Cairo University Faculty of Engineering Aerospace Department

AEROSPACE ENGINEERING

Task 2

Autopilot -AER 408

Dr.Osama Saaid

Team4

Name	sec	BN
Mohammed Ahmed Hassan Ahmed	2	37
Ibrahim Thabet Allam	1	1
Mohammed Hatem Mohammed Saeed	2	39
Mohammed Abd El-Mawgoud Ghoneam	2	43
Mohamed Hassan Gad Ali	2	41

Table of Contents

RBD Equations Solver

In this task we aim to solve the 12 rigid body equations of dynamics equations that we were derived in task1, the solver is built basically to compare the results of solution from 3 sources (our code, ODE45 and Simulink).

We will start by reviewing the 12 equations and the data given in this task, then we will show the figures and the code.

$$F_x = m(\dot{u} + qw - rv)$$

$$F_y = m(\dot{v} + ru - pw)$$

$$F_z = m(\dot{w} + pv - qu)$$

$$L = \dot{H}_x + qH_z - rH_y$$

$$M = \dot{H}_y + rH_x - pH_z$$

$$N = \dot{H}_z + pH_y - qH_x$$

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & S\varphi\, T\theta & C\varphi\, T\theta \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi/_{C\theta} & C\varphi/_{C\theta} \end{bmatrix} \begin{Bmatrix} P \\ Q \\ R \end{Bmatrix}$$

$$\begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\theta C_\theta & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}$$

Givens:

$$\boldsymbol{t_{final} = 15\ sec\ ,\ \ Forces = [10\ 5\ 9]\ N\ , Moments = [10\ 20\ 5] N.m}$$

$$\boldsymbol{Mass = 15\ Kg}$$

$$\boldsymbol{I = \begin{bmatrix} I_x & I_{xy} & I_{xz} \\ I_{yx} & I_y & I_{yz} \\ I_{zx} & I_{zy} & I_z \end{bmatrix} = \begin{bmatrix} 1 & -2 & -1 \\ -2 & 5 & -3 \\ -1 & -3 & 0.1 \end{bmatrix}}$$

IC's:

$$\boldsymbol{[u\ v\ w, p\ q\ r, \phi\ \theta\ \psi, x\ y\ z] = [10, 2, 0; \frac{2\pi}{180}, \frac{\pi}{180}\ 0; \frac{20\pi}{180}, \frac{15\pi}{180}, \frac{30\pi}{180};\ 2, 4, 7]}$$

Our solver code has main function called RBDSolver which takes the initial condition given and returns the 12 rates of change of each state, by integrating each one we get the states of the rigid body.

1- main script:

```
clc; clear; close all;
```

Inputs

```
[forces,Moments,Mass,I,timeSpan,h,IC_vec] = Input();

dt=h;

ICs=IC_vec;

g=9.81;

Mg=Mass*g;

t_initial=timeSpan(1);

t_final=timeSpan(2);

t_vec = t_initial:h:t_final;

n=length(t_vec);
```

Solving

```
Result = NaN(12,n);

Result(:,1) = ICs;

time = linspace(0, 15, n);

for i =2:n
    Result(:, i) = RBDSolver(Result(:, i-1), dt);
end
u_vec=Result(1,:);
v_vec=Result(2,:);
w_vec=Result(3,:);
p_vec=Result(4,:);
q_vec=Result(5,:);
r_vec=Result(6,:);
phi_vec=Result(7,:);
theta_vec=Result(8,:);
epsi_vec=Result(9,:);
xe_vec=Result(10,:);
ye_vec=Result(11,:);
ze_vec=Result(12,:);
```

Functions Used:

RBDSolver:

```
% RBD Solver is function that implements Runge-Kutta-4
```

```
% Algorithm in order to integrate 6 DOF set of equations
% with a give Inital Conditions ICs, for single dt time step.
function state = RBDSolver(ICs, dt)
    K = zeros(12, 4);

    K(:, 1) = dt*DOF6(0, ICs );
    K(:, 2) = dt*DOF6(0, ICs+0.5*K(:, 1) );
    K(:, 3) = dt*DOF6(0, ICs+0.5*K(:, 2) );
    K(:, 4) = dt*DOF6(0, ICs+K(:, 3) );

    state = ICs + (...
        K(:, 1)+...
        2*K(:, 2)+...
        2*K(:, 3)+...
        K(:, 4))/6;
end
```

3- Input:

```
function [Forces,Moments,Mass,I,timeSpan,dt,ICs] = Input()
%%Inputs
% Forces, Moments and Inertia
Forces = [10 5 9]';          % Vector
Moments = [10 20 5]';        % Vector
Mass = 15;
I = [1 -2 -1;
    -2 5 -3;
    -1 -3 0.1];
% Integration time span & Step
timeSpan = [0 15];
dt = 0.001;
% Initial Conditions
%[u; v; w; p; q; r; phi; theta; epsi; xe0; ye0; ze0]
ICs = [10; 2; 0; 2*pi/180; pi/180; 0; 20*pi/180; 15*pi/180; 30*pi/180; 2; 4; 7];

end
```

4- TCS function:

```
% Calculate Sin, Cos ,Tan for any set of three angles
% and return results in struct form for easy access in code.
function [S, C, T] = SCT(ICs)
    S = struct(...
        'phi', sin(ICs(1)),...
        'theta', sin(ICs(2)),...
        'epsi', sin(ICs(3))...
    );
    C = struct(...
        'phi', cos(ICs(1)),...
```

```
            'theta', cos(ICs(2)),...
            'epsi', cos(ICs(3))...
        );
        T = struct(...
            'phi', tan(ICs(1)),...
            'theta', tan(ICs(2)),...
            'epsi', tan(ICs(3))...
        );
end
```

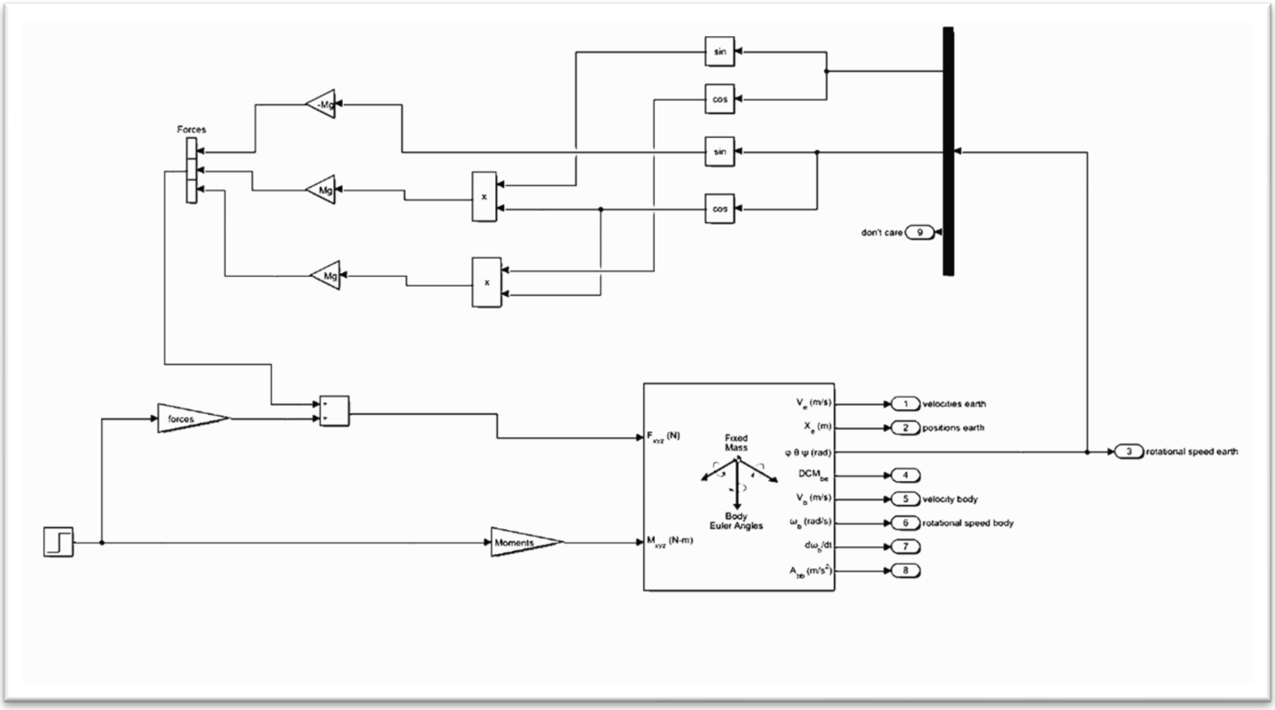
5- 6DOF function:

```
% Function return the set of 12 state of the six degree of freedom
% system after sub., with given ICs
function F = DOF6(~, ICs)
    [forces,Moments,Mass,Inertia,~,~,~] = Input();
    % (Sin, Cos, Tan) of (phi, theta, epsi)
    [S, C, T] = SCT(ICs(7:9));

    Forces = forces + Mass*9.81*[
        -S.theta;
        S.phi*C.theta;
        C.phi*C.theta;
    ];

    % (u, v, w) dot
    F(1:3, 1) = Forces/Mass - cross(...
        ICs(4:6, 1), ICs(1:3, 1)...
    );

    % (p, q, r) dot
    F(4:6, 1) = Inertia\(Moments - cross(...
        ICs(4:6, 1), Inertia * ICs(4:6, 1)...
    ));

    % (phi, theta, epsi) dot
    F(7:9, 1) = [
        1, S.phi*T.theta, C.phi*T.theta;
        0, C.phi, -S.phi;
        0, S.phi/C.theta, C.phi/C.theta;
    ] * ICs(4:6, 1);

    % (x, y, z) dot
    F(10:12, 1) = [
        C.theta*C.epsi, (S.phi*S.theta*C.epsi - C.phi*S.epsi),
(C.phi*S.theta*C.epsi + S.phi*S.epsi);
        C.theta*S.epsi, (S.phi*S.theta*S.epsi + C.phi*C.epsi),
(C.phi*S.theta*S.epsi - S.phi*C.epsi);
        -S.theta, S.phi*C.theta, C.phi*C.theta
```

```
    ] * ICs(1:3, 1);

end
```

Validation

we validated our results by using Simulink and ODE45



Solving using ODE45

```
[t, states] = ode45(@DOF6,timeSpan,ICs);

t_ODE=t';

u_ODE=states(:,1)';

v_ODE=states(:,2)';

w_ODE=states(:,3)';

p_ODE=states(:,4)';

q_ODE=states(:,5)';
```

```
r_ODE=states(:,6)';

phi_ODE=states(:,7)';

theta_ODE=states(:,8)';

epsi_ODE=states(:,9)';

xe_ODE=states(:,10)';

ye_ODE=states(:,11)';

ze_ODE=states(:,12);
```

solving using Simulink

```
%Run Smulink file
Sim_run=sim('simulink_test.slx');

%Extraxt Data from Simulink
%velocity body
Velocity_body=Sim_run.yout.getElement('velocity body');
%rotational speed body
Rotat_Velocity_body=Sim_run.yout.getElement('rotational speed body');
%rotational speed earth
Rotat_Velocity_earth=Sim_run.yout.getElement('rotational speed earth');
%positions earth
positions_earth=Sim_run.yout.getElement('positions earth');

t_sim=Velocity_body.Values.time;
u_sim=Velocity_body.Values.Data(:,1)';
v_sim=Velocity_body.Values.Data(:,2)';
w_sim=Velocity_body.Values.Data(:,3)';
p_sim=Rotat_Velocity_body.Values.Data(:,1)';
q_sim=Rotat_Velocity_body.Values.Data(:,2)';
r_sim=Rotat_Velocity_body.Values.Data(:,3)';
phi_sim=Rotat_Velocity_earth.Values.Data(:,1)';
theta_sim=Rotat_Velocity_earth.Values.Data(:,2)';
epsi_sim=Rotat_Velocity_earth.Values.Data(:,3)';
xe_sim=positions_earth.Values.Data(:,1)';
ye_sim=positions_earth.Values.Data(:,2)';
ze_sim=positions_earth.Values.Data(:,3)';
```

plots

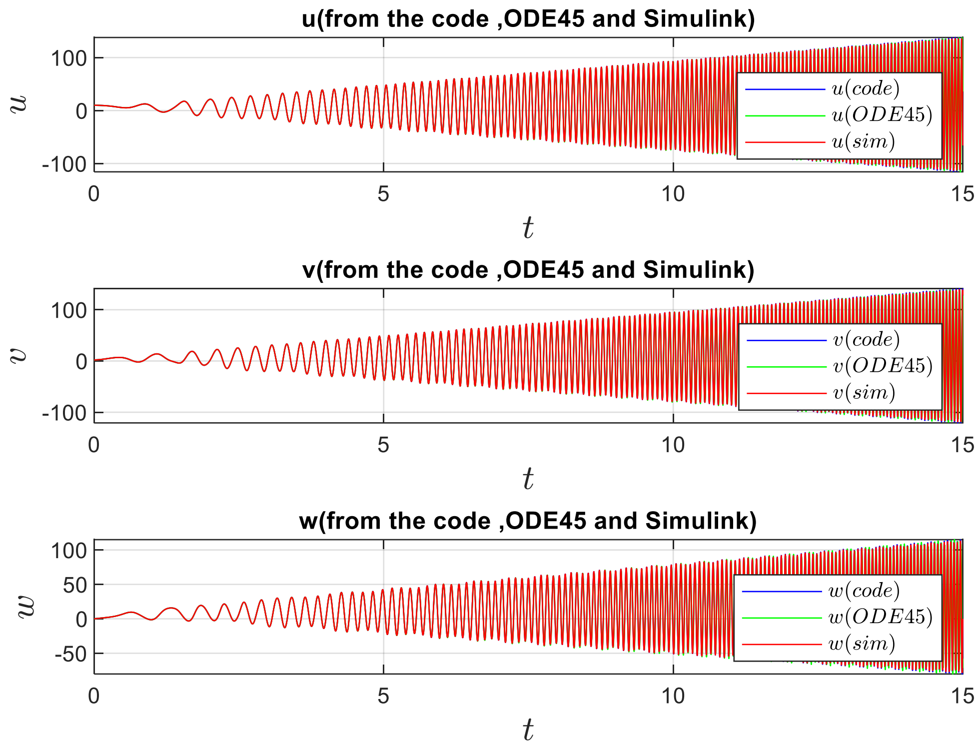
```
%velocity body
figure
subplot(3, 1,1)
plot(t_vec,u_vec,'b',t_ODE,u_ODE,'g',t_sim,u_sim,'r')
```

```
legend({'$u(code)$','$u(ODE45)$','$u(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$u$','Interpreter','latex','FontSize',13)
title('u(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 2)
plot(t_vec,v_vec,'b',t_ODE,v_ODE,'g',t_sim,v_sim,'r')
legend({'$v(code)$','$v(ODE45)$','$v(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$v$','Interpreter','latex','FontSize',13)
title('v(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 3)
plot(t_vec,w_vec,'b',t_ODE,w_ODE,'g',t_sim,w_sim,'r')
legend({'$w(code)$','$w(ODE45)$','$w(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$w$','Interpreter','latex','FontSize',13)
title('w(from the code ,ODE45 and Simulink)')
grid on


%rotational speed body
figure
subplot(3, 1,1)
plot(t_vec,p_vec,'b',t_ODE,p_ODE,'g',t_sim,p_sim,'r')
legend({'$p(code)$','$p(ODE45)$','$p(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$p$','Interpreter','latex','FontSize',13)
title('p(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 2)
plot(t_vec,q_vec,'b',t_ODE,q_ODE,'g',t_sim,q_sim,'r')
legend({'$q(code)$','$q(ODE45)$','$q(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$q$','Interpreter','latex','FontSize',13)
title('q(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 3)
plot(t_vec,r_vec,'b',t_ODE,r_ODE,'g',t_sim,r_sim,'r')
legend({'$r(code)$','$r(ODE45)$','$r(sim)$'},'Location','southeast','FontSize',8,..
.
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$r$','Interpreter','latex','FontSize',13)
title('r(from the code ,ODE45 and Simulink)')
```

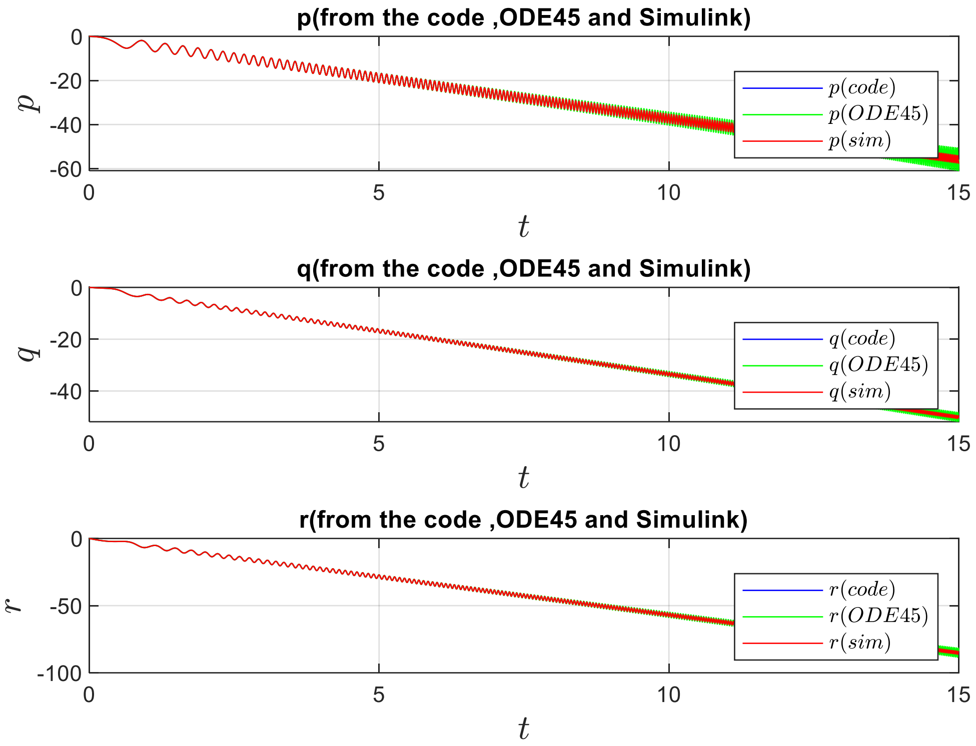
```
grid on


%rotational speed earth
figure
subplot(3, 1,1)
plot(t_vec,phi_vec,'b',t_ODE,phi_ODE,'g',t_sim,phi_sim,'r')
legend({'$\phi(code)$','$\phi(ODE45)$','$\phi(sim)$'},'Location','southeast','FontS
ize',8,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$\phi$','Interpreter','latex','FontSize',13)
title('\phi(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 2)
plot(t_vec,theta_vec,'b',t_ODE,theta_ODE,'g',t_sim,theta_sim,'r')
legend({'$\theta(code)$','$\theta(ODE45)$','$\theta(sim)$'},'Location','southeast',
'FontSize',8,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$\theta$','Interpreter','latex','FontSize',13)
title('\theta(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 3)
plot(t_vec,epsi_vec,'b',t_ODE,epsi_ODE,'g',t_sim,epsi_sim,'r')
legend({'$\psi(code)$','$\psi(ODE45)$','$\psi(sim)$'},'Location','southeast','FontS
ize',8,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$\psi$','Interpreter','latex','FontSize',13)
title('\psi(from the code ,ODE45 and Simulink)')
grid on


%positions earth
figure
subplot(3, 1,1)
plot(t_vec,xe_vec,'b',t_ODE,xe_ODE,'g',t_sim,xe_sim,'r')
legend({'$xe(code)$','$xe(ODE45)$','$xe(sim)$'},'Location','southeast','FontSize',8
,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$xe$','Interpreter','latex','FontSize',13)
title('xe(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 2)
plot(t_vec,ye_vec,'b',t_ODE,ye_ODE,'g',t_sim,ye_sim,'r')
legend({'$ye(code)$','$ye(ODE45)$','$ye(sim)$'},'Location','southeast','FontSize',8
,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$ye$','Interpreter','latex','FontSize',13)
title('ye(from the code ,ODE45 and Simulink)')
grid on
subplot(3,1, 3)
```

```
plot(t_vec,ze_vec,'b',t_ODE,ze_ODE,'g',t_sim,ze_sim,'r')
legend({'$ze(code)$','$ze(ODE45)$','$ze(sim)$'},'Location','southeast','FontSize',8
,...
    'Interpreter','latex')
xlabel('$t$','Interpreter','latex','FontSize',13)
ylabel('$ze$','Interpreter','latex','FontSize',13)
title('ze(from the code ,ODE45 and Simulink)')
grid on
```

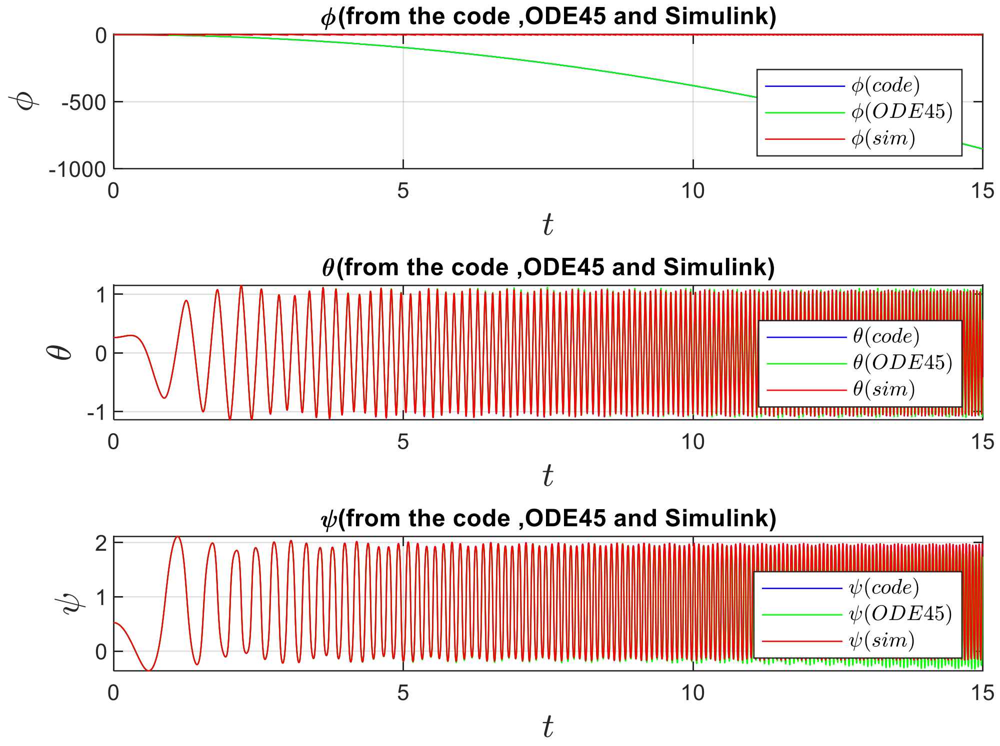
velocity with respect to body Axes:



Rotational speed W.r.t body axes



Rotational speed w.r.t Earth Axes



Position w.r.t Earth Axes