

For question1 we use following codes.

clc;

clear all;

theta1 = atan(2.5/2)-pi;

theta2 = pi/2;

theta3 =-atan(1.5/2);

theta\_all = [theta1;theta2;theta3];

L1 = [0,0];

L2 = [4,2];

L3 = [1,4];

pos1 = get\_pos(theta1,theta2,L1,L2);

pos2 = get\_pos(theta2,theta3,L2,L3);

pos3 = get\_pos(theta1,theta3,L1,L3);

Px = (pos1(1)+pos2(1)+pos3(1))/3;

Py = (pos1(2)+pos2(2)+pos3(2))/3;

p = [Px,Py];

len\_each = 100000;

percentage = zeros(1,10);

dis = zeros(10,len\_each);

for j = 1:1:10

sigma = j/10;

inside\_times = 0;

for i=1:1:len\_each

noised\_theta = theta\_all+normrnd(0,sigma,[3,1]);

%get 3 potential pos

pos12 = get\_pos(noised\_theta(1),noised\_theta(2),L1,L2);

pos23 = get\_pos(noised\_theta(2),noised\_theta(3),L2,L3);

pos13 = get\_pos(noised\_theta(1),noised\_theta(3),L1,L3);

c\_x = (pos12(1)+pos23(1)+pos13(1))/3;

c\_y = (pos12(2)+pos23(2)+pos13(2))/3;

c\_hat = [c\_x,c\_y];

S\_all = cal\_tri\_area(pos12,pos23,pos13);

S1 = cal\_tri\_area(p,pos23,pos13);

S2 = cal\_tri\_area(pos12,p,pos13);

S3 = cal\_tri\_area(pos12,pos23,p);

if(abs(S\_all-(S1+S2+S3))<10e-6)

inside\_times = inside\_times+1;

end

dis(j,i) = norm(c\_hat-p);

end

percentage(j) = inside\_times/len\_each;

end

mean\_dis = mean(dis,2);

figure(1)

plot(0.1:0.1:1,percentage);

xlabel("sigma");

ylabel("percentage of times");

title("percentage of times that P was inside the triangle versus sigma");

figure(2)

plot(0.1:0.1:1,mean\_dis);

xlabel("sigma");

ylabel("average dis from p to chat");

title("average distance of P to chat versus sigma");

The code includes 2 functions “get\_pos”, “cal\_tri\_area”

function pos = get\_pos(theta1,theta2,L1,L2)

%from L1 and L2 to get the pos of P

x1 = L1(1);

y1 = L1(2);

x2 = L2(1);

y2 = L2(2);

T1 = tan(pi/2-theta1);

T2 = tan(pi/2-theta2);

M1 = [T1,-1;

T2,-1];

V2 = [T1\*x1-y1;

T2\*x2-y2];

pos = inv(M1)\*V2;

end

function area = cal\_tri\_area(A1,A2,A3)

x1 = A1(1);

x2 = A2(1);

x3 = A3(1);

y1 = A1(2);

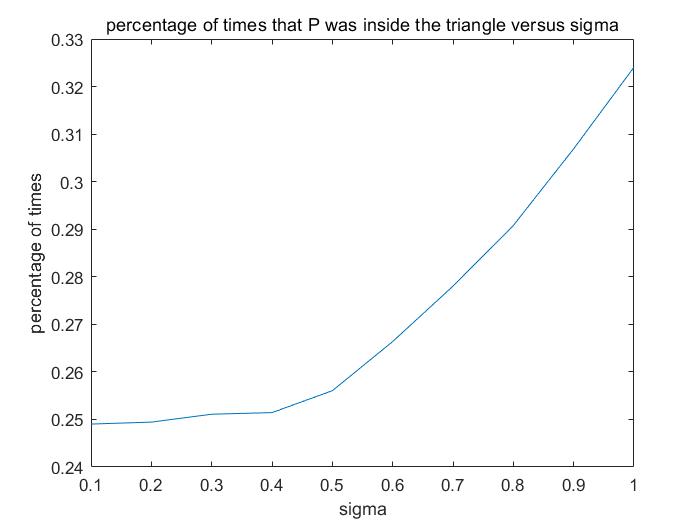
y2 = A2(2);

y3 = A3(2);

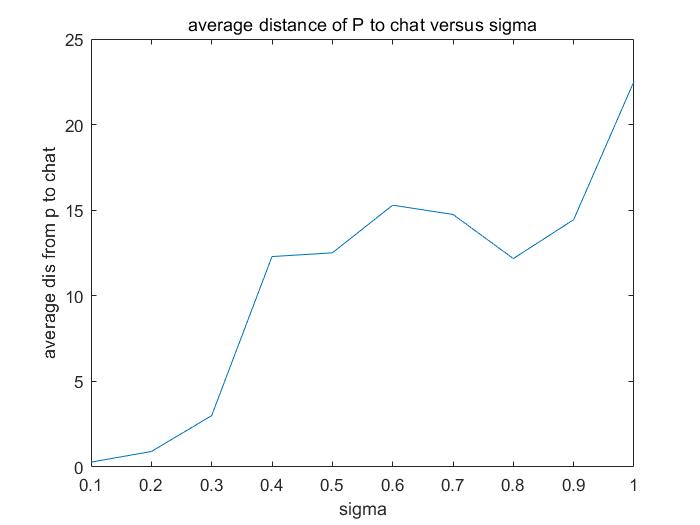
area = abs( (x1\*(y2-y3)+x2\*(y3-y1)+x3\*(y1-y2))/2 );

end

And the picture look like below:



P1: percentage that P was inside the triangle versus sigma



P2: average distance of P to c^ versus sigma

From the picture, we can observe that in general, the average distance from P to c^ is increasing and the percentage of times that P was inside the triangle is increasing with the increase of sigma.

Explanation: sigma represents the deviation from the true angle value. A large deviation means are far from where they should be. Then the triangle formed by these 3 points will become bigger and leads to the increasing possibility that P is inside the triangle. At the same time, as the triangle becomes bigger, the centroid of the triangle will move away from the true position P gradually, and leads to the increasing average distance.

For question2 we use the following code:

clc;

clear all;

load('AE584\_Midterm\_P2.mat');

L1 = [1.52,0,0]';

L2 = [0,0,0]';

star1 = [0,1,0]';

star2 = [0,0,1]';

P0 = [0.52 0 -1]';

P\_all = zeros(3,length(bearingL2St1));

P\_all(:,1) = P0;

time = 0:1:length(bearingL2St1);

for i=1:length(bearingL2St1)

theta = subAngL1L2(i);

phi1 = bearingL2St1(i);

phi2 = bearingL2St2(i);

P0 = [0.52 0 -1]';

fun = @(x) two\_star\_one\_angle(L1,L2,x,theta,phi1,star1,phi2,star2);

x0 = P\_all(:,i);

options = optimoptions('fminunc','OptimalityTolerance',10e-16);

[x\_ans,fval] = fminunc(fun,x0,options);

P\_all(:,i+1)=x\_ans;

end

figure(1)

subplot(3,1,1)

plot(time,P\_all(1,:));

title("position of spacecraft versus time on x-axis");

xlabel("time");

ylabel("position(AU)");

subplot(3,1,2)

plot(time,P\_all(2,:));

title("position of spacecraft versus time on y-axis");

xlabel("time");

ylabel("position(AU)");

subplot(3,1,3)

plot(time,P\_all(3,:));

title("position of spacecraft versus time on z-axis");

xlabel("time");

ylabel("position(AU)");

figure(2)

hold on

scatter3(P\_all(1,:),P\_all(2,:),P\_all(3,:),'filled');

scatter3(L1(1),L1(2),L1(3),'filled','r');

scatter3(L2(1),L2(2),L2(3),'filled','y');

hold off

title("3D plot of stars and spacecraft trajectory");

legend("spacecraft trajectory","Mar","Sun");

The code includes a function “two\_star\_one\_angle”:

function fun = two\_star\_one\_angle(L1,L2,x,theta,phi1,star1,phi2,star2)

eq1 = dot(x-L1,L2-L1)+norm(x-L1)...

\*norm(x-L2)\*cos(theta)-norm(x-L1)^2;

eq2 = dot(L2-x,star1)-cos(phi1)\*norm(L2-x);

eq3 = dot(L2-x,star2)-cos(phi2)\*norm(L2-x);

fun = eq1^2+eq2^2+eq3^2;

if(norm(x-L1)<0.1 || norm(x-L2)<0.1)

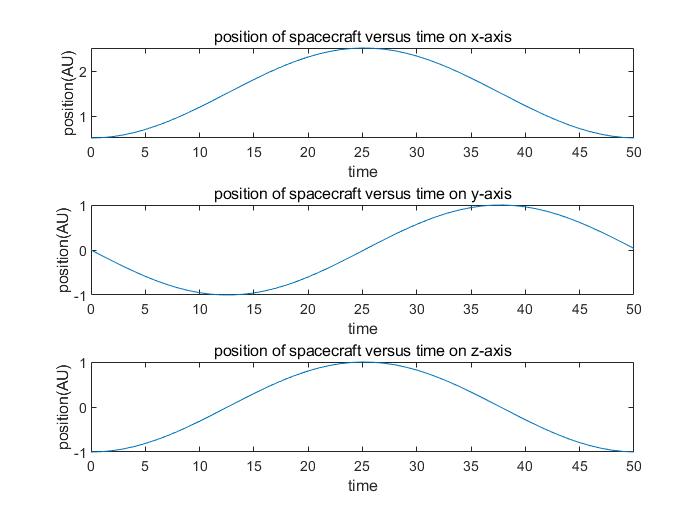
fun = fun+10001/(norm(x-L1)\*norm(x-L2)+0.01);

end

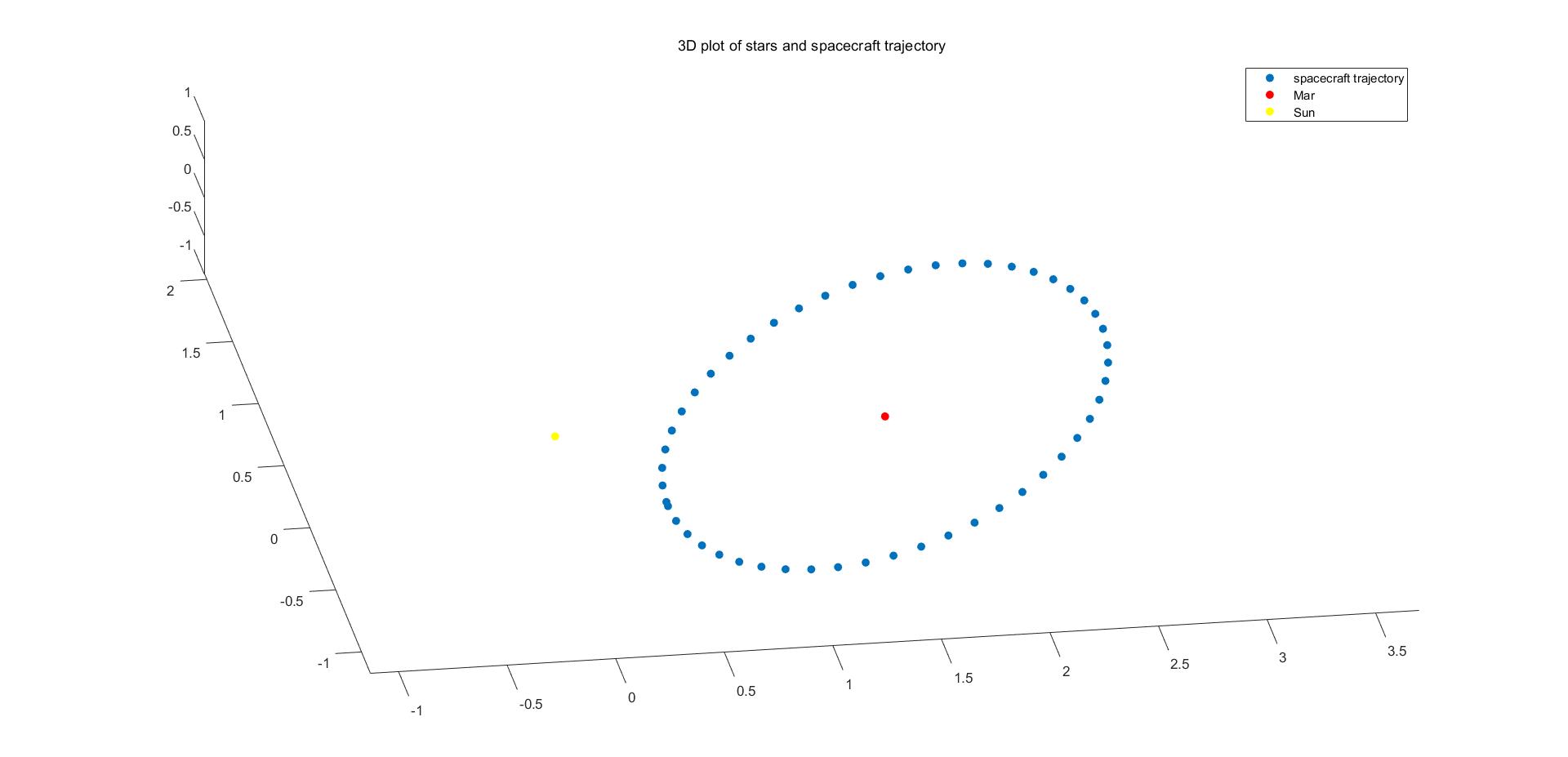
end

For question2 we use the following code:

And the pictures are shown below:



P3: x y z position of the spacecraft versus step K



P4: 3D trajectory of the spacecraft in solar system

For question3 we use the following code:

clc;

clear all;

time = 0:0.01:10;

phi\_E = 0;

theta\_E = pi/6;

psi\_E = 0;

%[~,phi\_e] = ode45(@(t,phi) sin(0.05\*t), time, phi\_E);

%[~,theta\_e] = ode45(@(t,theta) 0.3\*cos(0.01\*t), time, theta\_E);

%[~,psi\_e] = ode45(@(t,psi) 0.5\*sin(0.01\*t), time, psi\_E);

phi\_e = 20 \* (1-cos(0.05\*time));

theta\_e= 30\*sin(0.01\*time) + pi/6;

psi\_e = 50\*(1-cos(0.01\*time));

O\_EI = zeros(3,3,length(0:0.01:10));

for i = 1:1:length(0:0.01:10)

O\_EI(:,:,i) = zxz\_angle2mat(phi\_e(i),theta\_e(i),psi\_e(i));

end

%A = vec\_to\_mat(w0); % Some arbitrary matrix we will use

F0 = eye(3); % matrix initial value

odefun = @(t,y) deriv(t,y); % Anonymous derivative function with A

tspan = 0:0.01:10;

f0 = reshape(F0,[1,9])';

[T,F] = ode45(odefun,tspan,f0); % Pass in column vector initial value

F = reshape(F.',3,3,[]); % Reshape the output as a sequence of 3x3 matrices

O\_BI = F;

O\_BE = zeros(3,3,length(0:0.01:10));

for i = 1:1:length(0:0.01:10)

O\_BE(:,:,i) = O\_BI(:,:,i)\*O\_EI(:,:,i)';

end

figure(1)

hold on

for i=1:3

for j=1:3

subplot(3,3,3\*(i-1)+j);

plot(T,squeeze(O\_BE(i,j,:)),'color','#D95319','LineWidth',2);

%plot(T,o\_indexs\_2(j,:),'b');

txt = [int2str(3\*(i-1)+j),'th value of O\_{BE}'];

title(txt);

xlabel('time(s)')

end

end

hold off

O1\_solutions = zeros(6,length(O\_BE(1,1,:)));

for i=1:length(O\_BE)

[O1\_solutions(1:3,i),O1\_solutions(4:6,i)] = cal\_Eular(O\_BE(:,:,i));

end

T = 0:0.01:10;

figure(2)

subplot(3,1,1);

hold on

plot(T,O1\_solutions(1,:),'color','#D95319','LineWidth',2);

title('phi versus time');

xlabel('time(s)');

hold off

subplot(3,1,2);

hold on

plot(T,O1\_solutions(2,:),'color','#D95319','LineWidth',2);

title('theta versus time');

xlabel('time(s)')

hold off

subplot(3,1,3);

hold on

plot(T,O1\_solutions(3,:),'color','#D95319','LineWidth',2);

title('psi versus time');

xlabel('time(s)')

hold off

function dy = deriv(t,y)

A = vec\_to\_mat([cos(2\*t),cos(2\*t),0.025\*t]);

F = reshape(y,size(A)); % Reshape input y into matrix

FA = -A\*F; % Do the matrix multiply

dy = reshape(FA,[1,9])'; % Reshape output as a column vector

end

The code includes 3 functions “zxz\_angle2mat”, “cal\_Eular”, “vec\_to\_mat”

function O\_matrix = zxz\_angle2mat(phi,theta,psi)

r1 = [cos(phi), sin(phi), 0;

-sin(phi), cos(phi),0;

0, 0, 1];

r2 = [1, 0, 0;

0, cos(theta), sin(theta);

0, -sin(theta), cos(theta)];

r3 = [cos(psi), sin(psi), 0;

-sin(psi), cos(psi),0;

0, 0, 1];

O\_matrix = r3\*r2\*r1;

end

function [solution1, solution2] = cal\_Eular(o\_matrix)

%get orientation\_matrix in, Eular angles out

theta\_1 = -asin(o\_matrix(1,3));

theta\_2 = pi-theta\_1;

if(theta\_1<0)

theta\_2 = -pi-theta\_1;

end

Psi\_1 = atan2(o\_matrix(1,2)/cos(theta\_1),...

o\_matrix(1,1)/cos(theta\_1));

Psi\_2 = atan2(o\_matrix(1,2)/cos(theta\_2),...

o\_matrix(1,1)/cos(theta\_2));

Phi\_1 = atan2(o\_matrix(2,3)/cos(theta\_1),...

o\_matrix(3,3)/cos(theta\_1));

Phi\_2 = atan2(o\_matrix(2,3)/cos(theta\_2),...

o\_matrix(3,3)/cos(theta\_2));

solution1 = [Phi\_1,theta\_1,Psi\_1];

solution2 = [Phi\_2,theta\_2,Psi\_2];

end

function matrix = vec\_to\_mat(w)

wx = w(1);

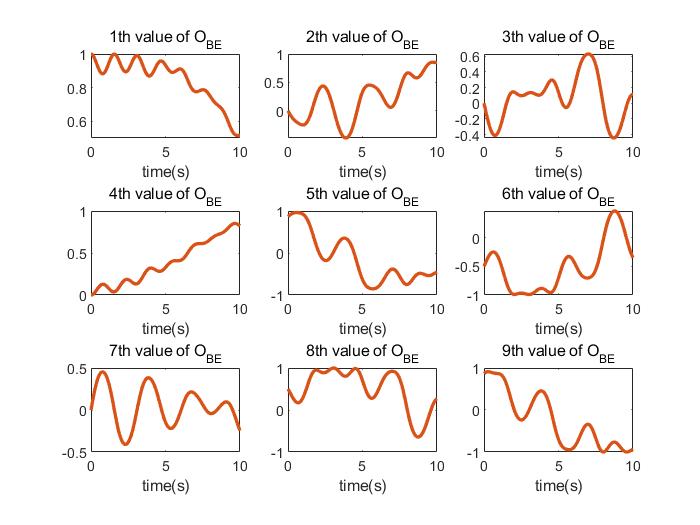
wy = w(2);

wz = w(3);

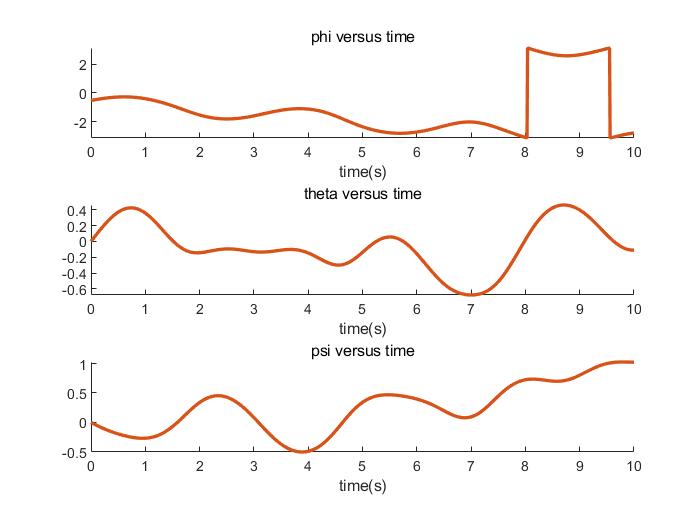
matrix = [0,-wz,wy;wz,0,-wx;-wy,wx,0];

end

And the pictures are shown below:



P5: all components of versus time



P6: versus time

For question4 we use the following code:

clc;

clear all;

time = 0:0.01:20;

phi = pi/6;

g = 9.80665;

O\_BA0 = [1, 0, 0;

0, cos(phi), sin(phi);

0, -sin(phi), cos(phi)]; % matrix initial value

v\_0 = [0;cos(phi);sin(phi)];

r\_0 = [1;0;0];

w\_BA = [0;0;1];

opts = odeset('RelTol',1e-5,'AbsTol',1e-16);

odefun = @(t,y) deriv(t,y); % Anonymous derivative function with A

tspan = time;

f0 = reshape(O\_BA0,[1,9])';

f0 = [f0;r\_0;v\_0];

[T,F] = ode45(odefun,tspan,f0,opts); % Pass in column vector initial value

%T = F';

O\_BA = F(:,1:9);

O\_BA = reshape(O\_BA.',3,3,[]); % Reshape the output as a sequence of 3x3 matrices

r\_cw = F(:,10:12);

figure(1)

hold on

for i=1:3

for j=1:3

subplot(3,3,3\*(i-1)+j);

plot(T,squeeze(O\_BA(i,j,:)));

%plot(T,o\_indexs\_2(j,:),'b');

txt = [int2str(3\*(i-1)+j),'th value of O\_{BA}'];

title(txt);

xlabel('time(s)')

end

end

hold off

figure(2)

hold on

for i=1:3

subplot(3,1,i);

plot(T,r\_cw(:,i));

txt = [int2str(i),'th component of r'];

title(txt);

xlabel('time(s)')

end

hold off

figure(3)

plot3(r\_cw(:,1),r\_cw(:,2),r\_cw(:,3));

title("3D trajectory of the center");

grid on;

axis equal;

xlabel("x")

zlabel("z")

ylabel("y")

function dy = deriv(t,y)

w\_x = vec\_to\_mat([0;0;1]);

O\_BA = reshape(y(1:9),size(w\_x)); % Reshape input y into matrix

O\_BA\_diff = -w\_x\*O\_BA; % Do the matrix multiply

dy1 = reshape(O\_BA\_diff,[1,9])'; % Reshape output as a column vector

dr = y(13:15);% same as velocity

g = 9.80665;

phi = pi/6;

a\_mean = [-1-g\*sin(phi)\*sin(t);

-g\*sin(phi)\*cos(t);

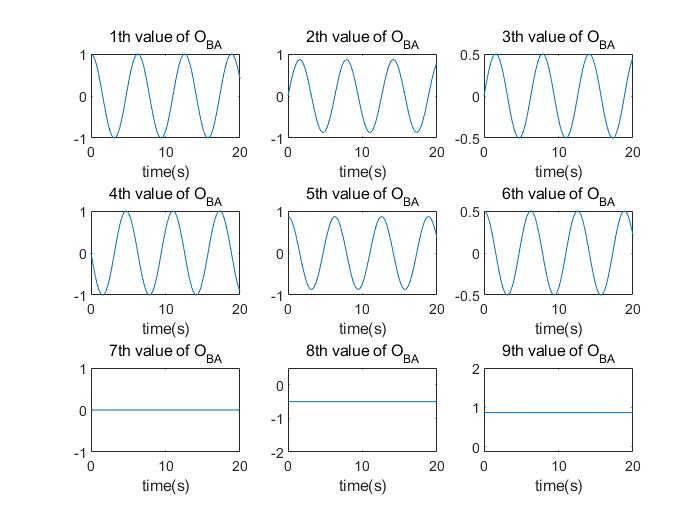
-g\*cos(phi)];

dv = O\_BA'\*a\_mean-[0;0;-g]; %acceleration

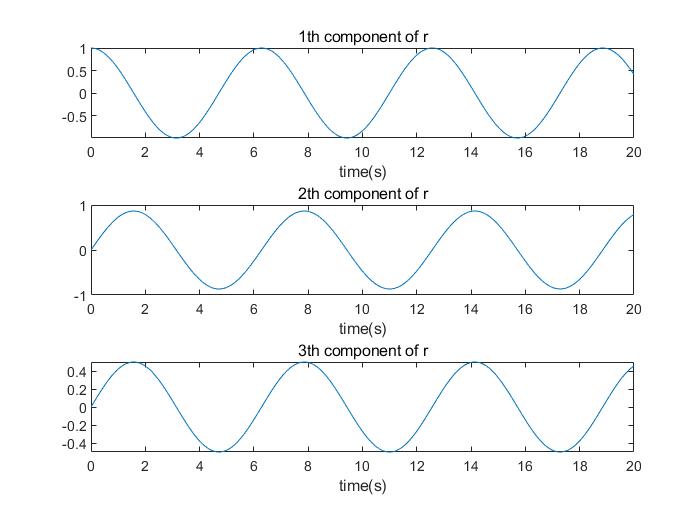
dy = [dy1;dr;dv];

end

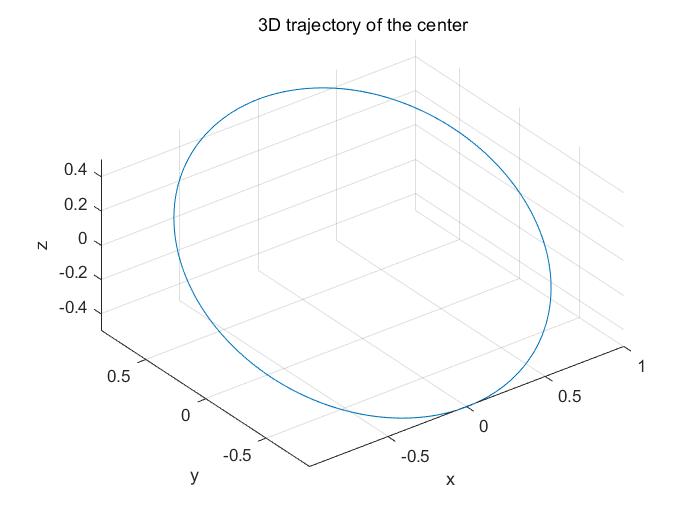
And the pictures are shown below:



P7: all components of versus time



P8: 3 components of versus time



P9: 3D trajectory of the center of mass of the quadcopter