f
        geted_lights+=(_refdot(dj1[i],_duichen(np.array([0,0,1]),center-dj1[i])))*f
        geted_lights+=(_refdot(dj2[i],_duichen(np.array([0,0,1]),center-dj2[i])))*f
        geted_lights+=(_refdot(dj3[i],_duichen(np.array([0,0,1]),center-dj3[i])))*f
    return geted_lights,all_lights
    # return ret
@njit# 计算三角形面积
def tr_square(a:np.ndarray,b:np.ndarray,c:np.ndarray):
    return abs(a[0]*(b[1]-c[1])+b[0]*(c[1]-a[1])+c[0]*(a[1]-b[1]))/2

class _节点:
    def __init__(self,name,pos,info_pos):
        self.name=name
        self.pos=np.array(pos)
        self.xpos,self.spos=np.array(info_pos[0]),np.array(info_pos[1])
        self.board=[]
        self.boarddots=dict()
    def move(self,x:float):# 移动节点（向上为正
        fxxl=self.spos-self.xpos
        fxxl/=pjfunc(fxxl)
        if fxxl[2]<0:
            fxxl*=-1
        dpos=fxxl*x
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
    def move_to_pwm(self):# 将自己贴到抛物面上（在伸缩范围内）
        dpos,distan=_dpos_to_pwm(self.pos)
        self.pos+=dpos
        self.spos+=dpos
        self.xpos+=dpos
        return distan
    def mean_distance2pwm(self):# 六边形的散点到抛物面的平均距离（自己在抛物面的下边为正）
        mnum=0
        mtotal_distance=0
        for i in self.boarddots:
            au,bu=_mean_distance2pwm(self.boarddots[i])
            mnum+=bu
            mtotal_distance+=au
        return mtotal_distance/mnum
    def __str__(self):
        return "<节点"+self.name+str(self.pos)+'>'
    def __repr__(self):
        return "<节点"+self.name+str(self.pos)+"下"+str(self.xpos)+"上"+str(self.spos)+'>'

d节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}# 所有的节点
# dr节点={i:_节点(i,data节点[i],data_info节点[i]) for i in data节点}
class _面板:
    def __init__(self,d1:_节点,d2:_节点,d3:_节点,ind:int):
        self.ds=[d1,d2,d3]
        self.bc=np.array([distance(d1.pos,d2.pos),distance(d2.pos,d3.pos),distance(d3.pos,d1.pos)])#
            初始三边长度
        self.ind=ind#索引
        d1.board.append(ind)
        d2.board.append(ind)
        d3.board.append(ind)
        self.expected_fxl=None
    def train(self):
        self.expected_fxl=_train(*(self.ds[i].pos for i in range(3)))
    def getallpos(self):# 将散点给到自己的三个顶点
        centerofball=root(lambda
            x:np.array([distance(self.ds[0].pos,x)-R,distance(self.ds[1].pos,x)-R,distance(self.ds[2].pos,x)-R])
        self.ds[0].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos),
        self.ds[1].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos),
        self.ds[2].boarddots[self.ind]=(getallpos2ball(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos),
    def refarea(self):
        centerofball=root(lambda
            x:np.array([distance(self.ds[0].pos,x)-R,distance(self.ds[1].pos,x)-R,distance(self.ds[2].pos,x)-R])
        percent = _refdots(
            getallpos2ball(_getallpos(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos),centerofball),
            getallpos2ball(_getallpos(self.ds[1].pos,self.ds[2].pos,self.ds[0].pos),centerofball),
            getallpos2ball(_getallpos(self.ds[2].pos,self.ds[0].pos,self.ds[1].pos),centerofball),
            centerofball
        )
        return percent[0]/percent[1],tr_square(self.ds[0].pos,self.ds[1].pos,self.ds[2].pos)

    def __str__(self):
        return "<面板:{}:{}:{}>".format(*(i.name for i in self.ds))
    def __repr__(self):
        return "<面板:{}:{}:{}>".format(*(i for i in self.ds))
d面板=[_面板(*(d节点[j] for j in i),ii) for ii,i in enumerate(data面板)]# 所有的面板

for i in d节点:# 坐标变换
    func(d节点[i].pos)
    func(d节点[i].spos)
    func(d节点[i].xpos)

in_r150节点=set()# 在新坐标系下位于口径300内的节点
@njit#平面上的点到原点的距离
def _f(a,b):
    return (a**2+b**2)**0.5
for i in d节点:
    if _f(d节点[i].pos[0],d节点[i].pos[1])<150:
        in_r150节点.add(i)# 统计150以内的点

@njit# 计算带有权重的17个点的平方和
def pingfang17(x:np.ndarray):
    ret=0.
    for i in range(17):
        if i==0:
            f=1/6
        elif i==1 or i==2 or i==3 or i==4 or i==5 or i==9 or i==12 or i==14 or i==15 or i==16:
            f=1/2
        else:
            f=1.
        ret+=min_distance_topwm(x[i])**2*f
    return ret

for i in d面板:
    i.getallpos()# 更新散点
pre=-100
while True:# 调整位置
    fang=0
    total=0# * 12+1/6
    for i in in_r150节点:
        d=d节点[i].mean_distance2pwm()
        d节点[i].move(d)
        for ooioo in d节点[i].board:
            d面板[ooioo].getallpos()
        fang+=d**2
        total+=1
    h=sqrt(fang/total)#*(12+1/6)
    print("均方根误差:",h)
    if abs(pre-h)<0.0001:
        break
    pre=h

totao_perc,all_area=0,0
for i in d面板:
    if i.ds[0].name in in_r150节点 and i.ds[1].name in in_r150节点 and i.ds[2].name in
         in_r150节点:
        percen,area=i.refarea()
        totao_perc+=percen*area
        all_area+=area
print("工作抛物面接收率:",totao_perc/all_area)
```

附录 C　支撑材料文件说明

README.txt 文件：对相应路径下的代码与文件说明。

result.xlsx 文件：问题二结果。

.py 文件：Python 运行代码。

附件 1.csv，附件 2.csv，附件 3.csv 文件：题中附录所给坐标等数据，用于被.py 文件读取。

.dir.dat.bak 文件：为 shelve 生成的数据存储文件，不需要主动运行。

.ipynb 文件：

“抛物线.ipynb”是计算理想抛物面的程序过程。

“点到抛物线的最短距离的公式.ipynb”是用 sympy 进行符号计算得到点到抛物面的最短距离的公式的过程。

“点到抛物线的球心径向距离.ipynb”是用 sympy 进行符号计算得到点到抛物面沿球心径向方向的的距离的公式的过程。