MAE 4733 - Automatic Control Systems: Final Project

Diego Colón

12-5-2020

```
clear all
```

Part 1

1.1 - State Space Model

Taking the eqautions provided in the exam can be expressed in a state space form ($x = f(x,u), y = \phi(x,h)$) , with state $x = \begin{bmatrix} h & \dot{h} & z & \dot{z} & \theta & \dot{\theta} \end{bmatrix}^T$ and input $u = \begin{bmatrix} F & \tau \end{bmatrix}^T$ as:

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{h} \\ \frac{F\cos(\theta)}{M+2m} - g \\ \dot{z} \\ -\frac{\mu \dot{z}}{M+2m} - \frac{F\sin(\theta)}{M+2m} \\ \dot{\theta} \\ \frac{\tau}{2md^2+J} \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{u_1\cos(x_5)}{M+2m} - g \\ x_4 \\ -\frac{\mu x_4}{M+2m} - \frac{u_1\sin(x_5)}{M+2m} \\ x_6 \\ \frac{u_2}{2md^2+J} \end{bmatrix}$$

$$y = \phi(x,u) = \begin{bmatrix} h \\ z \\ \theta \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix}$$

1.2 - Equilibium Trajectory

In order to create controllers, with the techniques learned in class, it is required to linearize the dynamics about an equilibrium trajectory. Equilibrium trajectories and corresponding measurementsare defined as:

$$(x^*(t), u^*(t)) \ni \dot{x}(t) = f(x^*(t), u^*(t)) = 0 \, \forall \, t \geq 0$$

$$y^*(t) = \phi(x^*(t), u^*(t))$$

```
%% Create all the system variables
syms s
syms h h_ast h_dot h_ddot z z_dot z_ddot theta theta_dot theta_ddot
syms g M m mu d J
syms F tau

%% Create symbols and values array for substitution
k = [g; M; m;  mu; d; J];
k_val = [9.81; 1; 0.25;  0.1; 0.3; 0.0042];

%% Write the VTOL Dynamics
h_ddot = -g +(cos(theta)*F)/(M+2*m);
z_ddot = -(mu*z_dot)/(M+2*m) - (sin(theta)*F)/(M+2*m);
theta_ddot = tau/(2*m*d^2 + J);

%% Create x, u, f(x,u) and phi(x,u)
x = [h; h_dot; z; z_dot; theta; theta_dot];
u = [F;tau];
process = [h_dot; h_ddot; z_dot; z_ddot; theta_dot; theta_ddot];
meas = [h; z; theta];

%% Solve for trajectory to linearize about
S = solve(process,[x;u]);
x_star = [S.h+10;S.h_dot;S.z;S.z_dot;S.theta;S.theta_dot]
```

```
x_star =
```

$$\begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```
u_star = [S.F;S.tau]
```

```
u_star =
```

$$\begin{pmatrix} g\,(M+2\,m) \\ 0 \end{pmatrix}$$

```
y_star= [S.h+10;S.z;S.theta]
```

y_star =

$$\begin{pmatrix} 10 \\ 0 \\ 0 \end{pmatrix}$$

1.3 - Linearize Dynamics

Once a equilibrium trajectory has been found, the dynamics must be linearized about them to express them in the form

$$\dot{x}_\delta = A\ x_\delta + B\ u_\delta$$
$$y_\delta = C\ x_\delta + D\ u_\delta$$

```
A = simplify(subs(jacobian(process,x),[x;u],[x_star;u_star]))
```

A =

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\mu}{M+2m} & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
B = simplify(subs(jacobian(process,u),[x;u],[x_star;u_star]))
```

B =

$$\begin{pmatrix} 0 & 0 \\ \frac{1}{M+2m} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{2md^2+J} \end{pmatrix}$$

```
C = simplify(subs(jacobian(meas,x),[x;u],[x_star;u_star]))
```

C =

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

```
D = simplify(subs(jacobian(meas,u),[x;u],[x_star;u_star]))
```

D =

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

```
x_star = subs(x_star,k,k_val);
u_star = subs(u_star,k,k_val);
```

1.4 - MIMO TF

Since bicopter is a multi-input, multi-output system, the system can be written as a MIMO tranfer function. In this case that transfer function is of the form

$$\begin{bmatrix} H(s) \\ Z(s) \\ \Theta(s) \end{bmatrix} = \begin{bmatrix} G_{\mathrm{HF}}(s) & G_{F\mathrm{T}}(s) \\ G_{\mathrm{ZF}}(s) & G_{Z\mathrm{T}}(s) \\ G_{\Theta F}(s) & G_{\Theta\mathrm{T}}(s) \end{bmatrix} \begin{bmatrix} F(s) \\ \mathrm{T}(s) \end{bmatrix}$$

```
G = collect(simplify(C*inv(s*eye(size(A, 1)) - A)*B + D),s)
```

G =

$$\begin{pmatrix} \frac{1}{(M+2m)\,s^2} & 0 \\ 0 & \frac{-M\,g-2\,g\,m}{(4\,d^2\,m^2+2\,M\,d^2\,m+2\,J\,m+J\,M)\,s^4+(2\,m\,\mu\,d^2+J\,\mu)\,s^3} \\ 0 & \frac{1}{(2\,m\,d^2+J)\,s^2} \end{pmatrix}$$

1.5 - Relevant Transfer Functions

Taking the nonzero transfer functions out of the MIMO transfer function

```
G_HF = collect(simplify(subs(G(1,1),k,k_val)),s)
```

G_HF =

$$\frac{2}{3\,s^2}$$

```
G_ZT = collect(simplify(subs(G(2,2),k,k_val)),s)
```

G_ZT =

$$-\frac{122625}{615\,s^4+41\,s^3}$$

```
G_ThetaT = collect(simplify(subs(G(3,2),k,k_val)),s)
```

G_ThetaT =

$$\frac{2500}{123\,s^2}$$

1.6 - Missing TF

Given the nature of the linearized dynimcs and that we would like to implement a outer-inner loop architecture, it is required to know the relationship between Z and Θ.

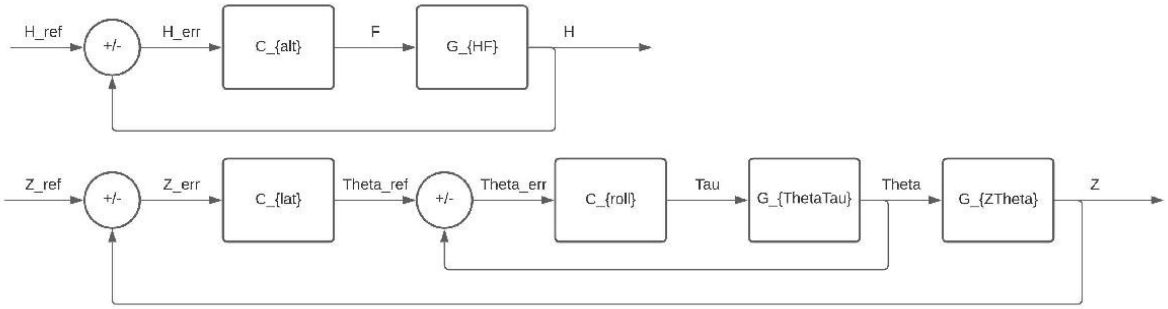
$$G_{Z\Theta}(s)=\frac{G_{ZT}(s)}{G_{\Theta T}(s)}$$

```
G_ZTheta = collect(simplify(G_ZT/G_ThetaT),s)
```

G_ZTheta =

$$-\frac{2943}{300\,s^2+20\,s}$$

1.7 - Controller Architecture



1.8 - Altitude Controller Design (Pole Placement)

```
syms Kp Kd Ki
Cp1 = Kp + Kd*s;
CL1 = collect(simplify((G_HF*Cp1)/(1 + G_HF*Cp1)),s)
```

CL1 =

$$\frac{(2\,\mathrm{Kd})\,s+2\,\mathrm{Kp}}{3\,s^2+(2\,\mathrm{Kd})\,s+2\,\mathrm{Kp}}$$

```
[Num,Den] = numden(G_HF);
G_HF= tf(sym2poly(Num),sym2poly(Den));

t_r = 2;
zeta = 0.707;

omega_n = (2.16*zeta+0.6)/t_r;
Kp_alt = (3/2)*omega_n^2;
Kd_alt = 3*zeta*omega_n;
C_alt = tf([Kd_alt,Kp_alt,0],[1,0])
```

```
C_alt =

  2.256 s^2 + 1.697 s
  -------------------
           s

Continuous-time transfer function.
```

```
CL_alt = minreal(feedback(C_alt*G_HF,1))
```

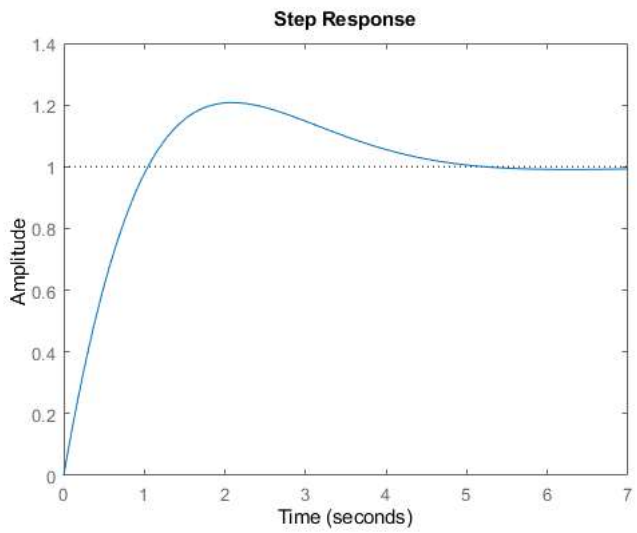
```
CL_alt =

     1.504 s + 1.131
  ----------------------
  s^2 + 1.504 s + 1.131

Continuous-time transfer function.
```

```
stepinfo(CL_alt)
```

```
ans = struct with fields:
        RiseTime: 0.7956
    SettlingTime: 4.6010
```

```
      SettlingMin: 0.9331
      SettlingMax: 1.2079
        Overshoot: 20.7910
       Undershoot: 0
             Peak: 1.2079
         PeakTime: 2.0823
```

```
step(CL_alt)
```




```
isstable(CL_alt)
```

```
ans = logical
   1
```

```
% System did not meet specificaltions, redesign iterations began.
% Final Design:

zeta = 0.8;
omega_n = 1;
Kp_alt = (3/2)*omega_n^2;
Kd_alt = (3/2)*(2*zeta*omega_n);
C_alt = tf([Kd_alt,Kp_alt,0],[1,0])
```

```
C_alt =

  2.4 s^2 + 1.5 s
  ---------------
        s

Continuous-time transfer function.
```

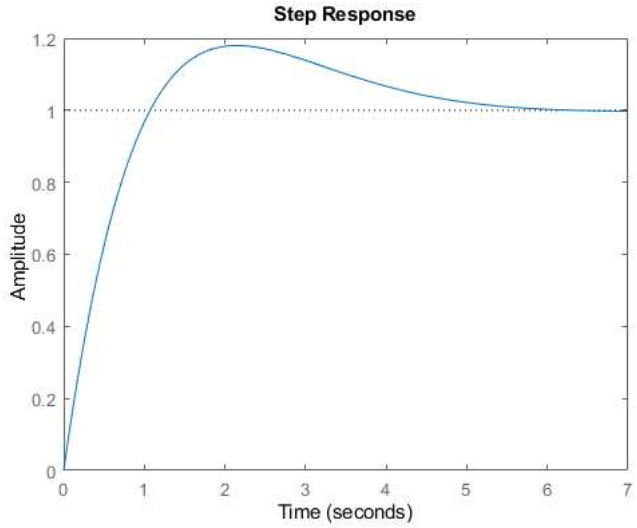
```
CL_alt = minreal(feedback(C_alt*G_HF,1))
```

```
CL_alt =

     1.6 s + 1
  ---------------
  s^2 + 1.6 s + 1

Continuous-time transfer function.
```

```
stepinfo(CL_alt)
```

```
ans = struct with fields:
        RiseTime: 0.8068
    SettlingTime: 5.0537
     SettlingMin: 0.9276
     SettlingMax: 1.1798
       Overshoot: 17.9763
      Undershoot: 0
            Peak: 1.1798
        PeakTime: 2.1299
```

```
step(CL_alt)
```




```
isstable(CL_alt)
```

```
ans = logical
   1
```

1.9.1 - Inner Loop Controller Design (Pole Placement)

```
syms Kp Kd Ki
Cp2 = Kp + Kd*s;
CL2 = collect(simplify((G_ThetaT*Cp2)/(1 + G_ThetaT*Cp2)),s)
```

```
CL2 =
(2500 Kd) s + 2500 Kp
---------------------
123 s^2 + (2500 Kd) s + 2500 Kp
```

```
[Num,Den] = numden(G_ThetaT);
G_ThetaTau = tf(sym2poly(Num),sym2poly(Den))
```

```
G_ThetaTau =

   2500
  -------
  123 s^2

Continuous-time transfer function.
```

```
t_r = 0.3;
M = 5;

zeta = abs(log(M/100))/sqrt(pi^2 + abs((log(M/100))^2));
omega_n = 1.8/t_r;

Kp_roll = (123/2500)*(omega_n^2);
Kd_roll = (123/2500)*(2*zeta*omega_n);
C_inner = tf([Kd_roll,Kp_roll,0],[1,0])
```

```
C_inner =

  0.4074 s^2 + 1.771 s
  --------------------
           s

Continuous-time transfer function.
```

```
CL_inner = minreal(feedback(C_inner*G_ThetaTau,1))
```

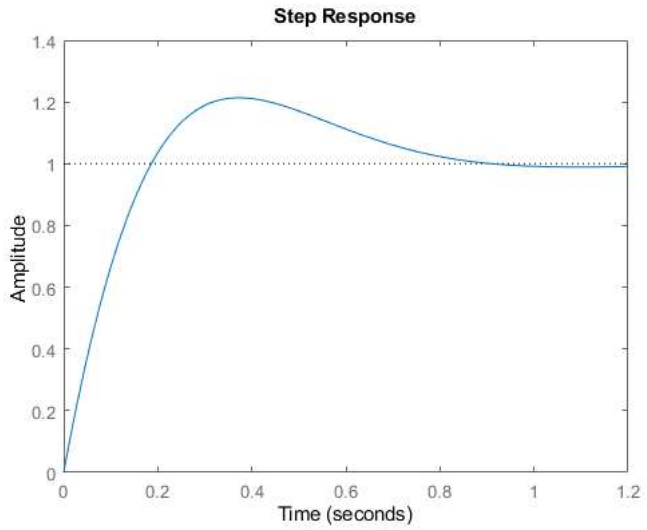
```
CL_inner =

     8.281 s + 36
  ------------------
  s^2 + 8.281 s + 36

Continuous-time transfer function.
```

```
stepinfo(CL_inner)
```

```
ans = struct with fields:
        RiseTime: 0.1423
    SettlingTime: 0.8111
```

```
      SettlingMin: 0.9038
      SettlingMax: 1.2136
        Overshoot: 21.3587
       Undershoot: 0
             Peak: 1.2136
         PeakTime: 0.3781
```

```
step(CL_inner)
```




```
isstable(CL_inner)
```

```
ans = logical
   1
```

```
% System did not meet specificaltions, redesign iterations began.
% Final Design:

a = -0.25;
b = -15;

Kp_roll = (a*b)/(2500/123);
Kd_roll = (-a-b)/(2500/123);
C_inner = tf([Kd_roll,Kp_roll,0],[1,0])
```

```
C_inner =

  0.7503 s^2 + 0.1845 s
  ---------------------
            s

Continuous-time transfer function.
```

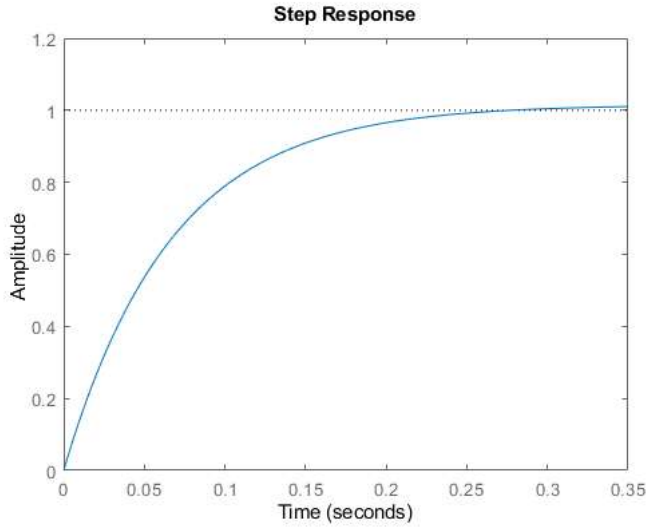
```
CL_inner = minreal(feedback(C_inner*G_ThetaTau,1))
```

```
CL_inner =

     15.25 s + 3.75
  --------------------
  s^2 + 15.25 s + 3.75

Continuous-time transfer function.
```

```
stepinfo(CL_inner)
```

```
ans = struct with fields:
        RiseTime: 0.1376
    SettlingTime: 0.2227
     SettlingMin: 0.9048
     SettlingMax: 1.0145
       Overshoot: 1.4507
      Undershoot: 0
            Peak: 1.0145
        PeakTime: 0.5557
```

```
step(CL_inner)
```




```
isstable(CL_inner)
```

```
ans = logical
   1
```

1.9.2 - Outer Loop Controller Design (Pole Placement)

```
syms Kp Kd Ki
Cp3 = Kp + Kd*s;
CL3 = collect(simplify((dcgain(CL_inner)*G_ZTheta*Cp1)/(1 + dcgain(CL_inner)*G_ZTheta*Cp1)),s)
```

```
CL3 =
```

$$\frac{(2943\,\mathrm{Kd})\,s + 2943\,\mathrm{Kp}}{-300\,s^2 + (2943\,\mathrm{Kd} - 20)\,s + 2943\,\mathrm{Kp}}$$

```
[Num,Den] = numden(G_ZTheta);
G_ZTheta = tf(dcgain(CL_inner)*sym2poly(Num),sym2poly(Den));

t_r = 3;
t_s = 8;

Kp_pos = (-300/2943)*(1.8/t_r)^2;
Kd_pos = (-300/2943)*(8/t_s) + 20/(-2943);
C_outer = tf([Kd_pos,Kp_pos,0],[1,0])
```

```
C_outer =

  -0.1087 s^2 - 0.0367 s
  ----------------------
            s

Continuous-time transfer function.
```

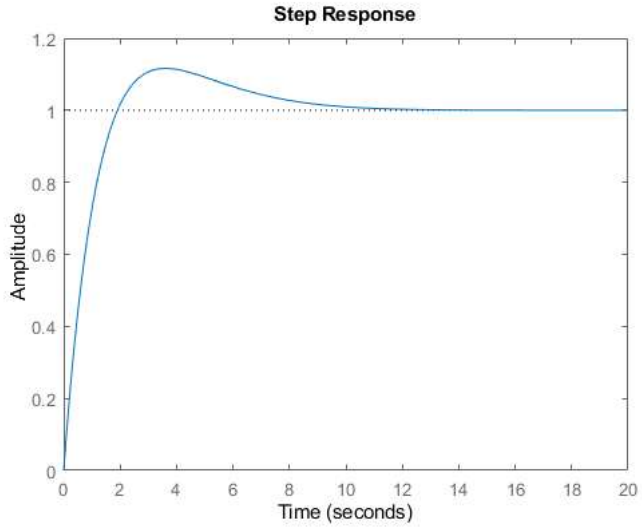
```
CL_outer = minreal(feedback(C_outer*G_ZTheta,1))
```

```
CL_outer =

     1.067 s + 0.36
  --------------------
  s^2 + 1.133 s + 0.36

Continuous-time transfer function.
```

```
stepinfo(CL_outer)
```

```
ans = struct with fields:
        RiseTime: 1.3778
    SettlingTime: 8.6187
     SettlingMin: 0.9192
     SettlingMax: 1.1163
       Overshoot: 11.6250
      Undershoot: 0
            Peak: 1.1163
        PeakTime: 3.5758
```

```
step(CL_outer)
```




```
isstable(CL_outer)
```

```
ans = logical
   1
```

```
% System did not meet specificaltions, redesign iterations began.
% Final Design:

Kp_pos = -0.002;
Kd_pos = -1;
C_outer = tf([Kd_pos,Kp_pos,0],[1,0])
```

```
C_outer =

  -s^2 - 0.002 s
  --------------
        s

Continuous-time transfer function.
```

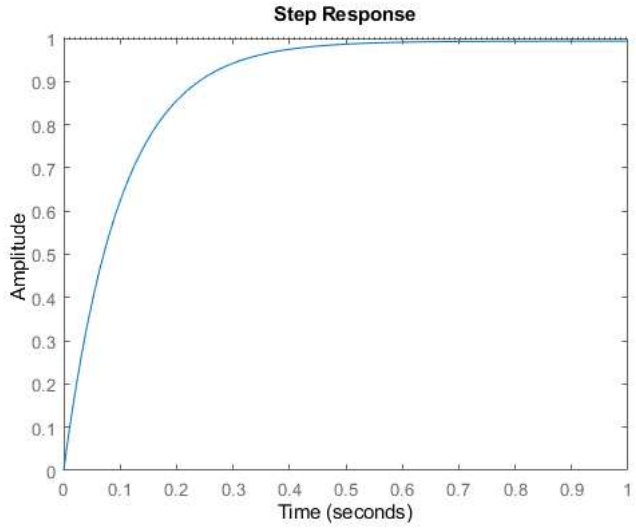
```
CL_outer = minreal(feedback(C_outer*G_ZTheta,1))
```

```
CL_outer =

     9.81 s + 0.01962
  -----------------------
  s^2 + 9.877 s + 0.01962

Continuous-time transfer function.
```

```
stepinfo(CL_outer)
```

```
ans = struct with fields:
        RiseTime: 0.2286
    SettlingTime: 0.4357
     SettlingMin: 0.9028
     SettlingMax: 0.9934
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9934
        PeakTime: 1.0680
```

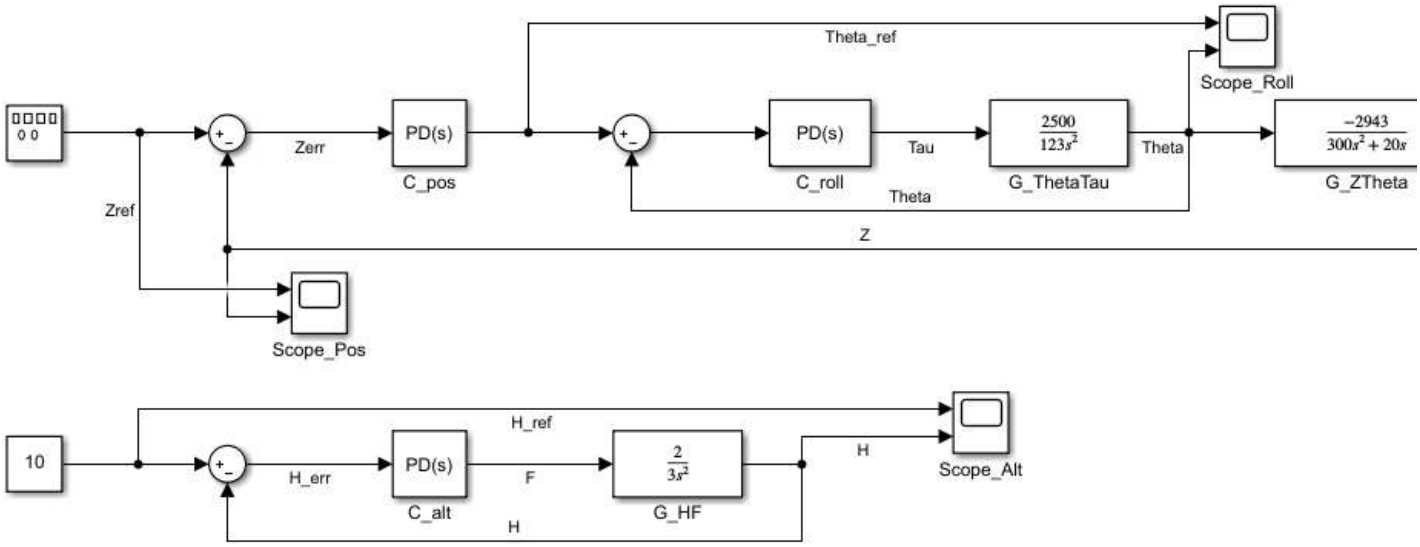
```
step(CL_outer)
```



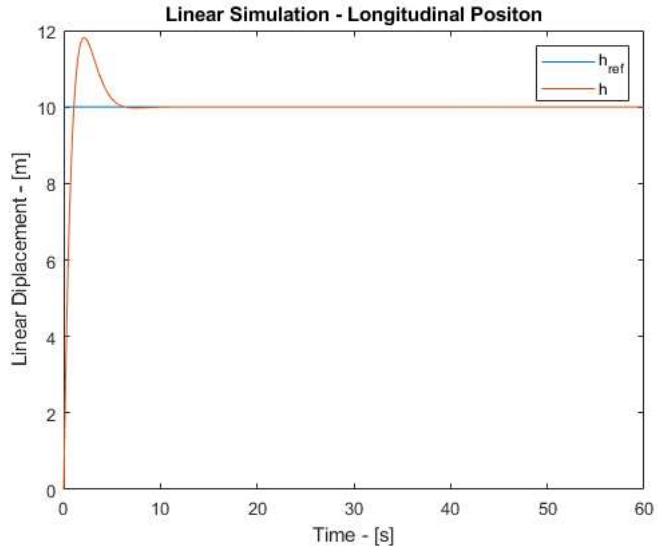

```
isstable(CL_outer)
```

```
ans = logical
   1
```

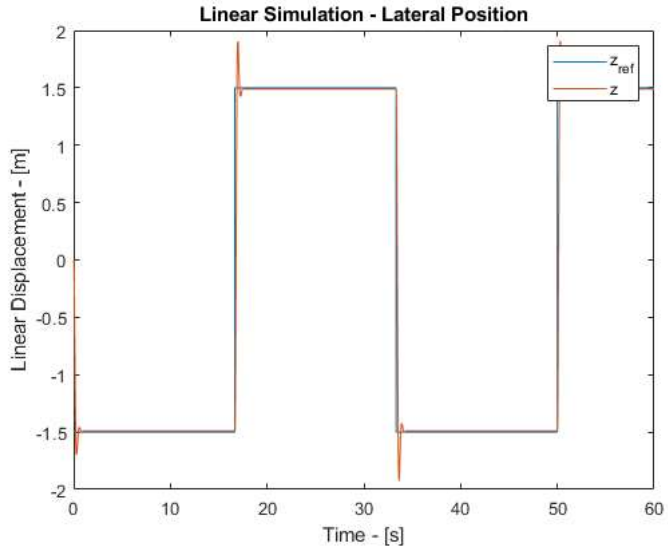
1.10 - Linearized Simulation



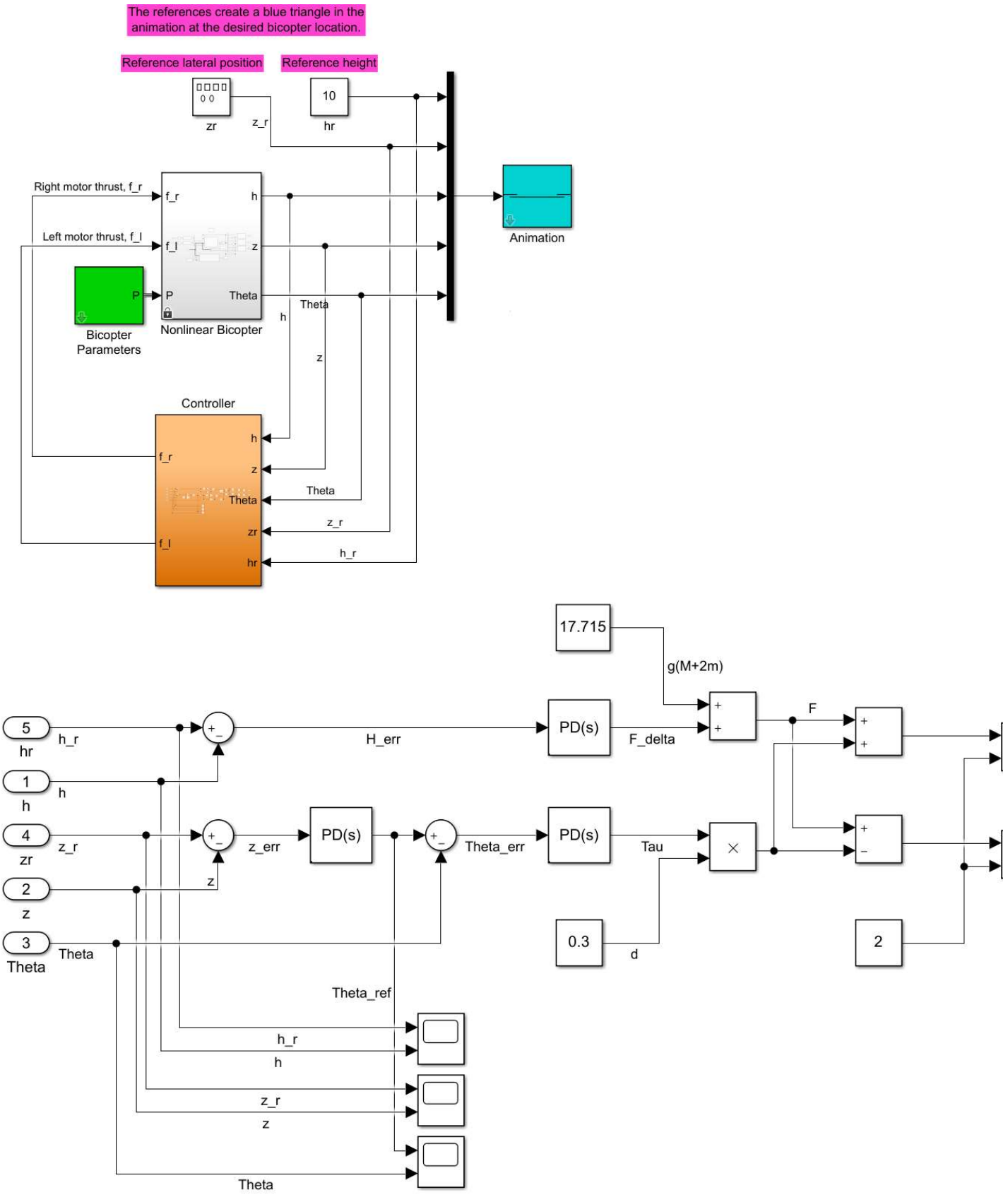

```
LS1 = sim('Part1_linsim',60);

plot(LS1.Alt.time,LS1.Alt.signals(1).values)
hold on
plot(LS1.Alt.time,LS1.Alt.signals(2).values)
hold off
title("Linear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(LS1.Pos.time,LS1.Pos.signals(1).values)
hold on
plot(LS1.Pos.time,LS1.Pos.signals(2).values)
hold off
title("Linear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```

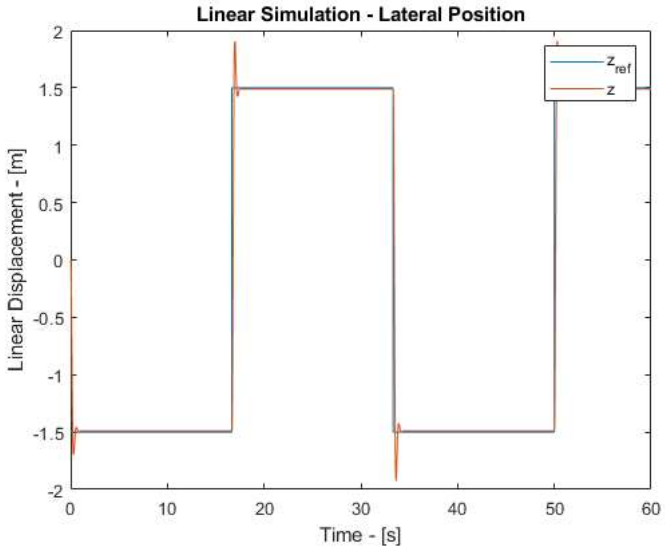


1.11 - Nonlinear Simulation

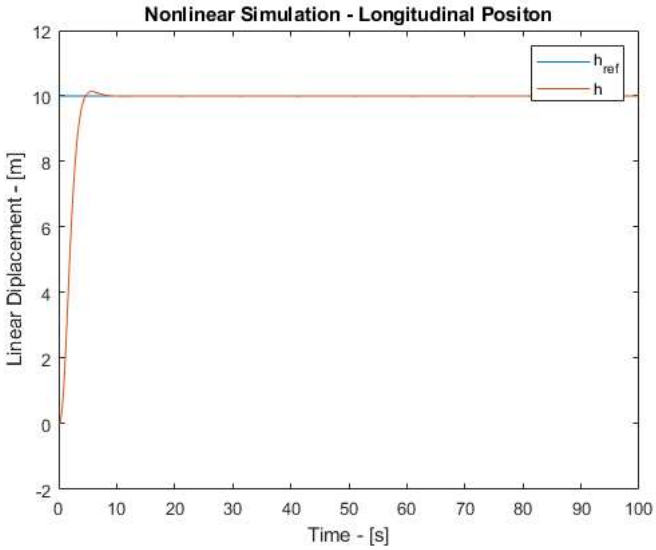



```
wind = 0;
variaciones = 0;
noise = 0;
NLS1 = sim('Part1_planarBicopter');
```

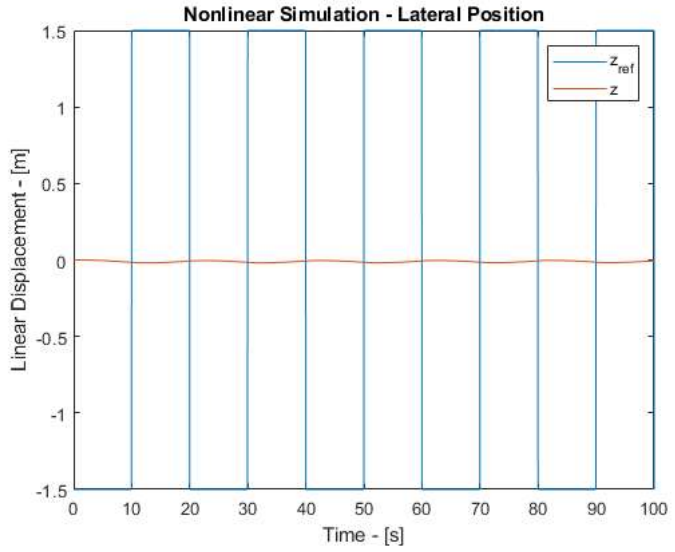
```
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS1.Alt.time,NLS1.Alt.signals(1).values)
hold on
plot(NLS1.Alt.time,NLS1.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```



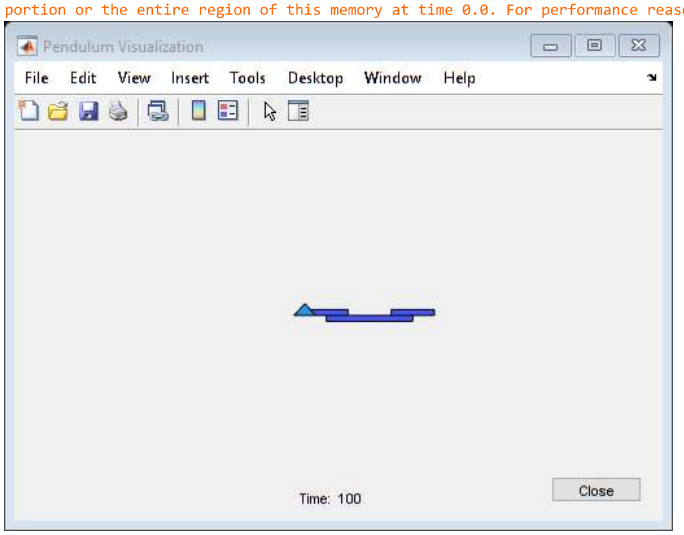

```
plot(NLS1.Pos.time,NLS1.Pos.signals(1).values)
hold on
plot(NLS1.Pos.time,NLS1.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



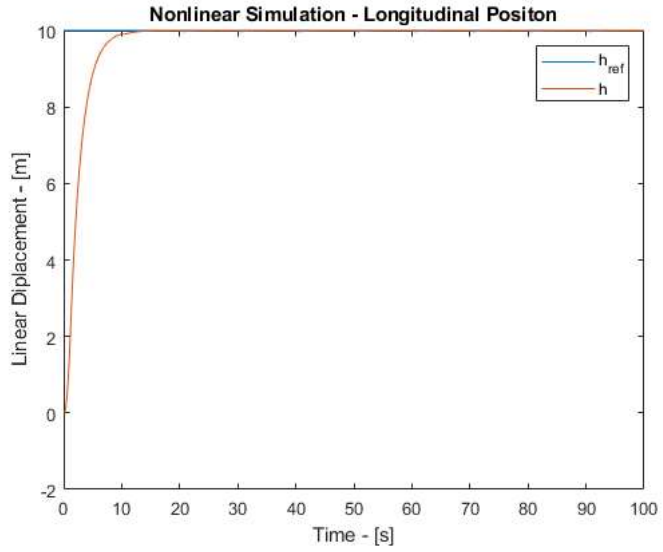
As it can be seen above, the Nonlinear simulation is unstable despite the linear simulation being stable. To solve this, the Linear controller gains were used as a starting point for tuning the controllers used in the nonlinear simulation. The final gains used are the following

```
Kp_alt = 50;
Kd_alt = 100;

Kp_roll = 10;
Kd_roll = 10;

Kp_pos = -2.5;
Kd_pos = -1.25;

wind = 0;
variaciones = 0;
noise = 0;
NLS1 = sim('Part1_planarBicopter');
```

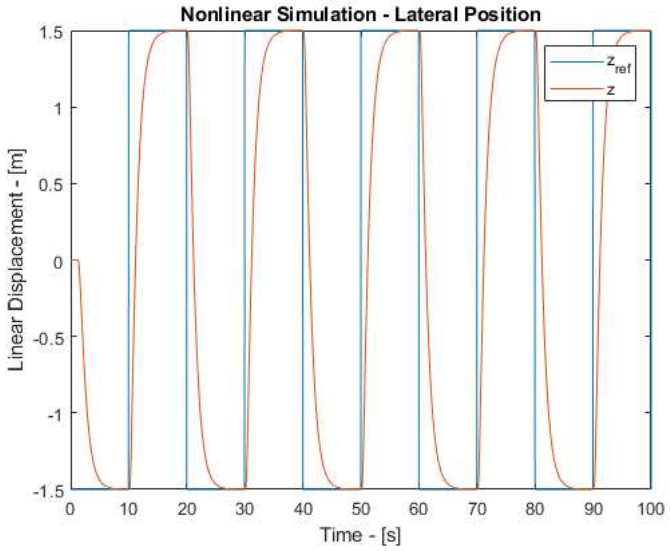
```
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS1.Alt.time,NLS1.Alt.signals(1).values)
hold on
plot(NLS1.Alt.time,NLS1.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(NLS1.Pos.time,NLS1.Pos.signals(1).values)
hold on
plot(NLS1.Pos.time,NLS1.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



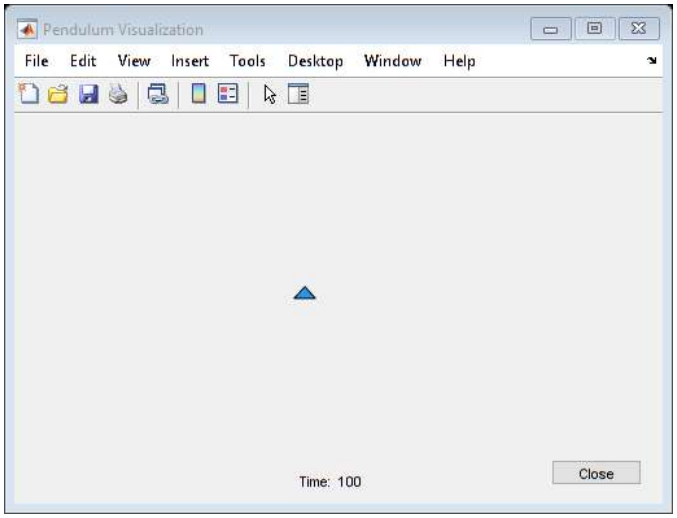
1.12 - Simulation Comparison

The controllers in the linear simulation are ableto track the reference trajectories with some overshoot. The same controllers when implemented in the nonlinear simulation do not track the lateral reference properly. This was expected because the controller was designed for a linearized system. The controllers had to be modified to track the lateral trajectory properly. After this modification, the system was able to track the reference trajectories properly.

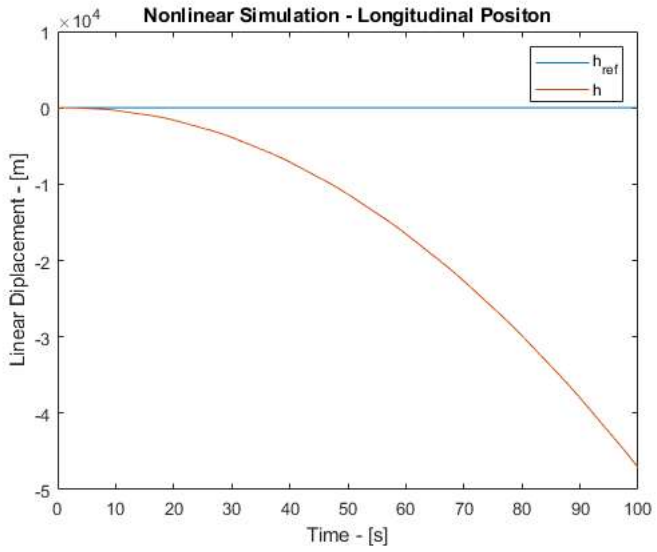
1.13 - Addition of Noise

```
wind = 0;
variaciones = 0;
noise = 1e-6;
NLS1 = sim('Part1_planarBicopter');
```

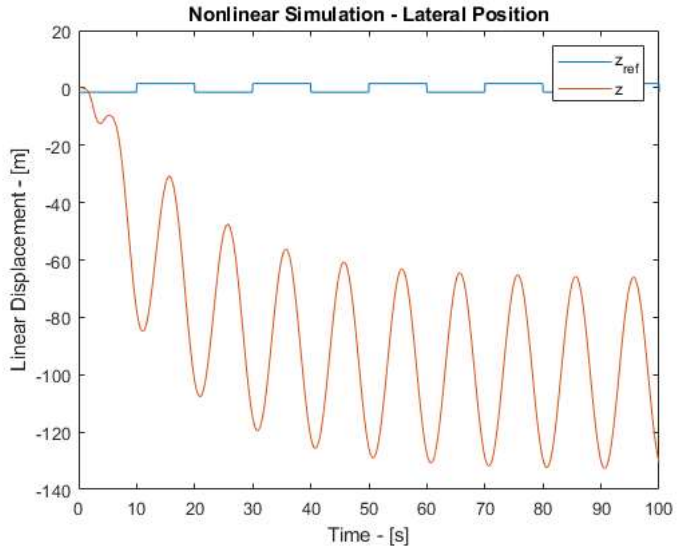
```
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS1.Alt.time,NLS1.Alt.signals(1).values)
hold on
plot(NLS1.Alt.time,NLS1.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```



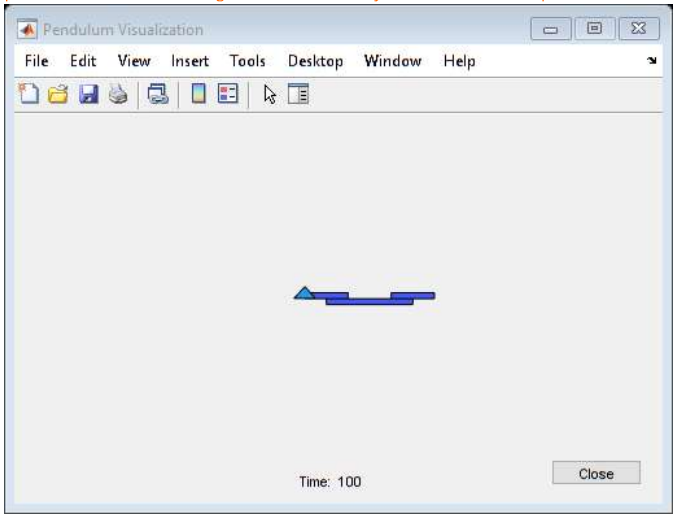

```
plot(NLS1.Pos.time,NLS1.Pos.signals(1).values)
hold on
plot(NLS1.Pos.time,NLS1.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



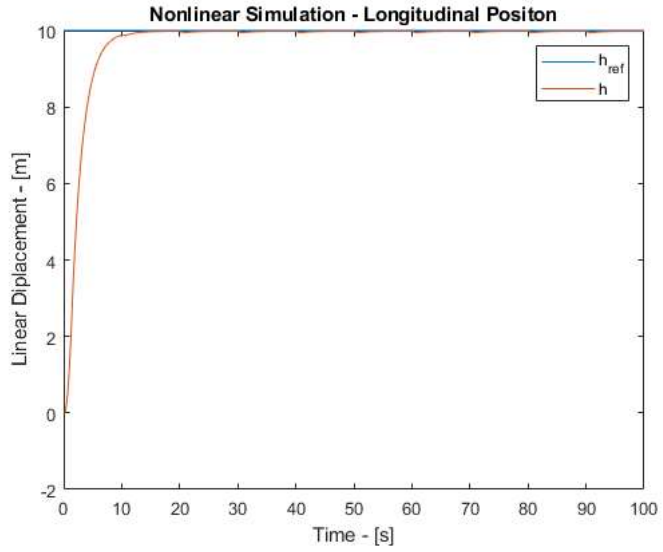
1.14 - Parameter Variations

```
wind = 0;
variaciones = 20;
noise = 0;
NLS1 = sim('Part1_planarBicopter');
```

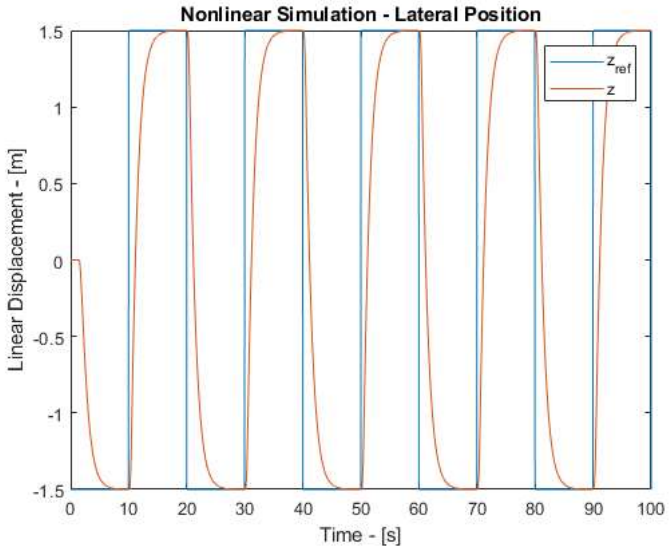
```
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS1.Alt.time,NLS1.Alt.signals(1).values)
hold on
plot(NLS1.Alt.time,NLS1.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```



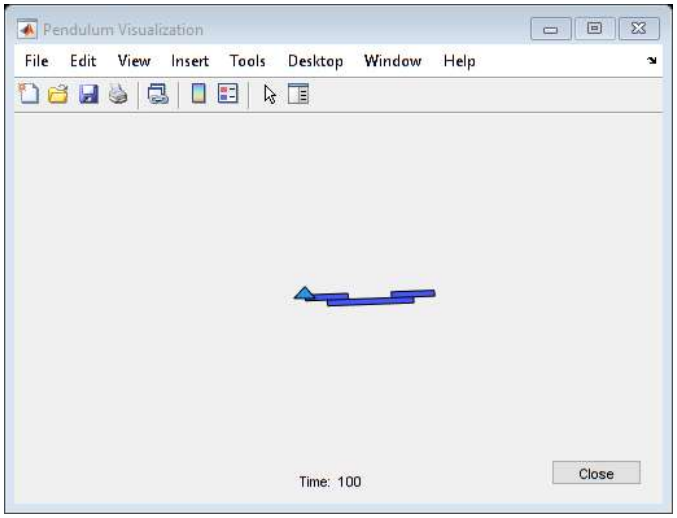

```
plot(NLS1.Pos.time,NLS1.Pos.signals(1).values)
hold on
plot(NLS1.Pos.time,NLS1.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



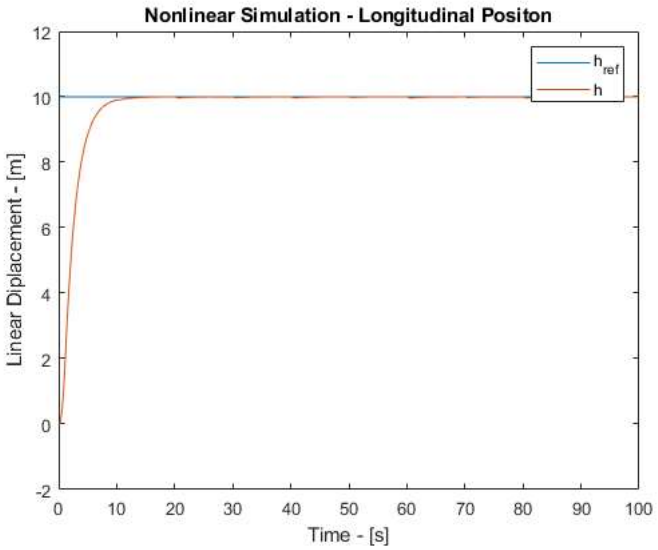
1.15 - Addition of Wind

```
wind = 1;
variaciones = 0;
noise = 0;
NLS1 = sim('Part1_planarBicopter');
```

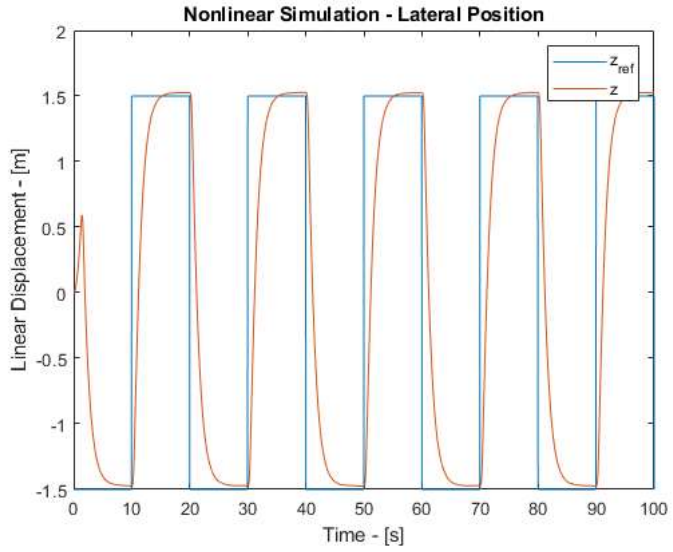
```
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part1_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part1_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS1.Alt.time,NLS1.Alt.signals(1).values)
hold on
plot(NLS1.Alt.time,NLS1.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(NLS1.Pos.time,NLS1.Pos.signals(1).values)
hold on
plot(NLS1.Pos.time,NLS1.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



Part 2

2.1 - Longitudinal (Loop Shaping)

Design a controller that generates linearized thrust commands and satisfies the following constraints:

- $\omega_B \leq 2 \frac{\text{rad}}{s}$
- $\epsilon_w \geq 0.1 \; \forall \; \omega_w \leq 0.01 \frac{\text{rad}}{s}$
- $\epsilon_{\text{no}} \geq 0.0005 \; \forall \; \omega_{\text{no}} \geq 100 \frac{\text{rad}}{s}$
- $\text{PM} \geq \frac{\pi}{3} \text{ rad}$

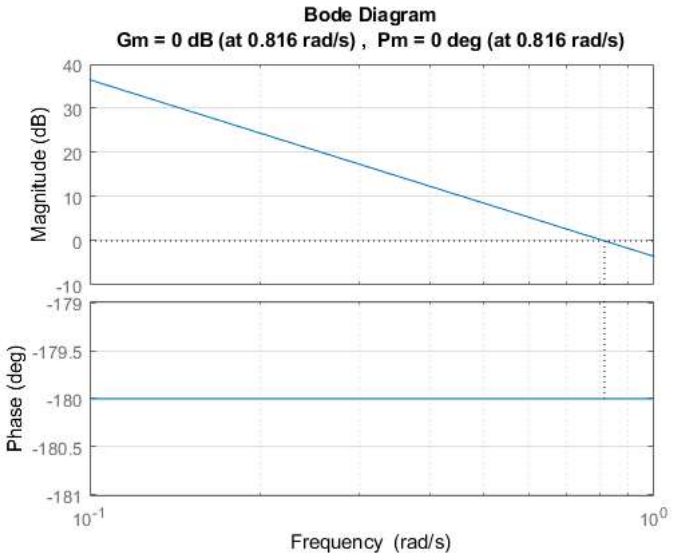
```
gain_w = 20*log10(1/0.1)
```

```
gain_w = 20
```

```
gain_no = 20*log10(0.0005)
```

```
gain_no = -66.0206
```

```
C_alt2 = 1;
margin(C_alt2*G_HF),grid
```



Base on the bode plot of the longitudinal dynamincs, the first step is to create a lead compensator to add phase to the system.

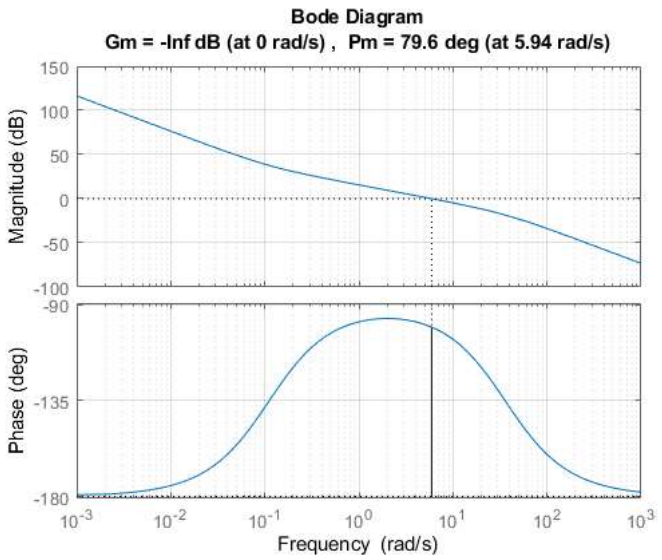
```
phi_lead = -80;
omega_lead = 2;

M = (1 + sin(phi_lead))/(1 - sin(phi_lead));
LC = tf([M, omega_lead*sqrt(M)],[1, omega_lead*sqrt(M)])
```

```
LC =

  326.3 s + 36.13
  ---------------
     s + 36.13

Continuous-time transfer function.
```

```
C_alt2 = C_alt2 * LC
```

```
C_alt2 =

  326.3 s + 36.13
  ---------------
     s + 36.13

Continuous-time transfer function.
```

```
margin(C_alt2*G_HF),grid
```



Now that the system meets and exceeds, the phase margins, it is time to meet the atenuation contraints. The disturbances were also met with the lead compensator.For the noise rejection, a low pass filter will be implemented

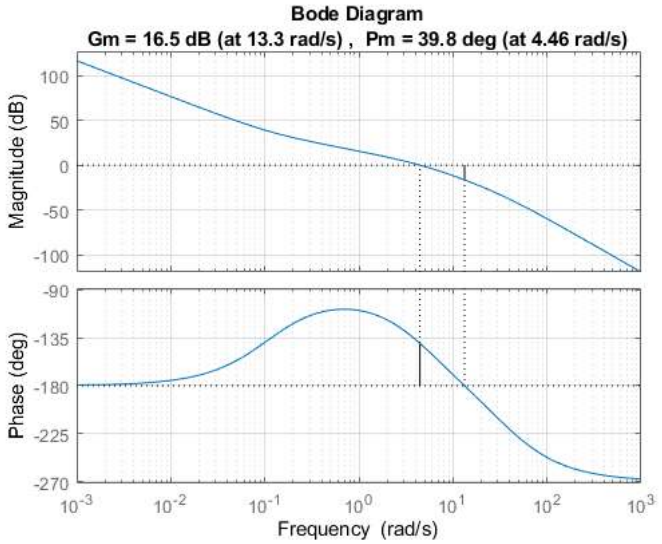
```
LPF = tf([5],[1,5])
```

```
LPF =

    5
  -----
  s + 5

Continuous-time transfer function.
```

```
C_alt2 = C_alt2 * LPF
```

```
C_alt2 =

     1631 s + 180.6
  ---------------------
  s^2 + 41.13 s + 180.6

Continuous-time transfer function.
```

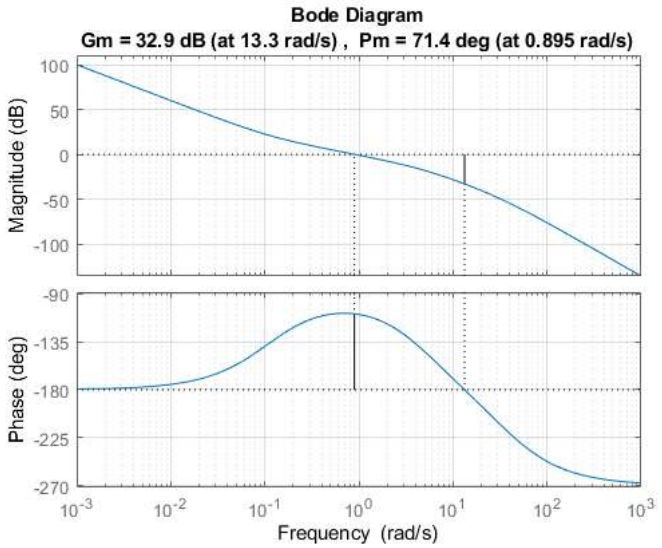
```
margin(C_alt2*G_HF),grid
```




```
P = 0.15;
C_alt2 = C_alt2 * P
```

```
C_alt2 =

     244.7 s + 27.09
  ---------------------
  s^2 + 41.13 s + 180.6

Continuous-time transfer function.
```

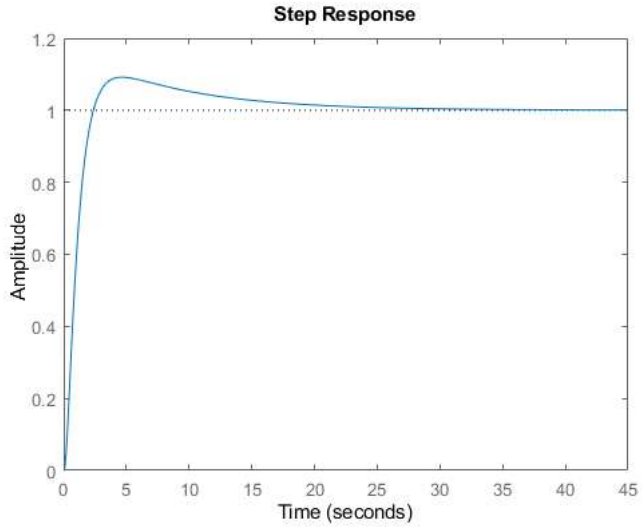
```
margin(C_alt2*G_HF),grid
```




```
CL_alt2 = minreal(feedback(C_alt2*G_HF,1))
```

```
CL_alt2 =

                163.1 s + 18.06
  ---------------------------------------------
  s^4 + 41.13 s^3 + 180.6 s^2 + 163.1 s + 18.06

Continuous-time transfer function.
```

```
step(CL_alt2)
```




```
stepinfo(CL_alt2)
```

```
ans = struct with fields:
        RiseTime: 1.5354
    SettlingTime: 17.5125
     SettlingMin: 0.9031
     SettlingMax: 1.0918
       Overshoot: 9.1770
      Undershoot: 0
            Peak: 1.0918
        PeakTime: 4.6396
```

```
isstable(CL_alt2)
```

```
ans = logical
   1
```

2.2.1 - Inner Loop

Design an inner loop controller that satisfies the following constraints:

- $\omega_B \geq 10 \ \frac{\text{rad}}{s}$
- $\epsilon_{\text{no}} \geq 0.0001 \ \forall \ \omega_{\text{no}} \geq 1000 \ \frac{\text{rad}}{s}$
- $\text{PM} \geq \frac{\pi}{3} \ \text{rad}$

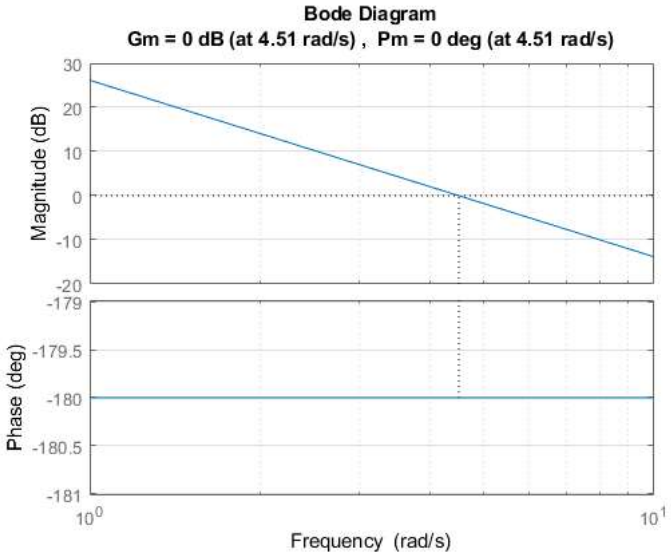
```
gain_no = 20*log10(0.0001)
```

```
gain_no = -80
```

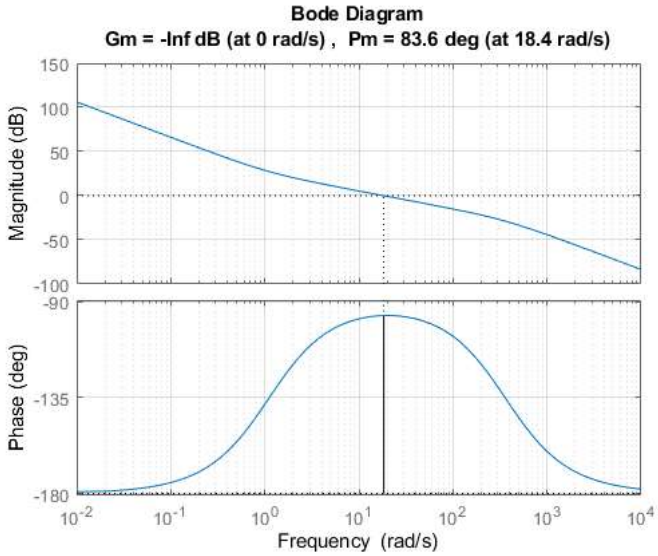
```
C_roll2 = 1
```

```
C_roll2 = 1
```

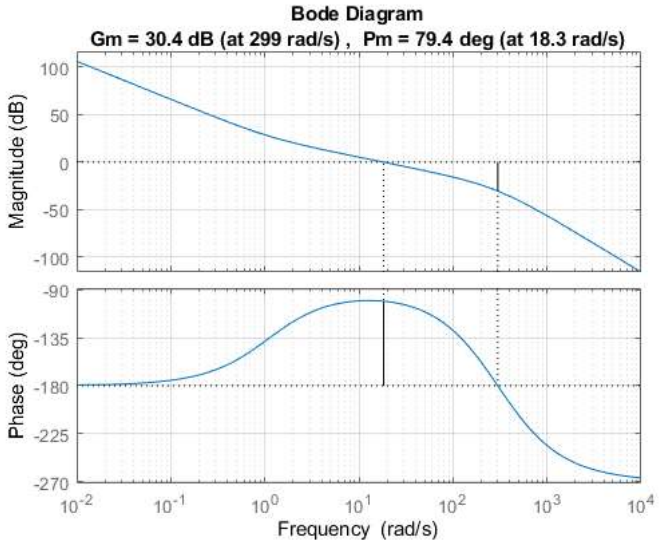
```
margin(C_roll2*G_ThetaTau),grid
```



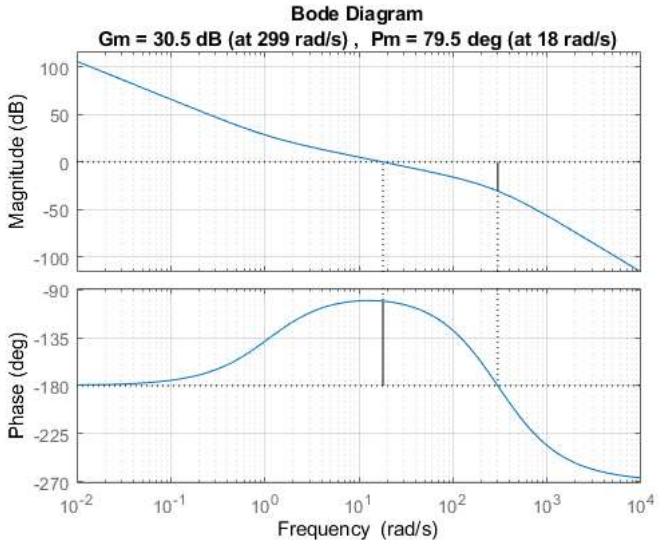

```
phi_lead = -80;
omega_lead = 20;

M = (1 + sin(phi_lead))/(1 - sin(phi_lead));
LC = tf([M, omega_lead*sqrt(M)],[1, omega_lead*sqrt(M)]);
C_roll2 = C_roll2 * LC;
margin(C_roll2*G_ThetaTau),grid
```




```
omega_c = 250;
C_roll2 = C_roll2 * tf([omega_c],[1, omega_c]);
margin(C_roll2*G_ThetaTau),grid
```



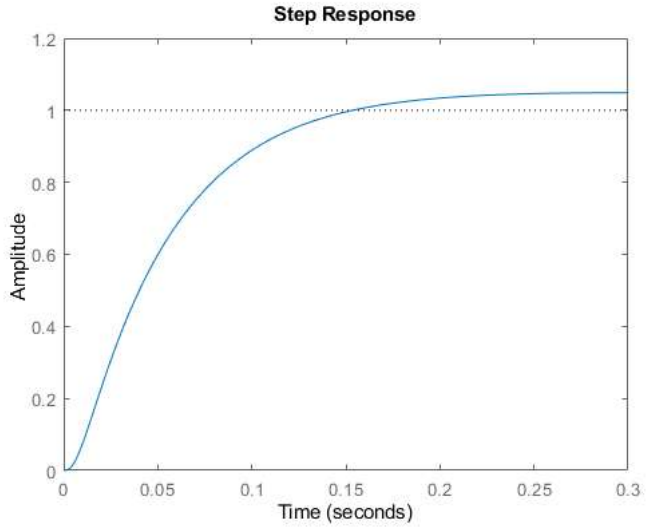

```
P = 30;
C_roll2 = C_roll2 * P;
margin(C_roll2*G_HF),grid
```




```
CL_roll2 = minreal(feedback(C_roll2*G_HF,1))
```

```
CL_roll2 =

                  1.631e06 s + 1.806e06
  ---------------------------------------------------
  s^4 + 611.3 s^3 + 9.031e04 s^2 + 1.631e06 s + 1.806e06

Continuous-time transfer function.
```

```
step(CL_roll2)
```




```
stepinfo(CL_roll2)
```

```
ans = struct with fields:
        RiseTime: 0.0916
    SettlingTime: NaN
     SettlingMin: 0.9011
     SettlingMax: 1.0488
       Overshoot: 4.8836
      Undershoot: 0
            Peak: 1.0488
        PeakTime: 0.3062
```

```
isstable(CL_roll2)
```

```
ans = logical
   1
```

2.2.2 - Outer Loop

Design an outer loop controller that satisfies the following constraints:

- $\omega_B \leq 1 \ \frac{\text{rad}}{s}$
- $\epsilon_w \geq 0.1 \ \forall \ \omega_w \leq 0.001 \frac{\text{rad}}{s}$
- $\epsilon_{\text{no}} \geq 0.0001 \ \forall \ \omega_{\text{no}} \geq 100 \ \frac{\text{rad}}{s}$
- $\text{PM} \geq \frac{\pi}{4} \ \text{rad}$

```
gain_w = 20*log10(1/0.1)
```

```
gain_w = 20
```

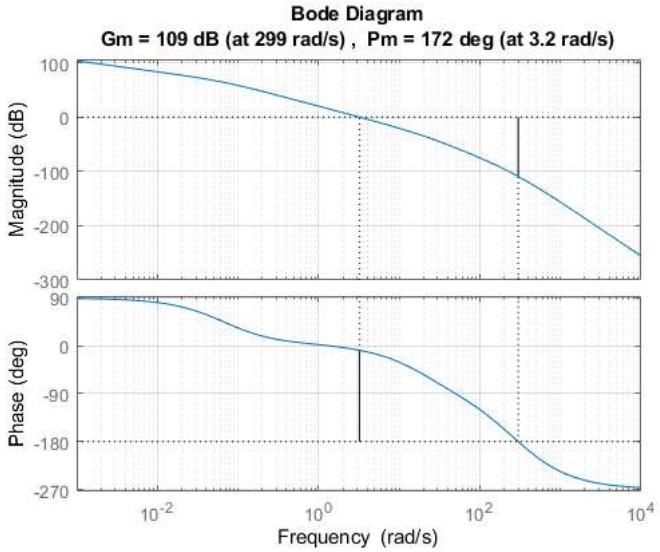
```
gain_no = 20*log10(0.0001)
```

```
gain_no = -80
```

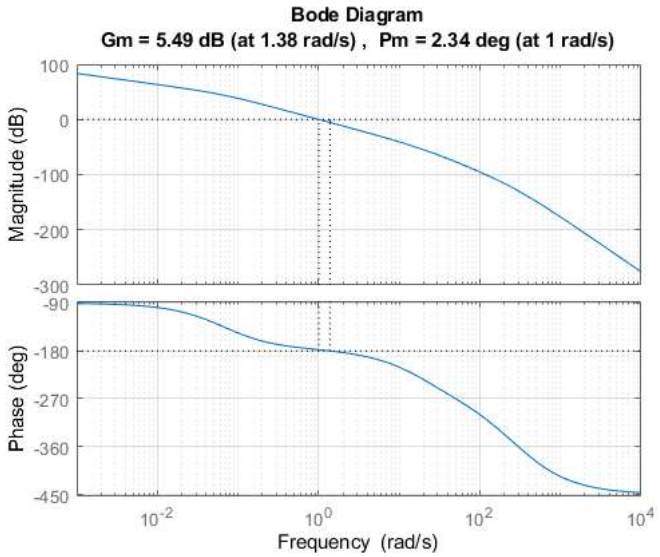
```
C_pos2 = 1
```

```
C_pos2 = 1
```

```
margin(C_pos2*CL_roll2*G_ZTheta),grid
```



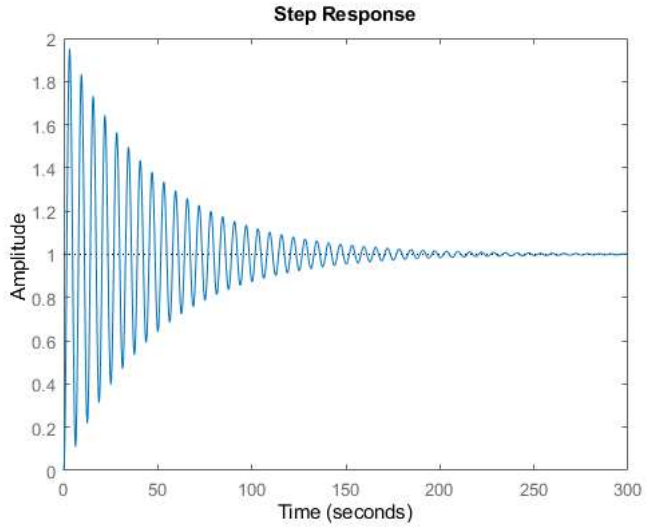

```
P = -0.1;
C_pos2 = C_pos2 * P;
margin(C_pos2*CL_roll2*G_ZTheta),grid
```




```
CL_pos2 = minreal(feedback(C_pos2*CL_roll2*G_ZTheta,1))
```

```
CL_pos2 =

                                1.6e06 s + 1.772e06
  ------------------------------------------------------------------------------------
  s^6 + 611.3 s^5 + 9.035e04 s^4 + 1.637e06 s^3 + 1.915e06 s^2 + 1.721e06 s + 1.772e06

Continuous-time transfer function.
```

```
step(CL_pos2)
```




```
stepinfo(CL_pos2)
```

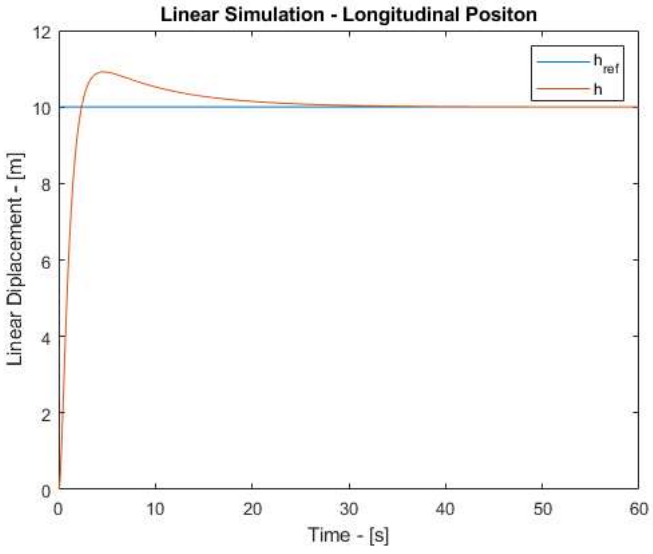
```
ans = struct with fields:
        RiseTime: 1.0472
    SettlingTime: 187.9602
     SettlingMin: 0.1095
     SettlingMax: 1.9508
       Overshoot: 95.0780
      Undershoot: 0
            Peak: 1.9508
        PeakTime: 3.1274
```

```
isstable(CL_pos2)
```

```
ans = logical
   1
```

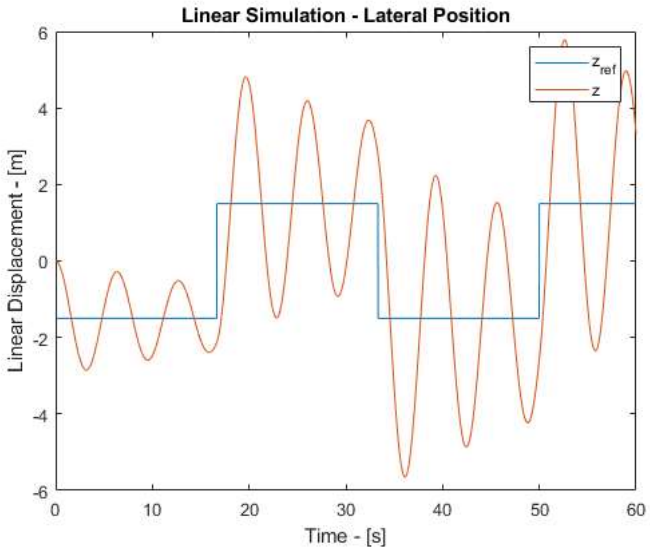
2.3 - Linear Simulation

```
LS2 = sim('Part2_linsim',60);
plot(LS2.Alt.time,LS2.Alt.signals(1).values)
hold on
plot(LS2.Alt.time,LS2.Alt.signals(2).values)
hold off
title("Linear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(LS2.Pos.time,LS2.Pos.signals(1).values)
hold on
plot(LS2.Pos.time,LS2.Pos.signals(2).values)
hold off
title("Linear Simulation - Lateral Position")
```

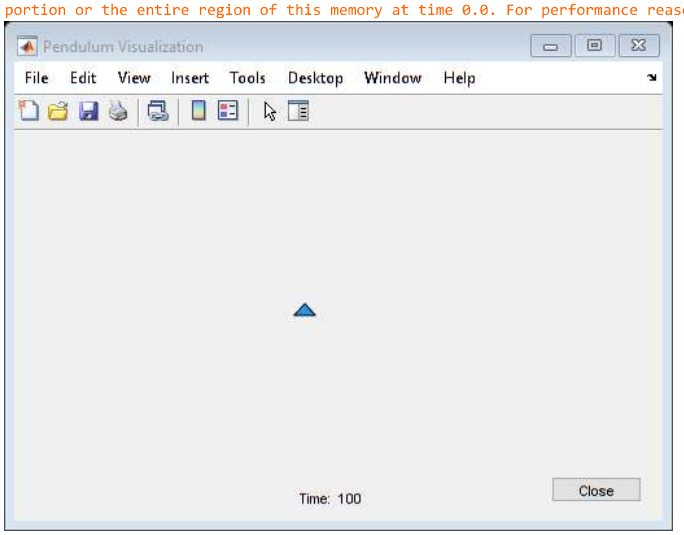
```
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



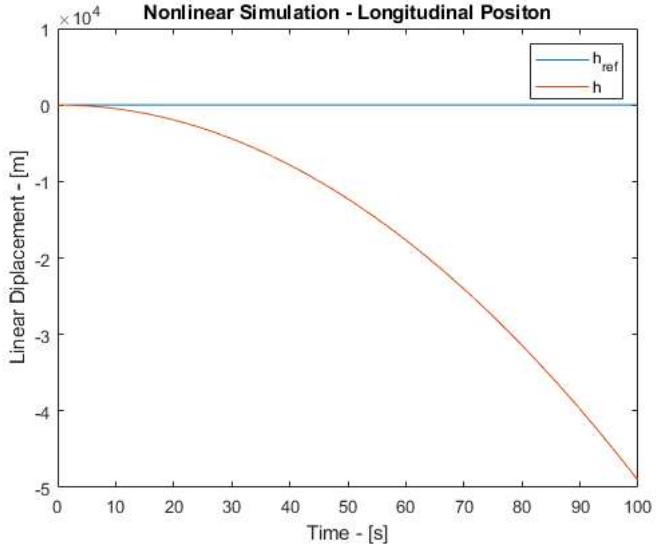
2.4 - Nonlinear Simulation

```
wind = 0;
variaciones = 0;
noise = 0;
NLS2 = sim('Part2_planarBicopter');
```

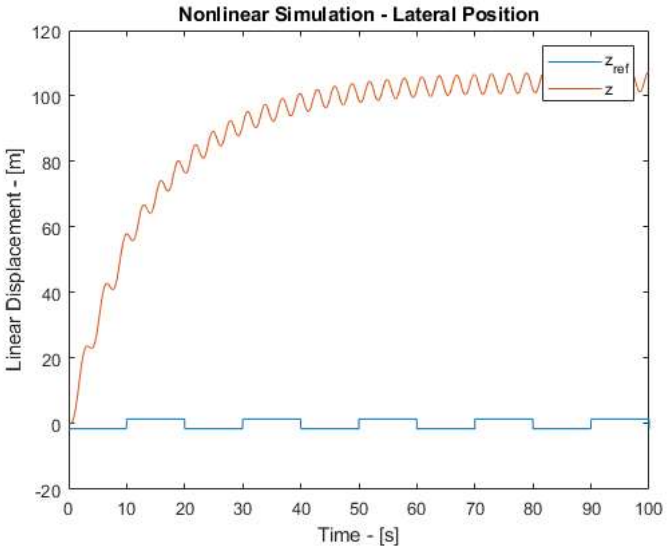
```
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS2.Alt.time,NLS2.Alt.signals(1).values)
hold on
plot(NLS2.Alt.time,NLS2.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(NLS2.Pos.time,NLS2.Pos.signals(1).values)
hold on
plot(NLS2.Pos.time,NLS2.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```

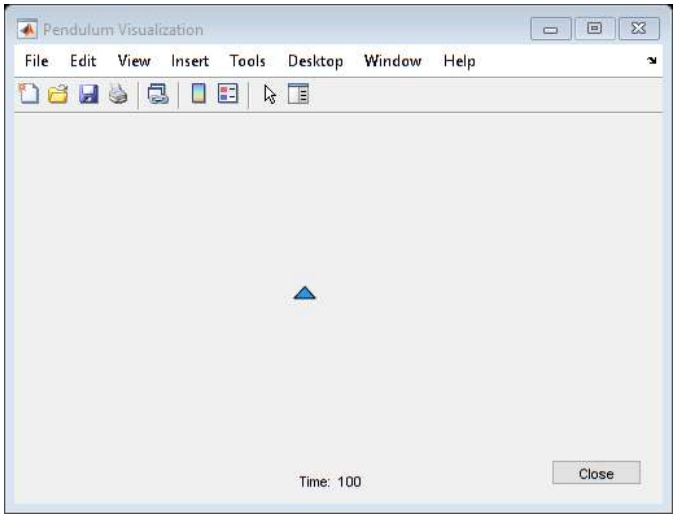


2.5 Comparison to Pole Placement

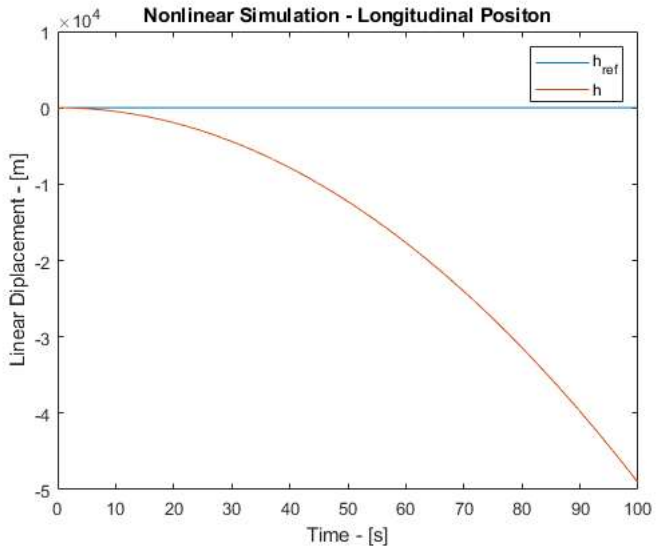
2.6 Addition of Noise

```
wind = 0;
variaciones = 0;
noise = 1e-6;
NLS2 = sim('Part2_planarBicopter');
```

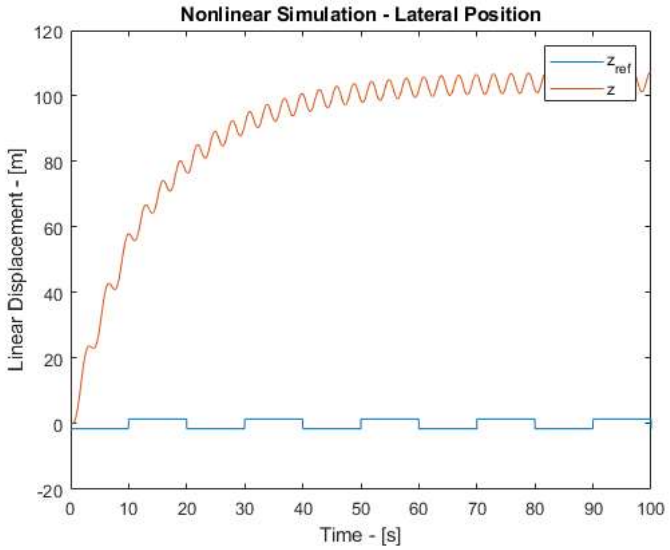
```
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS2.Alt.time,NLS2.Alt.signals(1).values)
hold on
plot(NLS2.Alt.time,NLS2.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```



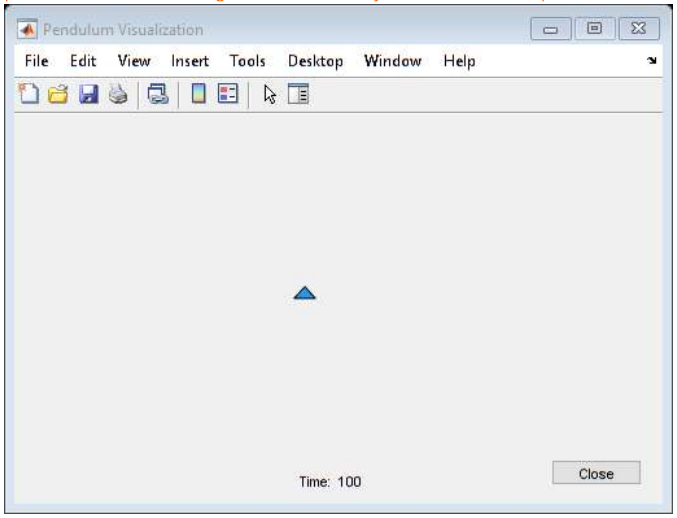

```
plot(NLS2.Pos.time,NLS2.Pos.signals(1).values)
hold on
plot(NLS2.Pos.time,NLS2.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



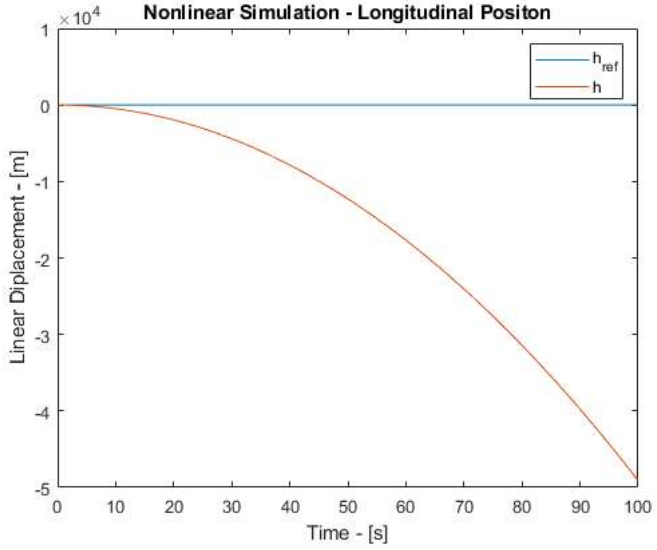
2.7 Parameter Variation

```
wind = 0;
variaciones = 20;
noise = 0;
NLS2 = sim('Part2_planarBicopter');
```

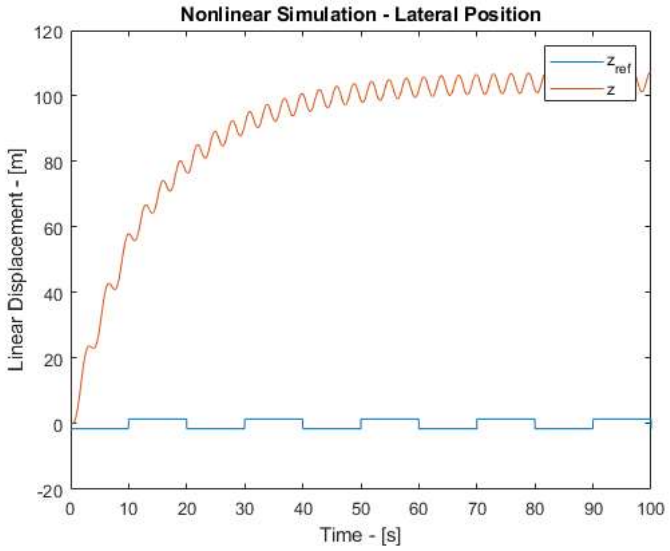
```
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS2.Alt.time,NLS2.Alt.signals(1).values)
hold on
plot(NLS2.Alt.time,NLS2.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```



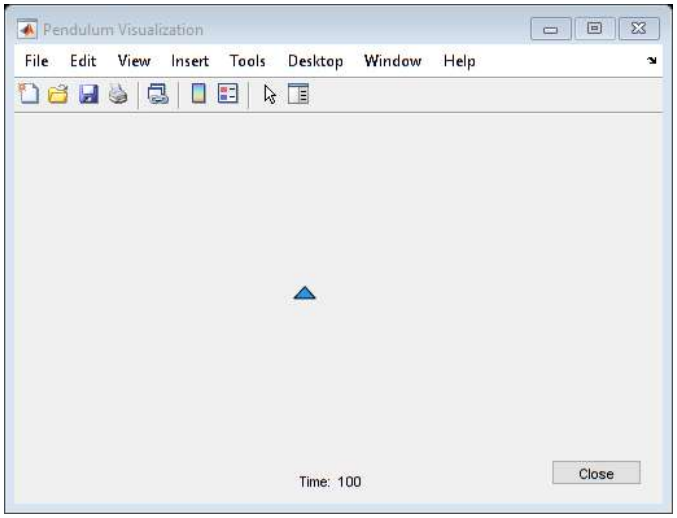

```
plot(NLS2.Pos.time,NLS2.Pos.signals(1).values)
hold on
plot(NLS2.Pos.time,NLS2.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



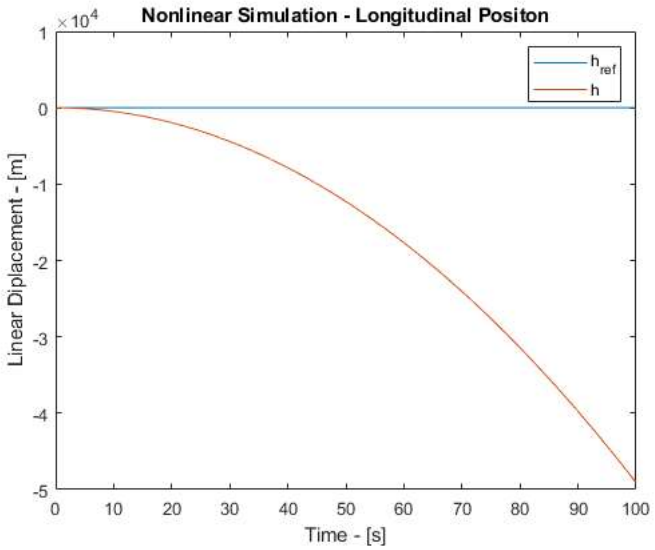
2.8 Addition of Wind

```
wind = 1;
variaciones = 0;
noise = 0;
NLS2 = sim('Part2_planarBicopter');
```

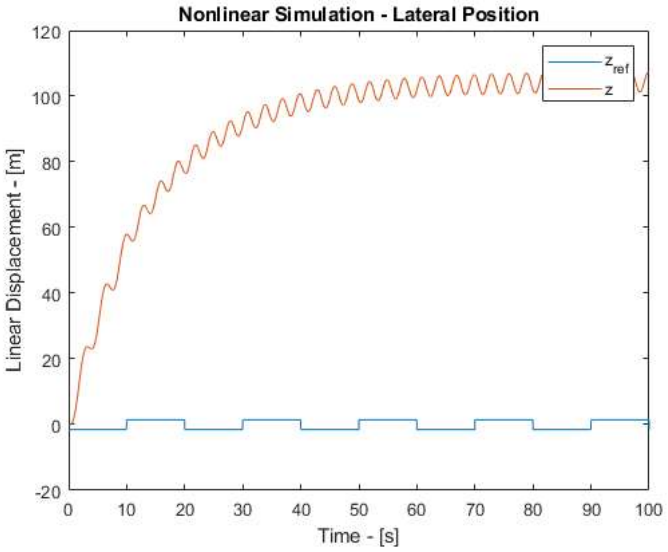
```
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Read' is reading from the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data S
occurrences of this diagnostic for this memory at other simulation time steps will be suppressed.
Warning: The block 'Part2_planarBicopter/Nonlinear Bicopter/Data Store Write' is writing to the data store 'Part2_planarBicopter/Nonlinear Bicopter/Data St
portion or the entire region of this memory at time 0.0. For performance reasons, occurrences of this diagnostic for this memory at other simulation time
```




```
plot(NLS2.Alt.time,NLS2.Alt.signals(1).values)
hold on
plot(NLS2.Alt.time,NLS2.Alt.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Longitudinal Positon")
legend("h_{ref}","h")
xlabel("Time - [s]")
ylabel("Linear Diplacement - [m]")
```




```
plot(NLS2.Pos.time,NLS2.Pos.signals(1).values)
hold on
plot(NLS2.Pos.time,NLS2.Pos.signals(2).values(:,:))
hold off
title("Nonlinear Simulation - Lateral Position")
legend("z_{ref}","z")
xlabel("Time - [s]")
ylabel("Linear Displacement - [m]")
```



2.9 Improvements