飞行器轨道动力学中的数学方法作业

1121820103　飞行器设计与工程　赵超

1. 利用轨道六根数和位置速度的转化关系（或 Lagrange 系数法），编写 Matlab 程序求解初值问题，并画出 2 个周期内的轨迹曲线。初始时刻卫星轨道参数 $E_2 = [a_2, e_2, i_2, \omega_2, \Omega_2, \varphi_2] = [10000\text{km}, 0, 10°, 20°, 30°, 60°]$ 。（必选）

解： 通过求解 Kepler 方程解初值问题。求解步骤如下：

(1) 已知初始时刻的真近点角 φ_0，由式(1)求出初始时刻的偏近点角 ψ_0 ;
(2) 由式(2)得到初始时刻 t_0 的平近点角 M_0 ;
(3) 由开普勒方程(3)，求得时刻 t 的偏近点角 ψ ，最后用方程得到时刻 t 的真近点角 φ 。

其中，真近点角 φ 与偏近点角 ψ 之间的转换关系为

$$\psi = 2\arctan\left(\sqrt{\frac{1-e}{1+e}}\tan\left(\frac{\varphi}{2}\right)\right) \tag{1}$$

飞行器平近点角为

$$M = \psi - e\sin(\psi) \tag{2}$$

描述飞行器位置和时间关系的开普勒方程为

$$n_0(t - t_0) = M - M_0 \tag{3}$$

绕地轨道示意如图 1所示，真近点角随时间变化曲线如图 2所示。

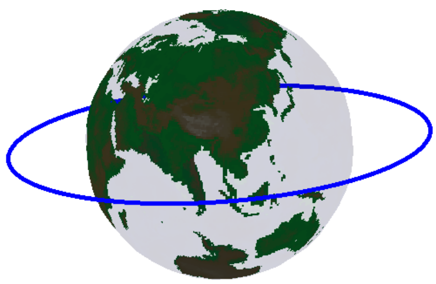

Figure 1: 绕地轨道

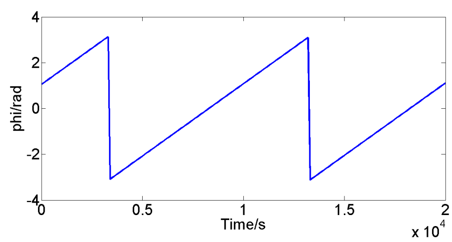

Figure 2: 真近点角随时间变化曲线

2. 利用长半轴长、或横向离心率、或飞行方向角求解固定时间 Lambert 问题，并应用到下面的轨道交会问题算例中。在初始时刻，目标卫星轨道参数 $E_2 = [a_2, e_2, i_2, \omega_2, \Omega_2, \varphi_2] = [10000\text{km}, 0, 10°, 20°, 30°, 60°]$，追踪卫星轨道参数 $E_1 = [a_1, e_1, i_1, \omega_1, \Omega_1, \varphi_1] = [9998\text{km}, 0, 9.95°, 20°, 30°, 59.9°]$，要求在 $t_f = 3000\text{s}$ 后完成轨道交会，利用 Matlab 编程求解需要的两次脉冲速度向量。（可选）

解： 采用长半轴长作为转移时间方程的自变量函数求解 Lambert 问题。椭圆轨道 Lagrange 时间方程可以写为如下形式

$$\sqrt{\mu}\Delta t = a^{3/2}[\alpha - \beta - (\sin\alpha - \sin\beta)] \tag{4}$$

其中，

$$\cos\alpha = 1 - \frac{s}{a}, \cos\beta = 1 - \frac{s-c}{a}, s = \frac{r_1 + r_2 + c}{2}$$

当 $a = s/2$ 时求得最小能量轨道时间 $t(a_{min})$, 若 $t(a_{min}) > t_f$, 则转移轨道为 short-path 轨道，否则为 long-path 轨道。

将 $t_f = \Delta t$ 带入公式(4)求解即可得到转移轨道的半长轴，由 Matlab计算可得 a 这样，我们就将轨道转移时间转换为转移轨道长半轴的单值函数，即 $\Delta t = \Delta t(a)$.

现采用割线法求解长半轴 a 使得 $\Delta t = t_f = 3000s$，a 的更新方程如式 5所示。

$$a_{n+1} = a_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{5}$$

经过多次迭代后，得到满足精度条件的长半轴 a.

现采用 Battin 方法[1][Battin, 1999] 求解初始时刻的速度向量 $\mathbf{v_1}$。

引入参数

$$\lambda s = \sqrt{r_1 r_2} \cos \frac{\theta}{2}$$

$$x^2 = 1 - a_m / a$$

$$\eta^2 = \frac{2a \sin^2 \psi}{s}, \psi = \frac{\alpha - \beta}{2}$$

则 $\mathbf{v_1}$ 矢量可表示为

$$\mathbf{v_1} = \frac{\sqrt{\mu}}{r_1} (\sigma_1 \mathbf{i}_{r1} + \sqrt{p} \mathbf{i}_h \times \mathbf{i}_{r1}) \tag{6}$$

其中，

$$p = \frac{r_1 r_2}{a_m \eta^2} \sin^2(\theta/2) \tag{7}$$

$$\sigma_1 = \frac{1}{\eta \sqrt{a_m}} [2\lambda a_m - r_1(\lambda + x\eta)] \tag{8}$$

将式 7和式 8带入到式 6中，即可得到 $\mathbf{v_1}$ 的表达式

$$\mathbf{v_1} = \frac{1}{\eta} \sqrt{\frac{\mu}{a_m}} \{[2\lambda \frac{a_m}{r_1} - (\lambda + x\eta)] \mathbf{i}_{r1} + \sqrt{\frac{r_2}{r_1}} \sin(\theta/2) \mathbf{i}_h \times \mathbf{i}_{r_1}\} \tag{9}$$

至此，Lambert 问题的基本求解结束。接下来，求解转移轨道偏心率矢量 $\mathbf{e}$，进而解得转移轨道的终止速度矢量 $\mathbf{v_2}$。

$$\mathbf{e} = \frac{1}{\mu} [(v_1^2 - \frac{\mu}{r_1}) \mathbf{r_1} - (\mathbf{r_1} \cdot \mathbf{v_1}) \mathbf{v_1}] \tag{10}$$

$$\mathbf{v_2} = \frac{\mu}{h^2} \mathbf{h} \times (\mathbf{e} + \frac{\mathbf{r_2}}{r_2}) \tag{11}$$

需要的速度脉冲向量 $\Delta \mathbf{v_1} = \mathbf{v_1} - \mathbf{v_{10}}, \Delta \mathbf{v_2} = \mathbf{v_2} - \mathbf{v_{20}}$。

采用实例[2][Curtis, 2005] 验证该算法的该正确性。$\mathbf{r_1} = 5000\mathbf{i} + 10,000\mathbf{j} + 2100\mathbf{k}$(Km), $\mathbf{r_2} = -14,600\mathbf{i} + 2500\mathbf{j} + 7000\mathbf{k}$(Km), 转移时间 $3600s$。采用 MATLAB 仿真得到

$$\mathbf{v_1} = -5.9925\mathbf{i} + 1.9254\mathbf{j} + 3.2456\mathbf{k}(\mathrm{Km/s})$$

$$\mathbf{v_2} = -3.3125\mathbf{i} - 4.1966\mathbf{j} - 0.38529\mathbf{k}(\mathrm{Km/s})$$

与书中通过 Gauss 解法得到的结果相同。

采用该算法解题目二中的算例，发现得到的离心率 $e = 0.999999999790312$，该离心率已经无法对 Kepler 方程迭代求解。求解速度向量如下。

$$\mathbf{v_{10}} = -5.9244\mathbf{i} - 2.1738\mathbf{j} + 0.1904\mathbf{k}(\mathrm{Km/s})$$

$$\mathbf{v_{20}} = -5.9287\mathbf{i} - 2.1636\mathbf{j} + 0.1913\mathbf{k}(\mathrm{Km/s})$$

[1] Richard H. Battin, *An introduction to the mathematics and methods of astrodynamics,1999,298-307*

[2] Howard D. Curtis, *Orbital Mechanics for Engineering Students, Third Edition, example5.2, 254*

$$\mathbf{v_1} = -1.4111\mathbf{i} + 3.9327\mathbf{j} + 0.7208\mathbf{k}(\mathrm{Km/s})$$
$$\mathbf{v_2} = 1.4197\mathbf{i} - 3.9306\mathbf{j} - 0.7258\mathbf{k}(\mathrm{Km/s})$$
$$\Delta\mathbf{v_1} = 4.5133\mathbf{i} + 6.1064\mathbf{j} + 0.5304\mathbf{k}(\mathrm{Km/s})$$
$$\Delta\mathbf{v_2} = 7.3485\mathbf{i} - 1.7671\mathbf{j} - 0.9171\mathbf{k}(\mathrm{Km/s})$$

* * *

问题解决过程中需要用到的 MATLAB 代码附录如下：

- function y = plot_t_phi(E_0) ，题目一主程序，用于绘制图 1和图 2。y 返回 0 时程序正常结束；
- function phi = Init_problem(E_0,t)，用于求解初值问题，给定轨道根数和时刻，输出当前时刻的真近点角；
- function [R V] = E2RV(E)，用于将轨道根数转换为位置和速度的列向量；
- function Visulization(P)，用于可视化一条地球轨道；
- function [V1,V2]=Lambert(R1,R2,t_f)，题目二主程序，用于求解 Lambert 问题；
- function t=x2t(x,s,c,t_f,t_m)，用于 Lambert 问题中的割线法迭代；
- function E = RV2E(R,V)，将轨道某一时刻的速度位置向量转化为轨道根数；

```
function y = plot_t_phi(E_0)
%Used to plot each vector phi versus vector t.
mu = 3.986e5;
T = 2*pi*sqrt(E_0(1)^3/mu);%Period
t = 0:100:ceil(T/100)*100;
phi = zeros(length(t),1);
R_mat = zeros(length(t),3);
V_mat = zeros(length(t),3);
E = E_0;
for jj = 1:length(t)
    phi(jj) = Init_problem(E_0,t(jj));%Init_problem()
    E(6) = phi(jj);
    [R V] = E2RV(E);%E2RV()
    R_mat(jj,:) = R';%Update R and V
    V_mat(jj,:) = V';
end
figure(1);
plot(t,phi);
figure(2);
Visulization(R_mat*1000);%Visulization()
y = 0;%Normal termination.
end
```

```
function phi = Init_problem(E_0,t)
%the main function to solve the initial problem
global n_0
mu = 3.986e5;
n_0 = sqrt(mu/E_0(1)^3);
M_0 = E2M(E_0);
M = n_0*t+M_0;
%M = psi - E(2)*sin(psi);
```

```
Kepler_fun = @(psi)(psi-E_0(2)*sin(psi)-M);
psi = fsolve(Kepler_fun,0);
phi = 2*atan(tan(psi/2)*sqrt((1+E_0(2))/(1-E_0(2))));
end
```

```
function [R V] = E2RV(E)
%convert E(a,e,i,omega,Omega,phi) to R and V(column vectors)
mu = 3.986e5;
Omega = E(5);
omega = E(4);
phi = E(6);
i = E(3);
a = E(1);
e = E(2);
M = [cos(Omega)*cos(omega+phi)-sin(Omega)*sin(omega+phi)*cos(i)
     sin(Omega)*cos(omega+phi)+cos(Omega)*sin(omega+phi)*cos(i)
     sin(omega+phi)*sin(i)];
Coe = a*(1-e^2)/(1+e*cos(phi));
R = Coe.*M;
Coe_prime_phi = a*e*(1-e^2)*sin(phi)/(1+e*cos(phi))^2;
M_prime_phi = [-cos(Omega)*sin(omega+phi)-sin(Omega)*cos(omega+phi)*cos(i)
     -sin(Omega)*sin(omega+phi)+cos(Omega)*cos(omega+phi)*cos(i)
     cos(omega+phi)*sin(i)];
R_prime_phi = Coe_prime_phi.*M+Coe.*M_prime_phi;
phi_prime_t = sqrt(mu)*(1+e*cos(phi))^2/(a*(1-e^2))^1.5;
V = R_prime_phi.*phi_prime_t;
end
```

```
function Visulization(P)
%visulize a geocentric orbit
col_time = length(P(:,1));
P_e = zeros(col_time,3);
CCC = [-1 0 0;0 -1 0;0 0 1];
%generate the position vector in given coordinate, in this script we give ...
    earth coordinate.
for jj = 1:col_time
    P_e(jj,:) = (CCC*[P(jj,1);P(jj,2);P(jj,3)]);
end
plot3(P_e(:,1),P_e(:,2),P_e(:,3),'LineWidth',.5);
hold on
r=6373000;%radius

%visulize the earth
load topo;
[x,y,z]=sphere;
s = surface(r*x,r*y,r*z,'facecolor','texturemap','cdata',topo);
view(0,105);
set(s,'edgecolor','none','facealpha','texture','alphadata',topo);
%set(s,'edgecolor','none');
set(s,'backfacelighting','unlit');
colormap(topomap1);
alpha('direct');
alphamap([0.1;1]);
brighten(.1);
axis off vis3d;
campos([2 13 10]);
camlight;
lighting gouraud;

axis equal
```

```
end
```

```
function [V1,V2]=Lambert(R1,R2,t_f)
%solve the Lambert problem.
%2015-06-06
tol=1e-10;  %Tolerence
mu = 398600;
r1 = norm(R1);
r2 = norm(R2);
C = R2-R1;
c = norm(C);
s = (r1+r2+c)/2;
theta = acos(dot(R1,R2)/(r1*r2));
lambda = sqrt(r1*r2)*cos(theta/2)/s;

%Mimimum-energy orbit
a_m=s/2;
beta_m = acos(1-(s-c)/a_m);
alpha_m = acos(1-s/a_m);
t_m = 1/sqrt(mu)*a_m^1.5*(alpha_m-beta_m-(sin(alpha_m)-sin(beta_m)));

%Initial Value.
x1 = -.99;
x2 = .99;
y1=x2t(x1,s,c,t_f,t_m);%x2t()
y2=x2t(x2,s,c,t_f,t_m);

%Newton iterations
err=1;
i=0;
while (err>tol) && (y1≠y2)
    i=i+1;
    x_new=x2-(x2-x1)*y2/(y2-y1);
    y_new=x2t(x_new,s,c,t_f,t_m);
    x1=x2;
    y1=y2;
    x2=x_new;
    y2=y_new;
    err=abs(x1-x_new);
end
disp(i);
x=x_new;
a=a_m/(1-x^2)

beta=2*asin(sqrt((s-c)/2/a));
alpha=acos(1-s/a);
if t_m < t_f
    % Long path
    alpha = 2*pi-alpha;
end
psi=(alpha-beta)/2;
eta=sqrt(2*a*sin(psi)^2/s);

p=(r1*r2*sin(theta/2)^2)/(a_m*eta^2);     %parameter of the solution
sigma1=(2*lambda*a_m-r1*(lambda+x*eta))/(eta*sqrt(a_m));
ih=cross(R1,R2)/norm(cross(R1,R2));

i_R1 = R1/norm(R1);
V1 = (sqrt(mu)/r1)*(sigma1*i_R1+sqrt(p)*cross(ih,i_R1));
v1 = norm(V1);
e_vec = (1/mu)*((v1^2-mu/r1)*R1-dot(R1,V1)*V1);
e = norm(e_vec)
```

```
h = cross(R1,V1);
V2 = (mu/norm(h)^2)*cross(h,(e_vec+R2/norm(R2)));

% E = RV2E(R1,V1);
% y = plot_t_phi(E);
end
```

```
function t=x2t(x,s,c,t_f,t_m)
%calculate the difference of t for the given variable x.
mu = 3.986e5;
a_m=s/2;
a=a_m/(1-x^2);
beta=2*asin(sqrt((s-c)/2/a));
alpha=2*acos(x);
if t_m < t_f
    % Long path
    alpha = 2*pi-alpha;
end
t=a^1.5*((alpha-sin(alpha))-(beta-sin(beta)))/sqrt(mu);
t = t-t_f;
end
```

```
function E = RV2E(R,V)
%convert vectors R and V to orbit elements.
%Adapted from Orbital Mechanics for Engineering Students, [Curtis, 2005].
mu = 398600;
r = norm(R);
v = norm(V);
vr = dot(R,V)/r;
H = cross(R,V);
h = norm(H);
i = acos(H(3)/h);
N = cross([0 0 1],H);
n = norm(N);
if n ≠ 0
    Omega = acos(N(1)/n);
    if N(2) < 0
        Omega = 2*pi - Omega;
    end
else
    Omega = 0;
end
E = 1/mu*((v^2 - mu/r)*R - r*vr*V);
e = norm(E);
if n ≠ 0
    if e > eps
        omega = acos(dot(N,E)/n/e);
        if E(3) < 0
            omega = 2*pi - omega;
        end
    else
        omega = 0;
    end
else
    omega = 0;
end
if e > eps
    phi = acos(dot(E,R)/e/r);
    if vr < 0
        phi = 2*pi - phi;
```

```
        end
else
        cp = cross(N,R);
        if cp(3) ≥ 0
                phi = acos(dot(N,R)/n/r);
        else
                phi = 2*pi - acos(dot(N,R)/n/r);
        end
end
a = h^2/mu/(1 - e^2);
E = [a e i omega Omega phi];
end
```