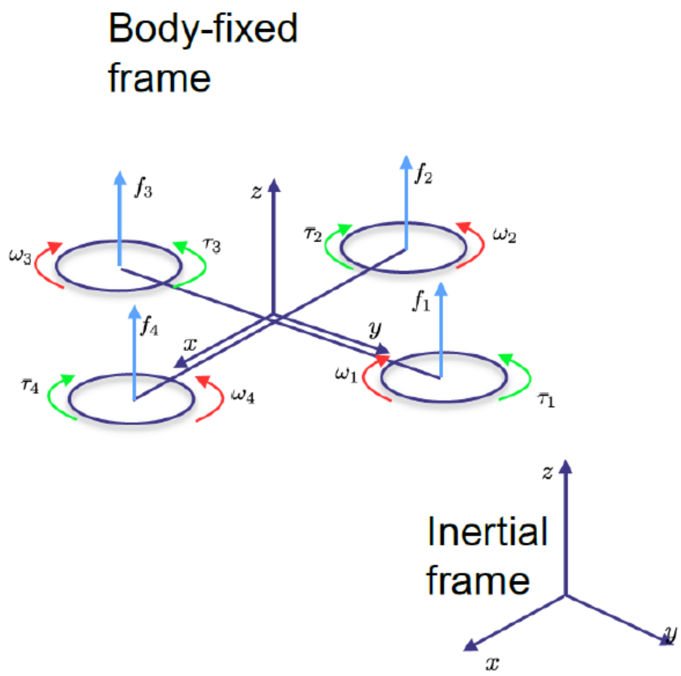
Part 2: Modeling




```
clear all
m = 0.5;    %[kg]           the drone mass in the center of gravity of the drone
L = 0.225;  %[m]            length of each arm of the quadrotor frame, the distance of the moto
k = 0.01;   %[N*s^2/rad^2]  the overall aerodynamic coefficient necessary to compute the lift c
b = 0.01;   %[Nm*s^2/rad^2] the overall aerodynamic coefficient necessary to compute the drag t
D = diag([0.01, 0.01, 0.01]);
            %[N*s/m]        matrix representing the the drag on the UAV moving in air with velo
Ixx = 3e-6; %[Nms^2/rad]    the moment of inertia on the principal axis x
Iyy = 3e-6; %[Nms^2/rad]    the moment of inertia on the principal axis y
Izz = 1e-5; %[Nms^2/rad]    the moment of inertia on the principal axis z
MoI = diag([Ixx, Iyy, Izz]);
g = [0, 0, -9.81]';
            % [m/s^2]       gravity acts along the z-axis of the inertial frame of reference
```

$\mathbf{p}[m]$ is the position of the CoG of the UAV w.r.t a fixed inertial frame of reference.

$\mathbf{p} = [x, y, z]^T$

$\Theta[rad]$ is the representation of the rotation of the body-fixed frame w.r.t the fixed inertial frame of reference, accroding to the Roll-Pitch-Yaw angular representation.

$\Theta = [\phi, \theta, \psi]^T$

2.1.1 Define the rotation matrix representing the orientation of the body-fixed frame w.r.t. the inertial frame.

Rotation from inertial frame to body-fixed frame

$$R_{ZYX}(\phi,\theta,\psi) = R_z(\phi)R_y(\theta)R_x(\psi)$$

$$R = \begin{bmatrix} \cos(\phi)\ \cos(\theta) & -\sin(\phi)\ \cos(\psi) + \cos(\phi)\ \sin(\theta)\ \sin(\psi) & \sin(\phi)\ \sin(\psi) + \cos(\phi) \\ \sin(\phi)\ \cos(\theta) & \cos(\phi)\ \cos(\psi) + \sin(\phi)\ \sin(\theta)\ \sin(\psi) & -\cos(\phi)\ \sin(\psi) + \sin(\phi) \\ -\sin(\theta) & \cos(\theta)\ \sin(\psi) & \cos(\theta)\ \cos(\end{bmatrix}$$

$$\mathbf{R}(\gamma) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix}$$

$$\vec{\omega} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \dot{\vec{\theta}}$$

2.1.2 Define the relation between the angular velocity Θ and the rotational velocity of the body-fixed frame ω.

From body-fixed to inerital frame

$$\dot{\theta} = [B(\theta)]\omega$$

$$\boxed{\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{L}_B^I \boldsymbol{\omega}_B}$$

From inerital frame to body-fixed frame

$$\omega = [B(\theta)]^{-1}\dot{\theta}$$

$$\boxed{\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{L}_I^B \dot{\boldsymbol{\Theta}}}$$

2.1.3 Write the linear and angular dynamic equation of the drone in compact form, and clearly show each component of the equation explicitly

Force

Propeller total trust

$$F_B = \sum_{i=1}^{4} f_i = k \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} \omega_i^2 \end{bmatrix}$$

Viscous damping

model friction as a force proportional to the linear velocity in each direction.

$$F_D = \begin{bmatrix} -k_d \dot{x} \\ -k_d \dot{y} \\ -k_d \dot{z} \end{bmatrix}$$

Total torque around z-axis of the body

$$\tau_\psi = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

Total torque on x-asis

$$\tau_\phi = \sum r \times T = L(k\omega_1^2 - k\omega_3^2) = Lk(\omega_1^2 - \omega_3^2)$$

Total torque on y-axis

$$\tau_\theta = \sum r \times T = L(k\omega_2^2 - k\omega_4^2) = Lk(\omega_2^2 - \omega_4^2)$$

Total body torque

$$\tau_B = \begin{bmatrix} Lk({\omega_1}^2 - {\omega_3}^2) \\ Lk({\omega_2}^2 - {\omega_4}^2) \\ b\left({\omega_1}^2 - {\omega_2}^2 + {\omega_3}^2 - {\omega_4}^2\right) \end{bmatrix}$$

Linear motion equations (in the inertial frame):

$$\mathbf{m}\ddot{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + RF_B + F_D \quad F_B = \sum_{i=1}^{4} f_i = k \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} \omega_i^2 \end{bmatrix}$$

$$F_D = \begin{bmatrix} -k_d\dot{x} \\ -k_d\dot{y} \\ -k_d\dot{z} \end{bmatrix}$$

Angular motion equations(in body-fixed frame):

$$\dot{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = I^{-1}(\tau_B - \omega \times (I\omega)) \quad I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}.$$

after simplification:

$$\dot{\omega} = \begin{bmatrix} \tau_\phi I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\psi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix}$$

$x_1 = [x, y, z]^T - position$

$x_2 = [\dot{x}, \dot{y}, \dot{z}]^T - linear\ \ velocity$

$x_3 = [\phi, \theta, \psi]^T - Roll - Pitch - Yaw\ \ angles$

$x_4 = [\omega_x, \omega_y, \omega_z]^T - angular\ \ velocity$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R T_B + \frac{1}{m} F_D$$

$$\dot{x}_3 = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix}^{-1} x_4$$

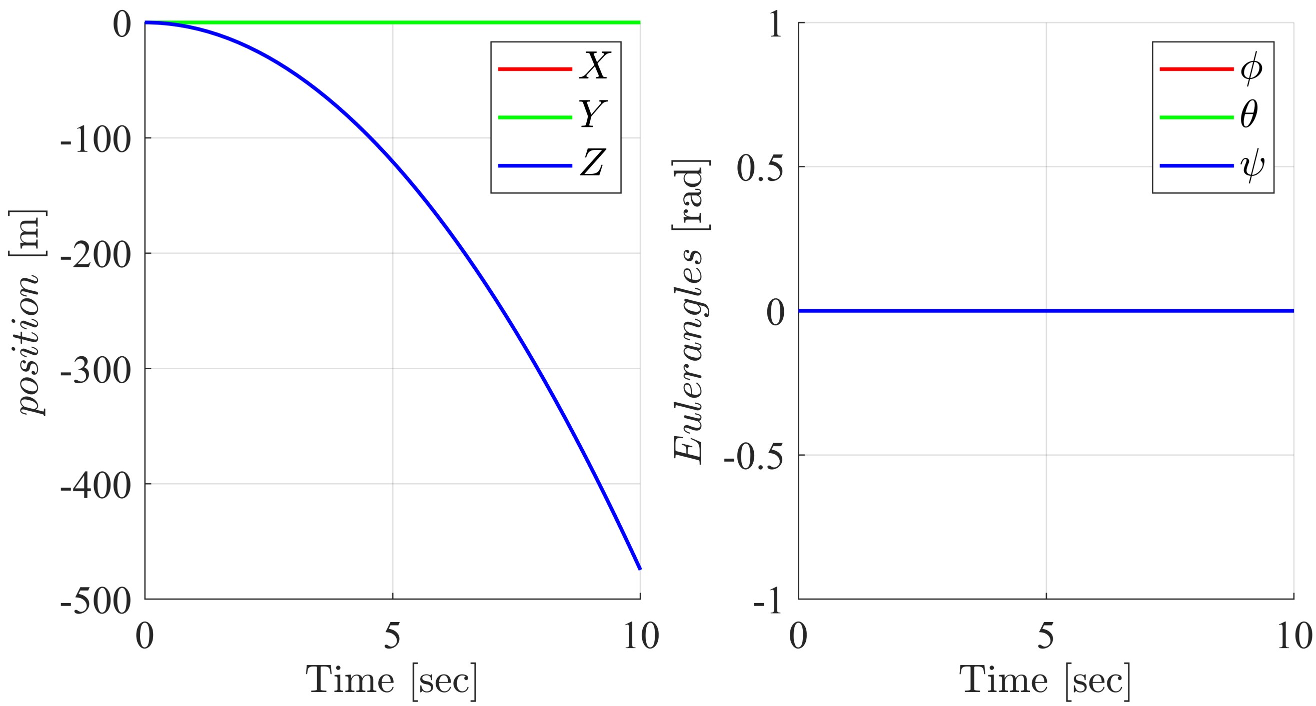
$$\dot{x}_4 = \begin{bmatrix} \tau_\phi {I_{xx}}^{-1} \\ \tau_\theta {I_{yy}}^{-1} \\ \tau_\psi {I_{zz}}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix}$$

2.1.4 Make a MATLAB/Simulink model of the drone, given initial conditions

```
% simTime
simTime = 10.0;

% [m] position of the CoG of the UAV w.r.t. a fixed inertial frame of reference
p = [0, 0, 0]';     %[x, y, z]

% [m/s] linear velocity of the CoG of the UAV w.r.t. a fixed inertial frame of reference
p_dot = [0, 0, 0]';

% [rad] the representation of the rotation of the body-fixed w.r.t. inertial frame
Theta = [0, 0, 0]';     %   [phi, theta, psi] = [roll, pitch, yaw]

% [rad/s] the angular velocity of the body-fixed frame w.r.t the inertial frame (expressed in
omega = [0, 0, 0]';

% [rad/s] vector of the angular speed of the four propellers
RPM = [0, 0, 0, 0]';
```

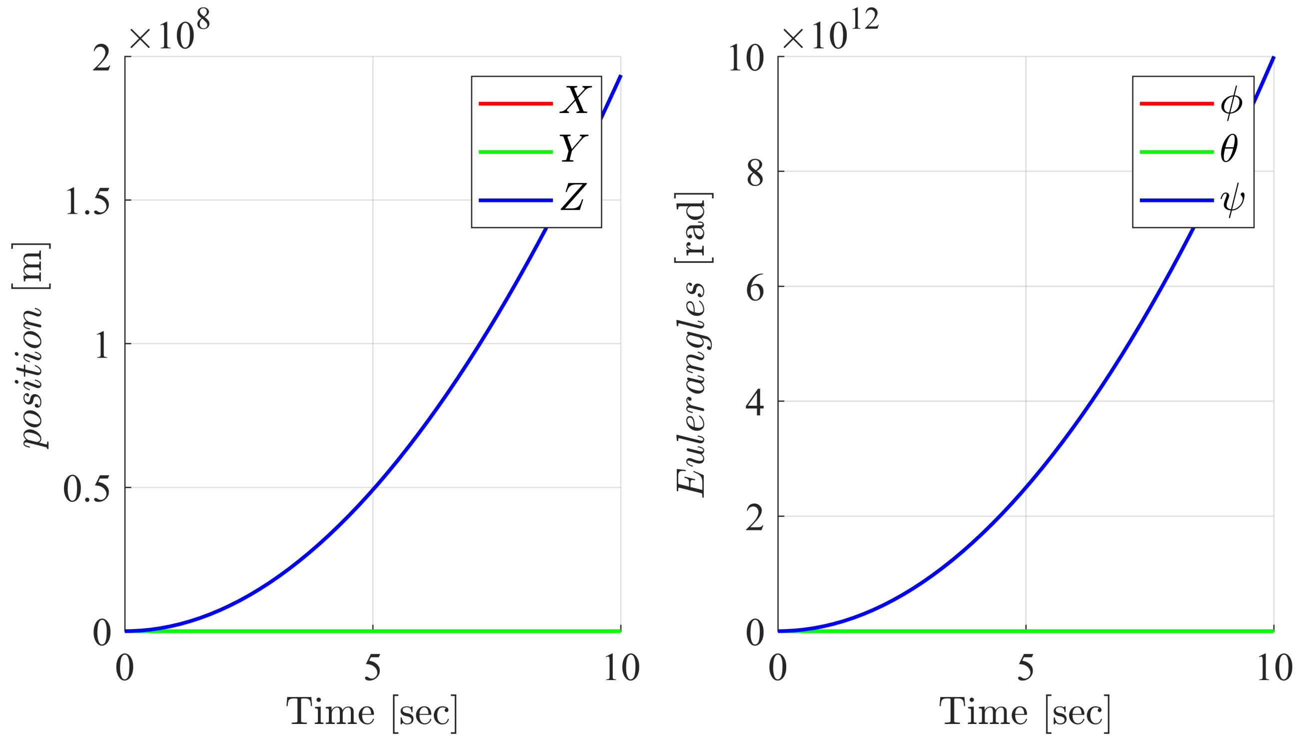
1) Make a plot of p and Θ, given Ω = [0, 0, 0, 0]T and explain the result

```
TakeStep = 1; % 1 for step input
RPM = [0, 0, 0, 0]';
simOut = sim("My_Nonlinear_new.slx");
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



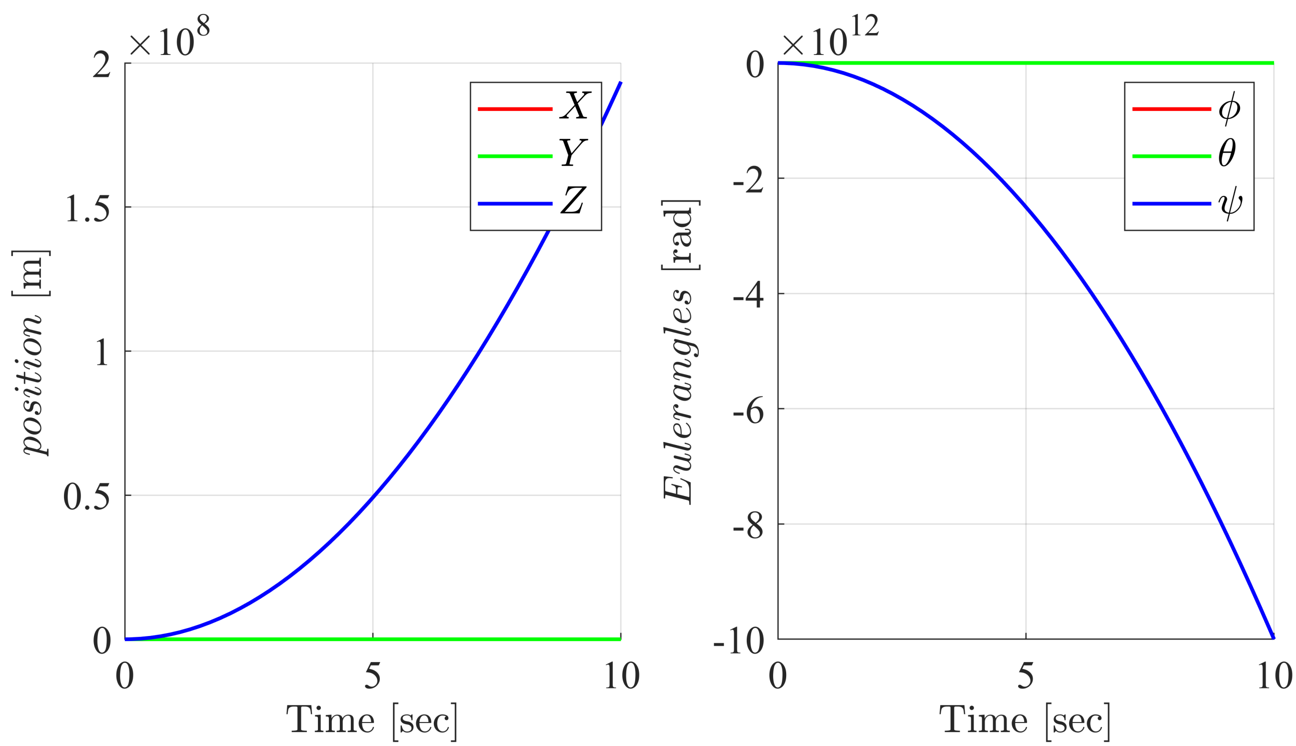
2) Make a plot of p and Θ, given Ω = [10000, 0, 10000, 0]T and explain the result

```
RPM = [10000, 0, 10000, 0]';
simOut = sim("My_Nonlinear_new.slx");
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



3) Make a plot of p and Θ, given $\Omega = [0, 10000, 0, 10000]T$ and explain the result

```
RPM = [0, 10000, 0, 10000]';
simOut = sim("My_Nonlinear_new.slx");
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



Exercise 2.2 UAV model using quaternions to represent the rotations

$$q_0 + q_1 i + q_2 j + q_3 k,$$

From quaternion to rotation matrix

$$\begin{aligned}
\mathbf{C}_{\mathcal{AD}} &= \mathbb{I}_{3\times 3} + 2\xi_0 \left[\check{\boldsymbol{\xi}}\right]_\times + 2\left[\check{\boldsymbol{\xi}}\right]_\times^2 = \left(2\xi_0^2 - 1\right)\mathbb{I}_{3\times 3} + 2\xi_0 \left[\check{\boldsymbol{\xi}}\right]_\times + 2\check{\boldsymbol{\xi}} \\
&= \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_0q_1 + 2q_2q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R_z(\psi)R_y(\theta)R_x(\phi)\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\
&= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}
\end{aligned}$$

From rotation matrix to Euler angles

$$\begin{aligned}
\psi &= \tan^{-1}\frac{C_{12}}{C_{11}} \\
\theta &= -\sin^{-1} C_{13} \\
\phi &= \tan^{-1}\frac{C_{23}}{C_{33}}
\end{aligned}$$

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{pmatrix} = \frac{1}{2} \begin{bmatrix} q_1 & q_4 & -q_3 & q_2 \\ q_2 & q_3 & q_4 & -q_1 \\ q_3 & -q_2 & q_1 & q_4 \\ q_4 & -q_1 & -q_2 & -q_3 \end{bmatrix} \begin{pmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}$$

$$\dot{\boldsymbol{\xi}}_{\mathcal{IB}} = \frac{1}{2}\mathbf{H}(\boldsymbol{\xi}_{\mathcal{IB}})^T {}_{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}} = \mathbf{E}_{R,quat}^{-1}\dot{\boldsymbol{\chi}}_{R,quat},$$

$$\begin{aligned} \mathbf{H}(\boldsymbol{\xi}) &= \begin{bmatrix} -\check{\boldsymbol{\xi}} & [\check{\boldsymbol{\xi}}]_\times + \xi_0 \mathbb{I}_{3\times 3} \end{bmatrix} \in \mathbb{R}^{3\times 4} \\ &= \begin{bmatrix} -\xi_1 & \xi_0 & -\xi_3 & \xi_2 \\ -\xi_2 & \xi_3 & \xi_0 & -\xi_1 \\ -\xi_3 & -\xi_2 & \xi_1 & \xi_0 \end{bmatrix}. \end{aligned}$$

$x_1 = [x, y, z]^T - position$

$x_2 = [\dot{x}, \dot{y}, \dot{z}]^T - linear\ \ velocity$

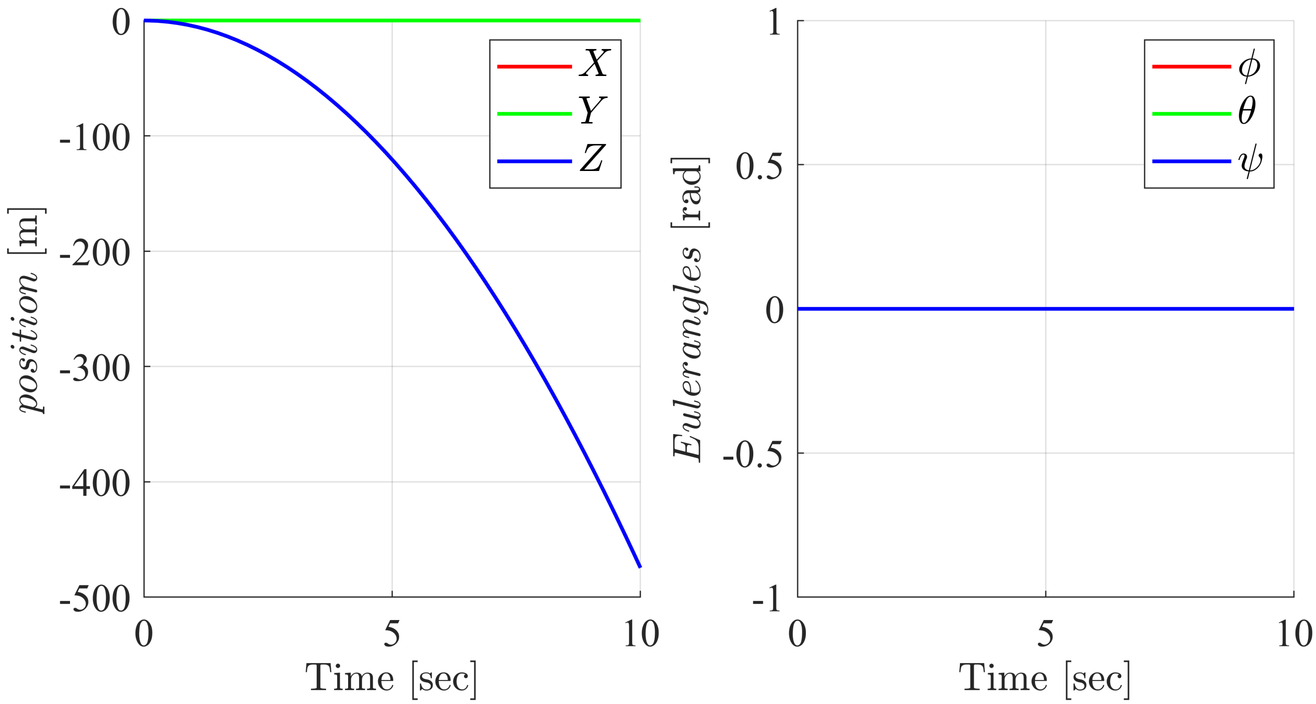
$x_3 = [q_0, q_1, q_2, q_3]^T - quaternion$

$x_4 = [\omega_x, \omega_y, \omega_z]^T - angular\ \ velocity$

```
% quaternion initial condition
quat = [1,0,0,0]';
```

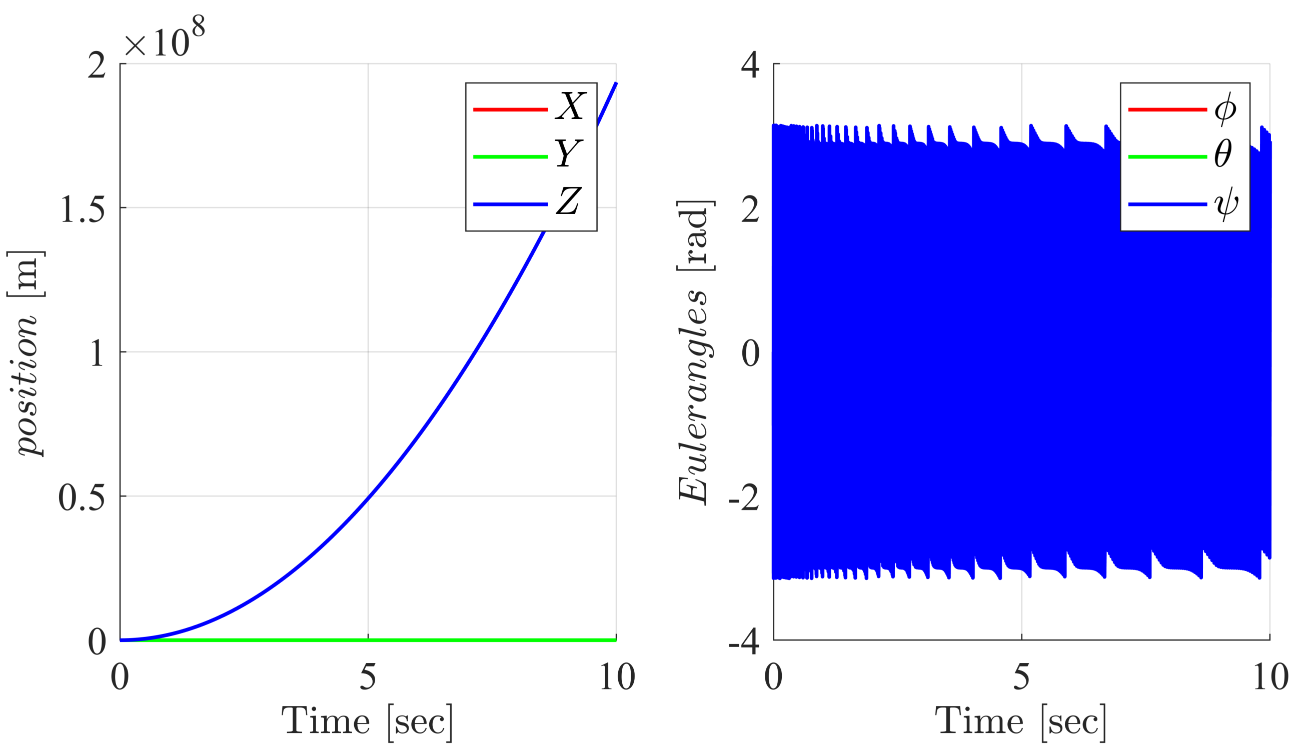
1) Make a plot of p and Θ, given Ω = [0, 0, 0, 0]T and explain the result

```
TakeStep = 1; %setting TakeStep 1 take the stepinput, setting TakeStep to 0 take the IN
RPM = [0, 0, 0, 0]';
simOut = sim('My_Nonlinear_Quaternions.slx');
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



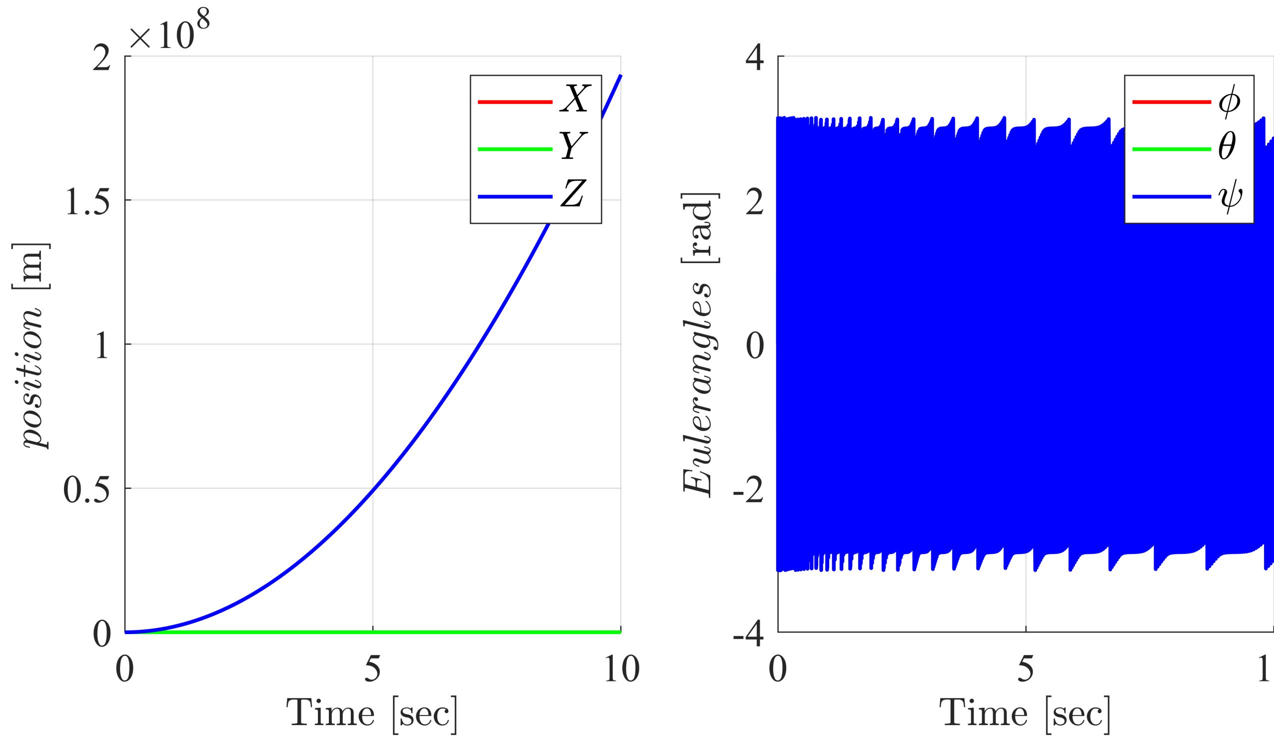
2) Make a plot of p and Θ, given Ω = [10000, 0, 10000, 0]T and explain the result

```
TakeStep = 1; %setting TakeStep 1 take the stepinput, setting TakeStep to 0 take the IN
RPM = [10000, 0, 10000, 0]';
simOut = sim('My_Nonlinear_Quaternions.slx');
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



3) Make a plot of p and Θ, given Ω = [0, 10000, 0, 10000]T and explain the result

```
TakeStep = 1; %setting TakeStep 1 take the stepinput, setting TakeStep to 0 take the IN
RPM = [0, 10000, 0, 10000]';
simOut = sim('My_Nonlinear_Quaternions.slx');
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```



Exercise 2.3 Linearize the dynamic model of the UAV in hovering conditions

```
x = Simulink.BlockDiagram.getInitialState('My_Nonlinear_new')
```

```
x =
Simulink.SimulationData.Dataset 'xFinal' with 4 elements

                        Name   BlockPath
                        ____   ________________________________________

    1  [1x1 State]      ''     ...r_new/Angular Acceleration Integrator
    2  [1x1 State]      ''     ...linear_new/Euler Angle Rate Itegrator
    3  [1x1 State]      ''     ...ar_new/Linera Acceleration Integrator
    4  [1x1 State]      ''     ...linear_new/Linera Velocity Integrator

  - Use braces { } to access, modify, or add elements using index.
```

Find hovering conditions numerically, using Matlab function *trim()*

```
TakeStep = 0;
p = [0,0,0]';
Theta = [0, 0, 0]';
x0 = [p;p_dot;Theta;omega]
```

```
x0 = 12×1
     0
     0
```

```
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

```
x0 = zeros(12,1);
u0 = ones(4,1);
y0 = [];

ix = [1 2 3 7 8 9];
iu = [];
iy = [];
[xss, uss, yss] = trim('Exercise_2_2_1_Nonlinear_new', x0, u0, y0, ix, iu, iy); % Exercise_2_2_
xss
```

```
xss = 12×1
10^-14 ×
    0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    0.1197
         0
         0
         0
   -0.0000
      ⋮
```

```
yss
```

```
yss = 6×1
10^-14 ×
    0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    0.1197
```

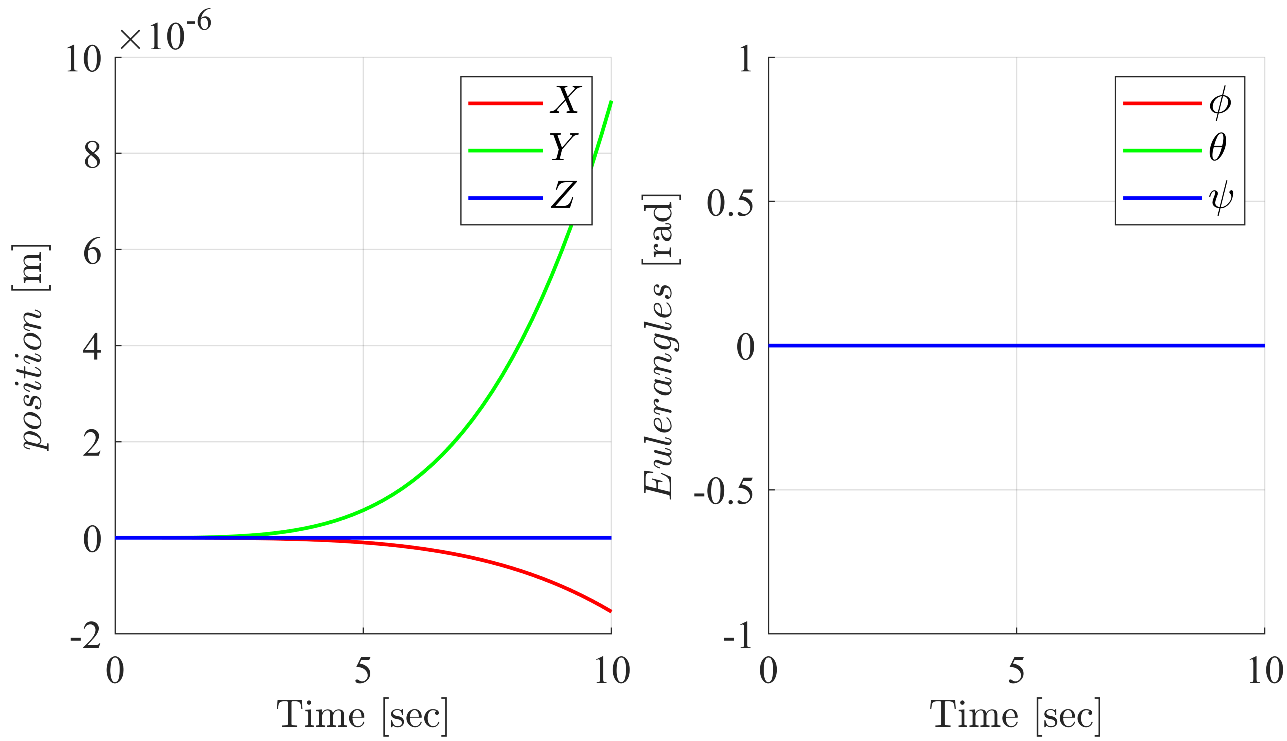
```
uss
```

```
uss = 4×1
   11.0736
   11.0736
   11.0736
   11.0736
```

Test the nonlinear model around operation point $\mathbf{P} = [0,0,0]^T,\ \Theta = [0,0,0]^T\ \ u = uss$

```
TakeStep = 1;
p = [0,0,0]';
```

```
Theta = [0, 0, 0]';
RPM = uss;
simOut = sim('My_Nonlinear_new.slx'); %My_Nonlinear_new
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
subplot(1,2,2)
xlim('auto')
ylim([-1,1])
```



Linearlize the nonlinear model around operation point

```
TakeStep = 0;
[Aa, Bb, Cc, Dd] = linmod('My_Nonlinear_new', x0, uss)
```

```
Aa = 12×12                                          Rows 3:12 | Columns 1:12
         0         0         0         0         0    1.0000         0         0
         0         0         0   -0.0100         0         0         0    9.8100
         0         0         0         0   -0.0100         0   -9.8100         0
         0         0         0         0         0   -0.0100         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
Bb = 12×4    Rows 3:12 | Columns 1:4
10^4 ×
         0         0         0         0
         0         0         0         0
         0         0         0         0
    0.0000    0.0000    0.0000    0.0000
         0         0         0         0
         0         0         0         0
         0         0         0         0
    1.6610         0   -1.6610         0
```

```
         0    1.6610         0   -1.6610
    2.2147   -2.2147    2.2147   -2.2147
Cc = 6×12
     1     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     0
Dd = 6×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

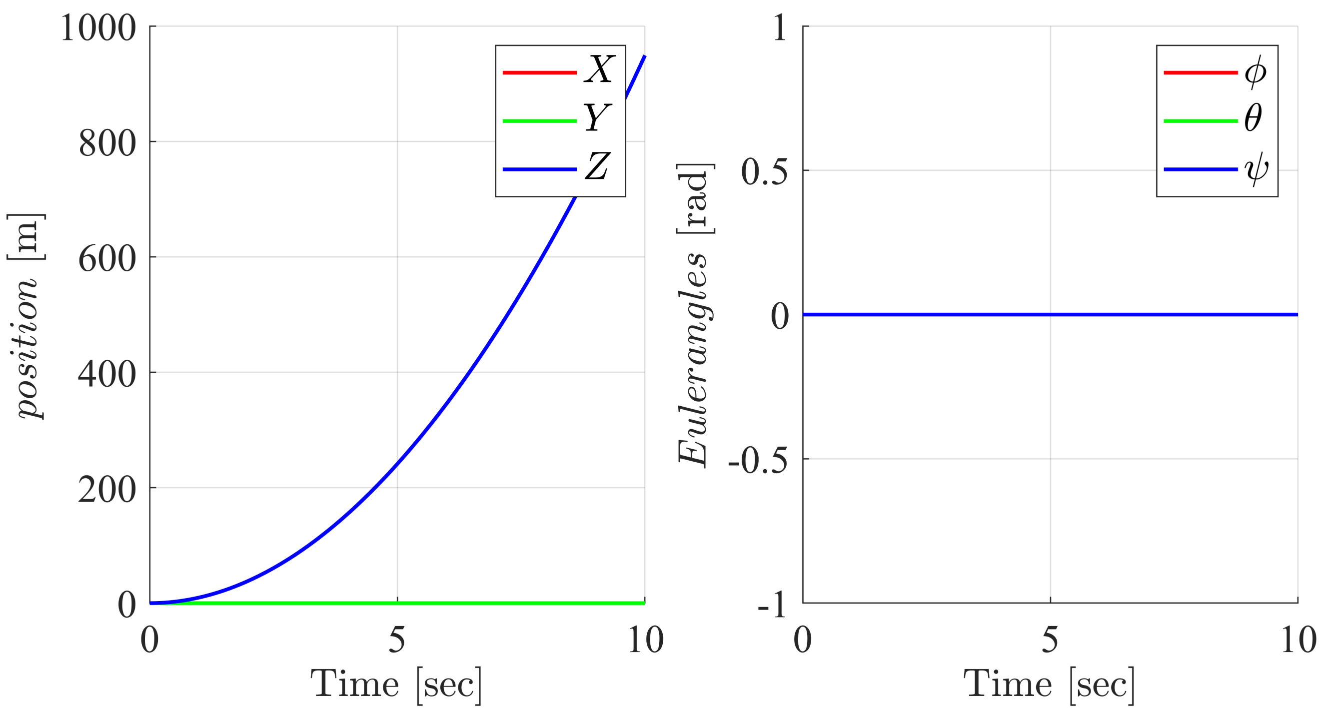
```
sys = ss(Aa, Bb, Cc, Dd)
```

```
sys =

  A = 
          x1     x2     x3     x4     x5     x6     x7     x8     x9    x10    x11    x12
   x1      0      0      0      1      0      0      0      0      0      0      0      0
   x2      0      0      0      0      1      0      0      0      0      0      0      0
   x3      0      0      0      0      0      1      0      0      0      0      0      0
   x4      0      0      0  -0.01      0      0      0   9.81      0      0      0      0
   x5      0      0      0      0  -0.01      0  -9.81      0      0      0      0      0
   x6      0      0      0      0      0  -0.01      0      0      0      0      0      0
   x7      0      0      0      0      0      0      0      0      0      1      0      0
   x8      0      0      0      0      0      0      0      0      0      0      1      0
   x9      0      0      0      0      0      0      0      0      0      0      0      1
   x10     0      0      0      0      0      0      0      0      0      0      0      0
   x11     0      0      0      0      0      0      0      0      0      0      0      0
   x12     0      0      0      0      0      0      0      0      0      0      0      0

  B = 
               u1          u2          u3          u4
   x1           0           0           0           0
   x2           0           0           0           0
   x3           0           0           0           0
   x4           0           0           0           0
   x5           0           0           0           0
   x6      0.4429      0.4429      0.4429      0.4429
   x7           0           0           0           0
   x8           0           0           0           0
   x9           0           0           0           0
   x10  1.661e+04           0  -1.661e+04           0
   x11          0   1.661e+04           0  -1.661e+04
   x12  2.215e+04  -2.215e+04   2.215e+04  -2.215e+04

  C = 
        x1   x2   x3   x4   x5   x6   x7   x8   x9  x10  x11  x12
   y1    1    0    0    0    0    0    0    0    0    0    0    0
   y2    0    1    0    0    0    0    0    0    0    0    0    0
   y3    0    0    1    0    0    0    0    0    0    0    0    0
   y4    0    0    0    0    0    0    1    0    0    0    0    0
   y5    0    0    0    0    0    0    0    1    0    0    0    0
   y6    0    0    0    0    0    0    0    0    1    0    0    0

  D = 
       u1  u2  u3  u4
   y1   0   0   0   0
   y2   0   0   0   0
```

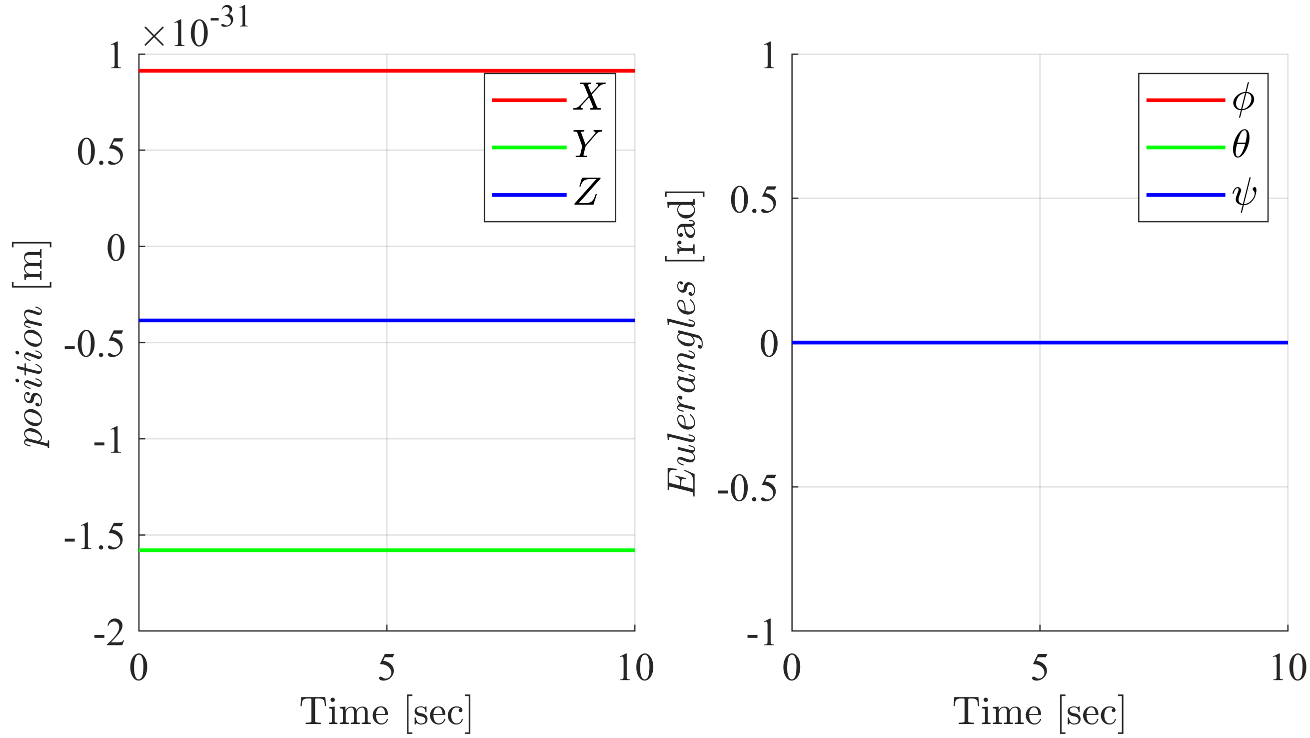
```
    y3   0   0   0   0
    y4   0   0   0   0
    y5   0   0   0   0
    y6   0   0   0   0

  Continuous-time state-space model.
```

Test the linear model around operation point $\mathbf{P} = [0,0,0]^T,\ \Theta = [0,0,0]^T,\ u = uss$

```
p = [0,0,0]';
Theta = [0, 0, 0]';
x0 = [p;p_dot;Theta;omega];

t = 0:0.1:10;
RPM = uss;
u = ones(length(t), 1) * RPM';
y = lsim(sys, u, t, x0);
plotSimOutput(t, y')
xlim('auto')
ylim([-1,1])
```

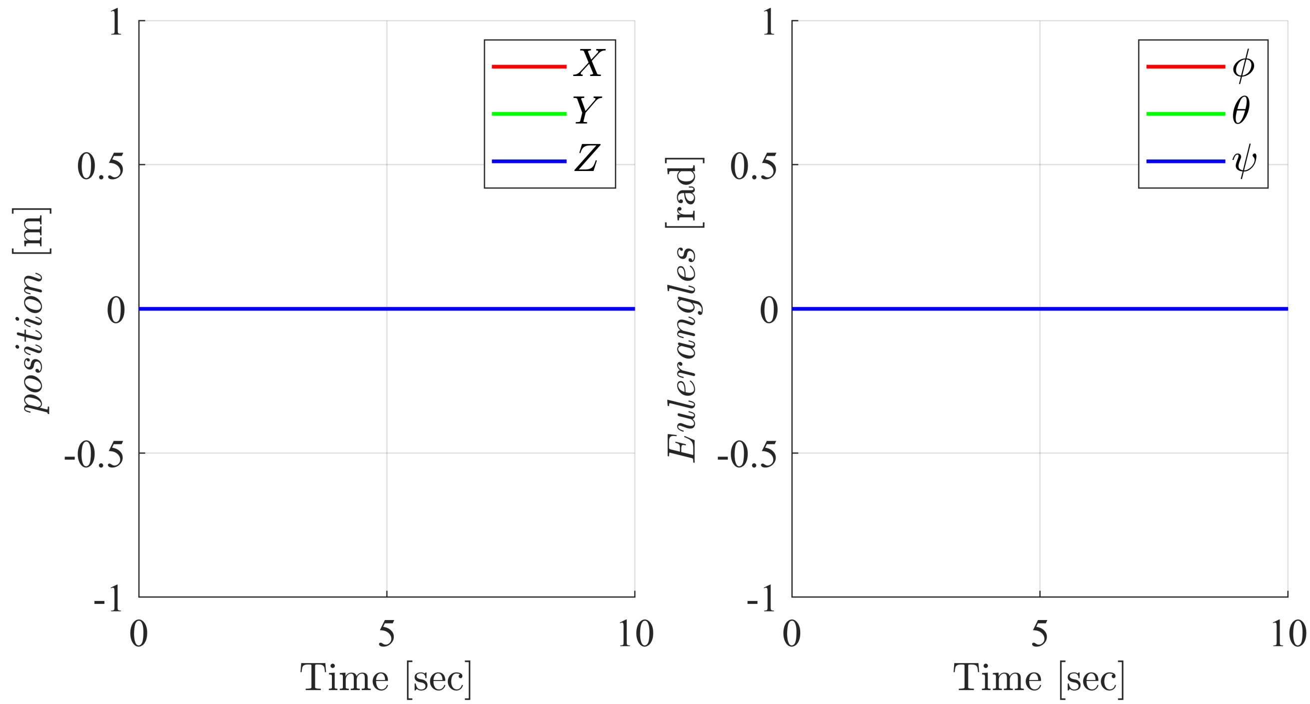



```
TakeStep = 1;
p = [0,0,0]';
Theta = [0, 0, 0]';
x0 = [p;p_dot;Theta;omega];
RPM = uss;
simOut = sim('My_Linear_new.slx'); %My_Linear_new Exercise_2_2_1_Linear %% difference is using
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
subplot(1,2,2)
xlim('auto')
ylim([-1,1])
```

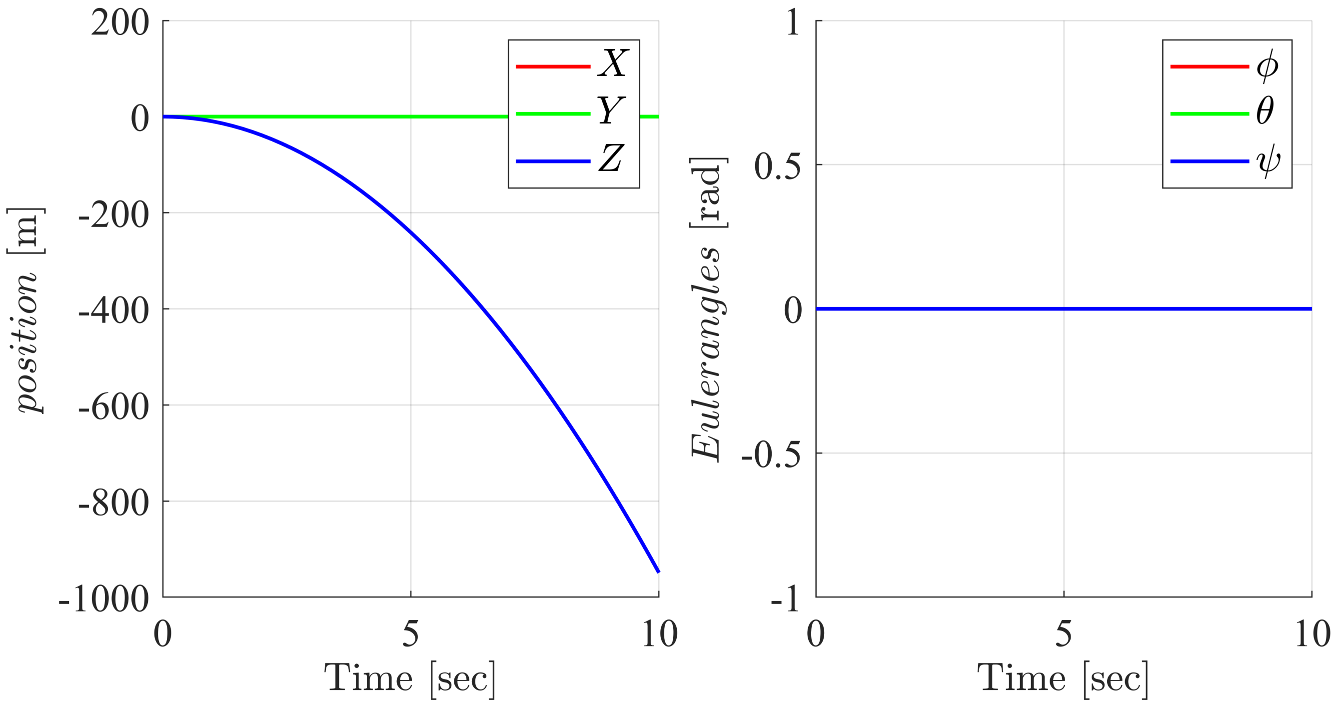


1) Make a plot of p and Θ, given Ω = [0, 0, 0, 0]T and explain the result

```
p = [0,0,0]';
Theta = [0, 0, 0]';
x0 = [p;p_dot;Theta;omega];

t = 0:0.1:10;
RPM = [0,0,0,0]';
u = ones(length(t), 1) * RPM';
y = lsim(sys, u, t, x0);
plotSimOutput(t, y')
xlim('auto')
ylim([-1,1])
```

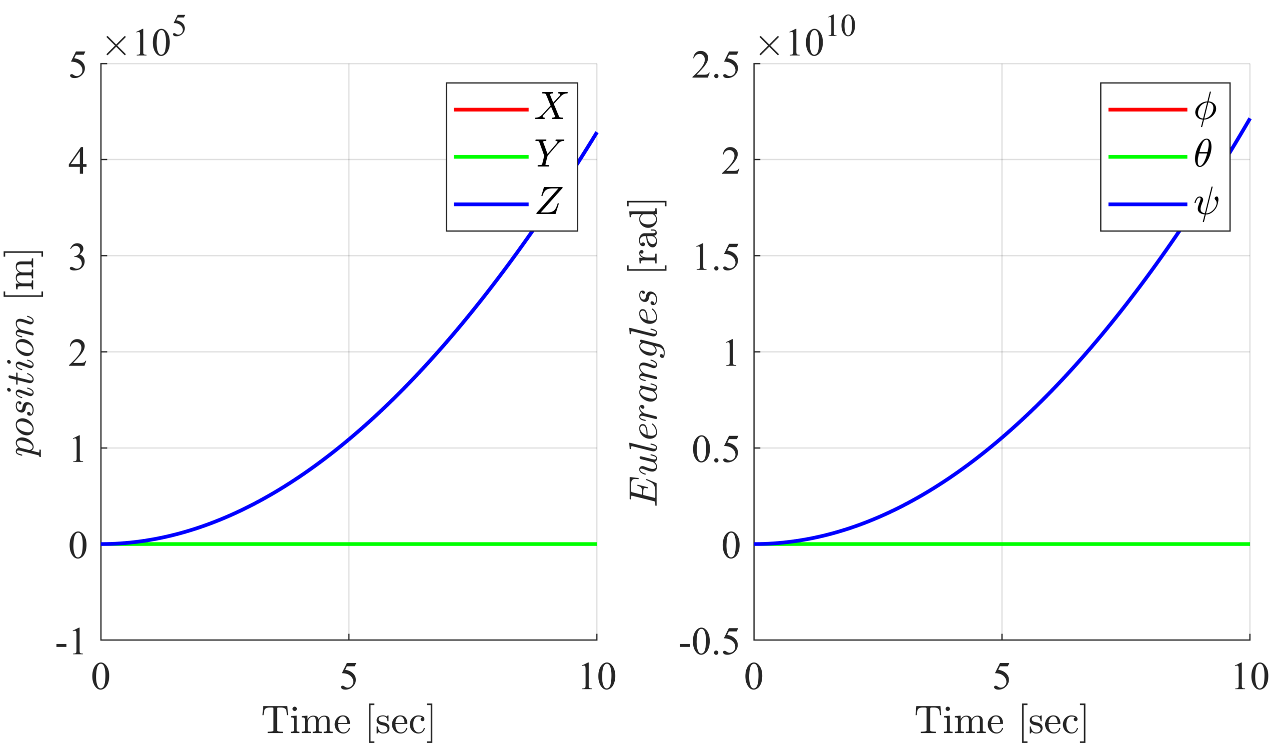



```
TakeStep = 1;
p = [0,0,0]';
Theta = [0, 0, 0]';
x0 = [p;p_dot;Theta;omega];
RPM = [0,0,0,0]';
simOut = sim('My_Linear_new.slx'); %My_Linear_new Exercise_2_2_1_Linear %% difference is using
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
subplot(1,2,2)
xlim('auto')
ylim([-1,1])
```



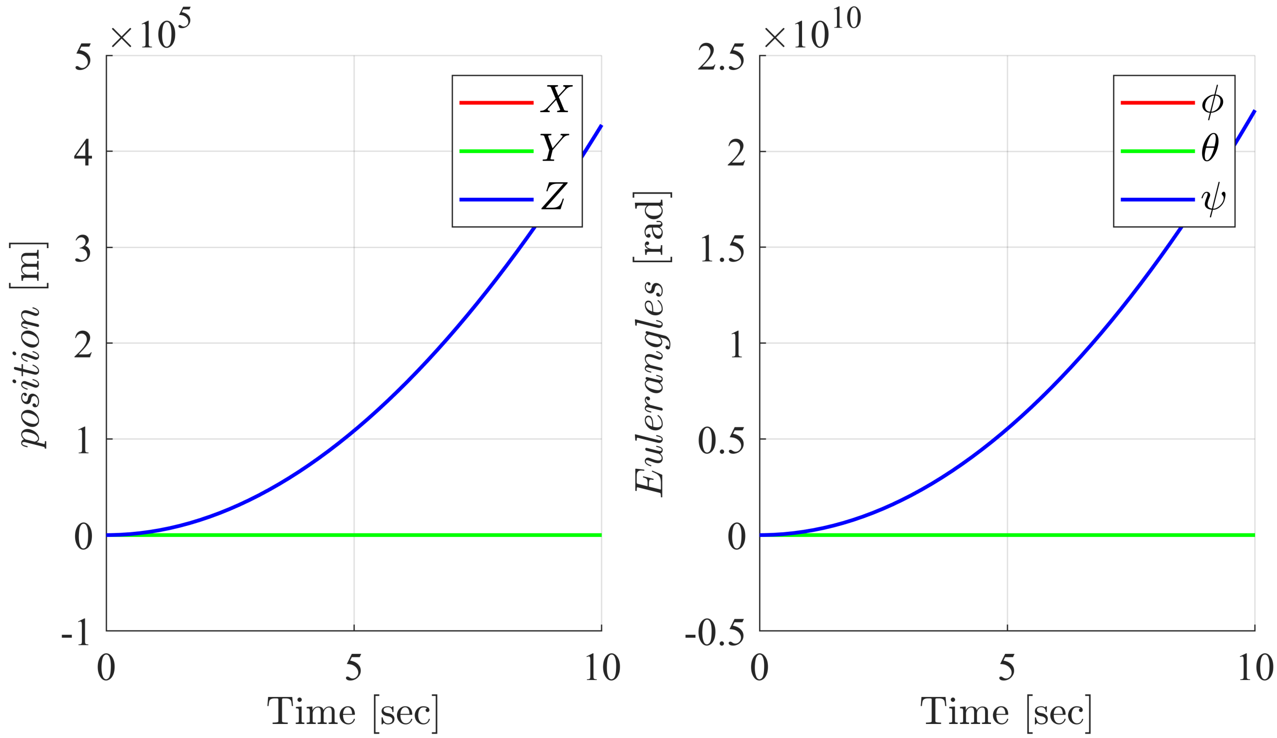
2) Make a plot of p and Θ, given Ω = [10000, 0, 10000, 0]T and explain the result

```
RPM = [10000,0,10000,0]';
u = ones(length(t), 1) * RPM';
y = lsim(sys, u, t, x0);
plotSimOutput(t, y')
```



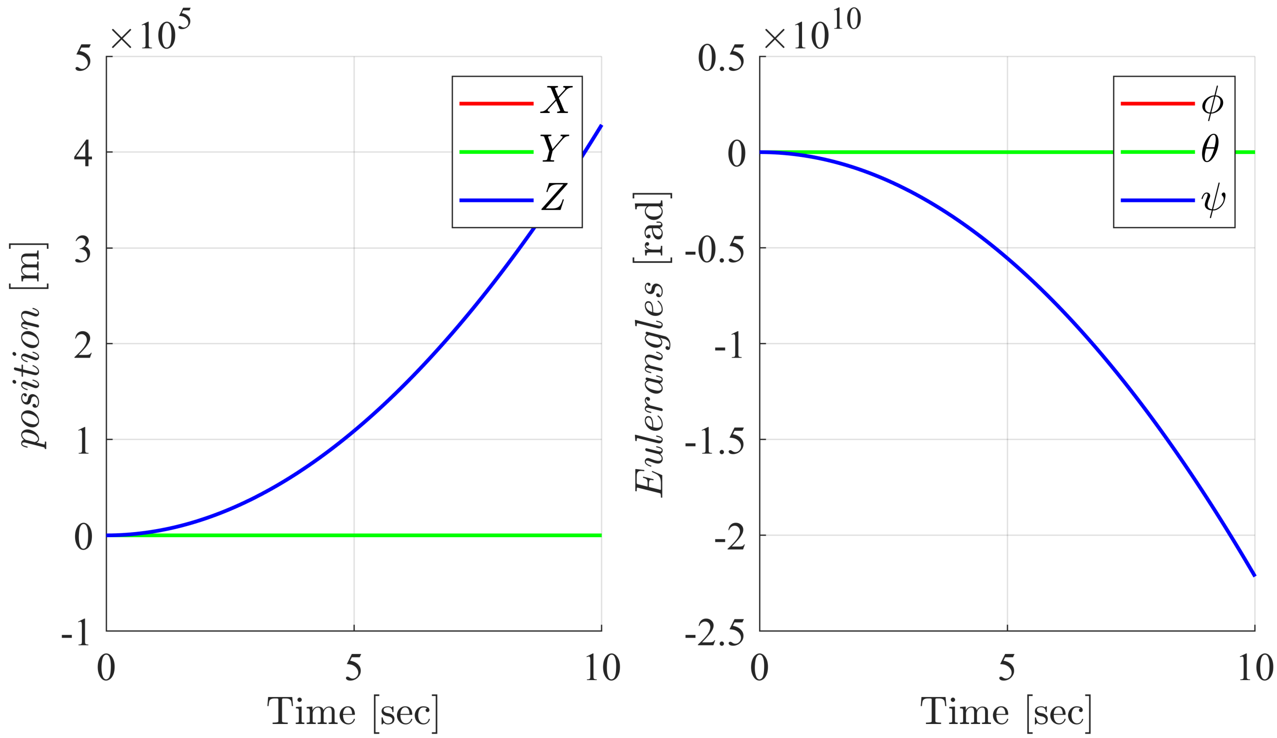

```
TakeStep = 1;
```

```
RPM = [10000,0,10000,0]';
simOut = sim('My_Linear_new.slx'); %My_Linear
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```

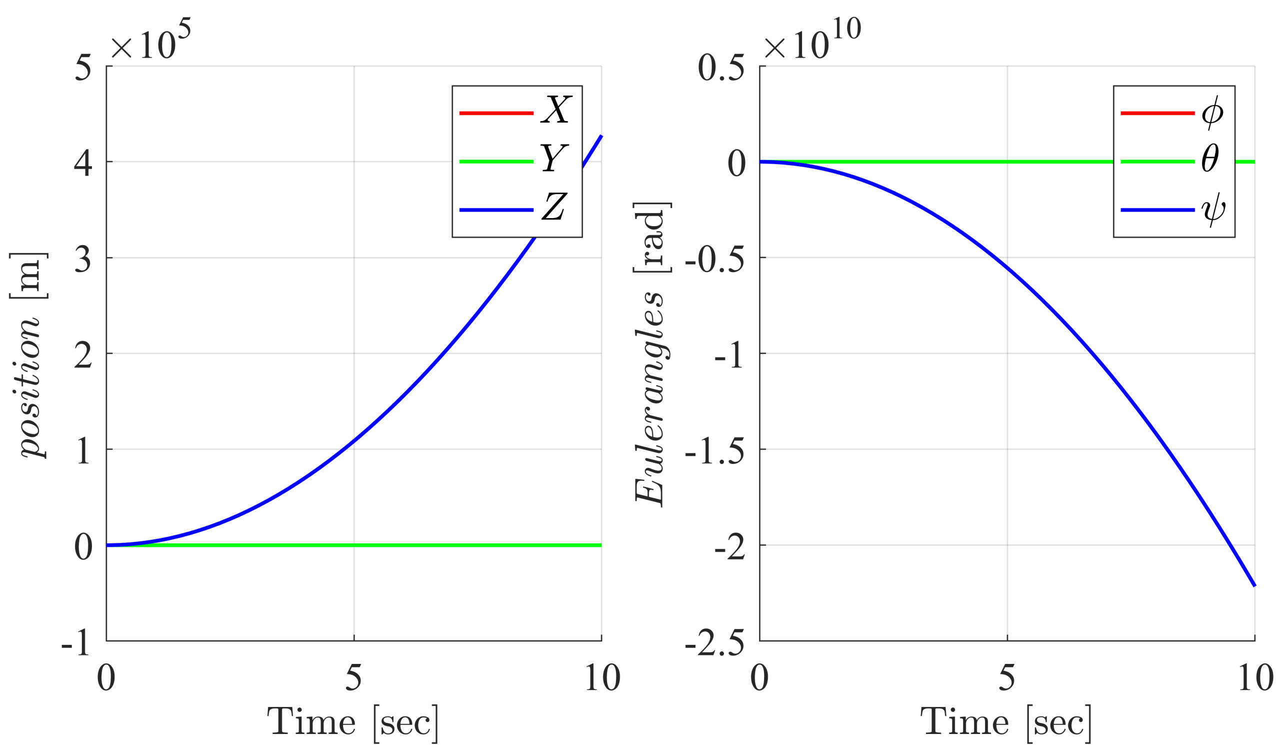


3) Make a plot of p and Θ, given Ω = [0, 10000, 0, 10000]T and explain the result

```
RPM = [0, 10000, 0, 10000]';
u = ones(length(t), 1) * RPM';
y = lsim(sys, u, t, x0);
plotSimOutput(t, y')
```




```
TakeStep = 1;
RPM = [0, 10000, 0, 10000]';
simOut = sim('My_Linear_new.slx'); %My_Linear
plotSimOutput(simOut.yout{1}.Values.Time, simOut.yout{1}.Values.Data)
```




```
function plotSimOutput(time, data)
```

```
    posX = data(1,:);
    posY = data(2,:);
    posZ = data(3,:);

    angPhi      = data(4,:);
    angTheta    = data(5,:);
    angPsi      = data(6,:);


    figure, h1 = subplot(1,2,1); set(h1,'FontName','times','FontSize',16)
    hold on, grid on
    plot(time, posX,'r', time, posY,'g', time, posZ,'b','LineWidth',1.5)
    xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
    ylabel('$position$ [m]','FontName','times','FontSize',16,'Interpreter','latex')
    legend('$X$', '$Y$', '$Z$', 'FontName','times','FontSize',16,'Interpreter','latex')
    h2 = subplot(1,2,2); set(h2,'FontName','times','FontSize',16)
    hold on, grid on
    plot(time, angPhi,'r', time, angTheta,'g', time, angPsi,'b', 'LineWidth',1.5)
    xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
    ylabel('$Euler angles$ [rad]','FontName','times','FontSize',16,'Interpreter','latex')
    legend('$\phi$', '$\theta$', '$\psi$', 'FontName','times','FontSize',16,'Interpreter','late
    set(gcf,'units','points','position',[0,0,600,300])
end
```