附录 **A** 计算程序

## 问题 1 计算程序

% 计算符合光照条件的点在angle\_1中的索引，存储在fit\_triangle 中

axis\_1 = zeros(4300,1); axis\_2 = zeros(4300,1); axis\_3 = zeros(4300,1); fit\_triangle = zeros(4300,3); belong\_index = 0;

for i = 1:4300

a = triangle(i,1); a = string(a);

axis\_1(i,1) = find(strcmp(a, point\_name)); b = triangle(i,2);

b = string(b);

axis\_2(i,1) = find(strcmp(b, point\_name)); c = triangle(i,3);

c = string(c);

axis\_3(i,1) = find(strcmp(c, point\_name));

end

% 找到符合半径条件的反射面板，有若三角形有两个点都在口径范围内，则视为满足条件

for i = 1:4300

mat\_angle\_1 = cell2mat(angle\_1(:,5));

% belong\_index = belong\_index + (abs(mat\_angle\_1(axis\_1(i,1),1)) > 60) + (abs(mat\_angle\_1(axis\_2(i,1),1)) > 60) + (abs(mat\_angle\_1(axis\_3(i,1),1)) > 60);

if axis\_1(i,1) <= length(update\_length(:,1)) && axis\_2(i,1) <= length(update\_length(:,1)) && axis\_3(i,1) <= length(update\_length(:,1))

fit\_triangle(i,1) = axis\_1(i,1); fit\_triangle(i,2) = axis\_2(i,1); fit\_triangle(i,3) = axis\_3(i,1);

end

end

% 去除不满足条件的三角面板（去除0值）

fit\_triangle = fit\_triangle(any(fit\_triangle,2),:);

% 根据update\_length伸长量来计算移动后的坐标索引,计算得到的索引存在new\_axis中

new\_axis = zeros(2226,3); r\_new\_axis = zeros(2226,3); dl\_index = 1:5;

index\_3 = zeros(length(dl\_index),1);

global o;

for i = 1:length(o)

if inv\_r\_base\_axis(i,1) < 0 o(i,2) = o(i,2) + 180;

end end

for i = 1:length(dl\_index)

for j = 1:length(update\_length) new\_axis(j,1) =

update\_length(j,dl\_index(i))\*cosd(angle\_1\_mat(j,1))\*cosd(angle\_1\_mat(j,2)); new\_axis(j,2) =

update\_length(j,dl\_index(i))\*cosd(angle\_1\_mat(j,1))\*sind(angle\_1\_mat(j,2)); new\_axis(j,3) = update\_length(j,dl\_index(i))\*sind(angle\_1\_mat(j,1));

end

[new\_angle, new\_line, temp\_a, temp\_b, temp\_c, index\_3(i,1)] = triangle\_calculator(fit\_triangle);

end

% global r\_mat;

% for i = 1:length(update\_length)

% r\_new\_axis(i,:) = new\_axis(i,:)\*r\_mat;

% end

%

% plot3(r\_new\_axis(:,1),r\_new\_axis(:,2),r\_new\_axis(:,3),'r\*');

% 生成一个有空位的元胞数组存储符合条件的结点，用于索引

% global angle\_1

% fit\_point\_name\_2 = cell(length(r\_new\_axis()),1);

% for i = 1:length(r\_new\_axis(:,1))

% if r\_new\_axis(i,1) ~= 0

% fit\_point\_name\_2(i,1) = angle\_1(i,1);

% end

% end

% fit\_point\_name\_2\_nan = fit\_point\_name\_2;

% 生成一个没有空位的元胞数组存储符合条件的节点，用于计数

% fit\_point\_name\_2(cellfun(@isempty,fit\_point\_name\_2)) = [];

% 计算了每次抛物线上下移动相等的距离所产生的新的伸长量，第一次迭代

surface\_1 = surface.'; o = zeros(2226,2); rad = zeros(2226,2); min\_index = -0.6;

max\_index = 0.6; for i = 1:2226

% 第一问

x = surface(i,1); y = surface(i,2); z = surface(i,3);

end

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1)); for i = 2:length(unique(rad(:,1)))

if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 能够满足正负0.6的顶点移动距离条件的顶点移动距离

success\_dl = zeros(num,1); success\_index = 1;

fail\_up = 0;

fail\_down = 0;

max\_move\_length = zeros(num,0); min\_move\_length = zeros(num,0); begin = 0.1;

pace = 0.1;

final = 0.6;

num = ((final - begin)/pace + 1)\*2; update\_length = zeros(2226,num); sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

figure;

for dl = begin:pace:final fail\_down = 0;

x = (-500:500);

% 顶点向上移动

z1 = z0 + dl;

% up\_result代表向上移动后，各主索节点的横坐标up\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue; else

z\_line = tan(unique\_rad(i))\*x;

% plot(x,z\_line); syms x\_1;

eqn = tan(unique\_rad(i))\*x\_1 == x\_1^2/(4\*(F-dl))+z1;

% 联立方程

S = solve(eqn, x\_1); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) up\_result(1,i) = S\_double(2,1);

else

up\_result(1,i) = S\_double(1,1);

end

end

end

up\_result(up\_result==0) = [];

% 根据满足条件的各点横坐标计算各点到c的距离，将多次的结果都存储在update\_length中, print\_up是各下拉锁移动距离

print\_up = zeros(length(up\_result),1); for i = 1:length(up\_result)

if i == 1

print\_up(i,1) = dl; else

print\_up(i,1) = up\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(angle\_1\_mat(:,1) - unique\_angle(i)) < 0.000001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32(num/2+dl/pace)) = up\_result(i)/cos(unique\_rad(i));

end

% 第一问得到706行的update\_length，第二问得到692行update\_length

% 对每一次各个长度的求和

sum(int32(num/2+dl/pace)) = sum(int32(num/2+dl/pace)) + abs(print\_up(i,1)); if abs(print\_up(i,1)) > 0.6

% 有超过界限的直接排除

fail\_up = 1; break;

end

end

end

% 假如促动器全部满足条件，视为成功,将dl存储,并且画出横坐标与伸长量print\_up的图,

% 并且记录最大移动量和最小移动量

if fail\_up == 0 success\_dl(success\_index,1) = dl; success\_index = success\_index + 1;

plot(up\_result(1,2:end), -print\_up(2:end,1)); hold on

max\_move\_length(int32(num/2+dl/pace),1) = max(print\_up(2:end,1)); min\_move\_length(int32(num/2+dl/pace),1) = min(print\_up(2:end,1));

end

% 将成功指标置为成功

fail\_up = 0;

% 顶点向下移动

z2 = z0 - dl;

down\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue;

else

z\_line = tan(unique\_rad(i))\*x; syms x\_2;

eqn = tan(unique\_rad(i))\*x\_2 == x\_2^2/(4\*(F+dl))+z2;

% 联立方程

S = solve(eqn, x\_2); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) down\_result(1,i) = S\_double(2,1);

else

down\_result(1,i) = S\_double(1,1);

end

end

end

down\_result(down\_result==0) = [];

% 促动器运动距离有超出范围的，失败, 继续下一次循环

print\_down = zeros(length(down\_result),1); for i = 1:length(down\_result)

if i == 1

print\_down(i,1) = -dl; else

print\_down(i,1) = down\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(angle\_1\_mat(:,1) - unique\_angle(i)) < 0.0001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32(num/2+1-dl/pace)) = abs(down\_result(i)/cos(unique\_rad(i)));

end

sum(int32(num/2+1-dl/pace)) = sum(int32(num/2+1-dl/pace)) + abs(print\_down(i,1)); if abs(print\_down(i,1)) > 0.6

fail\_down = 1; break;

end

end

end

% 假如促动器全部满足条件，视为成功,将dl存储,并且画出横坐标与伸长量print\_down的图,

% 并且记录最大移动量和最小移动量

if fail\_down == 0 success\_dl(success\_index,1) = -dl; success\_index = success\_index + 1;

plot(down\_result(1,2:end), -print\_down(2:end,1)); hold on

max\_move\_length(int32(num/2+1-dl/pace),1) = max(print\_down(2:end,1)); min\_move\_length(int32(num/2+1-dl/pace),1) = min(print\_down(2:end,1));

end

% 将成功指标置为成功

fail\_down = 0;

end hold off

%%

update\_length(all(update\_length==0,2),:) = [];

%%

% 加精计算-0.3 - -0.5的区间，第二次迭代

% 计算了每次抛物线上下移动相等的距离所产生的新的伸长量

surface\_1 = surface.'; o = zeros(2226,2); rad = zeros(2226,2); min\_index = -0.6;

max\_index = 0.6; for i = 1:2226

% 第一问

x = surface(i,1); y = surface(i,2); z = surface(i,3);

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

end

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1)); for i = 2:length(unique(rad(:,1)))

if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 能够满足正负0.6的顶点移动距离条件的顶点移动距离

begin = 0.3;

pace = 0.01;

final = 0.5;

num = int32((final - begin)/pace + 1) ; success\_dl = zeros(num,1); success\_index = 1;

fail\_up = 0;

fail\_down = 0;

max\_move\_length = zeros(num,0); min\_move\_length = zeros(num,0); update\_length = zeros(2226,num); sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

figure;

for dl = begin:pace:final fail\_down = 0;

x = (-500:500);

% 顶点向下移动

z2 = z0 - dl;

down\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue;

else

z\_line = tan(unique\_rad(i))\*x;

syms x\_2;

eqn = tan(unique\_rad(i))\*x\_2 == x\_2^2/(4\*(F+dl))+z2;

% 联立方程

S = solve(eqn, x\_2); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) down\_result(1,i) = S\_double(2,1);

else

down\_result(1,i) = S\_double(1,1);

end

end

end

down\_result(down\_result==0) = [];

% 促动器运动距离有超出范围的，失败, 继续下一次循环

print\_down = zeros(length(down\_result),1); for i = 1:length(down\_result)

if i == 1

print\_down(i,1) = -dl; else

print\_down(i,1) = down\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(angle\_1\_mat(:,1) - unique\_angle(i)) < 0.0001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32((dl - begin)/pace + 1)) = abs(down\_result(i)/cos(unique\_rad(i)));

end

sum(int32((dl - begin)/pace + 1)) = sum(int32((dl - begin)/pace + 1)) + abs(print\_down(i,1));

if abs(print\_down(i,1)) > 0.6 fail\_down = 1;

break;

end

end

end

% 假如促动器全部满足条件，视为成功,将dl存储,并且画出横坐标与伸长量print\_down的图,

% 并且记录最大移动量和最小移动量

if fail\_down == 0 success\_dl(success\_index,1) = -dl; success\_index = success\_index + 1;

plot(down\_result(1,2:end), -print\_down(2:end,1)); hold on

max\_move\_length(int32((dl - begin)/pace + 1),1) = max(print\_down(2:end,1)); min\_move\_length(int32((dl - begin)/pace + 1),1) = min(print\_down(2:end,1));

end

% 将成功指标置为成功

fail\_down = 0;

end

hold off

%%

update\_length(all(update\_length==0,2),:) = [];

%%

% 一些变量的初始化surface\_1 = surface.'; o = zeros(2226,2); rad = zeros(2226,2); for i = 1:2226

x = surface(i,1); y = surface(i,2); z = surface(i,3);

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

end

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1)); update\_length = zeros(2226,1); for i = 2:length(unique(rad(:,1)))

if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 首次成功的顶点移动距离

first\_success\_dl = 0;

begin = 0.1;

pace = 0.1;

final = 0.6;

num = ((final - begin)/pace + 1)\*2; sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

## 问题 2 计算程序

% 根据update\_length伸长量来计算移动后的坐标索引,计算得到的索引存在new\_axis中

% 每次进行基准球面的计算时，需要先运行此脚本

new\_axis = zeros(2226,3); r\_new\_axis = zeros(2226,3); dl\_index = 1:5;

index\_3 = zeros(length(dl\_index),1);

global o;

for i = 1:length(o)

if inv\_r\_base\_axis(i,1) < 0 o(i,2) = o(i,2) + 180;

end end

for j = 1:length(update\_length)

new\_axis(j,1) = update\_length(j,3)\*cosd(o(j,1))\*cosd(o(j,2)); new\_axis(j,2) = update\_length(j,3)\*cosd(o(j,1))\*sind(o(j,2)); new\_axis(j,3) = update\_length(j,3)\*sind(o(j,1));

end

% new\_axis(1,1) = 0;

% new\_axis(1,2) = 0;

% [new\_angle, new\_line, temp\_a, temp\_b, temp\_c, index\_3] = triangle\_calculator(fit\_triangle);

global r\_mat;

for i = 1:length(update\_length) r\_new\_axis(i,:) = new\_axis(i,:)\*r\_mat;

end

plot3(r\_new\_axis(:,1),r\_new\_axis(:,2),r\_new\_axis(:,3),'r\*');

% 生成一个有空位的元胞数组存储符合条件的结点，用于索引

global angle\_1

fit\_point\_name\_2 = cell(length(r\_new\_axis()),1); for i = 1:length(r\_new\_axis(:,1))

if r\_new\_axis(i,1) ~= 0 fit\_point\_name\_2(i,1) = angle\_1(i,1);

end

end

fit\_point\_name\_2\_nan = fit\_point\_name\_2;

% 生成一个没有空位的元胞数组存储符合条件的节点，用于计数

fit\_point\_name\_2(cellfun(@isempty,fit\_point\_name\_2)) = [];

% 根据update\_length伸长量来计算移动后的坐标索引,计算得到的索引存在new\_axis中

new\_axis = zeros(2226,3); r\_new\_axis = zeros(2226,3); dl\_index = 1:5;

index\_3 = zeros(length(dl\_index),1);

global o;

for i = 1:length(o)

if inv\_r\_base\_axis(i,1) < 0 o(i,2) = o(i,2) + 180;

end end

for i = 1:length(dl\_index)

for j = 1:length(update\_length)

new\_axis(j,1) = update\_length(j,dl\_index(i))\*cosd(o(j,1))\*cosd(o(j,2)); new\_axis(j,2) = update\_length(j,dl\_index(i))\*cosd(o(j,1))\*sind(o(j,2)); new\_axis(j,3) = update\_length(j,dl\_index(i))\*sind(o(j,1));

end

% new\_axis(1,1) = 0;

% new\_axis(1,2) = 0;

% [new\_angle, new\_line, temp\_a, temp\_b, temp\_c, index\_3] = triangle\_calculator(fit\_triangle);

end

global r\_mat;

for i = 1:length(update\_length) r\_new\_axis(i,:) = new\_axis(i,:)\*r\_mat;

end

plot3(r\_new\_axis(:,1),r\_new\_axis(:,2),r\_new\_axis(:,3),'r\*');

% 生成一个有空位的元胞数组存储符合条件的结点，用于索引

global angle\_1

fit\_point\_name\_2 = cell(length(r\_new\_axis()),1); for i = 1:length(r\_new\_axis(:,1))

if r\_new\_axis(i,1) ~= 0 fit\_point\_name\_2(i,1) = angle\_1(i,1);

end

end

fit\_point\_name\_2\_nan = fit\_point\_name\_2;

% 生成一个没有空位的元胞数组存储符合条件的节点，用于计数

fit\_point\_name\_2(cellfun(@isempty,fit\_point\_name\_2)) = [];

% 计算了每次抛物线上下移动相等的距离所产生的新的伸长量,第一次迭代

surface\_1 = surface.'; o = zeros(2226,2);

rad = zeros(2226,2); for i = 1:2226

% 处理第二问，需要将旋转坐标轴后的点坐标带入

x = inv\_r\_base\_axis(i,1); y = inv\_r\_base\_axis(i,2); z = inv\_r\_base\_axis(i,3);

end

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1)); for i = 2:length(unique(rad(:,1)))

if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 能够满足正负0.6的顶点移动距离条件的顶点移动距离

success\_dl = zeros(num,1); success\_index = 1;

fail\_up = 0;

fail\_down = 0;

max\_move\_length = zeros(num,0); min\_move\_length = zeros(num,0); begin = 0.1;

pace = 0.1;

final = 0.6;

num = ((final - begin)/pace + 1)\*2; update\_length = zeros(2226,num); sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

figure;

for dl = begin:pace:final fail\_down = 0;

x = (-500:500);

% 顶点向上移动

z1 = z0 + dl;

% z = x.^2/(4\*(F-dl))+z1;

% plot(x,z)

% hold on

% up\_result代表向上移动后，各主索节点的横坐标up\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue; else

z\_line = tan(unique\_rad(i))\*x;

% plot(x,z\_line); syms x\_1;

eqn = tan(unique\_rad(i))\*x\_1 == x\_1^2/(4\*(F-dl))+z1;

% 联立方程

S = solve(eqn, x\_1); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) up\_result(1,i) = S\_double(2,1);

else

up\_result(1,i) = S\_double(1,1);

end

end

end

up\_result(up\_result==0) = [];

% 根据满足条件的各点横坐标计算各点到c的距离，将多次的结果都存储在update\_length中, print\_up是各下拉锁移动距离的绝对值

print\_up = zeros(length(up\_result),1); for i = 1:length(up\_result)

print\_up(i,1) = up\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(o(:,1) - unique\_angle(i)) < 0.000001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32(num/2+dl/pace)) = abs(up\_result(i)/cos(unique\_rad(i)));

end

% 第一问得到706行的update\_length，第二问得到692行update\_length

% 对每一次各个长度的求和

sum(int32(num/2+dl/pace)) = sum(int32(num/2+dl/pace)) + abs(print\_up(i,1));

if abs(print\_up(i,1)) > 0.6 fail\_up = 1;

break;

end

end

% 假如促动器全部满足条件，视为成功

if fail\_up == 0 success\_dl(success\_index,1) = dl; success\_index = success\_index + 1;

plot(up\_result(1,2:end), -print\_up(2:end,1)); hold on

max\_move\_length(int32(num/2+dl/pace),1) = max(print\_up(2:end,1)); min\_move\_length(int32(num/2+dl/pace),1) = min(print\_up(2:end,1));

end

% 顶点向下移动

z2 = z0 - dl;

down\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue;

else

z\_line = tan(unique\_rad(i))\*x;

% plot(x,z\_line); syms x\_2;

eqn = tan(unique\_rad(i))\*x\_2 == x\_2^2/(4\*(F+dl))+z2;

% 联立方程

S = solve(eqn, x\_2); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) down\_result(1,i) = S\_double(2,1);

else

down\_result(1,i) = S\_double(1,1);

end

end

end

down\_result(down\_result==0) = [];

% 促动器运动距离有超出范围的，失败, 继续下一次循环

print\_down = zeros(length(down\_result),1); for i = 1:length(down\_result)

print\_down(i,1) = down\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(o(:,1) - unique\_angle(i)) < 0.0001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32(num/2+1-dl/pace)) =

abs(down\_result(i)/cos(unique\_rad(i)));

end

sum(int32(num/2+1-dl/pace)) = sum(int32(num/2+1-dl/pace)) + abs(print\_down(i,1)); if abs(print\_down(i,1)) > 0.6

fail\_down = 1; break;

end

end

% 假如促动器全部满足条件，视为成功

if fail\_down == 0 success\_dl(success\_index,1) = -dl; success\_index = success\_index + 1;

plot(down\_result(1,2:end), -print\_down(2:end,1)); hold on

max\_move\_length(int32(num/2+1-dl/pace),1) = max(print\_down(2:end,1)); min\_move\_length(int32(num/2+1-dl/pace),1) = min(print\_down(2:end,1));

end end

%%

% update\_length(all(update\_length==0,2),:) = [];

%%

% 加精计算-0.3 - -0.5的区间，第二次迭代

% 计算了每次抛物线上下移动相等的距离所产生的新的伸长量

surface\_1 = surface.'; o = zeros(2226,2); rad = zeros(2226,2); for i = 1:2226

% 处理第二问，需要将旋转坐标轴后的点坐标带入

x = inv\_r\_base\_axis(i,1); y = inv\_r\_base\_axis(i,2); z = inv\_r\_base\_axis(i,3);

end

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1));

for i = 2:length(unique(rad(:,1))) if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 能够满足正负0.6的顶点移动距离条件的顶点移动距离

success\_dl = zeros(num,1); success\_index = 1;

fail\_up = 0;

fail\_down = 0;

max\_move\_length = zeros(num,0); min\_move\_length = zeros(num,0); begin = 0.3;

pace = 0.01;

final = 0.5;

num = ((final - begin)/pace + 1)\*2; update\_length = zeros(2226,num); sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

figure;

for dl = begin:pace:final fail\_down = 0;

x = (-500:500);

% 顶点向下移动

z2 = z0 - dl;

down\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue;

else

z\_line = tan(unique\_rad(i))\*x;

% plot(x,z\_line); syms x\_2;

eqn = tan(unique\_rad(i))\*x\_2 == x\_2^2/(4\*(F+dl))+z2;

% 联立方程

S = solve(eqn, x\_2); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) down\_result(1,i) = S\_double(2,1);

else

down\_result(1,i) = S\_double(1,1);

end

end

end

down\_result(down\_result==0) = [];

% 促动器运动距离有超出范围的，失败, 继续下一次循环

print\_down = zeros(length(down\_result),1); for i = 1:length(down\_result)

print\_down(i,1) = down\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(o(:,1) - unique\_angle(i)) < 0.0001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32((dl - begin)/pace + 1)) = abs(down\_result(i)/cos(unique\_rad(i)));

end

sum(int32((dl - begin)/pace + 1)) = sum(int32((dl - begin)/pace + 1)) + abs(print\_down(i,1));

if abs(print\_down(i,1)) > 0.6 fail\_down = 1;

break;

end

end

% 假如促动器全部满足条件，视为成功

if fail\_down == 0 success\_dl(success\_index,1) = -dl; success\_index = success\_index + 1;

plot(down\_result(1,2:end), -print\_down(2:end,1)); hold on

max\_move\_length(int32((dl - begin)/pace + 1),1) = max(print\_down(2:end,1)); min\_move\_length(int32((dl - begin)/pace + 1),1) = min(print\_down(2:end,1));

end

% 将成功指标置为成功

fail\_down = 0;

end hold off

%%

update\_length(all(update\_length==0,2),:) = [];

%%

% 加精计算-0.3 - -0.5的区间，第二次迭代

% 计算了每次抛物线上下移动相等的距离所产生的新的伸长量

surface\_1 = surface.'; o = zeros(2226,2); rad = zeros(2226,2);

for i = 1:2226

% 处理第二问，需要将旋转坐标轴后的点坐标带入

x = inv\_r\_base\_axis(i,1); y = inv\_r\_base\_axis(i,2); z = inv\_r\_base\_axis(i,3);

end

beta\_d = asind(z/sqrt(x^2+y^2+z^2)); theta\_d = atand(y/x);

beta = asin(z/sqrt(x^2+y^2+z^2)); theta = atan(y/x);

o(i,1) = real(beta\_d); o(i,2) = real(theta\_d); rad(i,1) = real(beta); rad(i,2) = real(theta);

z0 = -300.4;

F = 139.8;

unique\_rad = unique(rad(:,1)); unique\_angle = unique(o(:,1)); for i = 2:length(unique(rad(:,1)))

if abs(unique\_rad(i)) < pi/3

% 由于光照区域角度是60度，所以大于一定角度的点直接舍去,根据角度索引筛出符合条件的点

% 将不在光照区域内的角度除去

unique\_rad(i) = 0;

unique\_angle(i) = 0; continue;

end

end

% 去除不满足的点unique\_rad(unique\_rad == 0) = []; unique\_angle(unique\_angle == 0) = [];

% 能够满足正负0.6的顶点移动距离条件的顶点移动距离

success\_dl = zeros(num,1); success\_index = 1;

fail\_up = 0;

fail\_down = 0;

max\_move\_length = zeros(num,0); min\_move\_length = zeros(num,0); begin = 0.3;

pace = 0.01;

final = 0.5;

num = ((final - begin)/pace + 1)\*2; update\_length = zeros(2226,num); sum = zeros(1,num);

show\_length = zeros(1,length(unique\_rad));

figure;

for dl = begin:pace:final fail\_down = 0;

x = (-500:500);

% 顶点向下移动

z2 = z0 - dl;

down\_result = zeros(1,length(unique\_rad)); for i = 1:length(unique\_rad)

if unique\_rad(i) == -pi/2 && unique\_rad(i) == pi/2

% 垂直的线不画出

continue;

else

z\_line = tan(unique\_rad(i))\*x;

% plot(x,z\_line); syms x\_2;

eqn = tan(unique\_rad(i))\*x\_2 == x\_2^2/(4\*(F+dl))+z2;

% 联立方程

S = solve(eqn, x\_2); S\_double = double(S);

if abs(S\_double(1,1)) > abs(S\_double(2,1)) down\_result(1,i) = S\_double(2,1);

else

down\_result(1,i) = S\_double(1,1);

end

end

end

down\_result(down\_result==0) = [];

% 促动器运动距离有超出范围的，失败, 继续下一次循环

print\_down = zeros(length(down\_result),1); for i = 1:length(down\_result)

print\_down(i,1) = down\_result(i)/cos(unique\_rad(i)) + z0;

% 检索当前角度的主索节点

angle\_x = find(abs(o(:,1) - unique\_angle(i)) < 0.0001); for j = 1:length(angle\_x)

update\_length(angle\_x(j,1),int32((dl - begin)/pace + 1)) = abs(down\_result(i)/cos(unique\_rad(i)));

end

sum(int32((dl - begin)/pace + 1)) = sum(int32((dl - begin)/pace + 1)) + abs(print\_down(i,1));

if abs(print\_down(i,1)) > 0.6 fail\_down = 1;

break;

end

end

% 假如促动器全部满足条件，视为成功

if fail\_down == 0

end

success\_dl(success\_index,1) = -dl; success\_index = success\_index + 1;

plot(down\_result(1,2:end), -print\_down(2:end,1)); hold on

max\_move\_length(int32((dl - begin)/pace + 1),1) = max(print\_down(2:end,1)); min\_move\_length(int32((dl - begin)/pace + 1),1) = min(print\_down(2:end,1));

% 将成功指标置为成功

fail\_down = 0;

end hold off

%%

update\_length(all(update\_length==0,2),:) = [];

%%

% 尝试将问题二转化为问题一，生成旋转矩阵，运行第二问的程序首先需要运行该脚本

beta\_2 = 78.169;

alpha\_2 =36.795;

r\_mat = [sind(alpha\_2), -cosd(alpha\_2), 0;

sind(beta\_2)\*cosd(alpha\_2), sind(beta\_2)\*sind(alpha\_2), -cosd(beta\_2); cosd(beta\_2)\*cosd(alpha\_2), cosd(beta\_2)\*sind(alpha\_2), sind(beta\_2)];

r\_base\_axis = zeros(length(base\_axis), 3); inv\_r\_base\_axis = zeros(length(base\_axis), 3); couple = zeros(length(base\_axis), 2);

index = 1;

base\_axis = cell2mat(angle\_1(:,2:4));

% 将原来的基准面向目标光线旋转

for i = 1:length(base\_axis) r\_base\_axis(i,:) = base\_axis(i,:)\*r\_mat;

end

% 将轴旋转至和目标光线平行，

for i = 1:length(base\_axis) inv\_r\_base\_axis(i,:) = base\_axis(i,:)/r\_mat;

end

% 验证是否配对

for i = 1:length(base\_axis) for j = 1:length(base\_axis)

if abs(r\_base\_axis(i,1) - base\_axis(j,1)) < 0.001 && abs(r\_base\_axis(i,2) - base\_axis(j,2))

< 0.001 ...

&& abs(r\_base\_axis(i,3) - base\_axis(j,3)) < 0.001 couple(index,1) = j;

couple(index,2) = i; index = index + 1;

end end

end

## 问题 3 计算程序

% 计算旋转坐标轴后基准球面的反射吸收率指标，方便与抛物面进行比较

axis\_1 = zeros(4300,1); axis\_2 = zeros(4300,1); axis\_3 = zeros(4300,1); fit\_triangle = zeros(4300,3); belong\_index = 0;

for i = 1:4300

a = triangle(i,1); a = string(a);

axis\_1(i,1) = find(strcmp(a, point\_name)); b = triangle(i,2);

b = string(b);

axis\_2(i,1) = find(strcmp(b, point\_name)); c = triangle(i,3);

c = string(c);

axis\_3(i,1) = find(strcmp(c, point\_name));

end

% 找到符合半径条件的反射面板，若三角形三个点全在光照区域内，视作有效顶点

for i = 1:4300

% 当旋转后的基准面的每个三角形都在光照区域时，这个三角形有效

if abs(o(axis\_1(i,1),1)) > 60 && abs(o(axis\_2(i,1),1)) > 60 && abs(o(axis\_3(i,1),1)) > 60 fit\_triangle(i,1) = axis\_1(i,1);

fit\_triangle(i,2) = axis\_2(i,1); fit\_triangle(i,3) = axis\_3(i,1);

end

% belong\_index = 0; end

% 去除不满足条件的三角面板（去除0值）

fit\_triangle = fit\_triangle(any(fit\_triangle,2),:); new\_axis = inv\_r\_base\_axis;

[new\_angle, new\_line, temp\_a, temp\_b, temp\_c, valid\_num, area\_c] = triangle\_calculator(fit\_triangle);

% 计算每个符合条件反射面板的法向量以及法向量与z轴的夹角

axis\_1 = zeros(4300,1); axis\_2 = zeros(4300,1); axis\_3 = zeros(4300,1); fit\_triangle = zeros(4300,3); belong\_index = 0;

for i = 1:4300

a = triangle(i,1); a = string(a);

axis\_1(i,1) = find(strcmp(a, point\_name)); b = triangle(i,2);

b = string(b);

axis\_2(i,1) = find(strcmp(b, point\_name)); c = triangle(i,3);

c = string(c);

axis\_3(i,1) = find(strcmp(c, point\_name));

end

% 找到符合半径条件的反射面板，若三角形三个点全在光照区域内，视作有效顶点

for i = 1:4300

% 等于0代表该点无效

pre\_1 = (new\_axis(axis\_1(i,1),2) == 0); pre\_2 = (new\_axis(axis\_2(i,1),2) == 0); pre\_3 = (new\_axis(axis\_3(i,1),2) == 0);

% mat\_angle\_1 = cell2mat(angle\_1(:,5));

% belong\_index = belong\_index + (abs(mat\_angle\_1(axis\_1(i,1),1)) > 60) + (abs(mat\_angle\_1(axis\_2(i,1),1)) > 60) + (abs(mat\_angle\_1(axis\_3(i,1),1)) > 60); if pre\_1 + pre\_2 + pre\_3 == 0

fit\_triangle(i,1) = axis\_1(i,1); fit\_triangle(i,2) = axis\_2(i,1); fit\_triangle(i,3) = axis\_3(i,1);

end

% belong\_index = 0; end

% 去除不满足条件的三角面板（去除0值）

fit\_triangle = fit\_triangle(any(fit\_triangle,2),:);

[new\_angle, new\_line, temp\_a, temp\_b, temp\_c, valid\_num, area\_r] = triangle\_calculator(fit\_triangle);

## 关键程序

% 根据输入的（n,3）的点列索引，在new\_axis中找到组成三角形的三个点，并且计算：各个三角形的法向量； 各个三角形反射光线方向向量；

% 各个三角形在馈源仓平面的投影，以及所有三角形与馈源仓交集区域的总面积（作为光线吸收率的重要指标）

% A = "A0";

% temp = string(new\_axis(:,1));

% [i,j] = find(strcmp(A, temp));

function [j\_angle, j\_line, temp\_a, temp\_b, temp\_c, valid\_num, sum\_area] = triangle\_calculator(new\_fit\_index)

global new\_axis;

j\_line = zeros(length(new\_fit\_index),3); light\_normal\_vector = zeros(length(new\_fit\_index(:,1)),3);

global temp\_a; global temp\_b; global temp\_c; global new\_axis;

x\_in = zeros(length(new\_fit\_index(:,1)),1); y\_in = zeros(length(new\_fit\_index(:,1)),1); z\_in = zeros(length(new\_fit\_index(:,1)),1); x\_zp = zeros(length(new\_fit\_index(:,1)),3); y\_zp = zeros(length(new\_fit\_index(:,1)),3); z\_p = -160.2;

sum\_area = 0;

valid\_num = 0;

D = zeros(length(new\_fit\_index),1); for i = 1:length(new\_fit\_index(:,1))

% start\_points = [new\_axis(new\_fit\_index(i,1),1:3);new\_axis(new\_fit\_index(i,2),1:3);new\_axis(new\_fit\_index(i,3),1:3)];

% end\_points = [new\_axis(new\_fit\_index(i,1),1:3);new\_axis(new\_fit\_index(i,2),1:3);new\_axis(new\_fit\_index(i,3),1:3)];

% X=[start\_points(:,1) end\_points(:,1)]';

% Y=[start\_points(:,2) end\_points(:,2)]';

% Z=[start\_points(:,3) end\_points(:,3)]';

% line(X, Y ,Z)

% 获取同一个面的三个点坐标

temp\_a(i,:) = new\_axis(new\_fit\_index(i,1),1:3); temp\_b(i,:) = new\_axis(new\_fit\_index(i,2),1:3); temp\_c(i,:) = new\_axis(new\_fit\_index(i,3),1:3);

% 求解平面的法向量

j\_line(i, 1) = (temp\_b(i,2) - temp\_a(i,2))\*(temp\_c(i,3) - temp\_a(i,3)) - (temp\_b(i,3) - temp\_a(i,3))\*(temp\_c(i,2) - temp\_a(i,2));

j\_line(i, 2) = (temp\_c(i,1) - temp\_a(i,1))\*(temp\_b(i,3) - temp\_a(i,3)) - (temp\_b(i,1) - temp\_a(i,1))\*(temp\_c(i,3) - temp\_a(i,3));

j\_line(i, 3) = (temp\_b(i,1) - temp\_a(i,1))\*(temp\_c(i,2) - temp\_a(i,2)) - (temp\_c(i,1) - temp\_a(i,1))\*(temp\_b(i,2) - temp\_a(i,2));

if j\_line(i,3) < 0

j\_line(i,:) = -j\_line(i,:);

end

D(i,1) = - (j\_line(i, 1)\*temp\_a(i,1) + j\_line(i, 2)\*temp\_a(i,2) + j\_line(i, 3)\*temp\_a(i,3));

end

% 求解法向量和z轴的夹角

j\_angle = zeros(length(new\_fit\_index(:,1)),1); for i = 1:length(new\_fit\_index(:,1))

j\_angle(i,1) = acosd(abs(j\_line(i,3)/sqrt(j\_line(i,1)^2 + j\_line(i,2)^2 + j\_line(i,3)^2)));

end

% 求解反射光线的方向

for i = 1:length(new\_fit\_index(:,1))

light\_normal\_vector(i,1) = j\_line(i, 1); light\_normal\_vector(i,2) = j\_line(i, 2);

light\_normal\_vector(i,3) = j\_line(i, 3) - sqrt(j\_line(i,1)^2 + j\_line(i,2)^2 + j\_line(i,3)^2)/(2\*cosd(j\_angle(i)));

% 求解交点

x\_in(i,1) = -(j\_line(i, 1)\*(j\_line(i, 3)\*z\_p + D(i,1))/(j\_line(i, 1)^2 + j\_line(i, 2)^2

+ j\_line(i, 3)\*light\_normal\_vector(i,3)));

y\_in(i,1) = -(j\_line(i, 2)\*(j\_line(i, 3)\*z\_p + D(i,1))/(j\_line(i, 1)^2 + j\_line(i, 2)^2

+ j\_line(i, 3)\*light\_normal\_vector(i,3))); z\_in(i,1) = ((j\_line(i, 1)^2 + j\_line(i, 2)^2)\*z\_p -

light\_normal\_vector(i,3)\*D(i,1))/(j\_line(i, 1)^2 + j\_line(i, 2)^2 + j\_line(i, 3)\*light\_normal\_vector(i,3));

% 通过公式查看交点是否在三角形内部，假如在内部则计数加1

% u = ((v1•v1)(v2•v0)-(v1•v0)(v2•v1)) / ((v0•v0)(v1•v1) - (v0•v1)(v1•v0))

%

% v = ((v0•v0)(v2•v1)-(v0•v1)(v2•v0)) / ((v0•v0)(v1•v1) - (v0•v1)(v1•v0)) v0 = temp\_c(i,:) - temp\_a(i,:);

v1 = temp\_b(i,:) - temp\_a(i,:);

v2 = [x\_in(i,1),y\_in(i,1),z\_in(i,1)] - temp\_a(i,:);

U = (dot(v1, v1)\*dot(v2, v0) - dot(v1, v0)\*dot(v2, v1)) / (dot(v0, v0)\*dot(v1, v1) - dot(v0, v1)\*dot(v1, v0));

V = (dot(v1, v1)\*dot(v2, v0) - dot(v1, v0)\*dot(v2, v1)) / (dot(v0, v0)\*dot(v1, v1) - dot(v0, v1)\*dot(v1, v0));

if U >= 0 && V >= 0 && U+V <= 1 valid\_num = valid\_num + 1;

end

end

% 计算经过三角形的反射光线到zp平面的投影

for i = 1:length(new\_fit\_index(:,1))

x\_zp(i,1) = light\_normal\_vector(i,1)/light\_normal\_vector(i,3)\*(z\_p - temp\_a(i,3)) + temp\_a(i,1);

x\_zp(i,2) = light\_normal\_vector(i,1)/light\_normal\_vector(i,3)\*(z\_p - temp\_b(i,3)) + temp\_b(i,1);

x\_zp(i,3) = light\_normal\_vector(i,1)/light\_normal\_vector(i,3)\*(z\_p - temp\_c(i,3)) + temp\_c(i,1);

y\_zp(i,1) = light\_normal\_vector(i,2)/light\_normal\_vector(i,3)\*(z\_p - temp\_a(i,3)) + temp\_a(i,2);

y\_zp(i,2) = light\_normal\_vector(i,2)/light\_normal\_vector(i,3)\*(z\_p - temp\_b(i,3)) + temp\_b(i,2);

y\_zp(i,3) = light\_normal\_vector(i,2)/light\_normal\_vector(i,3)\*(z\_p - temp\_c(i,3)) + temp\_c(i,2);

end

% 计算所有三角形与馈源仓的交集面积之和

circle\_x = zeros(1000,1); cirlce\_y = zeros(1000,1); theta = 2\*pi/1000;

for i = 1:1000

circle\_x(i,1) = 0.5\*cos((i-1)\*theta); cirlce\_y(i,1) = 0.5\*sin((i-1)\*theta);

end

circle = polyshape(circle\_x, cirlce\_y); for i = 1:length(new\_fit\_index(:,1))

poly1 = polyshape(x\_zp(i,:), y\_zp(i,:)); polyout = intersect(circle, poly1); single\_area = area(polyout);

single\_area = single\_area \* cosd(j\_angle(i))\*light\_normal\_vector(i,3)/... (sind(j\_angle(i))\*sqrt(norm(light\_normal\_vector(i,:))^2 -

light\_normal\_vector(i,3)^2) + cosd(j\_angle(i))\*light\_normal\_vector(i,3)); sum\_area = sum\_area + single\_area;

end

end

附录 **B ”result.xlsx”** 文件部分截图

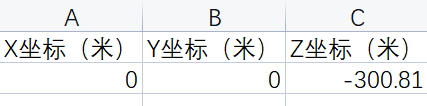
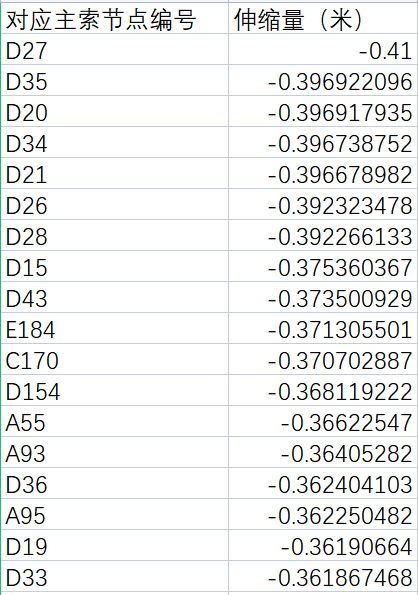


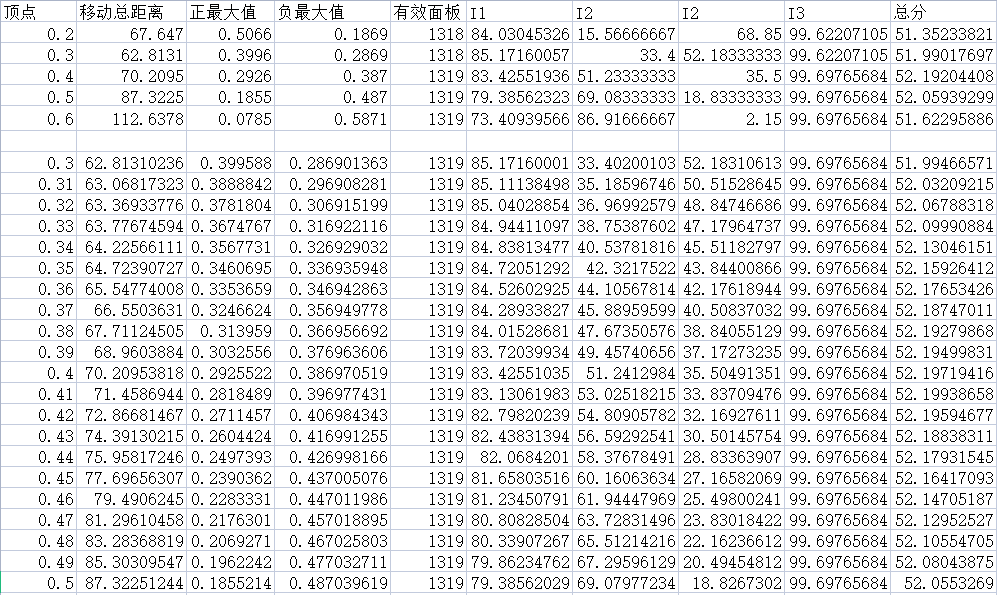
图 **7 result.xlsx**



* 1. **result.xlsx (b) result.xlsx**

图 **8**

# 附录 C 问题 1——各抛物面相关参数与评分表



## 图 9 问题 1 各抛物面相关参数与评分表

附录 **D** 问题 **2**——各抛物面相关参数与评分表

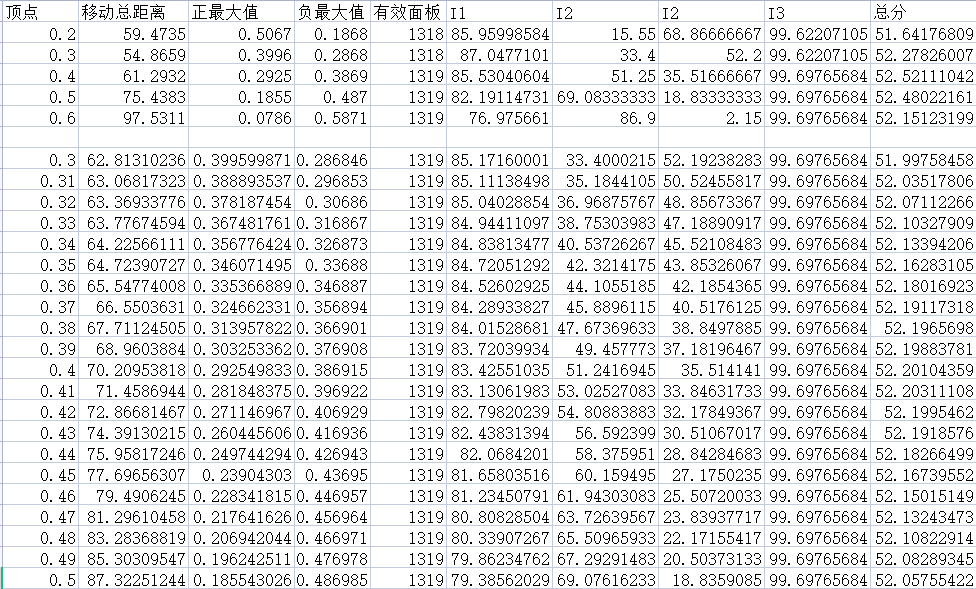


图 **10** 问题 **2** 各抛物面相关参数与评分表