Times New Roman Arial kai[AutoFakeBold]simkai.ttf song[AutoFakeBold]SimSun

4

FASTFAST

$x^2 + y^2 = 549.72(z + 300.63)$

, Zresult.xlsx

0.07%28.1%

-4

section

500FAST

FAST 300 500 300 1 FAST S SC P SC P

1 2226 6525 4300

2

3

4

5 0.07%

6

7 0±0.6m

8 S α β

$S\alpha = 0$,$\beta = 90$

S $\alpha = 36.795, \beta = 78.169$ 300 2 300

section

-
-
-
-
- ,

section

section

4.1

C P R', Gd $300.4 \pm 0.6m(1$, .

$$\frac{x^2}{2p} + \frac{y^2}{2p} = z + d \quad (\frac{p}{2} = d - R')$$

Ω.

d	-
p	
Ω	
Ω'	
Δ	
T	
η	

$u_i = (x_i, y_i, z_i),\ F(u) = 0$[1]

$$\Delta_i = F(u_i) = x_i^2 + y_i^2 - 2p(z_i + d)$$

$$min\sqrt{\frac{\sum_{i\in\Omega} F(u_i)^2}{n}} \quad (n = card\Omega)$$

-4 **1**

4.1.1 A

300mz150m1

$$x_i^2 + y_i^2 \leq 150^2 \quad i \in \Omega$$

Ω d_i,2

$$|d_i| \leq 0.6$$

Matlab+0.43 2 $\alpha = 0°\beta = 90°$

$$x^2 + y^2 = 2p(z + d) \quad p = 274.86; \quad d = 300.63$$

0.46m

-4 **2** **--**

4.2

CZ[1]T,$(x', y', z')^T(x, y, z)^T$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$(x^m)^2 + (y^m)^2 = 4f((z^m)^2 + h^m)$$

h^m [2], $u_j^r = (x_j^r, y_j^r, z_j^r)^T$,$\vec{v} = (\lambda, \varphi, \omega)^T$,

$$\begin{cases} x^2 + y^2 = 4f(z^2 + h^m) \\ \vec{p} = \vec{u_j^r} + t\vec{v} \quad j \in \Omega \end{cases}$$

$u_j^{r'} = (x_j^{r'}, y_j^{r'}, z_j^{r'})^T$,$(x', y', z')^T = T^{-1}u_j^{r'}$,

4.2.1

[2] 300 η,

$$\begin{cases} min\sqrt{\frac{\sum_{j\in\Omega}F(u_j)^2}{n}} \quad (n=card\Omega) \\ max \quad \eta \end{cases}$$

$$s.t.\begin{cases} x_j^2+y_j^2\le 150^2 \quad j\in\Omega \\ |d_j|\le 0.6 \end{cases}$$

η

-4 **3**

4.2.2 (B)

zZCS zα y $\frac{\pi}{2}-\beta$ 4

$\alpha=36.795°\quad\beta=78.169°$,

$$\begin{aligned} T &= R_1\cdot R_2 \\ &= \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix}\begin{pmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$\theta=\beta-\frac{\pi}{2}$

z 300 result.xlsx

Matlab'fsolve', $Levenberg-Marquardt$,

-4 **4**

4 . 3

1. 2.

4 . 3 . 1

$5AB$,CDz$A'B'ABZCDGH$

-4 **5**

4 . 3 . 2 C

Ω',Q_k,$Q_k^s=(x_k^s,y_k^s,z_k^s),(s=1,2,3)Q_k$

$$\tau_{Q_k}:\vec{v}-C\vec{Q}_k^1=\lambda Q_k^1\vec{Q}_k^2+\mu Q_k^1\vec{Q}_k^3$$

$M(m,n,r)$,$M'(m',n',r')$,

$$\begin{cases}\tau_{Q_k}(\frac{m+m'}{2},\frac{n+n'}{2},\frac{r+r'}{2})=0\\ \tau_{Q_k}(M)=\tau_{Q_k}(M')\end{cases}$$

M'Z$\tau_{Q_k}\Pi_{Q_k}$.

$$S_e=\sum_{Q_k\in\Omega'}\int_{P\in\Pi_{Q_k}}\tau_{Q_k}(p)$$

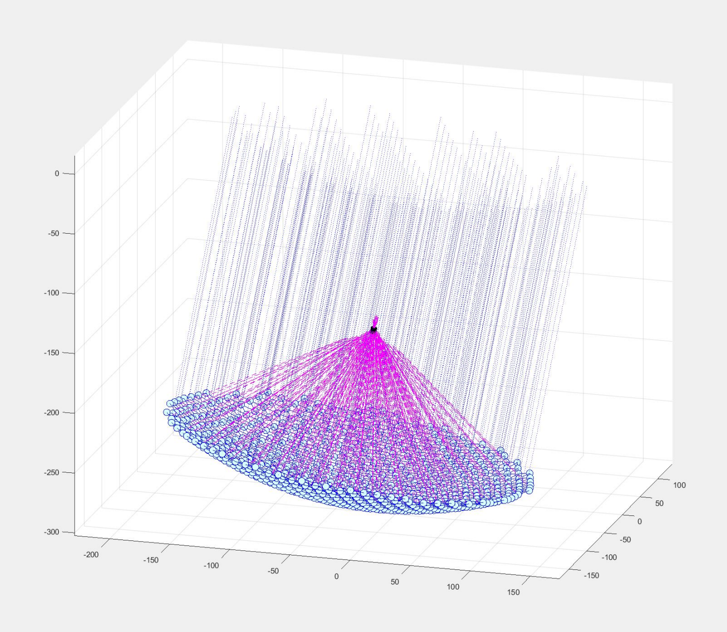
$$\eta=\frac{S_e}{\sum S_{Q_k}}$$

$$\Lambda=\frac{S_e'}{S_a}$$

$S_e'S_a$

MATLAB$\Lambda=5.20\%$.

6) $\eta=28.1\%$. 440.38%



-4 **6 MATLAB**

0.07%

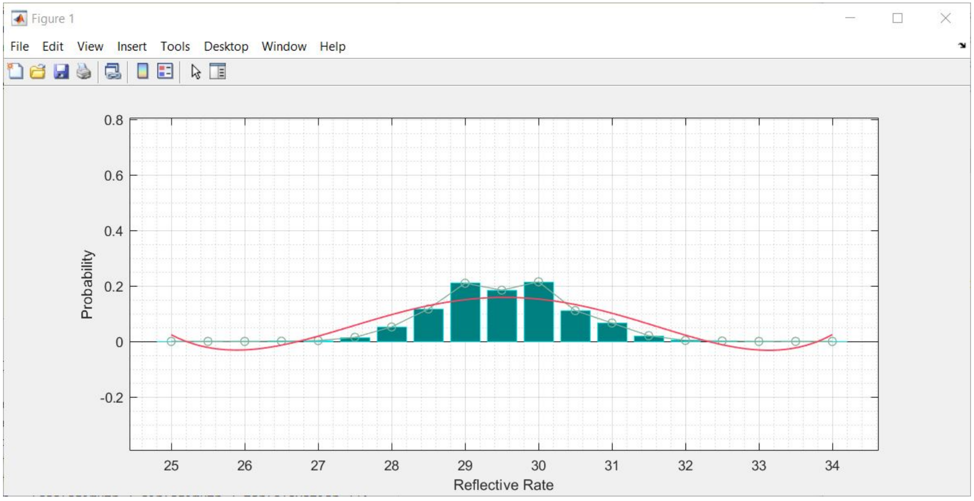
0.07% 0.07% 1360 7)

0.5%[28.6% 29.0%]

4.3.3

28.1% 5.20%28%

Carlo(1350).jpg Carlo(1350).jpg Carlo(1350).jpg Carlo(1350).jpg Carlo(1350).jpg
Carlo(1350).jpg Carlo(1350).jpg Carlo(1350).jpg Carlo(1350).jpg



-4 **7** **xy**

section

1.

2.LM

3.

1. 2.

section

[1],,.(2008).. (01),203-208. doi:CNKI:SUN:GCLX.0.2008-01-035.

[2],,.(2012)..(03),212-217. doi:CNKI:SUN:GCLX.0.2012-03-034.

[3],,,,,,.(2020).. ()(04),117-127. doi:10.19328/j.cnki.1006-1630.2020.04.016.

[4],,,,.(2015).. (04),378-390. doi:10.15940/j.cnki.0001-5245.2015.04.008.

[5]Qiu Y H. A novel design for a giant Arecibo-type spherical radio telescope with an active main reflector [J]. Monthly Notices of the Royal Astronomical Society, 1998, 301(3): 827830.

A T1–matlab

```
function Q1
%===================================
%   data1 column 4-6 are intersect oordinates
%   data1 column 7 is distance
%   data1 column 8 is bool, in = 1, out = 0
%   data1 column 9 is bool, positive = 1, negative = 0;
%
%   surface 4-6 is Centre of Mass coordinate
%   surface 7-9 is normal vector (normalized)
%   surface 10-12 is refraction vector
%===================================

    azimuth   =0;
    elevation = 90;
    delta     = .43;
    h_param   = 300.4 + delta;
    f_param   = h_param-300*(1-0.466);%-0.024;
    paraboloid_Z = @(x,y) (x.^2+y.^2)./4./f_param-h_param;
    numPtIn   = 0;
    peakCords = [];
    rayLength = [150 250];
    randomWalkLimit = 0.07/1.6/100;

    plotFlag = false;
    elseFlag = false;
    interval  = 1;
    inPtArray = [];
    translatedConnection = [];

    azimuth = azimuth/180*pi;
    elevation = elevation/180*pi;

    theta = elevation - pi/2;
    Transformer = [ cos(theta),  0,     sin(theta) ;...
                  0,             1,     0          ;...
```

```
              -sin(theta),   0,     cos(theta) ] *...
             [ cos(azimuth), sin(azimuth), 0   ;...
              -sin(azimuth), cos(azimuth), 0   ;...
              0,             0,            1   ];



data1 =
   csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix1.csv');
data2 =
   csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix2.csv');
%fprintf('data 2 processed, size = %d x %d\n',size(data2,1),size(data2,2));
%fprintf('data 1 processed, size = %d x %d\n',size(data1,1),size(data1,2));

[~,name,~] =
   xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\1.csv');
surface =
   xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3v2.csv');
%[~,connection,~] =
   xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3.csv');
%translatedConnection = zeros(length(connection)-1,3);
%
checkPoint(false);
%
transformData;
%
MSDtraverse(.01);

plotFlag = false;
%
processPointString;
%
loss = MSDloss;
%
revTransformData;

%
maxDist = 0;
```

```
index  = [];
for ii=1:interval:size(data1,1)
    if data1(ii,8) == 1
        if abs(data1(ii,7)) > maxDist && ii ~= 369 && ii ~= 612 && ii ~=
            752 && ii ~= 821
            maxDist = data1(ii,7);
            index = ii;
        end
    end
end

plotFlag = true;
%
plotPointString(data1,data2)
%          300
plot300perimeter;

%
[x_,y_,z_] = revTransform(0,0,-h_param);
peakCords = [x_,y_,z_];

%
fprintf('Max Correction = %.2f.\n',maxDist);
fprintf('Max Correction Index = %d.\n',index);
fprintf('Mean Square Deviation = %.2f\n',loss);
fprintf('Paraboloid Peak = [ %.3f, %.3f, %.3f
    ]\n',peakCords(1),peakCords(2),peakCords(3));
plot3(peakCords(1),peakCords(2),peakCords(3),'-o','Color','b','MarkerSize',15,...
'MarkerFaceColor','#D9FFFF')
fprintf('Verify:
    [820]%.2f->[821]%.2f->[822]%.2f.\n',data1(820,7),data1(821,7),data1(822,7));
%
plotParaboloid;

xlabel('X-axis (m)')
ylabel('Y-axis (m)')
zlabel('Z-axis (m)')
```

```
    %      excel
    write2excel


end
```

B T2-matlab

```
function Q2

%===================================
%   data1 column 4-6 are intersect oordinates
%   data1 column 7 is distance
%   data1 column 8 is bool, in = 1, out = 0
%   data1 column 9 is bool, positive = 1, negative = 0;
%
%   surface 4-6 is Centre of Mass coordinate
%   surface 7-9 is normal vector (normalized)
%   surface 10-12 is refraction vector
%===================================


    azimuth   = 36.795;
    elevation = 78.169;
    delta     = .43;
    h_param   = 300.4 + delta;
    f_param   = h_param-300*(1-0.466);%-0.024;
    paraboloid_Z = @(x,y) (x.^2+y.^2)./4./f_param-h_param;
    numPtIn   = 0;
    peakCords = [];
    rayLength = [150 250];
    randomWalkLimit = 0.07/1.6/100;


    plotFlag = false;
    elseFlag = false;
    interval  = 1;
    inPtArray = [];
```

```
translatedConnection = [];

azimuth = azimuth/180*pi;
elevation = elevation/180*pi;

theta = elevation - pi/2;
Transformer = [ cos(theta),  0,      sin(theta) ;...
               0,             1,      0          ;...
               -sin(theta),   0,      cos(theta) ] *...
             [ cos(azimuth), sin(azimuth), 0   ;...
               -sin(azimuth), cos(azimuth), 0  ;...
               0,             0,             1 ];



data1 =
    csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix1.csv');
data2 =
    csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix2.csv');
%fprintf('data 2 processed, size = %d x %d\n',size(data2,1),size(data2,2));
%fprintf('data 1 processed, size = %d x %d\n',size(data1,1),size(data1,2));

[~,name,~] =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\1.csv');
surface =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3v2.csv');
%[~,connection,~] =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3.csv');
%translatedConnection = zeros(length(connection)-1,3);

%
checkPoint(false);
%
transformData;
%
MSDtraverse(.01);

plotFlag = false;
```

```
%
processPointString;
%
loss = MSDloss;
%
revTransformData;

%
maxDist = 0;
index   = [];
for ii=1:interval:size(data1,1)
    if data1(ii,8) == 1
        if abs(data1(ii,7)) > maxDist && ii ~= 369 && ii ~= 612 && ii ~=
            752 && ii ~= 821
            maxDist = data1(ii,7);
            index = ii;
        end
    end
end

plotFlag = true;
%
plotPointString(data1,data2)
%           300
plot300perimeter;

%
[x_,y_,z_] = revTransform(0,0,-h_param);
peakCords = [x_,y_,z_];

%
fprintf('Max Correction = %.2f.\n',maxDist);
fprintf('Max Correction Index = %d.\n',index);
fprintf('Mean Square Deviation = %.2f\n',loss);
fprintf('Paraboloid Peak = [ %.3f, %.3f, %.3f
    ]\n',peakCords(1),peakCords(2),peakCords(3));
plot3(peakCords(1),peakCords(2),peakCords(3),'-o','Color','b','MarkerSize',15,...
    'MarkerFaceColor','#D9FFFF')
    fprintf('Verify:
        [820]%.2f->[821]%.2f->[822]%.2f.\n',data1(820,7),data1(821,7),data1(822,7));
    %
    plotParaboloid;

    xlabel('X-axis (m)')
    ylabel('Y-axis (m)')
    zlabel('Z-axis (m)')

    %       excel
    write2excel

end
```

C T3-matlab

```
   unction Q3
%===================================
%   data1 column 4-6 are intersect oordinates
%   data1 column 7 is distance
%   data1 column 8 is bool, in = 1, out = 0
%   data1 column 9 is bool, positive = 1, negative = 0;
%
%   surface 4-6 is Centre of Mass coordinate
%   surface 7-9 is normal vector (normalized)
%   surface 10-12 is refraction vector
%===================================

    azimuth   = 36.795;
    elevation = 78.169;
    delta     = .43;
    h_param   = 300.4 + delta;
    f_param   = h_param-300*(1-0.466);%-0.024;
    paraboloid_Z = @(x,y) (x.^2+y.^2)./4./f_param-h_param;
```

```
numPtIn   = 0;
peakCords = [];
rayLength = [150 250];
randomWalkLimit = 0.07/1.6/100;

plotFlag = false;
elseFlag = false;
interval  = 1;
inPtArray = [];
translatedConnection = [];

azimuth = azimuth/180*pi;
elevation = elevation/180*pi;

theta = elevation - pi/2;
Transformer = [ cos(theta),  0,      sin(theta) ;...
              0,             1,      0          ;...
              -sin(theta),   0,      cos(theta) ] *...
             [ cos(azimuth), sin(azimuth), 0   ;...
              -sin(azimuth), cos(azimuth), 0   ;...
              0,             0,            1   ];



data1 =
    csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix1.csv');
data2 =
    csvread('C:\Users\qeng1\Documents\Code\Modelling\data\A\appendix2.csv');
%fprintf('data 2 processed, size = %d x %d\n',size(data2,1),size(data2,2));
%fprintf('data 1 processed, size = %d x %d\n',size(data1,1),size(data1,2));

[~,name,~] =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\1.csv');
surface =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3v2.csv');
%[~,connection,~] =
    xlsread('C:\Users\qeng1\Documents\Code\Modelling\data\A\3.csv');
%translatedConnection = zeros(length(connection)-1,3);
```

D

```

    %
    MonteCarloSimulation(10000);
    %
    ReflecRateTraverse(.001);
    %
    checkPoint(false);
    %
    transformData;
    plotFlag = false;
    %
    processPointString;
    %
    revTransformData;
    plotFlag = true;
    %        300
    plot300perimeter;

    %    0   .07%
    randomizeCords;
    %
    plotMesh;
    %
    calcComNorm;
    %
    generateRay;
    %
    plotFeed;
    %
    calcRecvRate;

end
```

D

```
%                  Q1Q2Q3function                   - end

%===================================
%   data1 column 4-6 are intersect oordinates
%   data1 column 7 is distance
%   data1 column 8 is bool, in = 1, out = 0
%   data1 column 9 is bool, positive = 1, negative = 0;
%
%   surface 4-6 is Centre of Mass coordinate
%   surface 7-9 is normal vector (normalized)
%   surface 10-12 is refraction vector
%===================================


   function MonteCarloSimulation(times)
       range = 25:.5:34;
       clf
       axis([range(1) range(end) 0 1])
       FrequencyArray = zeros(1,length(range));
       ProbabilityArray = zeros(1,length(range));
       SumProbabilityArray = zeros(1,length(range));
       plotFlag = false;
       for i = 1:times

           if mod(i,10) == 0
               i
           end

           checkPoint(false);
           transformData;
           processPointString;
           revTransformData;
           randomizeCords;

           calcComNorm;
           generateRay;
           plotFeed;
           rate = calcRecvRate;
```

```
        FrequencyArray(ceil(2*(rate-range(1)))) =
            FrequencyArray(ceil(2*(rate-range(1)))) + 1;
        ProbabilityArray = FrequencyArray/i;
        for j = 1:length(range)
            SumProbabilityArray(j) = sum(ProbabilityArray(1:j));
        end
        clf
        bar(range,ProbabilityArray,'FaceColor',[0 .5 .5],'EdgeColor',[0 .9
            .9],'LineWidth',.5)
        hold on
        grid on
        grid minor
        xlabel('Reflective Rate')
        ylabel('Probability')

        plot(range,SumProbabilityArray,'o-','Color',[252 157
            154]/255,'LineWidth',.8)
        plot(range,ProbabilityArray,'o-','Color',[131 175
            155]/255,'LineWidth',.8)

%           simpleProbabilityArray = ProbabilityArray;
%           simpleRange = range;
%           for j = length(simpleProbabilityArray):-1:2
%              if  simpleProbabilityArray(j) == 0 &&
   simpleProbabilityArray(j-1) == 0
%                  simpleProbabilityArray(j) = [];
%                  simpleRange(j) = [];
%              end
%           end
%           for j = 1:length(simpleProbabilityArray)-1
%              if  simpleProbabilityArray(j) == 0 &&
   simpleProbabilityArray(j+1) == 0
%                  simpleProbabilityArray(j) = [];
%                  simpleRange(j) = [];
%              end
%           end
```

```
        p = polyfit(range,ProbabilityArray,5);

        x1 = linspace(range(1),range(end));
        y1 = polyval(p,x1);
        plot(x1,y1,’Color’,[255 66 93] / 255,’LineWidth’,1.1);

        pause(.01)
    end
end

function randomizeCords
    for i = 1:size(data1,1)
        if data1(i,8) == 1
            azimuth_ = rand*2*pi;
            elevation_ = (rand-0.5)*2*pi;
            vector = [cos(azimuth_),sin(azimuth_),tan(elevation_)];
            randomV =
                vector/norm(vector)*randomWalkLimit*rand*norm(data1(1,4:6)-data1(2,4:6));
            data1(i,4:6) = data1(i,4:6) + randomV;
        end
    end
end

function ReflecRateTraverse(interval_)
    plotFlag = false;
    rateArray = [];
    traverseArray = -0.14:interval_:0;
    for deltaH = traverseArray
        fprintf(’Current deltaH = %.3f\n’,deltaH);
        f_param = h_param-300*(1-0.466)+deltaH;
        checkPoint(false);
        transformData;
        processPointString;
        revTransformData;
        plot300perimeter;

        plotMesh;
        calcComNorm;
        generateRay;
        plotFeed;
        rate = calcRecvRate;
        rateArray = [rateArray,rate];
    end
    clf
    plot(traverseArray,rateArray);
    xlabel(’Correction on Z-axis’);
    ylabel(’Reflective Rate’);
    axis auto
end

function rate = calcRecvRate
    t=tic;
    v = [cos(azimuth) sin(azimuth) tan(elevation)];
    feedCentre = - v/norm(v) * 160.2;
    options =
        optimoptions(’fsolve’,’Display’,’none’,’Algorithm’,’levenberg-marquardt’);
    rayInCount = 0;
    rayOutCount = 0;

    for i = 1:size(surface,1)
        if surface(i,10) ~= 0
            S = fsolve(@equations,[0,0,0],options);
            % plot3(S(1),S(2),S(3),’o--r’);
            dist2centre = norm(S-feedCentre);
            if dist2centre <= .5
                rayInCount = rayInCount + 1;
            else
                rayOutCount = rayOutCount + 1;
            end
        end
    end
    toc(t)
    rate = rayInCount/rayOutCount*100;
    fprintf('Ray in count = %d, Ray out count =
        %d.\n',rayInCount,rayOutCount);
    fprintf('Reflective Rate = %.2f %%\n',rate);
    function e = equations(x)
        e(1) = dot(v,[x(1),x(2),x(3)]-feedCentre);
        e(2) =
            (x(1)-surface(i,4))*surface(i,11)-(x(2)-surface(i,5))*surface(i,10);
        e(3) =
            (x(1)-surface(i,4))*surface(i,12)-(x(3)-surface(i,6))*surface(i,10);
    end
end

function calcComNorm
    t=tic;
    for i = 1:size(surface,1)
        if sum(ismember(surface(i,1:3),inPtArray)) == 3
            cord1 = data1(surface(i,1),4:6);
            cord2 = data1(surface(i,2),4:6);
            cord3 = data1(surface(i,3),4:6);
            com = (cord1+cord2+cord3)/3;
            surface(i,4:6) = com;
            % calculate normal vector
            vector1 = cord1 - cord2;
            vector2 = cord2 - cord3;
            vecZ = 1;
            options =
                optimoptions('fsolve','Display','none','Algorithm','levenberg-marquardt');
            S = fsolve(@equations,[0,0,0],options);
            vecX = S(1);
            vecY = S(2);
            normalizedV = [vecX vecY vecZ]/norm([vecX vecY vecZ]);
            surface(i,7:9) = normalizedV;
        end
    end
    toc(t)
    function e = equations(x)
        e(1) = dot([x(1),x(2),vecZ],vector1);
```

```
       e(2) = dot([x(1),x(2),vecZ],vector2);
    end
end

function generateRay
    incidenceV = [cos(azimuth) sin(azimuth) tan(elevation)] /
       norm([cos(azimuth) sin(azimuth) tan(elevation)]);
    for i = 1:5:size(surface,1)
      if sum(ismember(surface(i,1:3),inPtArray)) == 3
          refractionV = incidenceV - 2 *
             dot(incidenceV,surface(i,7:9))*surface(i,7:9);
          refractionV = - refractionV / norm(refractionV);
          surface(i,10:12) = refractionV;
          % plot incidence ray
          if plotFlag && 1
             plot3([surface(i,4),surface(i,4)+rayLength(2)*incidenceV(1)],...
                 [surface(i,5),surface(i,5)+rayLength(2)*incidenceV(2)],...
                 [surface(i,6),surface(i,6)+rayLength(2)*incidenceV(3)],’:b’);
          end
          % plot refraction ray
          if plotFlag
             plot3([surface(i,4),surface(i,4)+rayLength(1)*refractionV(1)],...
                 [surface(i,5),surface(i,5)+rayLength(1)*refractionV(2)],...
                 [surface(i,6),surface(i,6)+rayLength(1)*refractionV(3)],’--m’);
          end
      end
    end
end

function plotFeed
    if plotFlag
       r = 2.5;
       valueX = @(z) z*cos(azimuth)/tan(elevation);
       valueY = @(z) z*sin(azimuth)/tan(elevation);
       valueZ = -157.5;
       X1 = [valueX(valueZ) valueY(valueZ) valueZ];
       valueZ = -162.5;
```

```
        X2 = [valueX(valueZ) valueY(valueZ) valueZ];
        length_cyl=norm(X2-X1);
        [x,y,z]=cylinder(r,100);
        z=z*length_cyl;
        %
        hold on;
        cylinderHdl=mesh(x,y,z);
        %
        unit_V=[0 0 1];
        angle_X1X2=acos(dot( unit_V,(X2-X1) )/( norm(unit_V)*norm(X2-X1))
            )*180/pi;
        %
        axis_rot=cross(unit_V,(X2-X1));
        %
        if angle_X1X2~=0 % Rotation is not needed if required direction is
            along X
            rotate(cylinderHdl,axis_rot,angle_X1X2,[0 0 0])
        end
        %
        set(cylinderHdl,'XData',get(cylinderHdl,'XData')+X1(1))
        set(cylinderHdl,'YData',get(cylinderHdl,'YData')+X1(2))
        set(cylinderHdl,'ZData',get(cylinderHdl,'ZData')+X1(3))
        %
        set(cylinderHdl,'FaceColor','k')
        set(cylinderHdl,'EdgeAlpha',0)
        alpha(cylinderHdl,1);
    end
end

function plotMesh
    %plot3(0,0,-200);
    hold on
    shading interp
    axis equal
    grid on
    if plotFlag
        for i = 1:size(surface,1)
```

```
            if
                abs(data1(surface(i,1),6)*data1(surface(i,2),6)*data1(surface(i,3),6))
                ~= 0
                plot3([data1(surface(i,1),4),data1(surface(i,2),4)],...
                    [data1(surface(i,1),5),data1(surface(i,2),5)],
                    [data1(surface(i,1),6),data1(surface(i,2),6)],'b');
                plot3([data1(surface(i,2),4),data1(surface(i,3),4)],...
                    [data1(surface(i,2),5),data1(surface(i,3),5)],...
                    [data1(surface(i,2),6),data1(surface(i,3),6)],'b');
                plot3([data1(surface(i,1),4),data1(surface(i,3),4)],...
                    [data1(surface(i,1),5),data1(surface(i,3),5)],...
                    [data1(surface(i,1),6),data1(surface(i,3),6)],'b');
            end
        end
        for i = 1:size(data1,1)
            if data1(i,8) == 1
                plot3(data1(i,4),data1(i,5),data1(i,6),'o','Color','b','MarkerSize',9,...
                'MarkerFaceColor','#D9FFFF');
            else
                %plot3(data1(i,1),data1(i,2),data1(i,3),'o','Color','m');
            end
        end
    end
end

function translation
    for i = 1:size(data1,1)
        nodeName = name{i+1,1};
        for j = 2:length(surface)
            for k =1:3
                if isequal(surface{j,k},nodeName)
                    translatedConnection(j-1,k) = i;
                end
            end
        end
        i
    end
    xlswrite('4.xlsx',translatedConnection);
end

function write2excel
        results = cell(length(inPtArray)+1,8);
        for j = 2:length(inPtArray)+1
            results{j,1} = name{inPtArray(j-1)+1,1};
            if data1(inPtArray(j-1),9) == 1
                results{j,2} = ['+'
                    num2str(round(data1(inPtArray(j-1),7),4))];
            else
                results{j,2} = ['-'
                    num2str(round(data1(inPtArray(j-1),7),4))];
            end
            results{j,3} = round(data1(inPtArray(j-1),4),4);
            results{j,4} = round(data1(inPtArray(j-1),5),4);
            results{j,5} = round(data1(inPtArray(j-1),6),4);
        end
        results{1,1} = '                       ';
        results{1,2} = '                 ';
        results{1,3} = '             X             ';
        results{1,4} = '             Y             ';
        results{1,5} = '             Z             ';
        results{1,6} = '          X          ';
        results{2,6} = round(peakCords(1),4);
        results{1,7} = '          Y          ';
        results{2,7} = round(peakCords(2),4);
        results{1,8} = '          Z          ';
        results{2,8} = round(peakCords(3),4);

        xlswrite('results.xlsx',results);
    end

function checkPoint(tfFlag)
   for i=1:interval:size(data1,1)
        m = [data1(i,1),data1(i,2),data1(i,3)];
        if ptInOrOut(m,tfFlag)
```

```
           numPtIn = numPtIn + 1;
           data1(i,8) = 1;
           inPtArray = [inPtArray,i];
       else
           data1(i,8) = 0;
       end
   end
end

function transformData
    for k=1:size(data1,1)
       [xp,yp,zp] = transform(data1(k,1),data1(k,2),data1(k,3));
       data1(k,1:3) = [xp,yp,zp];
    end
    for k=1:size(data2,1)
       [xp,yp,zp] = transform(data2(k,1),data2(k,2),data2(k,3));
       data2(k,1:3) = [xp,yp,zp];
    end
   for k=1:size(data2,1)
       [xp,yp,zp] = transform(data2(k,4),data2(k,5),data2(k,6));
       data2(k,4:6) = [xp,yp,zp];
   end
end

function revTransformData
    for k=1:size(data1,1)
       [xp,yp,zp] = revTransform(data1(k,1),data1(k,2),data1(k,3));
       data1(k,1:3) = [xp,yp,zp];
    end
    for k=1:size(data1,1)
       [xp,yp,zp] = revTransform(data1(k,4),data1(k,5),data1(k,6));
       data1(k,4:6) = [xp,yp,zp];
    end
    for k=1:size(data2,1)
       [xp,yp,zp] = revTransform(data2(k,1),data2(k,2),data2(k,3));
       data2(k,1:3) = [xp,yp,zp];
    end
```

```
      for k=1:size(data2,1)
           [xp,yp,zp] = revTransform(data2(k,4),data2(k,5),data2(k,6));
           data2(k,4:6) = [xp,yp,zp];
      end
end

function [x_,y_,z_] = transform(x,y,z)
   out = Transformer * [x,y,z]';
   x_ = out(1);y_ = out(2); z_ = out(3);
end

function [x_,y_,z_] = revTransform(x,y,z)
   out = inv(Transformer) * [x,y,z]';
   x_ = out(1);y_ = out(2); z_ = out(3);
end

function verify
    fprintf('Verify:
       [368]%.2f->[369]%.2f->[370]%.2f.\n',data1(368,7),data1(369,7),data1(370,7));
    fprintf('Verify:
       [611]%.2f->[612]%.2f->[613]%.2f.\n',data1(611,7),data1(612,7),data1(613,7));
end

function MSDtraverse(interval_)
    plotFlag = false;
    lossArray = [];
    distArray = [];
    clf
    for deltaV = -.6:interval_:.6
       fprintf('Current delta = %.2f\n',deltaV);
       delta = deltaV;
       h_param = 300.4 + delta;
       f_param = h_param-300*(1-0.466);

       processPointString;

       maxDist = 0;
```

```
        index  = [];
        for k=1:interval:size(data1,1)
            if data1(k,8) == 1
                if abs(data1(k,7)) > maxDist && k ~= 369 && k ~= 612 && k ~=
                    752 && k ~= 821
                    maxDist = data1(k,7);
                    index = k;
                end
            end
        end
        fprintf(’Max Correction = %.2f.\n’,maxDist);
        %fprintf(’Max Correction Index = %d.\n’,index);
        loss = MSDloss;
        lossArray = [lossArray,loss];
        distArray = [distArray,maxDist];
    end
    [AX,~,~] =
        plotyy(-.6:interval_:.6,lossArray,-.6:interval_:.6,distArray);
    title(’Correction offset on Z-axis’)
    set(get(AX(1),’Ylabel’),’String’,’Mean Squre Deviation’)
    set(get(AX(2),’Ylabel’),’String’,’Max Correction (m)’)
    grid on
end

function loss = MSDloss
    loss = 0;
    delta_c = [];
    for i=1:interval:size(data1,1)
    %
        if i == 369 || i == 612 || i == 752 || i == 821
            dist = data1(i+1,7);
            else
            dist = data1(i,7);
        end
        delta_c = [delta_c,dist];
    end
    loss = sqrt((delta_c*delta_c’)/numPtIn);
    %fprintf('Mean Square Deviation = %.2f.\n',loss);
end

function calcIntersectDist(index)
    % solve for the foot of the perpendicular point
    function e = equations(x)
        e(1) = (x(1)-data2(index,1))*(data2(index,2)-data2(index,5)) -
            (data2(index,1)-data2(index,4))*(x(2)-data2(index,2));
        e(2) = (x(2)-data2(index,2))*(data2(index,3)-data2(index,6)) -
            (data2(index,2)-data2(index,5))*(x(3)-data2(index,3));
        e(3) = x(1)^2+x(2)^2-4*f_param*(x(3)+h_param);
    end
    options = optimoptions('fsolve','Display','none');
    intersect = fsolve(@equations,data1(index,1:3),options);
    data1(index,4:6) = intersect;
    data1(index,7) = norm(data1(index,1:3)-intersect);
    if norm(data1(index,1:3)) > norm(data1(index,4:6))
        data1(index,9) = 1; % positive correction
    else
        data1(index,9) = 0; % negative correction
    end
    if plotFlag
        plot3(intersect(1),intersect(2),intersect(3),'*k');
        plot3([data1(index,1),data1(index,4)],[data1(index,2),data1(index,5)],...
        [data1(index,3),data1(index,6)],'b');
    end
end

function processPointString
    t = tic;
    %plot3(0,0,-200);
    hold on
    shading interp
    axis equal
    grid on
    for i=1:interval:size(data1,1)%*~testMode+100*testMode
        m = [data1(i,1),data1(i,2),data1(i,3)];
        if data1(i,8) == 1
            if plotFlag
                % point
                plot3(data1(i,1),data1(i,2),data1(i,3),'o--r');
                % string
                plot3([data2(i,1),data2(i,4)+(data2(i,4)-data2(i,1))*0],...
                [data2(i,2),data2(i,5)+(data2(i,5)-data2(i,2))*0],...
                [data2(i,3),data2(i,6)+(data2(i,6)-data2(i,3))*0],'--b');
            end
            % intersection point
            calcIntersectDist(i);
        else
            if elseFlag
                if plotFlag
                    % point
                    plot3(data1(i,1),data1(i,2),data1(i,3),'o--b');
                    % string
                    plot3([data2(i,1),data2(i,4)+(data2(i,4)-data2(i,1))*0],...
                    [data2(i,2),data2(i,5)+(data2(i,5)-data2(i,2))*0],...
                    [data2(i,3),data2(i,6)+(data2(i,6)-data2(i,3))*0],'--b');
                end
                % intersection point
                calcIntersectDist(i);
            end
        end
        %drawnow
        %pause(.01)
    end
    toc(t)
end

function plotPointString(data1_,data2_)
    t = tic;
    optionFlag = true;
    for i=1:interval:size(data1_,1)
        m = [data1_(i,1),data1_(i,2),data1_(i,3)];
        if data1_(i,8) == 1
            if plotFlag
                % point
                plot3(data1_(i,1),data1_(i,2),data1_(i,3),’o--r’);
                % string
                if i == 0
                    plot3([data2_(i,1),data2_(i,4)+(data2_(i,4)-data2_(i,1))*100],...
                    [data2_(i,2),data2_(i,5)+(data2_(i,5)-data2_(i,2))*100],...
                    [data2_(i,3),data2_(i,6)+(data2_(i,6)-data2_(i,3))*100],’--b’);
                else
                    plot3([data2_(i,1),data2_(i,4)+(data2_(i,4)-data2_(i,1))*0],...
                    [data2_(i,2),data2_(i,5)+(data2_(i,5)-data2_(i,2))*0],...
                    [data2_(i,3),data2_(i,6)+(data2_(i,6)-data2_(i,3))*0],’--b’);
                end
                % intersection point
                plot3(data1_(i,4),data1_(i,5),data1_(i,6),’*k’);
                plot3([data1_(i,1),data1_(i,4)],[data1_(i,2),data1_(i,5)],...
                [data1_(i,3),data1_(i,6)],’m’);
            end
        else
            % point
            if elseFlag
                if plotFlag
                    plot3(data1_(i,1),data1_(i,2),data1_(i,3),’o--b’);
                    % string
                    plot3([data2_(i,1),data2_(i,4)+(data2_(i,4)-data2_(i,1))*0],...
                    [data2_(i,2),data2_(i,5)+(data2_(i,5)-data2_(i,2))*0],...
                    [data2_(i,3),data2_(i,6)+(data2_(i,6)-data2_(i,3))*0],’--b’);
                    % intersection point
                    plot3(data1_(i,4),data1_(i,5),data1_(i,6),’*k’);
                    plot3([data1_(i,1),data1_(i,4)],...
                    [data1_(i,2),data1_(i,5)],...
                    [data1_(i,3),data1_(i,6)],’m’);
                end
            end
        end
        if optionFlag
            if plotFlag
```

```
                hold on
                shading interp
                axis equal
                grid on
            end
            optionFlag = false;
        end
    end
    toc(t)
end

function plotParaboloid
    if plotFlag
        arrayX = -300:1:300;
        arrayY = -300:1:300;
        [x,y]  = meshgrid(arrayX,arrayY);
        z = paraboloid_Z(x,y);
        meshHdl = mesh(x,y,z);
        alpha(meshHdl,.2);
    end
end

function output=rid369(input)
    output=input;
    output(369,:)=output(370,:);
end

function plot300perimeter
    % https://blog.csdn.net/weixin_44986426/article/details/114868368
    if plotFlag
        r = 150;
        valueX = @(z) z*cos(azimuth)/tan(elevation);
        valueY = @(z) z*sin(azimuth)/tan(elevation);
        valueZ = -180;
        X1 = [valueX(valueZ) valueY(valueZ) valueZ];
        valueZ = -300;
        X2 = [valueX(valueZ) valueY(valueZ) valueZ];
        length_cyl=norm(X2-X1);
        [x,y,z]=cylinder(r,100);
        z=z*length_cyl;
        %
        hold on;
        cylinderHdl=mesh(x,y,z);
        %
        unit_V=[0 0 1];
        angle_X1X2=acos(dot( unit_V,(X2-X1) )/( norm(unit_V)*norm(X2-X1))
            )*180/pi;
        %
        axis_rot=cross(unit_V,(X2-X1));
        %
        if angle_X1X2~=0 % Rotation is not needed if required direction is
            along X
            rotate(cylinderHdl,axis_rot,angle_X1X2,[0 0 0])
        end
        %
        set(cylinderHdl,'XData',get(cylinderHdl,'XData')+X1(1))
        set(cylinderHdl,'YData',get(cylinderHdl,'YData')+X1(2))
        set(cylinderHdl,'ZData',get(cylinderHdl,'ZData')+X1(3))
        %
        set(cylinderHdl,'FaceColor','k')
        set(cylinderHdl,'EdgeAlpha',0)
        alpha(cylinderHdl,0.3);
    end
end

function res = ptInOrOut(pt,tfFlag)
    if tfFlag
        v = [0 0 1];
    else
        v = [cos(azimuth) sin(azimuth) tan(elevation)];
    end
    if norm(cross(pt,v))/norm(v)>150
        res = false;
    else
```

```
        res = true;
    end
end
```