建模研究 FAST 望远镜主动反射面的变形问题

摘要

本文建立了 FAST 主索节点调整的数学模型，通过最优的方式调整主索节点位置，以求获得最大的接收比。

问题一中，我们确定了光束垂直入射情况下 FAST 调整的理想抛物面。我们认为**使主索节点调整幅度最小**对应的抛物面是最理想的。提出了一些基于直觉的假设，并以此为基础，引入**夹逼圆弧**的概念，利用**解析几何**方法解得理想抛物线的方程为：

$$z = \frac{x^2 + y^2}{561.28932} - 300.73593$$

问题二中，我们考虑了光束以一定角度入射的情况。我们通过**坐标变换**的方法，确定了新的理想抛物面的顶点坐标为：**（$-61.658909, 0, -294.3473456$）**。接着我们把各个主索节点沿径向移动至抛物面上，并筛选得到调节后在$300m$内的主索节点共**689**个，再通过坐标逆变换得到相应的各个点坐标和促动器的伸缩量变化，详见附录。

问题三中，我们计算了问题二情况下 FAST 的接收比并与调节前基准球面的接收比作比较。我们考虑了两种情况：第一种认为反射板是平面板；第二种认为反射板是球面板。针对第一种情况，我们计算得到的接收比为1.176%，相比于未调节时的1.168%变化不大。由于接收比较低，我们认为第二种情况更为合理。针对第二种情况，我们先计算了各个反射板移动后的球心坐标，进而得到口径范围内任意位置入射的光反射的法向量，接着计算反射光的方程，最后得到此光束经 FAST 反射后在焦平面上的位置。我们顺着平面板思路采用每块三角形板均匀的方法求出接收比为67.42%；用**蒙特卡洛模拟**的方法随机在口径范围内生成10000个点，通过接收成功率估算接收比。最终我们的结果是**63.15%**。前者需要的总体数据量稍大，后者更可信。通过解析几何求解基准球面的接受率为**5.25%**，用蒙特卡罗方法测试该值为5.41%。说明通过调节主索节点位置我们可以有效的提高 FAST 的性能。

最后，我们参考了其他文献和实际情况对我们的模型作出合理的评价。我们认为我们的模型有一定的参考价值，比较能反映出 FAST 调节的真实情况。

关键词：FAST 望远镜；曲面拟合；坐标变换；蒙特卡洛方法

一、问题背景与重述

1.1 问题背景

500米口径球面射电望远镜（简称 FAST），是我国具有自主知识产权的，目前世界上单口径最大、灵敏度最高的射电望远镜。其中的主动反射面既是它的核心结构，又是其主要创新点之一，同时还是工程设计的难点。

FAST 的主动反射面由在主索网上的若干三角形反射面板构成。在未工作状态（基准态）下，这些反射面板均位于基准球面上。当需要观测天体时，主动反射面便进入工作态，促动器会调节主索节点的位置，改变主索网的形状，进而带动反射面板，使它们组成工作抛物面。工作抛物面的形状与光滑的、以馈源舱所在位置为焦点的理想抛物面相似，所以反射面板会将观测天体发出的电磁波反射到馈源舱处，进而实现对天体的观察。

1.2 问题重述

问题一：当待观测天体位于$\alpha = 0°, \beta = 90°$的方向时，考虑反射面板的约束，确定与基准面变化不大、口径为$300m$且焦点位置与馈源舱位置重合的理想抛物面。

问题二：当待观测天体位于$\alpha = 36.795°, \beta = 78.169°$的位置时，先确定此时理想抛物面的方程，并给出该抛物面的顶点坐标，再建立调整反射面板位置与角度的模型。然后利用题目所给附件，给出经过调节后，$300m$口径内各主索节点的编号与位置坐标，以及相关的促动器的伸缩量。最后，将结果按照附件4的要求进行填写，并将结果文件保存。

问题三：在问题二的前提下，计算工作抛物面的信号接收比，并与基准球面的信号接收比进行比较。

二、名词解释与书写规则

2.1 默认坐标系

题目中所给出的、坐标原点为基准球面球心C的空间直角坐标系。如无特别说明，所有点的坐标、平面或曲面的方程均在该坐标系中书写。

2.2 理想抛物面

以馈源舱位置为焦点、光滑的、存在数学解析式的、最符合工程约束与题目条件的抛物面。它是有限大的。其口径为$300m$。

2.3 基准球面与工作抛物面

初始时主索节点位于球面，此时各反射面板形成了基准球面。通过调节主索节点拟合理想抛物面，各反射面板实际形成了工作抛物面。

2.4 夹逼圆弧

以坐标原点为圆心，将研究的抛物线夹在中间的两条圆弧。它们的半径分别为R_1、R_2。

2.5 坐标的书写

如无特别说明，各点的坐标均为在默认坐标系中的坐标。

所有坐标在书写时均省略长度单位“米（m）”。

部分坐标中含有符号。

2.6 S点、C点、P点与K点

S点：待观测天体所在的几何点。

C点：基准球面球心所在的几何点。

P点：馈源舱所在的几何点。

K点：理想抛物面的顶点。

2.7 焦平面

第一题条件下，以抛物面焦点（馈源舱最低点所在位置）的纵坐标，平行于xOy的平面。因为之后两题将进行坐标变换，半径$0.5m$的接收圆面一直在这个平面内。

三、问题分析

3.1 问题一的分析

根据抛物面性质，来自遥远星空平行入射的光会汇聚到焦点。因此，理想抛物面首先得以馈源舱为焦点。由解析几何的知识可知，在焦点位置（即馈源舱位置）给定的情况下，我们只能得到一个抛物面簇，而无法给出一个确定的抛物面。因此在问题一中，我们需要在该抛物面簇中找到最合理的、最符合工程约束条件的抛物面——我们称该抛物面为理想抛物面。

确定理想抛物面的工程约束条件主要如下：

1）调节后的抛物面口径达到$300m$；

2）任意促动器的顶端的径向位移均在$-0.6m$至$0.6m$之间；

3）调节后，主索节点之间的间距改变量不能超过0.07%，且越小越好。

经过估算，条件 3 比条件 1、2 更为苛刻。因此我们认为理想抛物面应当使节点的最大调节幅度度尽可能小，从而尽可能地满足条件 3。经过进一步的定量分析，理想抛物面与基准球面的位置关系应大致如图3.1.1所示：

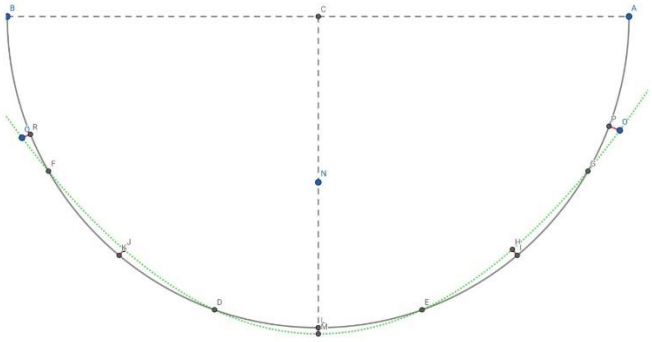

图 3.1.1

3.2 问题二的分析

1、坐标变换：问题二的光源角度相较于问题一发生了变化，我们采用旋转坐标系的方法。通过相对地旋转坐标系（如下图变为 $x^*y^*z^*$轴），光源依然沿z轴入射，这样能为主索节点位置的确定和之后的反射率计算带来简化。

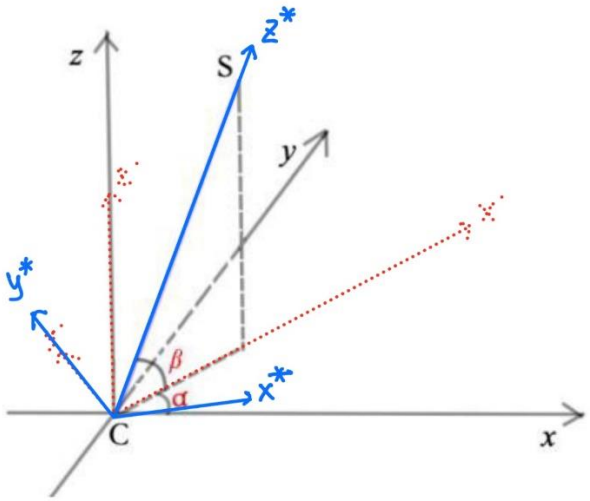

图 3.2.1

2、检验$300m$口径：问题二的理想抛物面除了需要考虑问题一所列的因素之外，因为光线是斜入射，我们需要考虑要求调节的$300m$口径的抛物面是否会超出原本望远镜的边缘。经检验，这个约束比较宽松。

3、拟合计算：我们需要移动我们的三角形反射板使它尽量贴合理想抛物面。我们首先把三角形反射板的顶点（也就是主索节点）移动到理想抛物面上，找到$300m$口径内的主索节点编号，得到新坐标系下主索节点坐标后，通过几何我们可以得出各促动器的伸缩量。不过，仅是主索节点贴在理想抛物面并不是最为贴合的情况。我们需要在此基础上把三角形板的位置进行适当的外移，以获得更好的拟合效果。

4、坐标逆变换：通过逆变换求得$300m$口径内主索节点在真实坐标系下的位置坐标。

3.3　问题三的分析

通过离散化的逐一考虑每一块反光板，计算出每一块反光板反射在焦平面的光束图像。根据题中所给信息，馈源舱只接收直径$1m$范围内的光束。根据馈源舱接收到的面积与反射光线达到接收平面的总面积之比，我们可以算出每一块反光板的接收比。我们假设光强是均匀入射的，因此我们给每一个反光板根据它相对光束的倾角和面积分配以不同的权重，把这些反光板接收比加权平均后我们可以得出总的接收比。最后我们把调节后的状态的接收比和基准态相比较就可以解决问题。

四、符号说明

符号	符号意义	数值与单位
R	基准球面半径（常量）	$300.4m$
D	基准球面口径（常量）	$500m$
α,β	描述观测天体的方位角	°
L	P点与C点之间的距离（常量）	$160.4136m$
D_1	理想抛物面口径（常量）	$300m$
L_1	理想抛物面的准平面与C点之间的距离（常量）	$441.05826m$
H	C点到理想抛物面边缘所处平面的距离（常量）	$260.64965m$
Θ	基准球面对C点的张角（常量）	$112.886°$
Θ_1	理想抛物面对C点的张角（常量）	$59.840°$
η	曲面的接收比	（无单位）

五、模型假设

在我们建立的模型中，用到了以下假设：

1）主索节点、下拉索和促动器都处在基准球面的径向上。

2）当促动器调节距离最小时，索网中的应力变化最小。

3）电磁波信号与反射信号沿直线传播，且不考虑干涉和衍射。

4）电磁波信号均匀入射，且可以被抽象为光线。

六、模型的建立与求解

6.1 问题一的求解

该问题的前提条件是待观测天体S位于$\alpha = 0°, \beta = 90°$的位置，即S位于z轴上。

首先，根据题目条件$F = 0.466R$，再根据附录一中主索节点的坐标得到$R = 300.4m$，可以得知馈源舱P与坐标原点C之间的距离为$L = F - R = 0.534R = 160.4136m$，所以得到此时馈源舱的坐标为$(0,0,-L)$。

由于理想抛物面与基准球面均关于z轴旋转对称，所以在确定理想抛物面的过程中，可以先将问题简化为xCz平面上的二维问题，用圆弧代替球面，用抛物线代替抛物面，然后按照一定的法则确定出理想的抛物线方程，进而确定在三维空间中的理想抛物面

方程。

接下来考虑简化后的二维问题。在xCz平面中，以x轴为横轴，以z轴为纵轴，以C为坐标原点，建立平面直角坐标系。于是在该坐标系中，P点的坐标为$(0,-L)$。令$z_0=-L$。

因为z轴是理想抛物线的对称轴，且沿z轴反方向入射的平行光经理想抛物线反射后应会聚于P点，所以理想抛物线必定是以P为焦点，以直线$z=z_1$（z_1为待定参数，$z_1<z_0$）为准线的抛物线的一部分。它关于z轴对称，且两个端点的间距为等于口径$D_1=300m$。只要确定参数z_1，便能确定理想抛物线。而要确定z_1，就必须给出一个约束条件——这个条件和理想抛物面的某条应具有的性质有关。

下面是对这条性质的讨论。

考虑到在反射面从基准球面变到工作抛物面的过程中，主索、下拉索和反射面板必定会有变形，进而它们内部的应变与应力状态会发生改变。又因为在像 FAST 望远镜这样的相当精密的大型设施中，应变与应力的变化应该尽可能地小，所以各个主索节点的位移量越小越好。于是，理想抛物面应当尽可能贴近基准球面——在二维问题中，则是理想抛物线应当尽可能贴近基准圆弧。

然后将这条性质反映到几何上。

设关于z轴对称的、以P为焦点的、两端间距为$300m$的部分抛物线位于两条夹逼圆弧C_1,C_2之间——这两条圆弧的圆心都是坐标原点C，且圆弧C_1的半径为R_1，圆弧C_2的半径为R_2（$R_1>R_2$）。图6.1.1是示意图。由于坐标原点C位于抛物线的开口内，所以抛物线的顶点应该在C_1上。当这条抛物线为理想抛物线时，应当满足条件：$M=\max\{|R_1-R|,|R_2-R|\}$最小。

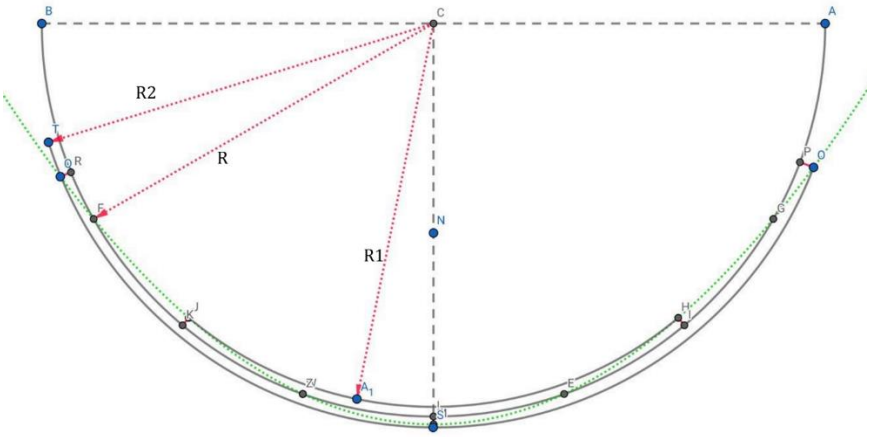

图 6.1.1

不难知道，当z_1变小时，(R_1-R)和(R_2-R)的值都会变大。所以当M最小时，必定有

$$\begin{cases} |R_1-R|=|R_2-R| \\ (R_1-R)(R_2-R)<0 \end{cases} \tag{6.1.1}$$

在得到上面的条件后，便可以计算参数z_1的值了。

首先利用参数z_1写出抛物线的方程：

$$z=\frac{x^2}{2(z_0-z_1)}+\frac{1}{2}(z_0+z_1) \tag{6.1.2}$$

设$R_1=R+\Delta R$，$R_2=R-\Delta R$。那么有

$$-R_1=\frac{1}{2}(z_0+z_1)$$

$$z_1=-2(R+\Delta R)-z_0 \tag{6.1.3}$$

将式(6.1.3)代入方程(6.1.2)，可以得到以ΔR为参数的抛物线方程：

$$z=\frac{x^2}{4(z_0+R+\Delta R)}-(R+\Delta R) \tag{6.1.4}$$

由于圆弧C_2应该与抛物线内切，所以联立抛物线与圆弧C_2的方程后，得到的一元二次方程

$$z^2+2(z_0-z_1)z-R_2^2+z_1^2-z_0^2=0$$

的判别式应该为零。进而可得方程

$$(R-\Delta R)^2=-4z_0(R+\Delta R+z_0)$$

解之得$\Delta R=0.33593m$，$z_1=-441.05826m$。然后根据方程(6.1.4)可得理想抛物线方程：

$$z=\frac{x^2}{561.28932}-300.73593 \tag{6.1.5}$$

将$x=\frac{D_1}{2}$代入上式，得到抛物线端点的纵坐标$z_2=-260.64965m$。又因为

$$\sqrt{\frac{D_1^2}{4}+z_2^2}-R=0.32951m<\Delta R$$

所以抛物线端点在夹逼圆弧之间，我们得到的结果是正确的。

于是我们解决了简化后的二维问题。根据旋转对称性，用(x^2+y^2)替换x^2即可得到理想抛物面的方程：

$$z=\frac{x^2+y^2}{561.28932}-300.73593 \tag{6.1.6}$$

由于$\Delta R<0.6m$，所以该抛物面处在可调节范围内。

为了方便后续问题的描述，我们记理想抛物面的准平面与C点之间的距离为L_1。根据之前的论述不难得知，$L_1 = 441.05826m$。

C点到理想抛物面边缘平面的距离$H = \left|\frac{150^2}{561.28932} - 300.73593\right| m = 260.64965m$。

6.2 问题二的求解

该问题的前提条件是待观测天体S位于$\alpha = 36.795°, \beta = 78.169°$的位置。由于该问题较为复杂，故接下来分模块进行介绍。

1）理想抛物面的方程与顶点坐标

由于对称性，当待观测天体S的方位改变时，理想抛物面的形状不会改变，它只会绕坐标原点C转动，使得其旋转轴位于直线SC上。所以，坐标原点、理想抛物面的焦点与准平面之间的相对位置不会发生改变。

由于理想抛物面的准平面与直线SC垂直，与坐标原点C之间的距离为L_1，且与z轴的负半轴相交；馈源舱P在直线SC上，且$\overrightarrow{CP}$与$\overrightarrow{CS}$方向相反；矢量$\overrightarrow{CS}$对应的单位矢量为$(\cos\alpha\cos\beta, \sin\alpha\cos\beta, \sin\beta)$，所以此时$P$点坐标为

$$(-L\cos\alpha\cos\beta, -L\sin\alpha\cos\beta, -L\sin\beta)$$

理想抛物线的准平面的方程为

$$\cos\alpha\cos\beta\cdot x + \sin\alpha\cos\beta\cdot y + \sin\beta\cdot z + L_1 = 0$$

于是可以得到此时理想抛物面的方程：

$$(\cos\alpha\cos\beta\cdot x + \sin\alpha\cos\beta\cdot y + \sin\beta\cdot z + L_1)^2 = (x + L\cos\alpha\cos\beta)^2 + (y + L\sin\alpha\cos\beta)^2 + (z + L\sin\beta)^2 \quad (6.2.1a)$$

将数值代入得：

$$(0.16418x + 0.12280y + 0.97876z + 441.05826)^2 = (x + 26.33689)^2 + (y + 19.69893)^2 + (z + 157.00587)^2 \quad (6.2.1b)$$

此时理想抛物面的顶点坐标为$(-49.37518, -36.93063, -294.34728)$。

2）理想抛物面与主动反射面（基准球面）之间的位置关系

在研究主索节点的移动与工作抛物面前，我们需要知道式$(6.2.1b)$表示的理想抛物面与主动反射面之间的位置关系，从而得知理想抛物面是否超出了主动反射面的边缘。

由于式(6.2.1b)与式(6.1.6)所表示的抛物面的大小、形状完全一致，所以我们利用式(6.1.6)研究口径为300m的理想抛物面的几何特征。

记Θ为基准球面对C点的张角，Θ_1理想抛物面对C点的张角。那么有

$$\frac{\Theta}{2}=\arcsin\frac{D}{2R}=56.443°$$

$$\frac{\Theta_1}{2}=\arctan\frac{D_1}{2H}=29.920°$$

由于

$$\frac{\Theta_1}{2}+90°-\beta=41.751°<\frac{\Theta}{2}$$

所以此时理想抛物面未超出主动反射面的边缘。

3）主索节点的调整与促动器的伸缩

显然，在问题二的背景下，理想抛物面的方程(6.2.1b)相当复杂，既不便于我们确定需调整主索节点，也不便于进行其他工作。于是，我们采取下列步骤来解决问题。

第一步，假想理想抛物面不动，将待观测天体S和主动反射面绕C点进行旋转，使得旋转后的S点位于$\alpha=0°,\beta=90°$的位置。记录下旋转后的主索节点位置。

第二步，根据得到的主索节点的新位置和方程(6.1.6)，确定需要调节的主索节点，然后将这些节点调节到理想抛物面上。记录调节后的主索节点位置，并求出相关促动器的伸缩量。

第三步，将S点和调整后的主动反射面绕点C逆向旋转，使它们回到实际位置。此时得到的主索节点位置即为要求得到的结果。

下面对以上步骤进行更细致的说明。

3.1）第一步与第三步的处理方法

在上述步骤中，第一步与第三步涉及到了点的坐标变换问题。为了便于统一计算，我们需要给出坐标变换矩阵。又因为第三步中的旋转过程与第一步完全相反，所以第三步中使用到的坐标变换矩阵是第一步中的坐标变换矩阵的逆矩阵。

下求第一步中的坐标变换矩阵J_1。

为了让点S可以旋转到$\alpha=0°,\beta=90°$的位置上，我们需要先让整个空间绕z轴顺时针旋转（逆着z轴正方向看）$\theta_1=36.795°$，再绕y轴逆时针旋转（顺着y轴正方向看）$\theta_2=90°-78.169°=11.831°$。这两次操作分别对应两个坐标变换矩阵

$$G_1=\begin{bmatrix}\cos\theta_1 & \sin\theta_1 & 0\\ -\sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{bmatrix},\quad G_2=\begin{bmatrix}\cos\theta_2 & 0 & -\sin\theta_2\\ 0 & 1 & 0\\ \sin\theta_2 & 0 & \cos\theta_2\end{bmatrix}$$

那么坐标变换矩阵J_1就是上述两个矩阵相乘的结果，即：

$$J_1=G_2G_1=\begin{bmatrix}\cos\theta_1\cos\theta_2 & \sin\theta_1\cos\theta_2 & -\sin\theta_2\\ -\sin\theta_1 & \cos\theta_1 & 0\\ \cos\theta_1\sin\theta_2 & \sin\theta_1\sin\theta_2 & \cos\theta_2\end{bmatrix} \tag{6.2.2}$$

将主索节点的坐标与矩阵J_1相乘，即可得到旋转后的主索节点坐标。

在得到J_1后，由于J_1是正交矩阵，故第三步中的坐标变换矩阵J_2就容易得到：

$$J_2=J_1^{-1}=J_1^T=\begin{bmatrix}\cos\theta_1\cos\theta_2 & -\sin\theta_1 & \cos\theta_1\sin\theta_2\\ \sin\theta_1\cos\theta_2 & \cos\theta_1 & \sin\theta_1\sin\theta_2\\ -\sin\theta_2 & 0 & \cos\theta_2\end{bmatrix} \tag{6.2.3}$$

将第二步中得到的主索节点的坐标与矩阵J_2相乘，即可得到在默认坐标系中，调节后的主索节点坐标。

3.2）第二步的处理方法

在这一步中，所有主索节点的坐标都是经第一步旋转后的坐标。

首先，由于理想抛物线的口径是$300m$，所以我们构造了一个圆柱面

$$\Gamma: x^2+y^2=150^2 \tag{6.2.4}$$

来筛选待调节的主索节点。在我们的模型中，待调节的节点是所有位于圆柱面Γ内的节点。

在得到待调节的主索节点后，接下来便是将它们移动到构造好的理想抛物面上。由于主索节点的位置是通过促动器的径向伸缩调节的，所以我们认为主索节点也是沿径向移动的——即对任意主索节点Q，它只能在直线CQ上移动。

设主索节点Q的坐标是(x_Q,y_Q,z_Q)，那么根据上面的假设，它在调节后的坐标可以表示为$(\lambda_Q x_Q,\lambda_Q y_Q,\lambda_Q z_Q)$。由于坐标$(x_Q,y_Q,z_Q)$是已知的，所以只需要求出待定参数$\lambda_Q$的值，即可知道节点$Q$在调节后的坐标。

因为Q在调节后位于方程(6.1.6)表示的抛物面上，所以求参数λ_Q的方程是下面的一元二次方程：

$$\lambda_Q z_Q=\frac{\lambda_Q^2\left({x_Q}^2+y_Q^2\right)}{561.28932}-300.73593 \tag{6.2.5}$$

又因为主索节点沿径向调整的距离应该远小于R，所以参数λ_Q的值应为

$$\lambda_Q = \frac{z_Q + \sqrt{z_Q^2 + 4 \times \frac{300.73593}{561.28932}\left({x_Q}^2 + y_Q^2\right)}}{\frac{2}{561.28932}\left({x_Q}^2 + y_Q^2\right)} \tag{6.2.6}$$

由于Q是任意节点，所以计算式(6.2.6)是普适的。

到这里为止，我们已经得到了所有需要调节的主索节点，并算出了它们经过调节后的坐标。接下来便是求促动器的伸缩量。

根据我们的假设，连接主索节点Q的下拉索和促动器都在直线CQ上。因为下拉索的长度不会变化，所以促动器的伸长（缩短）量等于线段CQ的缩短（伸长）量。记促动器的伸长量为Δl_Q，于是有

$$\Delta l_Q = \left(1 - \lambda_Q\right)\sqrt{{x_Q}^2 + y_Q^2 + z_Q^2} \tag{6.2.7}$$

综上，我们已经解决了问题二。

6.3 问题三的求解

因为题目中主索节点构成三角网格控制了反射面板的移动，我们先入为主地认为反射面板是一个个平面三角形；但在计算出接收比并参考了一些文献之后[1]，我们意识到反射面板可以采用特殊的工艺制造成球面。这样得到的接收比远高于平面板，于是我们利用蒙特卡罗模拟，分析了球面板的接收比，并得到了较为满意的结果。

1）反射面板是平面板的情况

step1：确定需要用到的反射面板。

因为题中要求比较的是“$300m$口径内反射面的反射信号”，这里有争议的地方是处于正好$300m$边界上的反射面板是否应该被计算。经过统一，我们认为题目要求视为完全在$300m$之内，也即若有一个主索节点不在$300m$内，该面板就不纳入计算。在这种前提下，我们纳入计算的反射面板数是1295块（若相反有一个主索节点在$300m$内就计算，反射面板数为1469块，这在之后球面板假设下的蒙特卡罗模拟算法中有用到，这里不详细介绍）

step2：计算单个面板权重

根据假设 4，光是均匀入射的，因此一块面板反射光量与该板水平投影面积成正比，

因此计算面板水平投影面积作为权，最后算出单个面板接收比后加权即可得总体接收比。

step3：计算接收比

因为是平面三角形，反射光在焦平面也形成一个三角形。只要计算入射到三个顶点的光线在焦平面上的位置，将这三个位置构成的平面三角形的面积记为S_1，它与半径$0.5m$的接收圆的重合面积为S_2，单块面板的接收比就是S_2/S_1。

因为已经进行坐标变换，相当于光源竖直入射，设向量$\vec{a}$的方向与入射光方向一致，由主索节点向量叉乘可以计算出反射面板的单位法向量$\vec{n}$，根据几何关系可得出射光的方向向量$\vec{b}$（这一思想在球面板时一样通用，不同的只是单位法向量的计算方式和入射点的选取）。通过入射点与反射光方向向量可以算出到达焦平面的位置。

最终计算得：调整后馈源仓的接收比为1.176%，对比基准球面接收比1.168%，改进不明显，接收比都太低。

2）反射面板是球面板的情况

2.1）基准球面时馈源仓接收比

由于基准球面、馈源舱有效区域是关于z轴旋转对称的，所以我们可以先在xCz平面中计算出可以反射到馈源舱有效区域中的电磁波的入射范围，然后确定接收比。

由于信号经反射后会衰减，故以下只考虑一次反射。

设基准圆弧最低点为A点，X是基准圆弧上的某点，$\angle ACX = \theta$。图 X 给出了电磁波的反射路径。根据图6.3.1不难看出，电磁波被反射后，它的传播路径直线l_X与直线$z = -L$的交点的横坐标与θ之间的关系为

$$x(\theta) = R\sin\theta - (R\cos\theta - L)\tan 2\theta \qquad (6.3.1)$$

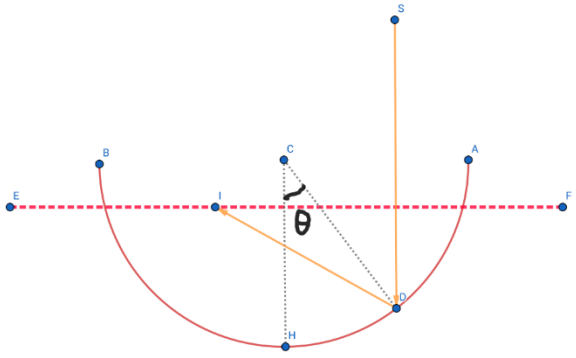

图 6.3.1

由于对称性，我们只需要考虑$0 \le \theta \le \frac{\Theta}{2}$的情况。借助程序打表计算可知，当$x \in (-0.5,0.5)$时，$\theta \in [0,0.025) \cup (0.349,0.368)$。又因为

$$\arcsin\frac{150}{300.4} = 0.523 > 0.368$$

进而可以算出口径为$300m$的基准球面的接收比为

$$\eta = \frac{300.4^2}{150^2} \times (\sin^2 0.025 + \sin^2 0.368 - \sin^2 0.349) = 5.258\% \qquad (6.3.2)$$

2.2）工作抛物面时馈源仓接收比

方法一：每块三角形板均匀打点

参照之前平面板假设下的思路，计算每块三角形板（只需选内部的1295块）的权重，只需找到在三角形板中均匀打点的方法[3]，然后在每块板中均匀打1000个点即可。因为运行限制，真实程序改在了每板100个点，运行时间大概在3小时左右。

最终，得到的接收比为67.42%。

方法二：整体打点

因为每块板中均匀打点的数据量要求大（1295×1000每个点判断能否接收需要解线性方程），我们选择在整个半径$150m$的圆板中打10000个点，统计接收比。因为要涵盖$300m$口径的圆，得首先取所有含有$300m$内主索节点的反射面板（共1469块）。然后需要解决判断点落在哪个三角形中，这里只需判断各点与三角形任意两顶点组成三角形面积是否等于三角形总面积。最后需要每个点在圆板内均匀[2]， 10000个点大概要3小时左右。

从图6.3.2中我们看到，最终实验图像趋于收敛，接收比为63.51%。

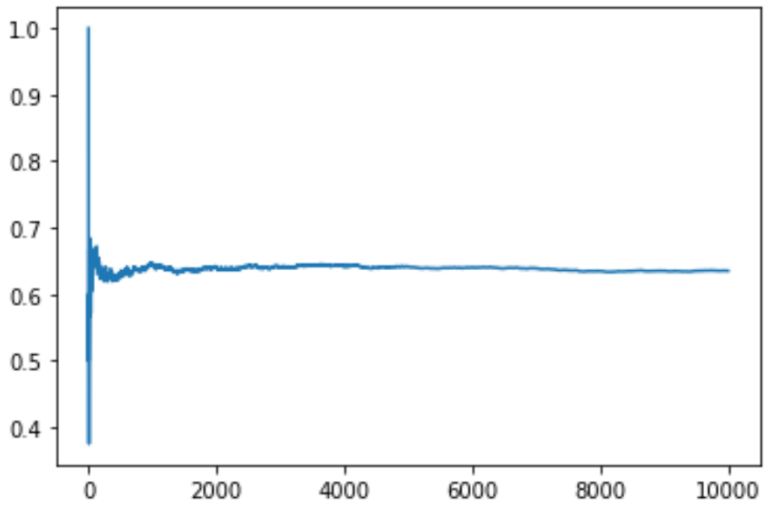

图 6.3.2

3）两个接收比的比较（球面板）

根据上述结果可以得知，调整后的接收比远大于调整前，馈源舱有效区域接收到的信号的强度大幅增加，在$300m$口径内接收比达到65%左右。

七、模型误差分析

本模型的误差主要由以下因素产生：

1）理想抛物面的拟合方法，存在反射面板与理想抛物面更接近的方式；

2）促动器的伸缩方向不一定完全沿球心（存在偏差）；

3）蒙特卡洛方法计算接收比时测试点的数量不足

八、模型评价

8.1 模型的合理之处与优点

如前文所述，本文针对 FAST 的运行原理建立了较为细致的模型。总的来说，本模型有以下优点：

1）结果合理：主动反射面经本模型调整后，接收比从5%左右提高到60%以上，符合大幅提高接收信号强度的工程要求。

2）逻辑清晰：本模型每一步都有足够的数学支撑，逻辑推导，模型中涉及到的计算原理相当简单，可以较快地得到结果。

3）方法巧妙：运用数学推导，完成理想抛物面的解析求解和基准球面接收比的解析求解；蒙特卡洛模拟的方法计算出接收比，有效地简化问题。

4）考虑全面：理想抛物面的选取综合分析了材料形变最小，300m 口径主索节点分类考虑了临界节点，接收比综合对比了平面、球面板及调节前后。

8.2 局限与不足

由于时间紧张、已知条件不足，在建立本模型的过程中，我们考虑的因素较少，建立出来的模型也存在一些瑕疵。

1）理想抛物面并非工程上的最优解

在参考论文[1]中，作者具体考虑了三种抛物面的情况：第一种方法是抛物面顶点和基准面底部重合；第二种是使抛物面相对基准面移动的最大距离最小化来确定抛物面，也就是本文的情况；第三种方法是使抛物面的边缘与基准面重合。作者针对这三种情况做了应力上的分析和模拟，最终得出结论，第二种方法相对产生的应力较大，不能成为最理想的抛物面。因此，如果要考虑到力学因素，本文以促动器变化范围最小为依据产生的理想抛物面需要进一步改进。

2）未能满足节点间距离的限制

题目中说，调节后相邻主索节点的变化幅度不超过 0.07%。这是一个很强的限制条件。根据本文的调节方案，需要沿径向把主索节点调节到抛物面上。但是，就算本文使用调节距离最小的理想抛物面，依然无法满足主索节点变化幅度不超过 0.07%这个约束。根据参考论文[1]中所述，主索节点会因内部应力的作用产生一个沿切向的位移，使之满足约束调件。

然而由于题目中未给出有关力学的信息，我们也未找见更详细的资料，故很难将模型在该方向上优化。

3）拟合理想抛物面的方法有待改进

本文采用把主索节点沿径向移动至理想抛物面上的方法进行拟合。然而，这并不是最佳的拟合方案。我们还可以选择在此基础上使每一个主索节点移动一个微小的位移，使反射板更贴近理想抛物面，如下图所示。根据参考论文[1]，我们可以在每一块反射面板上随机选取100个点，计算所有点相对于理想抛物面的偏移平方之和。以此作为损失函数，使其最小化得到主索节点的最佳位置。

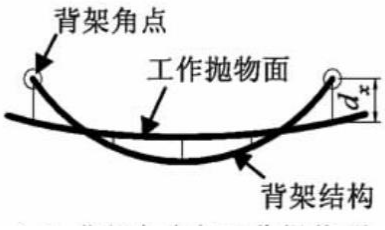

图 8.2.1

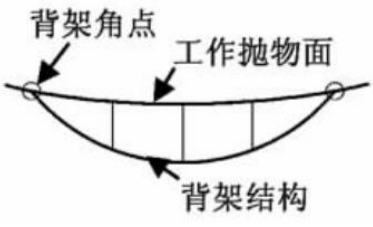

图 8.2.2

（图片摘自参考文献[1]）

4）计算接收比的方法有待改进

本文采用蒙特卡洛模拟的方法，由于时间原因，只选取了 10000 个点进行测试。这难免会带来一定程度的误差。之后可以通过增大测试点数量或者对每一块板的反射情况进行数学严格的计算以获得更加精确的结果。

九、参考文献

[1] 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.

[2] 李旭东,赵雪娇.矩形和椭圆内均匀分布随机点定理及应用[J].成都理工大学学报(自然科学版),2012,39(05):555-558.

[3] JieFeiLau，在几何图形中均匀随机取点算法总结及 Delaunay 三角剖分算法介绍 https://blog.csdn.net/u014028063/article/details/84314780

十、附录

附录 1

Python 代码:
python3.8 用 vscode 打开
代码运行需要导入附件一和附件三，涵盖了第二题、第三题、三维作图的运行单元，分块运行结果在输出框中，
其中第三题采用蒙特卡罗模拟，共取 10000 个点，运行速度慢，结果较为准确，球面板工作抛物面接收比为 63.51%，球面板基准面接收比 5.41%
对比球面板基准面物理计算所得 5.25%。

```
# %%
#第二问
#坐标变换
from numpy import *
import sympy
import pandas as pd
from scipy.optimize import fsolve
# %%
df1=pd.read_csv('./1.csv',encoding='gbk')
df2=pd.read_csv('./2.csv',encoding='gbk')
df3=pd.read_csv('./3.csv',encoding='gbk')
# %%
#画三维图
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
ar=array(df1)[:,1:]
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(ar[:,0],ar[:,1],ar[:,2],'o',markersize=1)
# %%
#坐标变换函数：原坐标到新坐标
def changexy_0to1(p0=[0,0,0],a=36.795,b=78.169):
    p1=[0,0,0]
    a=a/180*pi
    b=b/180*pi
    p0=array(p0)
    A=matrix([[cos(a)*sin(b),sin(a)*sin(b), -cos(b)],
```

```
            [-sin(a),cos(a),0],
            [cos(a)*cos(b),cos(b)*sin(a),sin(b)]
            ])
    p1=dot(A,p0.T)
    p1=array(p1)
    return p1[0]
#新坐标到原坐标
def changexy_1to0(p1=[0,0,0],a=36.795,b=78.169):
    p0=[0,0,0]
    a=a/180*pi
    b=b/180*pi
    p1=array(p1)
    A=matrix([[cos(a)*sin(b),sin(a)*sin(b), -cos(b)],
            [-sin(a),cos(a),0],
            [cos(a)*cos(b),cos(b)*sin(a),sin(b)]
            ])
    p0=dot(A.I,p1.T)
    p0=array(p0)
    return p0[0]

#把点 p 调整到抛物曲面上新的坐标点
def fit(p):
    np=[0,0,0]
    l=0
    k = sympy.symbols('k', positive = True)
    f= ((k*p[0])**2+(k*p[1])**2)/561.28934-300.735935-k*p[2]  #垂直入射下的抛物线方程
    val=sympy.solve([f], [k])
    if type(val)==dict :
        val=list(val.values())
    if type(val)==list:
        val=val[0]
    for vali in val:
        if vali>0:
            l=vali
            np[0]=l*p[0]
            np[1]=l*p[1]
            np[2]=l*p[2]
            break
    return np,l
# %%
# 在新坐标系下生成位置 newloc（满足 300m 条件的主索节点）
ar=array(df1)
newloc=[]
k_list=[]
id_list=[]
i=0
for p in ar:
    np=changexy_0to1(list(p[1:]))
    id=p[0]
    nnp,k=fit(np)
    if abs(nnp[0]**2+nnp[1]**2)<=150**2:
        newloc.append(nnp)
        k_list.append(k)
        id_list.append(id)
        i+=1
        print(i)
ndf=pd.DataFrame(newloc,columns=['x','y','z']) #ndf 是新坐标系下调整后的主索节点坐标
ndf['k']=k_list #k 是伸缩比例
ndf['id']=id_list #id 是主索节点编号
# %%
# 在新坐标系下生成位置 alnewloc（扩大 300m 条件）
alar=array(df1)
alnewloc=[]
alk_list=[]
alid_list=[]
i=0
for p in alar:
    np=changexy_0to1(list(p[1:]))
    id=p[0]
    if abs(np[0]**2+np[1]**2)<=170**2:
        nnp,k=fit(np)
        alnewloc.append(nnp)
        alk_list.append(k)
        alid_list.append(id)
```

```
            i+=1
            print(i)
        alndf=pd.DataFrame(alnewloc,columns=['x','y','z'])
        alndf['k']=alk_list
        alndf['id']=alid_list
        # %%
        #ndf 坐标变换回原坐标系，并计算伸缩量 生成第二题结果：df
        df1.index=df1['节点编号']
        loc=[]
        ssl=[]
        for nep,id in zip(newloc,id_list):
            p=changexy_1to0(nep)
            loc.append(p)
            ssl.append(-
pow(p[0]**2+p[1]**2+p[2]**2,0.5)+pow(df1.loc[id].values[1]**2+df1.loc[id].values[2]**2+df1.loc[id].values[3]**2,0.5))
        df=pd.DataFrame(loc,columns=['x','y','z'])
        df['k']=k_list
        df['id']=id_list
        df['ssl']=ssl #ssl 是伸缩量
        #df 是原坐标系下主索节点的位置和伸缩量信息
        # %%
        #抛物面顶点坐标
        dingdian=changexy_0to1([0,0,-300.73563])

        # %%
        df


        # %%
        #第三题


        # %%
        #提取出涉及到的三角形列表 df3_n(只要有点在 300m 口径内)
        def in_df(dingdian):
            if dingdian in array(df['id']):
                return True
            else:
                return False
        triangles=[]
        for dingdians in array(df3):
            flag=0
            for dingdian in dingdians:
                if in_df(dingdian):
                    flag +=1
            if flag>=1:
                triangles.append(dingdians)

        df3_n=pd.DataFrame(triangles,columns=['1','2','3'])
        # %%
        #在 df3_n 上加上球心的位置 1 号坐标系
        ndf.index=ndf['id']
        alndf.index=alndf['id']
        center=[]

        def find_c(X,arg):
            [x1,y1,z1,x2,y2,z2,x3,y3,z3,r]=arg
            x = float(X[0])
            y = float(X[1])
            z = float(X[2])
            return [
                (x-x1)**2+(y-y1)**2+(z-z1)**2-r**2,
                (x-x2)**2+(y-y2)**2+(z-z2)**2-r**2,
                (x-x3)**2+(y-y3)**2+(z-z3)**2-r**2
            ]
        i=0
        center=[]
        for dingdians in array(df3_n):
            p1=alndf.loc[dingdians[0]][:3]
            p2=alndf.loc[dingdians[1]][:3]
            p3=alndf.loc[dingdians[2]][:3]
            cp=fsolve(find_c,[0,0,0],[p1[0],p1[1],p1[2],p2[0],p2[1],p2[2],p3[0],p3[1],p3[2],300.4])
            center.append(cp)
            print(i)
```

```
    i+=1 #进度
df3_n['center']=center
df3_n



# %%
#计算接收率（当平面算）
#输入三个点，返回反射后投影平面的二维坐标
def touyingdian(p1,p2,p3):
    p1=array(p1)
    p2=array(p2)
    p3=array(p3)
    n=array([0,0,0])
    a=p1-p2
    b=p1-p3
    x,y,z,q = sympy.symbols('x y z q')
    val=sympy.solve([a[0]*x+a[1]*y+a[2]*z,b[0]*x+b[1]*y+b[2]*z,z**2+x**2+y**2-1], [x, y, z])
    n=array(val[0])
    I=array([0,0,-1])
    R=I-2*dot(I,n)*n #求反射向量
    val=[]
    for p in [p1,p2,p3]:
        vali=sympy.solve([q*R[0]-x+p[0],q*R[1]-y+p[1],q*R[2]-z+p[2],z+160.2], [x, y, z, q])
        val.append(list(vali.values())[0:2])
    [np1,np2,np3]=val
    return [np1,np2,np3]
shadow_sets=[]
S=[]
for i in range(len(df3_n)):
    three_ps=[]
    three_ps_shadow=[]
    for ip in df3_n.iloc[i].values[:3]:
        p=df1.loc[ip].values[1:]
        three_ps.append(p)
    three_ps_shadow=touyingdian(three_ps[0],three_ps[1],three_ps[2])
    shadow_sets.append(three_ps_shadow)
    q1=array(three_ps_shadow[0])-array(three_ps_shadow[1])
    q2=array(three_ps_shadow[2])-array(three_ps_shadow[1])
    Q=matrix([q1,q2])
    S.append(abs(q1[0]*q2[1]-q1[1]*q2[0])/2)#每个三角形的投影面积
    print(i)#进度
# %%
ratio=pi*0.5**2*len(S)/array(S).sum()
ratio #接收比

# %%
#考虑反射板是曲面
#判断点是否在三角形内
def IsTrangleOrArea(x1,y1,x2,y2,x3,y3):
    return abs((x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)) / 2.0)
def IsInside(x1,y1,x2,y2,x3,y3,x,y):
    #三角形 ABC 的面积
    ABC = IsTrangleOrArea(x1,y1,x2,y2,x3,y3)
    # 三角形 PBC 的面积。。。
    PBC = IsTrangleOrArea(x,y,x2,y2,x3,y3)
    PAC = IsTrangleOrArea(x1,y1,x,y,x3,y3)
    PAB = IsTrangleOrArea(x1,y1,x2,y2,x,y)
    return (-ABC + PBC + PAC + PAB<0.01)
def find_z(xy,cp,r=300.4):
    x=xy[0]
    y=xy[1]
    z=sympy.symbols('z')
    f=(x-cp[0])**2+(y-cp[1])**2+(z-cp[2])**2-r**2
    ans=sympy.solve([f],[z])
    a=[]
    for an in ans:
        a.append(an[0])
    return array(a).min()

def touyingdian2(xy):
    cp=[999,999,999]
    for sanjiaoxing in array(df3_n):
```

```
            A=alndf.loc[sanjiaoxing[0]][:3].values
            B=alndf.loc[sanjiaoxing[1]][:3].values
            C=alndf.loc[sanjiaoxing[2]][:3].values
            if IsInside(A[0],A[1],B[0],B[1],C[0],C[1],xy[0],xy[1]):
                cp=sanjiaoxing[3]
                break
        z=find_z(xy,cp)
        p=array([xy[0],xy[1],z])
        n=array(cp)-p
        n=n/pow(n[0]**2+n[1]**2+n[2]**2,0.5)
        I=array([0,0,-1])
        R=I-2*dot(I,n)*n #求反射向量
        x,y,z,q = sympy.symbols('x y z q')
        val=sympy.solve([q*R[0]-x+p[0],q*R[1]-y+p[1],q*R[2]-z+p[2],z+160.2], [x, y, z, q])
        return list(val.values())[0:2]
    # %%
    #随机点试验
    samples_num = 10000
    t = random.random(size=samples_num) * 2 * pi - pi
    r = random.random(size=samples_num) ** 0.5
    x_list = cos(t)*150.2*r
    y_list = sin(t)*150.2*r
    count=0
    sum=0
    ratio_list=[]
    for x,y in zip(x_list,y_list):
        sum+=1
        p=touyingdian2([x,y])
        if p[0]**2+p[1]**2<=0.25:
            count+=1
        ratio=count/sum
        ratio_list.append(ratio)
        print(sum,ratio)
    # %%
    #接收率
    ratio=count/sum
    ratio

    # %%
    #画图
    import matplotlib.pyplot as plt
    plt.plot(range(1,10001),ratio_list)

    # %%
    #对球面蒙特卡洛模拟
    def touyingdian3(xy):
        cp=[999,999,999]
        for sanjiaoxing in array(df3):
            A=df1.loc[sanjiaoxing[0]][1:4].values
            B=df1.loc[sanjiaoxing[1]][1:4].values
            C=df1.loc[sanjiaoxing[2]][1:4].values
            if IsInside(A[0],A[1],B[0],B[1],C[0],C[1],xy[0],xy[1]):
                cp=[0,0,0]
                break
        z=find_z(xy,cp)
        p=array([xy[0],xy[1],z])
        n=array(cp)-p
        n=n/pow(n[0]**2+n[1]**2+n[2]**2,0.5)
        I=array([0,0,-1])
        R=I-2*dot(I,n)*n #求反射向量
        x,y,z,q = sympy.symbols('x y z q')
        val=sympy.solve([q*R[0]-x+p[0],q*R[1]-y+p[1],q*R[2]-z+p[2],z+160.2], [x, y, z, q])
        return [val[x],val[y]]
    samples_num = 10000
    t = random.random(size=samples_num) * 2 * pi - pi
    r = random.random(size=samples_num) ** 0.5
    x_list3 = cos(t)*150.2*r
    y_list3 = sin(t)*150.2*r
    count=0
    sum=0
    ratio_list3=[]
    for x,y in zip(x_list3,y_list3):
        sum+=1
        p=touyingdian3([x,y])
```

```
            if p[0]**2+p[1]**2<=0.25:
                count+=1
            ratio3=count/sum
            ratio_list3.append(ratio)
            print(sum,ratio3)
        # %%
        ratio_list3
```

[4]

附录 2

matlab 源代码：

用的是 MATLAB R2020a

运行 matlab 程序前，先导入 workarea_initial，数据来自于题给附件 1 和附件 3.

依次运行 q1，q2，q3，q3_ori，q3_arcsurface（movepoint、isreceive、solvecenter 是辅助函数），得到第一题（q1）第二题（q2）

第三题（q3 得到平面板工作抛物面接收比为 1.176%，对比 q3_ori 得到的平面板基准面接收比 1.168%）结果，

q3_arcsurface 运行一次得到球面板工作抛物面接收比为 67.42%（虽然最后不采用这个数据）；

其中第一、三题结果直接输出到 command window，第二题结果自动填入到 附录 4 excel 表格；

但第三题球面板的计算采用 300m 内每块三角板均匀打点的模型，所需数据大，运行速度慢，在每块采样点达到 1000 时，准确率才可以保证，

不过目前每平方米 1 个点，就需要计算机运行 2 个小时左右，因此采用整体打点的模型更好。

```
function tr = isreceive(x0,y0,z0,p1,p2,p3,o)
    r=300.4;
    a=[0 0 -1];
    n=o-[x0,y0,z0];
    nnorm=n/norm(n);
    b=a-2*dot(a,nnorm)*nnorm;
    syms lambda;
    f=b(3)*lambda+z0+0.534*r;
    s=solve(f);
    k=double(s);
    out=[x0 y0 z0]+k*b;
    if out(1)^2+out(2)^2<=1/4
        tr=1;
    else
        tr=0;
    end


end

function [targ,k1,ecp] = movepoint(x)
k1=0; %k1 的类型是数值
y=[0 0 0]; %y 的类型是数值矩阵
digits(6);
syms k;
x1=k*x;
f=x1(3)-(x1(1)^2+x1(2)^2)/561.28934+300.735935;
s=solve(f);
if length(s)==2
    if s(1)>0
        k1=double((s(1)));
    else
        k1=double((s(2)));
    end
else
    k1=double(s);
end

ecp=(1-k1)*sqrt(x(1)^2+x(2)^2+x(3)^2); %以促动器趋向球心为正方向
y=k1*x;
targ=double(y);
end

syms dx;
```

```
r=300.4;
x1=150;
p=2*(0.466*r+dx);
ya=x1^2/2/p-dx-r;
f1=sqrt(x1^2+ya^2)-r;
f2=dx;
f3=r-sqrt(-p^2+2*p*r+2*p*dx);
n=100;
x=0;
for t=0.3359:0.000001:0.336
   a=abs(double(subs(f1,t)));
   b=abs(double(subs(f2,t)));
   c=abs(double(subs(f3,t)));
   m=max([a,b,c]);
   if m<n
      n=m;
      x=t;
   end
end
P=4*(0.466*r+x);

fprintf('最终平面抛物线方程为 y=x^2/%f-%f\n',P,r+x);
fprintf('最终空间抛物面方程为 z=x^2+y^2/%f-%f\n',P,r+x);

%nodep 及 1 2 3 是先选点再移动到工作抛物面的结果 共有 692 个点
%nodepx1 及 2x 3x 是先移动到工作抛物面再选点的结果 共 689 个点 据题意，最终选取 689 个点。
alpha=36.795*2*pi/360;
beta=78.169*2*pi/360;
a=[cos(-alpha) -sin(-alpha) 0;sin(-alpha) cos(-alpha) 0;0 0 1];
b=[cos(beta-pi/2) 0 sin(beta-pi/2);0 1 0;-sin(beta-pi/2) 0 cos(beta-pi/2)];
%导入 2226*1 nodenum0 附件 1 中主索节点编号 字符串阵
%导入 2226*3 nodeori 附件 1 中节点位置 数值矩阵

nodep=zeros(2226,3);%新坐标下全部节点位置
nodep1=zeros(1000,3);%新坐标下筛选后点位置
nodep2=zeros(692,3);%新坐标下筛选后调工作面 点位置
nodep3=zeros(692,3);%原来坐标下筛选后调工作面 点位置
d=zeros(2226,1);%记录平面内距离
k=zeros(692,1);%记录伸缩比例
ecp=zeros(692,1);%expansion and contraction quantity 伸缩量
n=1; %先选点再移动的 300m 内主索节点数

for i=1:2226
   nodep(i,:)=transpose(b*a*(transpose(nodeori(i,:))));
   d(i,1)=sqrt(nodep(i,1)^2+nodep(i,2)^2);
   if d(i,1)<=150
      nodenum(n,1)=nodenum0(i,1);%筛选后主索节点编号
      nodep1(n,:)=nodep(i,:);
      [nodep2(n,:) k(n,1) ecp(n,1)]=movepoint(nodep1(n,:));
      nodep3(n,:)=transpose(inv(a)*inv(b)*transpose(nodep2(n,:)));
      n=n+1;%1 到 n-1 都有
   end
end

nodepx1=zeros(2226,3);%新坐标下移动到工作抛物面的主索节点位置
ecp0=zeros(2226,1);%全部的伸缩量
k0=zeros(2226,1);%全部的伸缩比例
for i=1:2226
   [nodepx1(i,:) k0(i,1) ecp0(i,1)]=movepoint(nodep(i,:));
end
nux=0;
nodep2x=zeros(689,3);
nodep3x=zeros(689,3);
ecpx=zeros(689,1);
for i=1:2226
   if nodepx1(i,1)^2+nodepx1(i,2)^2<=150^2
      nux=nux+1;
      ecpx(nux)=ecp0(i);
      nodenumx(nux,:)=nodenum0(i,:);
      nodep2x(nux,:)=nodepx1(i,:);
      nodep3x(nux,:)=transpose(inv(a)*inv(b)*transpose(nodep2x(nux,:)));
   end
```

```
end
vert=transpose(inv(b)*inv(a)*transpose([0 0 -300.7360]));
writematrix(vert,'附件 4.xlsx','sheet',1,'Range','A2:C2')
xlswrite('附件 4.xlsx',nodenumx,'调整后主索节点编号及坐标','A2:A690')
writematrix(nodep3x,'附件 4.xlsx','sheet',2,'Range','B2:D690')
xlswrite('附件 4.xlsx',nodenumx,'促动器顶端伸缩量','A2:A690')
writematrix(ecpx,'附件 4.xlsx','sheet',3,'Range','B2:B690')

%导入 4300*3 的反射面板主索节点 字符串数组 panel
%nodep2 是新坐标下筛选后调到工作抛物面时的各主索节点位置 数值矩阵
n=692;%q2 中已求出的 300m 内主索节点个数
p1=[0 0 0];%满足 300m 内三角板第一个顶点
p2=[0 0 0];%满足 300m 内三角板第二个顶点
p3=[0 0 0];%满足 300m 内三角板第三个顶点
out1=[0 0 0];%反射到焦平面的第一个顶点
out2=[0 0 0];%反射到焦平面的第二个顶点
out3=[0 0 0];%反射到焦平面的第三个顶点
S=zeros(4000,1); %总面积(通光量)，最后要加的权
S_per=zeros(4000,1);%每块板的接收比
pa=0; %完全在 300m 内面板数
emit=[0 0 0];%反射光方向
so=[0 0 0];%解方程结果
lam=[0 0 0];%解方程结果
F=-0.534*300.4; %焦平面纵坐标
S_pa=0;%一块板反射光打在焦平面上面积
S_re=0;%反射光被馈源仓接收面积
nothere=0;
for i=1:4300
    flag=1;
    for j=1:n
        if strcmp(panel(i,1),nodenum(j,1))
            flag=flag+1;
            p1=nodep2(j,:);
        end
        if strcmp(panel(i,2),nodenum(j,1))
            flag=flag+1;
            p2=nodep2(j,:);
        end
        if strcmp(panel(i,3),nodenum(j,1))
            flag=flag+1;
            p3=nodep2(j,:);
        end
        if flag==4
            break
        end
    end
    if flag~=4
        %disp('不在');
        nothere=nothere+1;
        continue
    else
        pa=pa+1;
        S(pa)=1/2*abs(p1(1)*p2(2)+p2(1)*p3(2)+p3(1)*p1(2)-p2(1)*p1(2)-p3(1)*p2(2)-p1(1)*p3(2));
        %已知三点求水平面投影面积公式
        ab=p2-p1; %ac 边向量
        ac=p3-p1; %ac 边向量
        nv=cross(ab,ac);%法向量
        nvnorm=nv/norm(nv);%单位法向量
        emit=[0 0 -1]-2*dot([0 0 -1],nvnorm)*nvnorm;%出射光线方向
        syms lambda
        f1=p1(3)+emit(3)*lambda-F;
        so(1)=solve(f1);
        f2=p2(3)+emit(3)*lambda-F;
        so(2)=solve(f2);
        f3=p3(3)+emit(3)*lambda-F;
        so(3)=solve(f3);
        lam=double(so);
        out1=lam(1)*emit+p1;
        out2=lam(2)*emit+p2;
        out3=lam(3)*emit+p3;
```

```
        S_pa=1/2*abs(out1(1)*out2(2)+out2(1)*out3(2)+out3(1)*out1(2)-out2(1)*out1(2)-out3(1)*out2(2)-out1(1)*out3(2));

        S_re=pi/4;
        S_per(pa)=S_re/S_pa;
    end

end

final_re=sum(S.*S_per)/sum(S);
disp(final_re);

%导入 4300*3 的反射面板主索节点 字符串数组
%nodep1 是新坐标下筛选后在原来基准面各主索节点位置 数值矩阵
n=692;%q2 中已求出的 300m 内主索节点个数
p1=[0 0 0];%满足 300m 内三角板第一个顶点
p2=[0 0 0];%满足 300m 内三角板第二个顶点
p3=[0 0 0];%满足 300m 内三角板第三个顶点
out1=[0 0 0];%反射到焦平面的第一个顶点
out2=[0 0 0];%反射到焦平面的第二个顶点
out3=[0 0 0];%反射到焦平面的第三个顶点
S=zeros(4000,1); %总面积(通光量)，最后要加的权
S_per=zeros(4000,1);%每块板的接收比
pa=0; %在 300m 内面板数
emit=[0 0 0];%反射光方向
so=[0 0 0];%解方程结果
lam=[0 0 0];%解方程结果
F=-0.534*300; %焦平面纵坐标
S_pa=0;%一块板反射光打在焦平面上面积
S_re=0;%反射光被馈源仓接收面积
nothere=0;
for i=1:4300
    flag=1;
    for j=1:n
        if strcmp(panel(i,1),nodenum(j,1))
            flag=flag+1;
            p1=nodep1(j,:);
        end
        if strcmp(panel(i,2),nodenum(j,1))
            flag=flag+1;
            p2=nodep1(j,:);
        end
        if strcmp(panel(i,3),nodenum(j,1))
            flag=flag+1;
            p3=nodep1(j,:);
        end
        if flag==4
            break
        end
    end
    if flag~=4
        %disp('不在');
        nothere=nothere+1;
        continue
    else
        pa=pa+1;
        S(pa)=1/2*abs(p1(1)*p2(2)+p2(1)*p3(2)+p3(1)*p1(2)-p2(1)*p1(2)-p3(1)*p2(2)-p1(1)*p3(2));
        %已知三点求水平面投影面积公式
        ab=p2-p1; %ac 边向量
        ac=p3-p1; %ac 边向量
        nv=cross(ab,ac);%法向量
        nvnorm=nv/norm(nv);%单位法向量
        emit=[0 0 -1]-2*dot([0 0 -1],nvnorm)*nvnorm;%出射光线方向
        syms lambda
        f1=p1(3)+emit(3)*lambda-F;
        so(1)=solve(f1);
        f2=p2(3)+emit(3)*lambda-F;
        so(2)=solve(f2);
        f3=p3(3)+emit(3)*lambda-F;
        so(3)=solve(f3);
        lam=double(so);
        out1=lam(1)*emit+p1;
```

```
        out2=lam(2)*emit+p2;
        out3=lam(3)*emit+p3;
        S_pa=1/2*abs(out1(1)*out2(2)+out2(1)*out3(2)+out3(1)*out1(2)-out2(1)*out1(2)-out3(1)*out2(2)-out1(1)*out3(2));
        S_re=pi/4;
        S_per(pa)=S_re/S_pa;
    end

end

ori_re=sum(S.*S_per)/sum(S);
disp(ori_re);

function p = solvecenter(p1,p2,p3,r)
    syms x y z;

    f1=norm([x y z]-p1)-r;
    f2=norm([x y z]-p2)-r;
    f3=norm([x y z]-p3)-r;
    s=solve(f1,f2,f3,[x,y,z]);
    if s.z(1)>max([p1(3),p2(3),p3(3)])
        p(1)=double(s.x(1));
        p(2)=double(s.y(1));
        p(3)=double(s.z(1));
    else
        p(1)=double(s.x(2));
        p(2)=double(s.y(2));
        p(3)=double(s.z(2));
    end

end
```