CONTROL OF AIRCRAFT

LONGITUDINAL AUTOPILOT

Jean-Pierre NOUAILLE

October 4, 2020

CONTROL OF AIRCRAFT

ROLE OF THE AUTOPILOT

The autopilot tasks are:

- to stabilize an unstable aircraft; The stabilization of an unstable aircraft is done at the cost of a higher bandwidth autopilot, and necessitates actuators and sensors with high bandwidth (which could lead to problems of feasibility).
- to improve the response of a stable aircraft.
- to control the states of the aircraft to their commanded value calculated by the guidance function.

Performances and robustness

The autopilot must maintain a minimum level of performances (stability, response time, overshoot, bandwidth . . .) despite:

- a variation of aerodynamic forces and moments which vary with altitude, Mach, incidence, sideslip, fin deflection,
- off-centered thrust,
- variation of mass, center of gravity position and inertia (because of fuel consumption, booster jettison. . .);
- uncertainties in aircraft model (mass, center of gravity position, inertia, aerodynamic coefficients, sensors and actuators behavior, representativity of the model at high frequency. . .);
- disturbances (wind, flexible modes, wind gusts);
- noises (sensors measurement noise (coming from inertial measurement unit, guidance sensors, etc)). . .

LAPLACE TRANSFORM

DEFINITION

$$\mathcal{L}(f(t)) = \int_0^{\infty} f(\tau) e^{-s\tau} d\tau$$

Permits to transform linear differential equations into polynomials rational fractions of s. $\mathcal{L}\left(\frac{df}{dt}\right) = s\mathcal{L}(f(t)) - f(0)$ so the Laplace transform of a linear differential equation of the first order, while supposing null initial conditions could be written as:

$$\mathcal{L}\left(\frac{df}{dt} + af\right) = s\mathcal{L}(f) + \mathcal{L}(f) = \mathcal{F}(s+a) = \mathcal{L}(e) = \mathcal{E}$$

TRANSFER FUNCTION

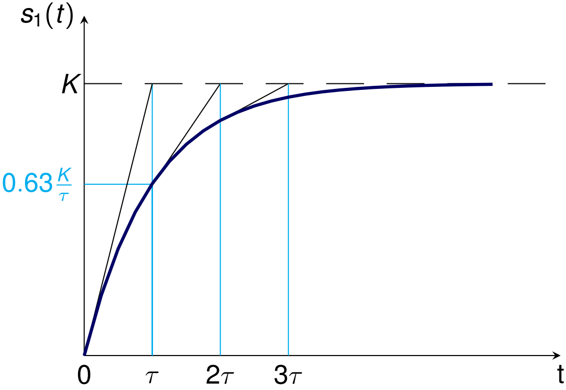
Once the Laplace transform applied to differential equation, we can write:

$$\frac{\mathcal{F}}{\mathcal{E}}(s) = \frac{1}{s+a}$$

$\frac{\mathcal{F}}{\mathcal{E}}(s)$ is the transfer function linking $\mathcal{E}$ to $\mathcal{F}$, in the frequency domain.

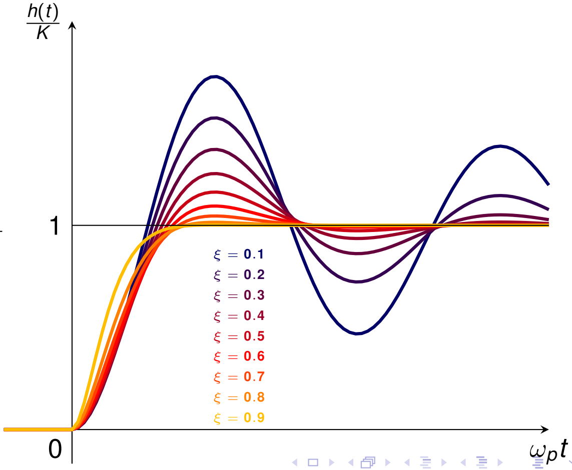
Step response of first order systems

$$\mathcal{F}(s) = \frac{K}{1+\tau s}$$



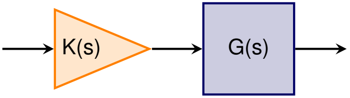
STEP RESPONSE OF SECOND ORDER SYSTEMS

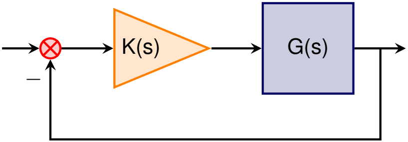
$$\mathcal{F}(s) = \frac{K}{\frac{s^2}{\omega_p^2} + \frac{2\xi}{\omega_p}s + 1}$$



Open loop and closed loop

Let G(s) be the transfer function of the aircraft, linking for example an aerodynamic fin deflection to acceleration realized by the aircraft





$$K(s)G(s)$$

$$\frac{K(s)G(s)}{1 + K(s)G(s)}$$

STATE SPACE REPRESENTATION

Exemple
If we have a differential equation:

$$\frac{d^2y}{dt^2} + a\frac{dy}{dt} + by = u$$

By defining $x_1 = \dfrac{dy}{dt} = \dot{y}$ and $x_2 = y$, we have

$$\begin{cases} \dot{x}_1 & = -a\dot{y} - by + u = -ax_1 - bx_2 + u \\ \dot{x}_2 & = x_1 \end{cases}$$

With matrix notation:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -a & -b \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u$$

we have $y = x_2$, so

$$y = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u$$

By defining $X = (x_1 \quad x_2)^T$, we have:

$$\begin{array}{ccccc} \dot{X} & = & AX & + & Bu \\ y & = & CX & + & Du \end{array}$$

This is the state space representation of the system.

Stability of the commanded system

Input-Ouput stability of a linear system

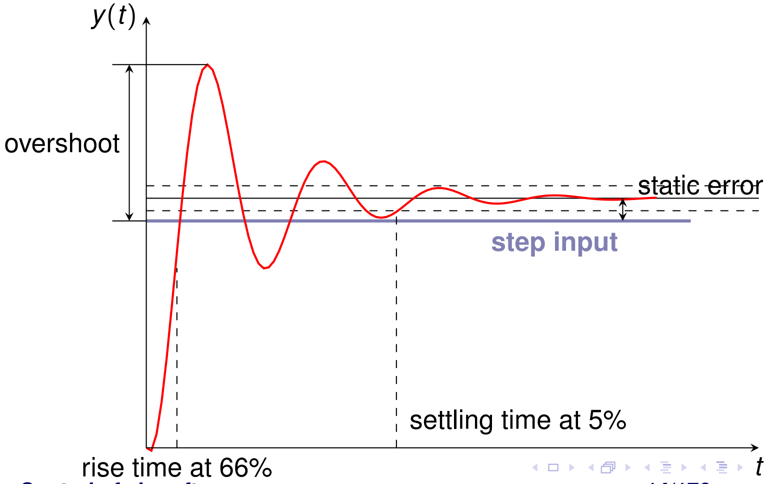
The commanded system is stable if and only if the poles (roots of the denominator of the transfer function) of the closed loop system have their real part strictly negative.

Intern stability of a linear system

The commanded system is stable if and only if the eigenvalues of the state matrix *A* have all their real part negative (*A* being the state matrix of the closed loop system).

Then the solutions of the differential equation, of the form $ae^{\lambda_i t}$, with λ_i pole of the system, won't diverge for $t \to \infty$.

TIME RESPONSE PERFORMANCES



BODE DIAGRAM

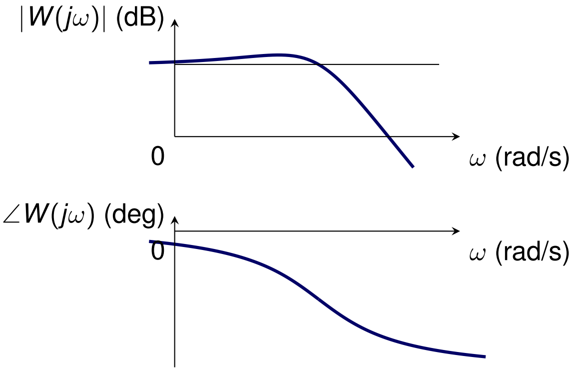
We replace, in the transfer function, the Laplace variable s by $j\omega$. The transfer function becomes a complex number, for which we can determine:

- its module, the gain;
- its argument, the phase.

The Bode plot describes the steady state response of a system to sinusoid inputs, for various frequencies.
Second order system

$$\frac{\Gamma_e}{\Gamma_c} = \frac{K}{\frac{s^2}{\omega_0} + 2\frac{\xi}{\omega_0}s + 1}$$

BODE DIAGRAM



BLACK-NICHOLS DIAGRAM

This is the graph of gain as a function of phase, for each frequency. We represent on the diagram the open loop transfer function of the system and this diagram allows us to visualize robustness margin.

Root locus Evans diagram

This the location of the poles of the closed loop system, obtained by varying a parameter (generally a gain) in a SISO transfer function.
On the abscissa axis we put the real part of the root, and on the ordinate axis the imaginary part of the root.
While the gain vary from 0 to $+\infty$ in the system, the root locus are plotted on the diagram.

CONTROL OF AIRCRAFT

PITCH TRANSFER FUNCTION

We define the following coefficients

$$z_\alpha = \frac{QS_{Réf}CN_\alpha}{mV}$$

$$z_{\delta_m} = \frac{QS_{Réf}CN_{\delta_m}}{mV}$$

$$m_\alpha = \frac{\ell_{ref}QS_{Réf}Cm_\alpha}{I_{YY}}$$

$$m_{\delta_m} = \frac{\ell_{ref}QS_{Réf}Cm_{\delta_m}}{I_{YY}}$$

$$m_q = \frac{\ell_{ref}QS_{Réf}\frac{\ell_{ref}}{V_a}Cm_q}{I_{YY}}$$

with $Q = \frac{1}{2}\rho V_a^2$ being the dynamic pressure.

$$T_\alpha = \frac{m_{\delta_m}}{-m_\alpha z_{\delta_m} + m_{\delta_m} z_\alpha}$$

$$K_3 = \frac{-m_\alpha z_{\delta_m} + m_{\delta_m} z_\alpha}{-m_q z_\alpha - m_\alpha}$$

$$\omega_z = \frac{\sqrt{-z_{\delta_m}(-m_\alpha z_{\delta_m} + m_{\delta_m} z_\alpha)}}{z_{\delta_m}}$$

$$\omega_{af} = \sqrt{-(m_q z_\alpha + m_\alpha)}$$

$$\xi_{af} = -\frac{1}{2}\frac{m_q - z_\alpha}{\omega_{af}}$$

$$K1 = \frac{V_a(-m_\alpha z_{\delta_m} + m_{\delta_m} z_\alpha)}{-m_q z_\alpha - m_\alpha}$$

PITCH TRANSFER FUNCTIONS

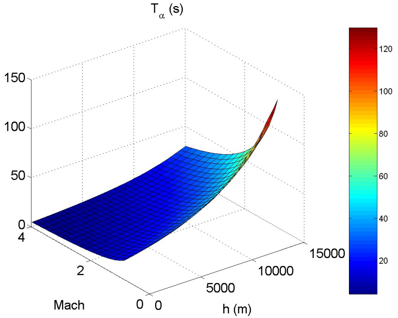
Finally, the transfer function between fin deflection (elevator for pitch axis) and pitch acceleration (for the incidence oscillation mode) is written

$$\frac{\Gamma_z}{\delta_m} = \frac{K_1(-\frac{s^2}{\omega_z^2} + 1)}{\frac{s^2}{\omega_{af}^2} + 2\xi_{af}\frac{s}{\omega_{af}} + 1}$$

and the transfer function between fin deflection and rotation speed is

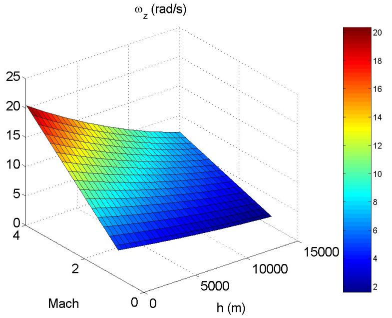
$$\frac{q}{\delta_m} = \frac{K_3(T_\alpha s + 1)}{\frac{s^2}{\omega_{af}^2} + 2\xi_{af}\frac{s}{\omega_{af}} + 1}$$

VARIATION OF T_α WITH MACH AND ALTITUDE



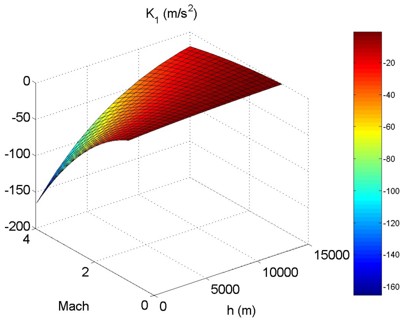
T_α increases with altitude and decreases with Mach.
T_α is the incidence lag, meaning the lag between an aircraft rotation and a change in speed orientation.

VARIATION OF ω_z WITH MACH AND ALTITUDE



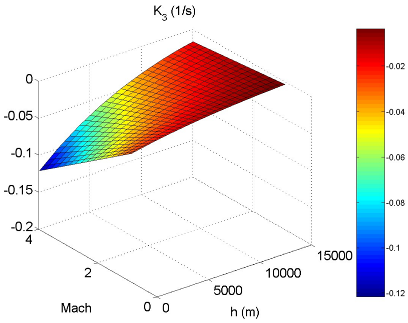
ω_z decreases with altitude and increases with Mach.
When ω_z decreases, the "non minimum phase shift effect" increases.

VARIATION OF K_1 WITH MACH AND ALTITUDE



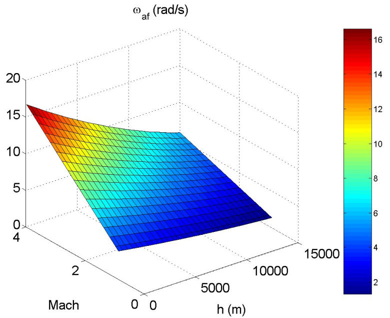
$|K_1|$ decreases with altitude and increases with Mach.
K_1 is the static gain between fin deflection and realized acceleration.
So maneuverability decreases with increasing altitude and decreasing Mach.

VARIATION OF K_3 WITH MACH AND ALTITUDE



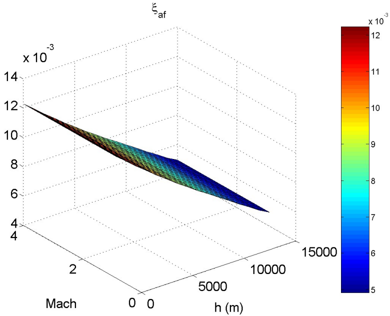
$|K_3|$ decreases with altitude and increases with Mach.
K_3 is the static gain between fin deflection and realized rotation speed.

VARIATION OF ω_{af} WITH MACH AND ALTITUDE



ω_{af} decreases with altitude and increases with Mach.
ω_{af} is the proper pulsation of the aircraft without controller. It is inversely proportional to oscillations duration (open loop response)

VARIATION OF ξ_{af} WITH MACH AND ALTITUDE



ξ_{af} decreases with altitude and weakly influenced by variations of Mach (but in general decreases with Mach).

ξ_{af} is the open loop damping ratio of the aircraft. The open loop aircraft is less damped at high altitude.

CONTROL ON THE WHOLE FLIGHT DOMAIN

Taking into account the large variability of characteristics of the transfer function with flight conditions:

- Choice of flight points (Mach, incidence), center of gravity and inertia, the most challenging for aircraft stability;
- For these flight points, a controller synthesis is made for each axis.

The embedded autopilot will possess the same structure as each individual controller but with linearly interpolated gains as a function of:

- altitude
- Mach
- mass (possibly)

This approach is called “gain scheduling”.

Open loop controller

Suppose we want to control the aircraft with an open loop controller.
If we command the aircraft with a fin deflection $\delta_m = \dfrac{\Gamma_c}{K_1}$, we will get as output, after a transient phase, an achieved acceleration equal to Γ_c.
But the response is still weakly damped, and all the more so as the aircraft is flying at high altitude.
Moreover, if the real gain K_1 is different from the model, the achieved acceleration could be quite different from wished acceleration.
For these reasons, a better solution is to use measurements and add feedback(s) to the controller.

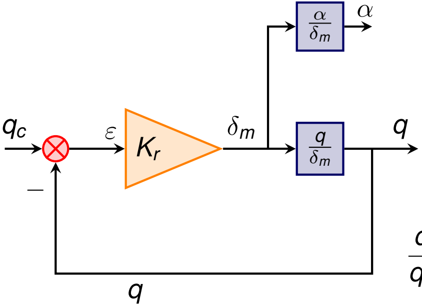
CLOSED LOOP CONTROLLER

Among the 2 weakly damped oscillatory modes (phugoid and short period) the phugoid mode is not the main concern and will be treated as a disturbance by the guidance controller.
First, we are going to add q feedback loop, which acts on the fastest mode, the short period mode:

- If the aircraft is aerodynamically unstable, this feedback loop can stabilize the aircraft.
- In order to obtain a good damping ratio for the closed loop via pitch fin actuator (aileron), the controller will use the derivative of the pitch angle, meaning q. The use of the derivative has a stabilizing effect (quicker oscillation attenuation, reduction of overshoot).
- Moreover, this is a parameter accessible to measurement by using a gyrometer, which has a dynamics meeting the control requirements.

CONTROLLER WITH GYROMETRIC FEEDBACK

We use the aircraft model seen in the modeling course. K_r is the gain of the controller (here a proportional controller).

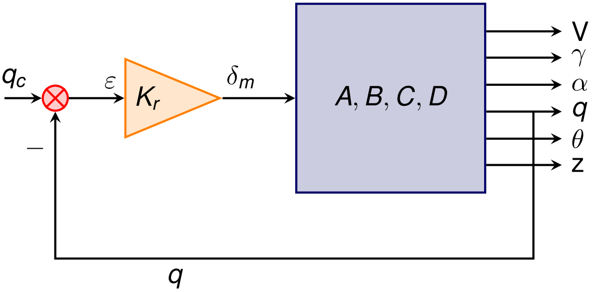


$$\frac{q}{q_c} = \frac{K_r \frac{q}{\delta_m}}{1 + K_r \frac{q}{\delta_m}}$$

$$\frac{q}{\delta_m} = \frac{K_3(T_\alpha s + 1)}{\frac{s^2}{\omega_{af}^2} + 2\xi_{af}\frac{s}{\omega_{af}} + 1}$$

$$\frac{q}{q_c} = \frac{K_r K_3\,(T_\alpha s + 1)}{\frac{s^2}{\omega_{af}^2} + 2\xi_{af}\frac{s}{\omega_{af}} + 1 + K_r K_3(T_\alpha s + 1)}$$

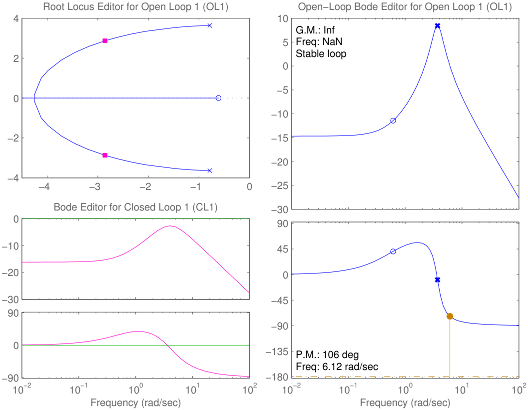
One could use the state space representation instead of the transfer function.



We can tune the controller gain by using sisotool, a tool that allows to tune a proportional controller (a simple gain in the direct chain) of single input single output (SISO) system. This tool provides:

- the current value of the gain
- the closed loop step response (time response)
- the open loop bode diagram (frequency response)
- the Evans root locus (the real and imaginary parts of the system closed loop poles for the current gain, and a curve giving all the possible positions of the closed loop poles of the system when the gain varies from 0 to ∞)
- the values of the real and imaginary parts of the closed loop poles of the system, with their damping ratio and their proper pulsation
- for the step response, the overshoot (OS) and the settling time to within 5%
- for the bode diagram, the gain margin (GM) and the phase margin (PM)

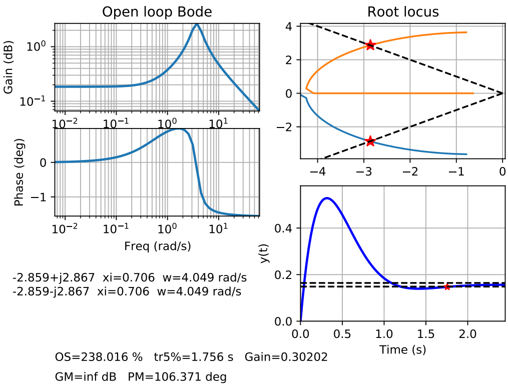
With Matlab



$\forall K_r < 0$ the poles stay in the left half plane and the closed loop with K_r is stable.

With Python (iPython) and sisopy31 (note the trick to tune a negative gain):

```
sisotool(-TqDm_tf)
```



$$\frac{q}{q_c} = \frac{K_r K_3 \left(T_\alpha s + 1\right)}{\left(1 + K_r K_3\right) \left(\frac{s^2}{(1+K_r K_3)\omega_{af}^2} + \frac{\left(\frac{2\xi_{af}}{\omega_{af}} + K_r K_3 T_\alpha\right)}{1+K_r K_3} s + 1\right)}$$

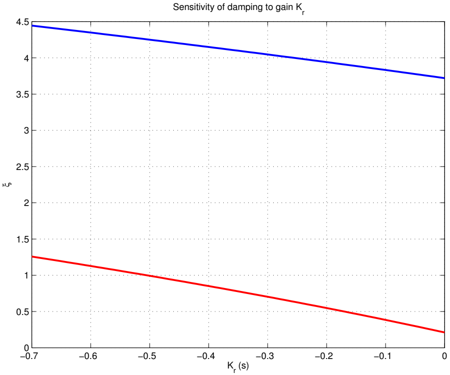
By identification with a second order at denominator,

THE NEW PROPER PULSATION IS

$$\omega = \omega_{af} \sqrt{1 + K_r\ K_3}$$

THE NEW DAMPING RATIO IS THEN

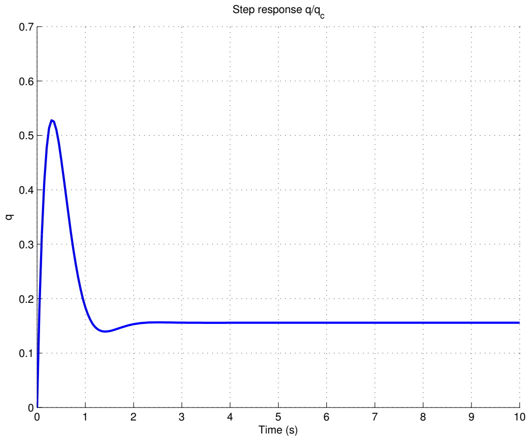
$$\xi = \frac{\omega}{2} \left(\frac{\frac{2\xi_{af}}{\omega_{af}} + K_r\ K_3\ T_\alpha}{1 + K_r\ K_3} \right)$$



ω variation is relatively small with K_r compare to ξ variation with K_r.
So K_r allows to fix the closed loop damping ratio of the controlled aircraft.

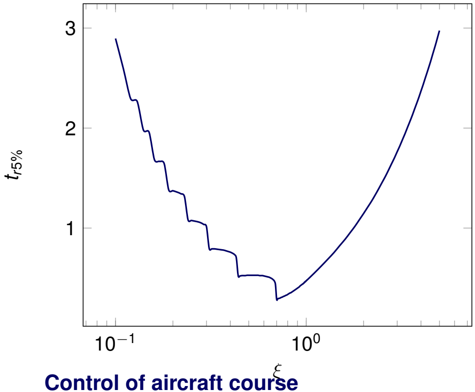
We choose $K_r = -0.302$ to have a damping ratio of $\xi = 0.707 = \frac{1}{\sqrt{2}}$.

Step response q/q_c



Why choosing a damping ration of 0.7? For a second order system, with a transfer function

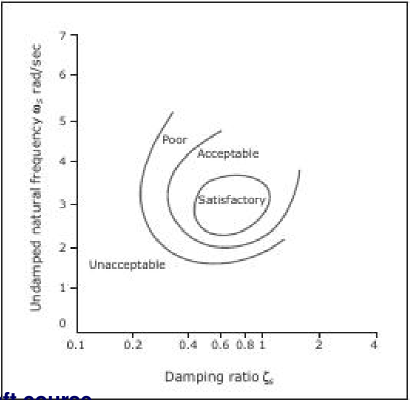
$$f(s) = \frac{1}{\frac{s^2}{\omega_n} + 2\xi\frac{s}{\omega_n} + 1}$$



We can, for a given ω_n, which is 10 rad/s in this example, plot the settling time to within 5% as a function of damping ratio ξ.
The settling time to within 5% is minimal for $\xi \approx 0.7$.

FLYING QUALITY OF AN AIRCRAFT

May serve to specify control requirements to chose the values of the closed loop damping ratio (around 0.7) and the proper pulsation around 3 rad/s.

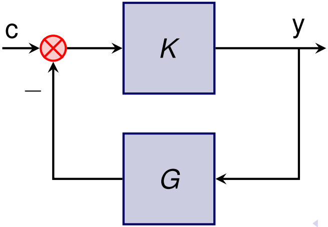


FEEDBACK COMMAND

The `feedback` command allows to create a closed loop system. Let $K(s)$ and $G(s)$ be the transfer functions of 2 systems K and G.
The command

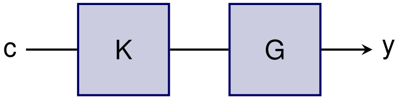
```
syscl=control.feedback(K,G)
```

creates the closed loop system `syscl` transfer function of the following system (note that by default, the feedback is negative). Note that we could also have used state space representations instead of transfer functions.



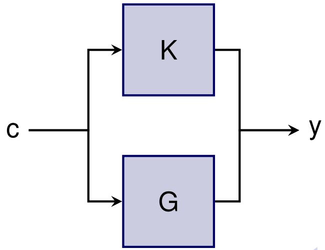
SERIES COMMAND

The `series` command allows to create a new system constituted of the 2 systems mounted in series.
Let $K(s)$ and $G(s)$ be the transfer function of 2 systems K and G.
The command `syscl=control.series(K,G)` gives the transfer function of the following system (between c and y). Note that we could have used the state space representations instead of the transfer function.



For the interconnections of systems, the control toolbox also provides the following commands:

- `control.matlab.append`: to create an augmented system by combining multiple systems
- `control.matlab.connect`: to connect inputs and outputs for example of an augmented systems
- `control.matlab.parallel`: returns the resulting system from mounting multiple systems in parallel



REPRESENTING THE Q FEEDBACK CLOSED LOOP

With the command `feedback` we can build the closed system (aircraft model + q feedback loop):

```
>>figure
>>Kr=-0.302;
>>ftbf=feedback(Kr*TqDm_tf,1);
>>step(ftbf)
>>damp(ftbf)
      Eigenvalue           Damping      Freq. (rad/s)

 -2.86e+00 + 2.87e+00i     7.06e-01        4.05e+00
 -2.86e+00 - 2.87e+00i     7.06e-01        4.05e+00
>>figure
>>sys2=feedback(Kr,TqDm_tf);
>>alpha_qc=series(sys2,TaDm_tf);
>>step(alpha_qc)
```

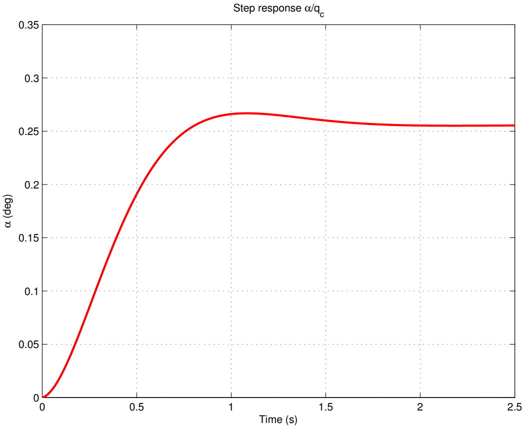
And with Python:

```
figure(4)
Yqcl,Tqcl=control.matlab.step(closedLoopq_qc,arange(0,5,0.01))
plot(Tqcl,Yqcl,'b',lw=2)
plot([0,Tqcl[-1]],[Yqcl[-1],Yqcl[-1]],'k--',lw=1)
plot([0,Tqcl[-1]],[1.05*Yqcl[-1],1.05*Yqcl[-1]],'k--',lw=1)
plot([0,Tqcl[-1]],[0.95*Yqcl[-1],0.95*Yqcl[-1]],'k--',lw=1)
minorticks_on()
grid(b=True, which='both')
title(r'Step response $q/q_c$')
xlabel('Time (s)')
ylabel(r'$q$ (rad/s)')

Osqcl,Trqcl,Tsqcl=step_info(Tqcl,Yqcl)
yyqcl=interp1d(Tqcl,Yqcl)
plot(Tsqcl,yyqcl(Tsqcl),'rs')
text(Tsqcl,yyqcl(Tsqcl)-0.02,Tsq)
print('q Settling time 5%%     = %f s'%Tsqcl)
```

```
sys2=control.feedback(control.tf(Kq,1),TqDm_tf)
alpha_q_tf=control.series(sys2,TaDm_tf)
Yalqcl,Talqcl=control.matlab.step(alpha_q_tf,arange(0,5,0.01))
figure(5)
plot(Talqcl,Yalqcl,'b',lw=2)
grid(True)
title(r'Step response $\alpha/q_c$')
xlabel('Time (s)')
ylabel(r'$\alpha$ (rad/s)')
```

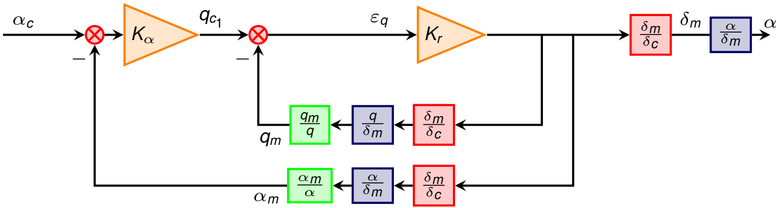
STEP RESPONSE α/q_c

Step response α/q_c



LOOP BY LOOP APPROACH: SISOTOOL

Two loop controller: q and α (sensors and actuators models have been added). Note that we keep the previous tuning of the q feedback loop: K_r is kept unchanged.



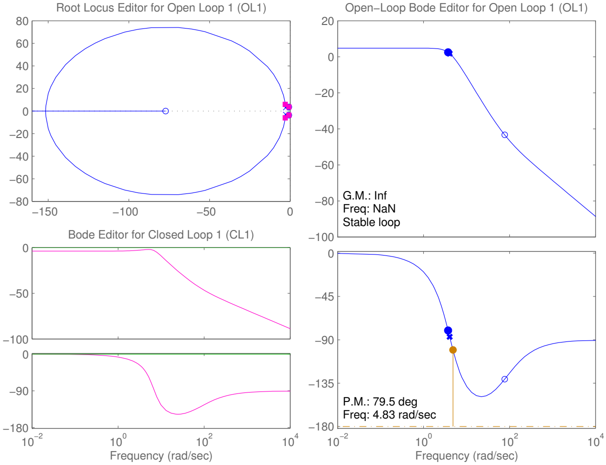
$$K_\alpha = 6.78$$

in order to have

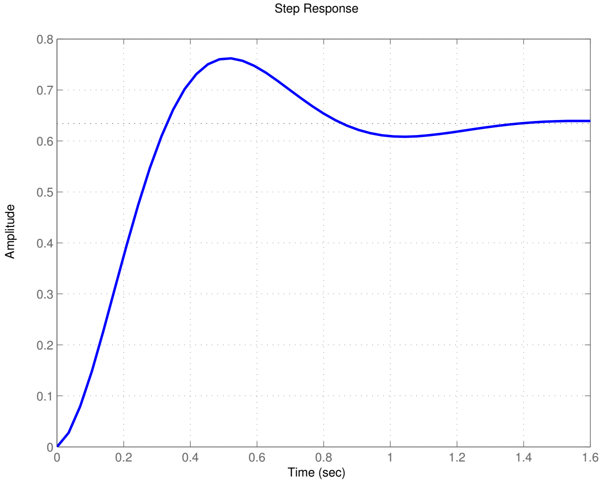
$$\xi = 0.45$$

$$K_r = -0.302$$

$\forall K_\alpha > 0$, the closed loop system is stable



Step response with feedback in q and α



CONTROL AND GUIDANCE MODES

- Control
  - Stabilization and servo-control around the center of gravity
  - Base modes:
    - incidence hold,
    - acceleration hold,
    - pitch angle hold or
    - flight path angle hold
- Guidance
  - Flight scheduling management and control of movements of the center of gravity of the aircraft, calculation of the commands sent to the control loop
  - Superior modes:
    - heading hold,
    - altitude hold,
    - go towards an objective which can be fix or mobile.

There exists 3 control loops for the longitudinal plane:

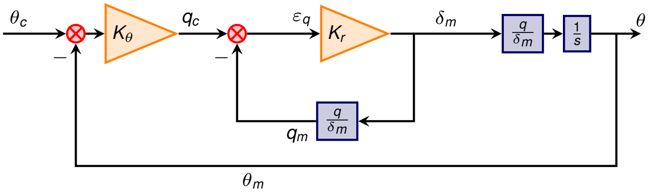
- the fin control loop: servo-control of the deflection or the angular speed of the fin
- the mean control loop for base modes
- the external loop for guidance modes

General method

- The synthesis is done from the internal loop to the most external loop (one by one, and at each step, the tuning of the preceding internal loop is preserved).
- We assumed that sensors and actuators are perfect.
- The control laws are linear (in practice, saturation will be introduced in the control loops).

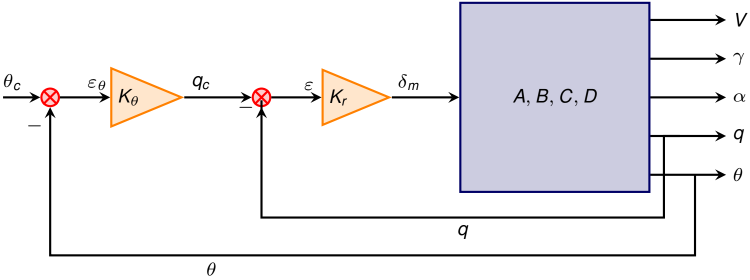
PITCH ANGLE HOLD MODE

Instead of adding a second loop which controls α, the second controller uses the pitch angle θ.



PITCH ANGLE HOLD MODE

The state space representation can be used equivalently.



$$\frac{\delta_{mc}}{q_c} = \frac{K_r\left(\frac{s^2}{\omega_{AF}^2} + \frac{2\xi_{AF}}{\omega_{AF}}s + 1\right)}{\frac{s^2}{\omega_{AF}^2} + \left(\frac{2\xi_{AF}}{\omega_{AF}} + k_r K_3 T_\alpha\right)s + 1 + k_r K_3}$$

$$T_{\theta BO} = \frac{\theta}{\theta_c - \theta} = K_\theta \frac{1}{s}\frac{\delta_{mc}}{q_c}\frac{q}{\delta_m}$$

$$T_{\theta BO} = \frac{K_\theta}{s}\frac{K_r K_3\left(T_\alpha s + 1\right)}{\frac{s^2}{\omega_{AF}^2} + \left(\frac{2\xi_{AF}}{\omega_{AF}} + k_r K_3 T_\alpha\right)s + 1 + k_r K_3}$$

$$T_{\theta BF} = \frac{\theta}{\theta_c} = \frac{T_{\theta BO}}{1 + T_{\theta BO}}$$

$$T_{\theta BF} = \frac{K_\theta K_r K_3 \left(T_\alpha s + 1\right)}{\frac{s^3}{\omega_{AF}} + \left(\frac{2\xi_{AF}}{\omega_{AF}} + K_r K_3 T_\alpha\right) s^2 + \left(1 + K_r K_3 + K_\theta K_r K_3 T_\alpha\right) s + K_\theta K_r K_3}$$

We can write

$$T_{\theta BF} = \frac{T_{\theta BO}}{1 + f_\theta g_\theta(s)}$$

so that g_θ is normalized, meaning with a coefficient equal to one for terms of highest order in s at the numerator and the denominator.

$$T_{\theta BO} = K_\theta K_r K_3 T_\alpha \frac{\left(s + \frac{1}{T_\alpha}\right)}{s^3 + \left(2\xi_{AF}\omega_{AF} + k_r K_3 T_\alpha \omega_{AF}^2\right) s^2 + \left(1 + k_r K_3\right)\omega_{AF}^2 s}$$

$$T_{\theta BO} = l_\theta g_\theta(s) = K_\theta K_r K_3 T_\alpha g_\theta(s)$$

l_θ is then the gain used for drawing the root locus.
It is possible to put $g_\theta(s)$ under the form

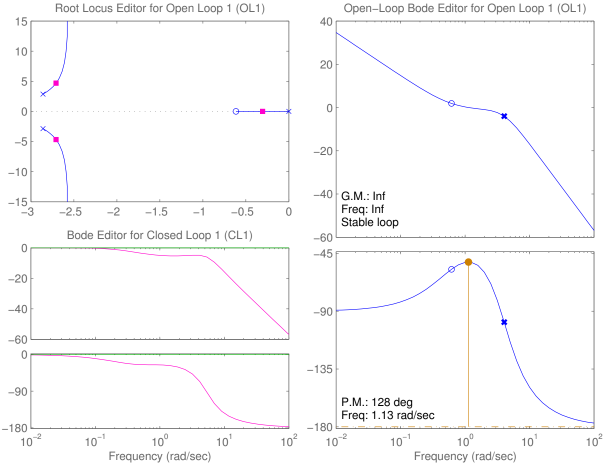
$$g_\theta(s) = \frac{s - z_1}{s(s - p1)(s - p2)}$$

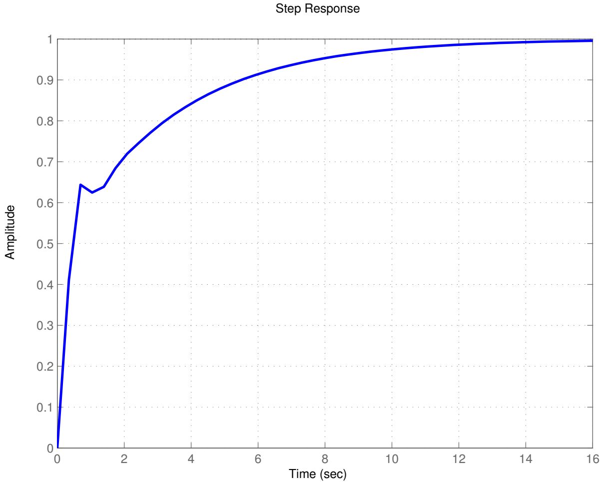
$g_\theta(s)$ possesses p=3 poles and z=1 zero.
The Evans root locus possesses p-z=2 infinite branches, with directions $\dfrac{2\lambda + 1}{p - z}\pi$ with $\lambda \in [0, p - z - 1]$, or $\theta_a = \pm\dfrac{\pi}{2}$
The intersection point of branches asymptotes with real axis is

$$\sigma_a = \frac{\sum_{j=1}^{p} p_j - \sum_{i=1}^{z} z_i}{p - z} = \frac{0 + p_1 + p_2 - z_1}{2}$$

For k=0, the locus begin at the poles and with increasing k arrive either to the zeros or go to infinity when k tends towards infinity.

```
ktheta=1;
disp('Ttheta')
Ttheta=tf(1,[1 0])*TqDm_bf
T = sisoinit(1);                % single-loop configuration with
                                % C in the forward path
T.G.Value = Ttheta;             % model for plant G
T.C.Value = tf(ktheta,1);       % initial compensator value
T.OL1.View = {'rlocus','bode'}; % views for tuning Open Loop OL1
sisotool(T)
% ktheta obtained with sisotool to have damping ratio of 0.5
ktheta=3.48;
Ttheta_bo=ktheta*Ttheta;
%Ttheta_bf0=Ttheta_bo/(1+Ttheta_bo);
disp('Ttheta_bf0')
Ttheta_bf0=feedback(Ttheta_bo,1)
figure(20)
clf
step(Ttheta_bf0)
damp(Ttheta_bf0)
grid on
title('Step response \theta/\delta_m')
```


Root Locus Editor for Open Loop 1 (OL1)
Open−Loop Bode Editor for Open Loop 1 (OL1)
Bode Editor for Closed Loop 1 (CL1)
G.M.: Inf
Freq: Inf
Stable loop
P.M.: 128 deg
Freq: 1.13 rad/sec
Frequency (rad/sec)
Frequency (rad/sec)


Step Response
Amplitude
Time (sec)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16

```
>> Ttheta

Transfer function:
     4.149 s + 2.553
-------------------------
s^3 + 5.718 s^2 + 16.39 s

>> Ttheta_bf0

Transfer function:
         14.44 s + 8.884
---------------------------------
s^3 + 5.718 s^2 + 30.83 s + 8.884

>> damp(Ttheta_bf0)

       Eigenvalue             Damping     Freq. (rad/s)

 -3.04e-01                    1.00e+00       3.04e-01
 -2.71e+00 + 4.68e+00i        5.01e-01       5.40e+00
 -2.71e+00 - 4.68e+00i        5.01e-01       5.40e+00
```

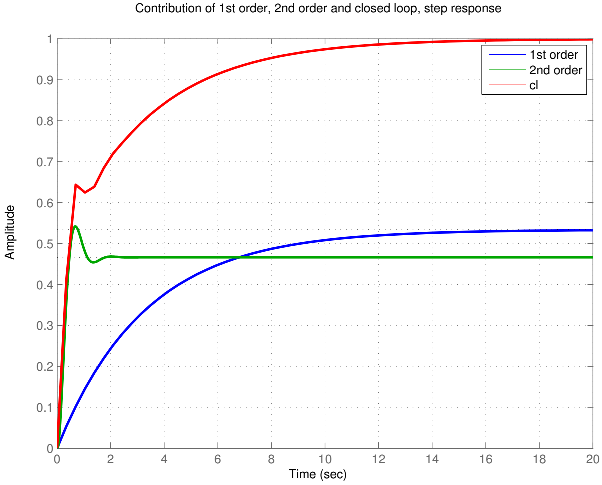
- $\forall K_\theta$ the pitch angle hold mode is stable.
- $\forall K_\theta$ the pitch angle hold always has an aperiodic mode and a pseudo-periodic mode.
- For the pseudo-periodic mode, the damping ratio ξ_θ decreases when K_θ increases.
- The maximum value of ξ_θ is fixed by the setting of the damping ratio of the q feedback loop.
- The choice of K_θ is a compromise between the will to move away the real pole far from imaginary axis to decrease the response time and on the contrary to not get it too far away from the imaginary axis to have a sufficient damping ratio (not too low).

Let's take a look at how the first order pole (aperiodic) and the second order pole (pseudo-periodic) influence the step response. We are going to decompose the transfer function into a sum of fraction, each fraction being linked to a pole.

The mode of the first order fixes the dynamics of the system.

```
% Decomposition in simple elements
[nTheta_bf0,dTheta_bf0]=tfdata(ss(Ttheta_bf0),'v');
% fraction expansion
[r,p,k]=residue(nTheta_bf0,dTheta_bf0)
r2=[r(1);r(2)];
p2=[p(1);p(2)];
% transformation in num,den for the 2 conjugate poles
[n2,d2]=residue(r2,p2,k);
disp('Transfer function of the second order')
Ttheta_bf0_ord2=tf(n2,d2)
disp('Transfer function of the first order')
[n1,d1]=residue(r(3),p(3),k);
Ttheta_bf0_ord1=tf(n1,d1)

figure(22)
step(Ttheta_bf0_ord1,Ttheta_bf0_ord2,Ttheta_bf0)
title('Contribution 1st order and 2nd order and step response')
legend('1st order','2nd order','cl')
grid on
```

```
r =
  -0.0812 - 1.5024i
  -0.0812 + 1.5024i
   0.1625
p =
  -2.7069 + 4.6752i
  -2.7069 - 4.6752i
  -0.3044
k =
     []
second order transfer function
Transfer function:
  -0.1625 s + 13.61
---------------------
s^2 + 5.414 s + 29.18

First order transfer function
Transfer function:
  0.1625
----------
s + 0.3044
```


Contribution of 1st order, 2nd order and closed loop, step response
1st order
2nd order
cl
Amplitude
Time (sec)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16
18
20

The corresponding Python code is:

```
# q controller gain
Kq=-0.302
# q feedback loop transfer function
closedLoopq_qc=control.feedback(control.series(control.tf(Kq
    ,1),TqDm_tf,1))
# theta is the intergral of q
Ttheta=control.series(control.tf(1,[1,0]),closedLoopq_qc)
# theta controller gain
ktheta=3.48
# theta and q controlled loops transfer function
Tthetabf=control.feedback(ktheta*Ttheta,tf(1,1),-1)
```

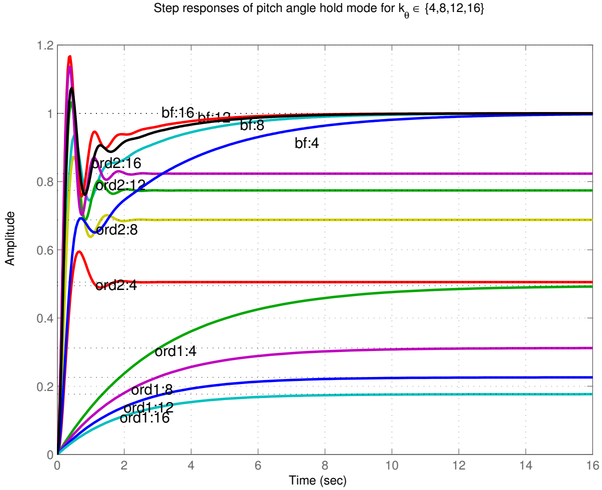
```
# fraction expansion of the theta and q closed loop
r,p,k=scipy.signal.residue(Tthetabf.num[0][0],Tthetabf.den
    [0][0])

r2=np.array([r[1],r[2]])
p2=np.array([p[1],p[2]])
n2,d2=scipy.signal.invres(r2,p2,k)
Tthetabf_2ndorder=control.tf(n2,d2)
print("Tthetabf_2ndorder")
print(Tthetabf_2ndorder)
r1=[r[0]]
p1=[p[0]]
n1,d1=scipy.signal.invres(r1,p1,k)
Tthetabf_1storder=control.tf(n1,d1)
print("Tthetabf_1storder")
print(Tthetabf_1storder)
```

- The step response clearly shows that the mode of the 1st order is dominant.
- The settling time to within 5 % is large, of about 7 s.
- We will study the sensitivity of the responses to variations of K_θ.

How this behavior varies with different values of the θ gain k_θ?

```
figure(23)
Tktheta=[4 8 12 16];
for ii=1:4
    Ktheta=Tktheta(ii);
    Ttheta_bo=Ktheta*Ttheta;
    Ttheta_bf=feedback(Ttheta_bo,1);
    step(Ttheta_bf,25); hold on; grid on;
    title(['Step responses of pitch angle hold'...
           ' for k_\theta \in \{4,8,12,16\}'])
    [nTheta_bf,dTheta_bf]=tfdata(ss(Ttheta_bf),'v');
    % fraction expansion
    [r,p,k]=residue(nTheta_bf,dTheta_bf)
    r2=[r(1);r(2)];
    p2=[p(1);p(2)];
    % back to num,den for the 2 conjugate poles
    [n2,d2]=residue(r2,p2,k);
```

```
    disp('Transfer function of the second order')
    Ttheta_bf_ord2=tf(n2,d2)
    disp('Transfer function of the first order')
    [n1,d1]=residue(r(3),p(3),k);
    Ttheta_bf_ord1=tf(n1,d1)
    step(Ttheta_bf_ord1,Ttheta_bf_ord2,25)

    [y,x]=step(Ttheta_bf_ord1);
    text(x(10),y(10),sprintf('ord1:%.0f',Ktheta),'FontSize',12)
    [y,x]=step(Ttheta_bf_ord2);
    text(x(30),y(30),sprintf('ord2:%.0f',Ktheta),'FontSize',12)
    [y,x]=step(Ttheta_bf);
    text(x(50),y(50),sprintf('bf:%.0f',Ktheta),'FontSize',12)
end
```


Step responses of pitch angle hold mode for $k_\theta \in \{4,8,12,16\}$
Amplitude
Time (sec)
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
bf:16
bf:12
bf:8
bf:4
ord2:16
ord2:12
ord2:8
ord2:4
ord1:4
ord1:8
ord1:12
ord1:16

When K_θ increases:

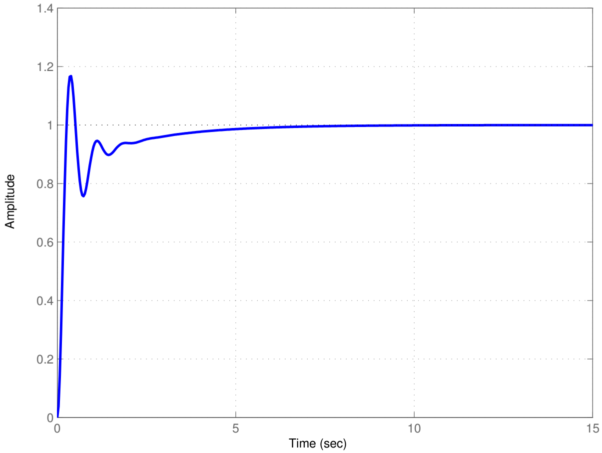
- The static gain of the contribution of the aperiodic mode decreases.
- The static gain of the contribution of the pseudo-periodic mode increases.
- The damping ratio of the contribution of the pseudo periodic mode decreases.
- The settling time to within 5% of the global response decreases.

```
% Choice of a gain ktheta=16
Ktheta=16;
disp('Ttheta_bo=')
Ttheta_bo=Ktheta*Ttheta
disp('Ttheta_bf=')
Ttheta_bf=feedback(Ttheta_bo,1);
damp(Ttheta_bf)
figure(24)
step(Ttheta_bf,15);
grid on
title('Step response for the pitch angle hold mode for K_\theta=16')
```

```
Ttheta_bo=
Transfer function:
     66.38 s + 40.85
-------------------------
s^3 + 5.718 s^2 + 16.39 s

Ttheta_bf=
       Eigenvalue           Damping     Freq. (rad/s)

 -5.10e-01                  1.00e+00       5.10e-01
 -2.60e+00 + 8.56e+00i      2.91e-01       8.95e+00
 -2.60e+00 - 8.56e+00i      2.91e-01       8.95e+00
```

The gain margin is infinite and the phase margin is 39.3 deg at a pulsation of 8.04 rad/s

Step responses of pitch angle hold mode for K_θ=16

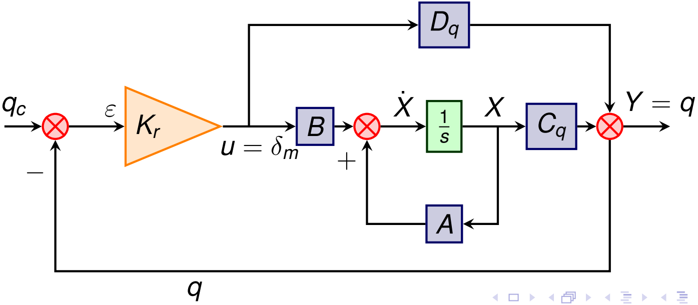


Decoupled Model

$$\begin{pmatrix} \dot{V} \\ \dot{\gamma} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -X_V & -X_\gamma & -X_\alpha & 0 & 0 & 0 \\ Z_V & 0 & Z_\alpha & 0 & 0 & 0 \\ -Z_V & 0 & -Z_\alpha & 1 & 0 & 0 \\ 0 & 0 & m_\alpha & m_q & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & V_{éq} & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V \\ \gamma \\ \alpha \\ q \\ \theta \end{pmatrix} + \begin{pmatrix} 0 \\ Z_{\delta_m} \\ -Z_{\delta_m} \\ m_{\delta_m} \\ 0 \end{pmatrix}$$

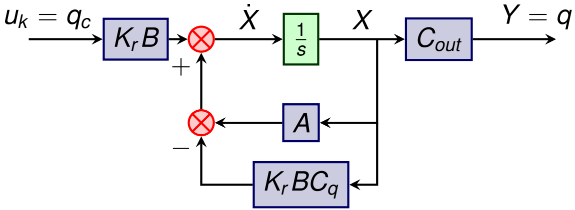
STATE SPACE REPRESENTATION WITH GYROMETRIC FEEDBACK

The approach is to tune the gain loop by loop. When an inner loop gain is fixed, we need to write the state space representation of the inner loop (process+controller) which is going to be used at the next stage.



STATE SPACE REPRESENTATION WITH GYROMETRIC FEEDBACK

D is null, the system can be put under the form



STATE SPACE REPRESENTATION WITH GYROMETRIC FEEDBACK

We now have a new state space representation A_k, B_k, C_k, D_k with

$$A_k = A - K_r B C_q$$

$$B_k = K_r B$$

$$C_k = C_{out}$$

$$D_k = K_r D$$

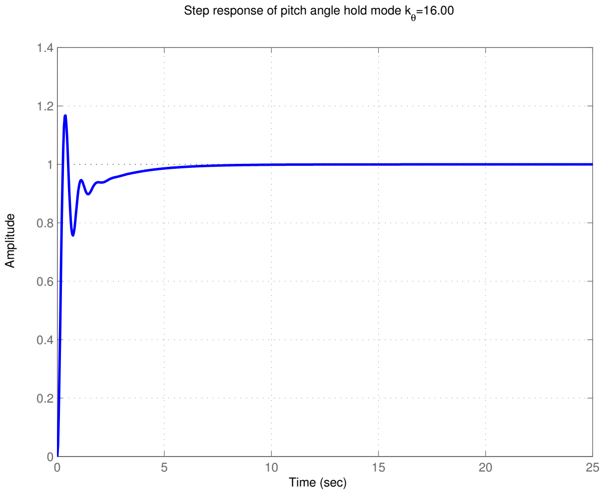
so D_k is null. If we want to have only q as an output, then $C_{out} = C_q$, but for the tuning of other outer loops, C_{out} could be replaced by C_θ or C_γ for example.

```
ktheta=16;
Kr=-0.302
Atheta0=[-Xv -Xgamma  0       0  0
          Zv  0       0       0  0
          0   0      -Zalpha  1  0
          0   0       malpha  mq 0
          0   0       0       1  0]
Btheta0=[0;Zdelta;-Zdelta;mdelta;0];
% X=[V gamma alpha q theta]'
Cq=[0 0 0 1 0];
Dq=0;
Atheta1=Atheta0-Kr*Btheta0*Cq;
Btheta1=Kr*Btheta0;
% X=[V gamma alpha q theta]'
Ctheta=[0 0 0 0 1];
Dtheta=0;
Atheta=Atheta1-Ktheta*Btheta1*Ctheta;
Btheta=Ktheta*Btheta1;
Athetabis=Atheta0-Btheta0*[0 0 0 Kr Kr*Ktheta];
Ctheta2=eye(5);
Dtheta2=zeros(5,1);
```

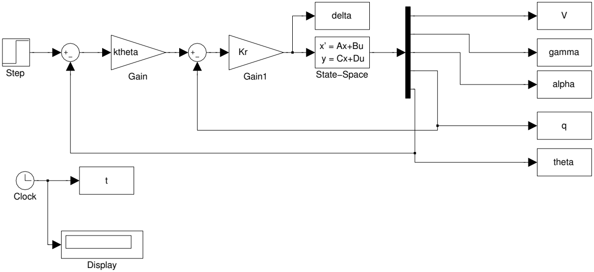
```
Ttheta_bf_ss=ss(Atheta,Btheta,Ctheta,Dtheta);
Ttheta_bf=tf(Ttheta_bf_ss)
disp('Characteristics of v and gamma modes')
damp(Ttheta_bf(1,1))
disp('Characteristics of alpha, q et theta modes')
damp(Ttheta_bf(5,1))
figure(26)
step(Ttheta_bf(5,1),25)
grid on
s=sprintf('Step response of pitch angle hold mode k_\\theta=%.2f',...
          ktheta)
title(s)
```

```
Transfer function from input to output...
           0.03149 s^3 + 0.04941 s^2 + 0.4358 s + 1.304e-16
 #1:  ----------------------------------------------------------
      s^5 + 5.733 s^4 + 82.86 s^3 + 42.08 s^2 + 0.8133 s + 0.106

      -0.8688 s^4 - 1.376 s^3 - 12.05 s^2 - 0.1762 s + 2.528e-15
 #2:  ----------------------------------------------------------
      s^5 + 5.733 s^4 + 82.86 s^3 + 42.08 s^2 + 0.8133 s + 0.106

       0.8688 s^2 + 67.06 s + 2.42e-13
 #3:  ---------------------------------
      s^3 + 5.718 s^2 + 82.78 s + 40.85

             66.38 s^2 + 40.85 s
 #4:  ---------------------------------
      s^3 + 5.718 s^2 + 82.78 s + 40.85

               66.38 s + 40.85
 #5:  ---------------------------------
      s^3 + 5.718 s^2 + 82.78 s + 40.85
```

```
Characteristics of v and gamma modes

      Eigenvalue           Damping     Freq. (rad/s)

 -7.33e-03 + 5.04e-02i     1.44e-01       5.09e-02
 -7.33e-03 - 5.04e-02i     1.44e-01       5.09e-02
 -5.10e-01                 1.00e+00       5.10e-01
 -2.60e+00 + 8.56e+00i     2.91e-01       8.95e+00
 -2.60e+00 - 8.56e+00i     2.91e-01       8.95e+00

Characteristics of alpha, q et theta modes

      Eigenvalue           Damping     Freq. (rad/s)

 -5.10e-01                 1.00e+00       5.10e-01
 -2.60e+00 + 8.56e+00i     2.91e-01       8.95e+00
 -2.60e+00 - 8.56e+00i     2.91e-01       8.95e+00
```

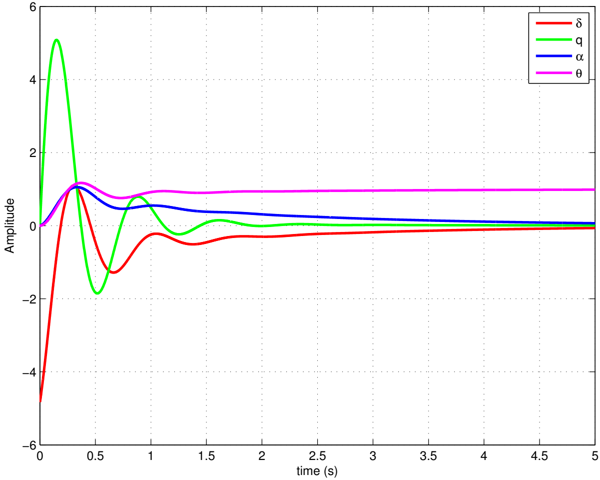

Step response of pitch angle hold mode k_θ=16.00
Amplitude
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25

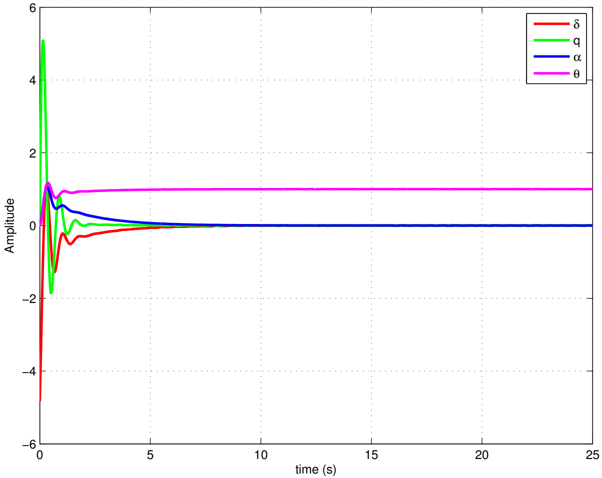
Simulation with Simulink

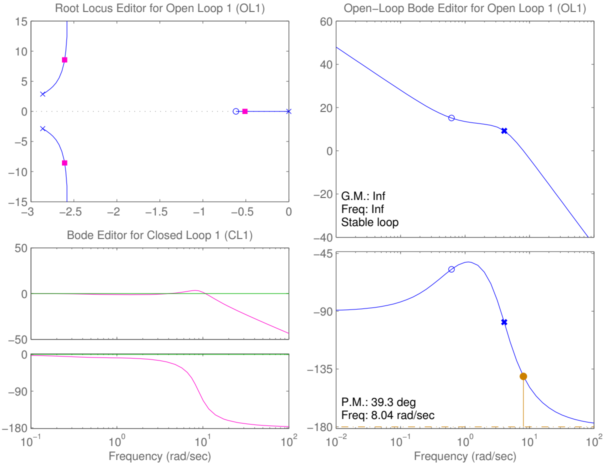


Use of Simulink from command line in Matlab

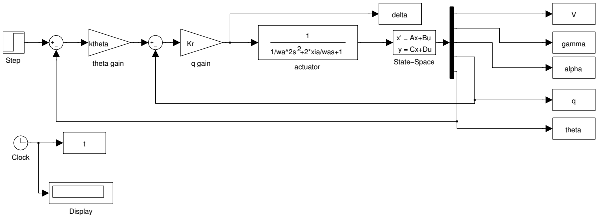
```
% maximum simulation duration fixed to within 5 s
tsim=5;
% launch of the simulink model file pitch_angle_hold.mdl
sim('pitch_angle_hold')
figure(27)
plot(t,delta,'r',t,q,'g',t,alpha,'b',t,theta,'m');
grid on
legend('\delta','q','\alpha','\theta')
xlabel('time (s)')
ylabel('Amplitude')
% maximum simulation duration fixed at 25 s
tsim=25;
sim('pitch_angle_hold')
figure(28)
plot(t,delta,'r',t,q,'g',t,alpha,'b',t,theta,'m');
grid on
legend('\delta','q','\alpha','\theta')
xlabel('time (s)')
ylabel('Amplitude')
```


δ
q
α
θ
Amplitude
time (s)
6
4
2
0
−2
−4
−6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


δ
q
α
θ
Amplitude
time (s)
6
4
2
0
−2
−4
−6
0
5
10
15
20
25


Root Locus Editor for Open Loop 1 (OL1)
15
10
5
0
−5
−10
−15
−3
−2.5
−2
−1.5
−1
−0.5
0
Bode Editor for Closed Loop 1 (CL1)
50
0
−50
0
−90
−180
Frequency (rad/sec)
Open−Loop Bode Editor for Open Loop 1 (OL1)
60
40
20
0
−20
−40
G.M.: Inf
Freq: Inf
Stable loop
−45
−90
−135
−180
P.M.: 39.3 deg
Freq: 8.04 rad/sec
Frequency (rad/sec)

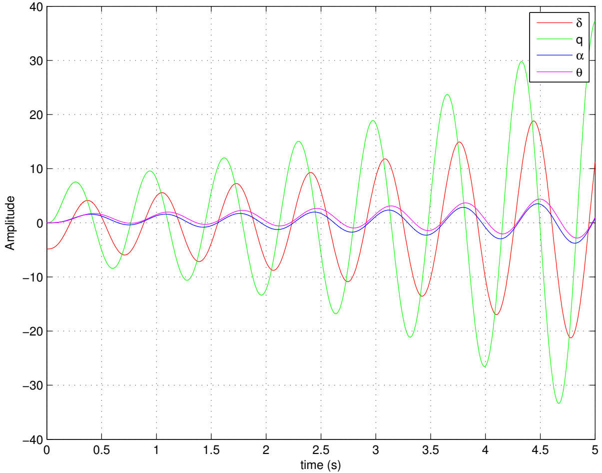
Addition of an actuator in process model



The added actuator has a proper pulsation of 3 Hz and a damping ratio of 0.7

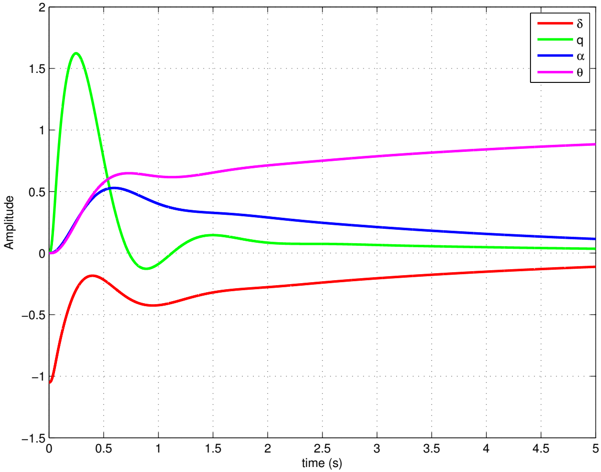
```
tsim=5;
wa=3*2*pi;
xia=0.7;
sim('tpitch_angle_hold_act')
figure(290)
clf
plot(t,delta,'r',t,q,'g',t,alpha,'b',t,theta,'m');
grid on
legend('\delta','q','\alpha','\theta')
xlabel('time (s)')
ylabel('Amplitude')
```

Closed loop step response with the actuator

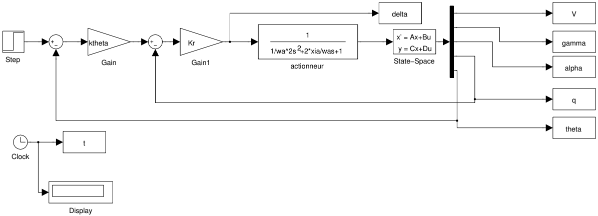


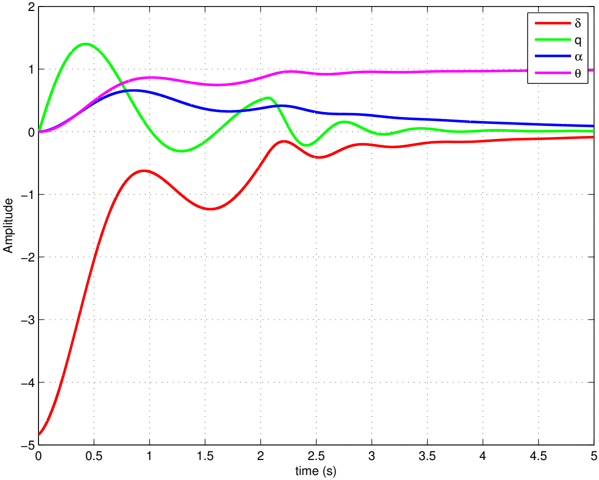
The closed loop bandwidth of the system without the actuator is 12 rad/s, which is 2 Hz. But we need the actuator bandwidth to be largely greater than the bandwidth of the system.
With a faster actuator having a proper frequency of 10 Hz and a damping ratio of 0.7, we obtain

```
tsim=5;
wa=10*2*pi;
xia=0.7;
sim('tpitch_angle_hold_act')
figure(290)
clf
plot(t,delta,'r',t,q,'g',t,alpha,'b',t,theta,'m');
grid on
legend('\delta','q','\alpha','\theta')
xlabel('time (s)')
ylabel('Amplitude')
```


δ
q
α
θ
Amplitude
time (s)
2
1.5
1
0.5
0
−0.5
−1
−1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Add of a saturation at actuator output




δ
q
α
θ
Amplitude
time (s)
2
1
0
−1
−2
−3
−4
−5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

With Python, there seems to be no exact equivalent of Simulink but we can obtain the step response using the control package and its connection functions.

```
Kq=-0.302
ktheta=16
wact=2*pi*10
xiact=0.7
delta_m_delta_mc=control.tf(1,[1/wact**2,2*xiact/wact,1])
sysat=ss(sys.A,sys.B[:,0:1],sys.C,sys.D[:,0:1])
sys_act=series(delta_m_delta_mc,sysat)

# state vector X=[[deltadot,100*delta,V,gamma,alpha,q,theta,
    z]].T
C_q=matrix([[0,0,0,0,0,1,0,0]])
D_q=np.zeros((1,1))
A_q=sys_act.A-sys_act.B*Kq*C_q
B_q=sys_act.B*Kq
C_theta=matrix([[0,0,0,0,0,0,1,0]])
D_theta=np.zeros((1,1))
sysqbf=ss(A_q,B_q,C_theta,D_theta)
```

```
A_theta=A_q-B_q*ktheta*C_theta
B_theta=B_q*ktheta

systhetabf=ss(A_theta,B_theta,C_theta,D_theta)

Ytheta,Ttheta=control.matlab.step(systhetabf,arange
    (0,15,0.05))
figure(9)
plot(Ttheta,Ytheta,'b',lw=2)
grid(True)
title(r'Step response $\theta/\theta_c$ with actuator (
    bandwidth '+str(wact/2/pi)+' Hz)')
xlabel('Time (s)')
ylabel(r'$\theta$ (rad)')
```

NON LINEARITIES AND FILTERS IN THE AUTOPILOT

- Saturations
  - respect of aircraft constraints, of motors operation (load factor, maximum incidence and sideslip, . . .
  - actuators
  - sensors

  For proportional integral controllers, there is a need to introduce a desaturation mechanism (to cancel integration when the signal is saturated), such as an anti-windup filter.
- Filters
  - noises
  - flexible modes
  - add of phase advance/delay controller to shape the controller response with the objective of increasing robustness margin.

ROBUSTNESS MARGIN

- Gain margin
  "gain distance" between the open loop transfer function and the critical point $\{-180°, 0\ dB\}$
  This is the maximum value that the open loop gain could be increased without destabilizing the system.
- Phase margin
  "Phase distance" between the open loop transfer function and the critical point $\{-180°, 0\ dB\}$
  This is the maximum value that the open loop phase could be increased without destabilizing the system.

ROBUSTNESS MARGIN

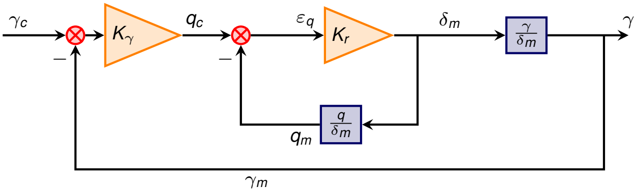
- Delay margin
  equals the phase margin divided by the pulsation to which is found the phase margin:

$$Delay\ margin = \frac{Phase\ margin}{Pulsation\ of\ phase\ margin}$$

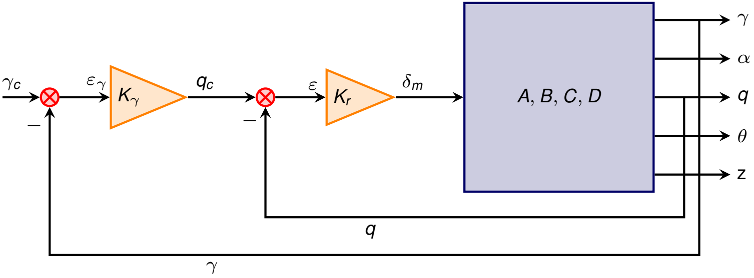
  This is the maximum pure delay that could be introduced in the loop without destabilizing the system.

FLIGHT PATH ANGLE HOLD MODE

- For analysis purpose, the decoupled model is used.
- The aircraft has a servo control for thrust (auto throttle) so that $\frac{dV}{dt} = 0$
- Required performances: $\xi = 0.5$ and an optimized settling time to within 5%.
- We will use a state vector of length 5. But, the variables used are different from those previously used: We will use $X = \begin{pmatrix} \gamma & \alpha & q & \theta & z \end{pmatrix}^T$ instead of $X = \begin{pmatrix} V & \gamma & \alpha & q & \theta \end{pmatrix}^T$


γ_c
K_γ
q_c
ε_q
K_r
δ_m
$\frac{\gamma}{\delta m}$
γ
$-$
$-$
$\frac{q}{\delta m}$
q_m
γ_m

The state space representation could also be used.



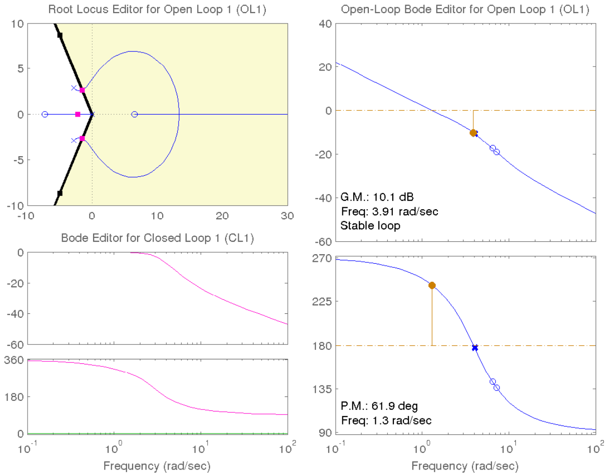
```
% state vector = [gamma alpha q theta z]'
Agam0=[0      Zalpha 0   0 0;
       0     -Zalpha 1   0 0;
       0      malpha mq  0 0;
       0      0      1   0 0;
       V_eq   0      0   0 0];

Bgam=[Zdelta;-Zdelta;mdelta;0;0];
Cgam=eye(5);
Dgam=zeros(5,1);

Kr=-0.302
Cq=[0 0 1 0 0];
Agamk=Agam0-Bgam*Kr*Cq;
Bgamk=Kr*Bgam
Tgam_bo_ss=ss(Agamk,Bgamk,Cgam,Dgam);
Tgam_bo=tf(Tgam_bo_ss);
disp('Transfer function gamma/delta_m')
Tgam_bo(1,1)
disp('Characteristics of gamma/delta_m modes')
damp(Tgam_bo(1,1))
```

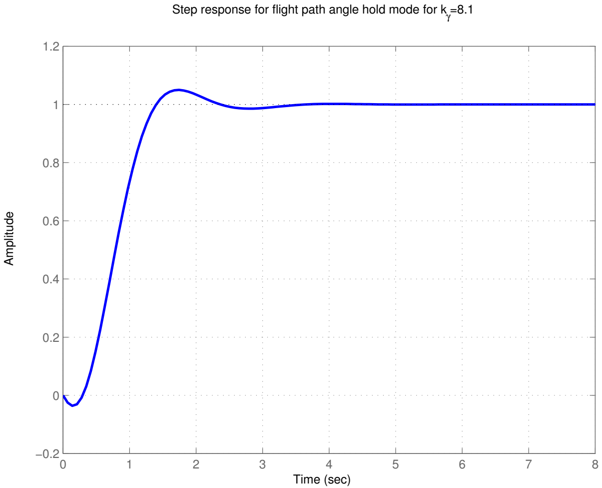
```
T = sisoinit(1);          % single-loop configuration with
                          % C in the forward path
T.G.Value = Tgam_bo(1,1);       % model for plant G
T.C.Value = tf(kgamma,1);  % initial compensator value
T.OL1.View = {'rlocus','bode'};  % views for tuning Open Loop OL1
sisotool(T)
```

```
% choice of kgamma in sisotool to have xi=0.5
kgamma=8.11
disp('Tgam_BO=')
Tgam_BO=kgamma*Tgam_bo(1,1)
disp('Tgam_BF=')
Tgam_bf=feedback(Tgam_BO,1)
damp(Tgam_bf)
figure(30)
set(gcf,'name','step kgamma feedback')
step(Tgam_bf,8)
grid on
s=sprintf('Step response for flight path angle hold mode for k_\\gamma=%.1f',..
          kgamma)
title(s)
```

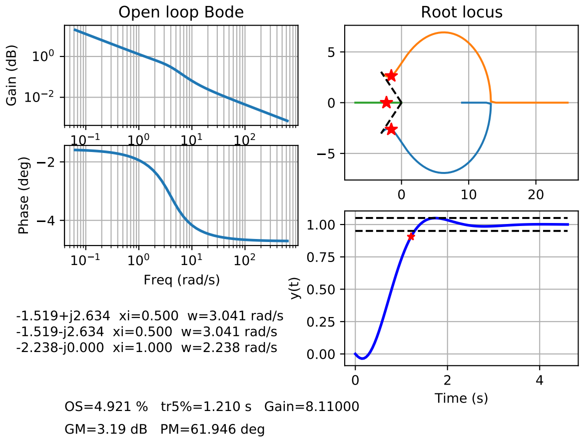
```
Transfer function gamma/delta_m

Transfer function:
-0.0543 s^2 - 0.0424 s + 2.553
------------------------------
  s^3 + 5.718 s^2 + 16.39 s

Characteristics of gamma/delta_m modes

       Eigenvalue           Damping     Freq. (rad/s)

  0.00e+00                -1.00e+00       0.00e+00
 -2.86e+00 + 2.87e+00i     7.06e-01       4.05e+00
 -2.86e+00 - 2.87e+00i     7.06e-01       4.05e+00

Tgam_BO=

Transfer function:
-0.4404 s^2 - 0.3439 s + 20.74
------------------------------
  s^3 + 5.724 s^2 + 16.42 s
```

```
Tgam_BF=

Transfer function:
 -0.4404 s^2 - 0.3439 s + 20.7
--------------------------------
s^3 + 5.278 s^2 + 16.05 s + 20.7


       Eigenvalue           Damping     Freq. (rad/s)

 -2.24e+00                  1.00e+00       2.24e+00
 -1.52e+00 + 2.63e+00i      5.00e-01       3.04e+00
 -1.52e+00 - 2.63e+00i      5.00e-01       3.04e+00
```


Root Locus Editor for Open Loop 1 (OL1)
Open-Loop Bode Editor for Open Loop 1 (OL1)
Bode Editor for Closed Loop 1 (CL1)
G.M.: 10.1 dB
Freq: 3.91 rad/sec
Stable loop
P.M.: 61.9 deg
Freq: 1.3 rad/sec
Frequency (rad/sec)
Frequency (rad/sec)


Step response for flight path angle hold mode for k_γ=8.1
Amplitude
Time (sec)
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
0
1
2
3
4
5
6
7
8

Under Python, with `sisotool` from `sisopy31`



The performances of the closed loop system are

- a settling time to within 5% of 1.2 s
- an overshoot of 4.9%
- a damping ratio of 0.5

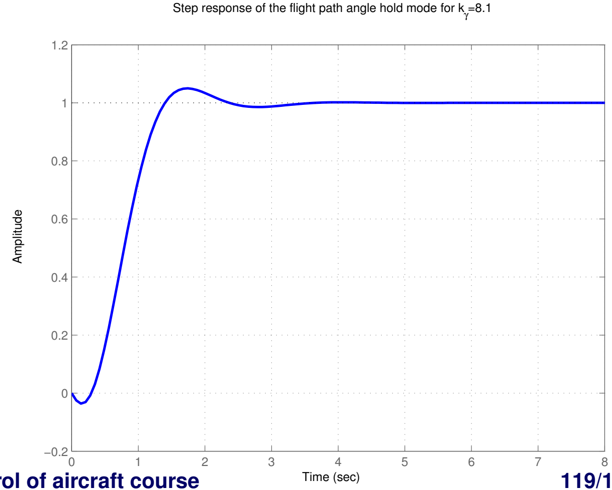
We build the state space representation of the closed loop for the γ feedback:

```
Cgamma=[1 0 0 0 0];
Agambf=Agamk-Bgamk*kgamma*Cgamma;
% which gives Agambf=Agam0-Bgam*[kgamma*Kr 0 Kr 0 0];
Bgambf=kgamma*Bgamk;
Tgam_bf_ss=ss(Agambf,Bgambf,Cgam,Dgam);
Tgam_bfb=tf(Tgam_bf_ss)
disp('Characteristics of the modes of gamma/delta_m')
damp(Tgam_bfb(1,1))

figure(31)
clf
set(gcf,'name','step kgamma feedback')
step(Tgam_bfb(1,1),8)
grid on
s=sprintf('Step response of the flight path angle hold mode'...
  ' for k_\\gamma=%.1f',kgamma);
title(s)
```

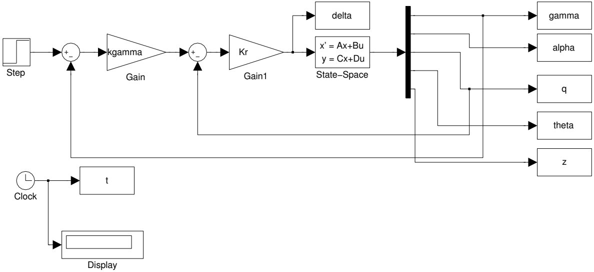
```
Transfer function from input to output...
       -0.4404 s^2 - 0.3439 s + 20.7
 #1:  --------------------------------
      s^3 + 5.278 s^2 + 16.05 s + 20.7

      0.4404 s^2 + 33.99 s - 1.091e-14
 #2:  --------------------------------
      s^3 + 5.278 s^2 + 16.05 s + 20.7

       33.65 s^2 + 20.7 s + 6.896e-15
 #3:  --------------------------------
      s^3 + 5.278 s^2 + 16.05 s + 20.7

         33.65 s^2 + 20.7 s + 1.704e-13
 #4:  ------------------------------------
      s^4 + 5.278 s^3 + 16.05 s^2 + 20.7 s
```

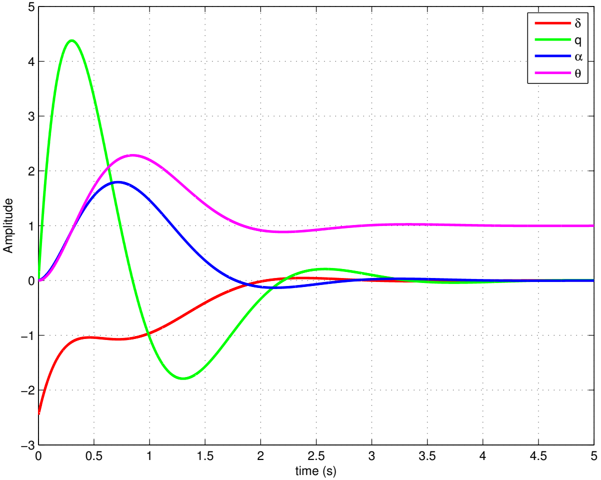
```
          -119.2 s^2 - 93.07 s + 5604
 #5:  ------------------------------------
      s^4 + 5.278 s^3 + 16.05 s^2 + 20.7 s

Characteristics of the modes of gamma/delta_m

       Eigenvalue          Damping     Freq. (rad/s)

 -2.24e+00                 1.00e+00       2.24e+00
 -1.52e+00 + 2.63e+00i     5.00e-01       3.04e+00
 -1.52e+00 - 2.63e+00i     5.00e-01       3.04e+00
```

The step response with the transfer function is the same as the step response of the state space system.



Simulation with Simulink




δ
q
α
θ
Amplitude
time (s)
5
4
3
2
1
0
−1
−2
−3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Filtering and state estimation

Filtering and estimation are necessary in an operating autopilot:

- The measurements are provided with noise, bias... and the measurements have to be filtered so that the controller can behave properly.
- It may happen that we want to control a variable of the state vector that is not measured. And estimator, using a model of the system and by correcting the outputs with the measurements can provide estimates of these missing variables.

One of the most used estimator and filter is the Kalman filter and its variants.

KALMAN FILTER MODEL

The Kalman filter/estimator is a linear estimator that can get an estimation $\hat{X}$ of the state vector X such as the variance of estimation error $E = \hat{X} - X$ is minimum. State model describing the behavior of X with process noise w and measure noise v on Y.

$$\dot{X} = AX + Bu + w$$

$$Y = CX + v$$

w and v Gaussian white noise with covariance matrices Q and R.

$$Q = cov(w) = \begin{pmatrix} Q_{11} & Q_{12} & \dots & Q_{1n} \\ Q_{21} & Q_{22} & \dots & Q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n1} & Q_{n2} & \dots & Q_{nn} \end{pmatrix}$$

with $Q_{ij} = E([w_i - E(w_i)][w_j - E(w_j)])$

CONTINUOUS KALMAN FILTER

We can show that the best estimate of X is given by

$$\dot{\hat{x}} = A\hat{x} + BU + K(y - C\hat{x})$$

$A\hat{x} + BU$ is the propagated state with the model and $K(y - C\hat{x})$ is the correction which takes into account the measurements.
With the innovation gain

$$K = PC^T R^{-1}$$

P being the covariance matrix of the error estimation vector, given by the Ricatti equation

$$\dot{P} = AP + PA^T + Q - PC^T R^{-1} CP$$

DISCRETE KALMAN FILTER

An implementation of the Kalman filter on a computer uses the discrete form of the Kalman filter.
Let's suppose the sample time is T_s

$$X(t) = \Phi(t)X(t0)$$

$$\Phi(t) = \mathcal{L}^{-1}\left((sI - A)^{-1}\right) = e^{At}$$

$$\Phi_k = \Phi(T_s) \approx I + AT_s$$

$$G_k = \int_0^{T_s} \Phi(\tau)Bd\tau$$

$$Q_k = \int_0^{T_s} \Phi(\tau)Q\Phi^T(\tau)d\tau$$

DISCRETE KALMAN FILTER

Time update of the estimate of the state vector

$$\hat{x}_{k|k-1} = \Phi_k \hat{x}_{k-1|k-1} + G_k u_{k-1}$$

Time update of the estimate of the state covariance matrix

$$P_{k|k-1} = \Phi_k P_{k-1|k-1} \Phi_k^T + Q_k$$

$$S_k = C P_{k|k-1} C^T + R_k$$

Kalman gain

$$K_k = P_{k|k-1} C^T S_k^{-1}$$

Measurement update of the state vector

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - C\hat{x}_{k|k-1})$$

Measurement update of the state covariance matrix.

$$P_{k|k} = (I - K_k C) P_{k|k-1}$$

KALMAN FILTER EXAMPLE

We use the matrices A_r, B_r of our aircraft model (with state vector $X = \begin{pmatrix} V & \gamma & \alpha & q \end{pmatrix}^T$) and we want to filter measurements of rotation speed q and acceleration Γ_z and to estimate the unmeasured state variables (V, γ, α)

In the following Python simulation the parameters of noise are

- for Γ_z a noise power of $1^2 \times 0.01\ (m/s^2)^2/Hz$ (Gaussian noise $\mu = 0$ $sigma = 1\ m/s^2$) and a sample time of 0.01 s
  and
- for q a noise power of $0.1^2 \times 0.01\ (rad)^2/Hz$ (Gaussian noise $\mu = 0$ $sigma = 0.1\ rad/s$) and a sample time of 0.01 s

The python code for the linear discrete Kalman filter is given here

```
#!/usr/bin/python
# coding: utf-8

from __future__ import unicode_literals
from matplotlib.pyplot import *
import control
from control.matlab import *
from math import *
from scipy.interpolate import interp1d
from pylab import *
from matplotlib.widgets import Slider
import numpy as np
from atm_std import *
import scipy.interpolate
from sisopy31 import *
```

```
# State vector
# xk=[V gamma alpha q]'
# command
# uk=deltam
# measurement
# yk=[q Gammaz]'

Ar=np.array([[ -0.0147, -0.0362, -0.0011, 0],
[0.0716, 0, 0.7884, 0],
[-0.0716, 0, -0.7884, 1.0000],
[0, 0, -13.2485, -0.7808]])
Br=np.array([[ 0],[ 0.1798], [-0.1798], [-13.7591]])
```

```
Zalpha=0.7884
Zdelta=0.1798
V_eq = 270.6800
Cr=np.array([[0, 0, V_eq*Zalpha, 0],
[0, 0, 0, 1]])
Dr=np.array([[V_eq*Zdelta],[0]])

sigmaGammanoise=0.5
sigmaqnoise=0.05
QN=np.diag([0.0, 0.0, sigmaGammanoise/Zalpha/V_eq,
    sigmaqnoise])**2
RN=np.diag([sigmaGammanoise**2, sigmaqnoise**2])
X0=np.array([0.0, 0.0, 0.0, 0.0])
P0=QN
Ts=0.01
phik=eye(4)+Ar*Ts
Gk=(eye(4)*Ts+Ar*Ts**2/2).dot(Br)
Qk=QN*Ts+(Ar.dot(QN)+QN.dot(np.transpose(Ar)))*Ts**2/2+(Ar.
    dot(QN)).dot(np.transpose(Ar))*Ts**3/3
Rk=RN
```

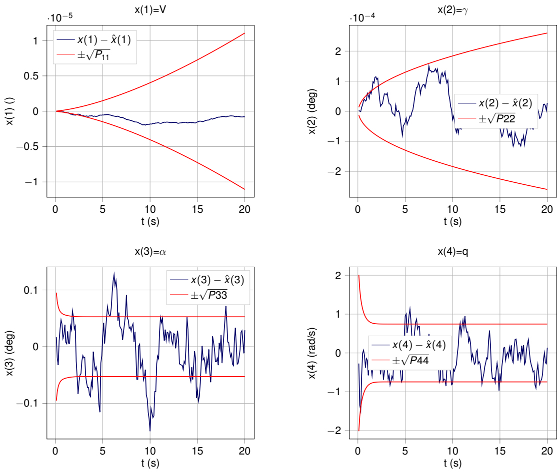
```
# initialization of save arrays
T=[]
X0=[]
X1=[]
X2=[]
X3=[]
Xhat0=[]
Xhat1=[]
Xhat2=[]
Xhat3=[]
Uk=[]
Yk1=[]
Yk2=[]
Ykr1=[]
Ykr2=[]
```

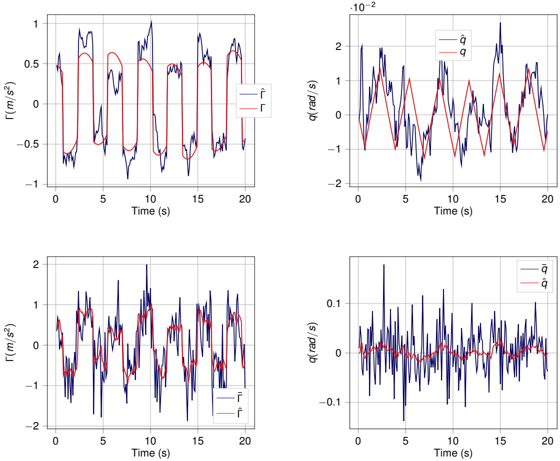
```
Ykhat1=[]
Ykhat2=[]
srP00=[]
srP11=[]
srP22=[]
srP33=[]
err0=[]
err1=[]
err2=[]
err3=[]
Pk=P0
t=0
xhatk=np.array([[0],[0],[0],[0]])
x=np.array([[0],[0],[0],[0]])
```

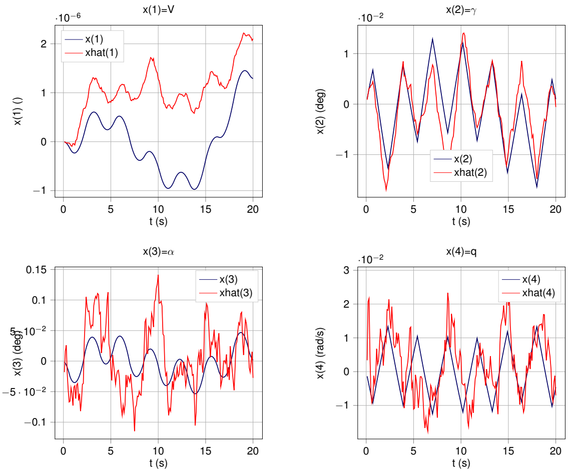
```
# sampling time
Ts=0.1
for i in range(0,200):
  t=t+Ts
  uk=np.array([[0.01*sign(cos(2*pi/(2*pi/2)*t))]])
  xhatk=phik.dot(xhatk)+Gk.dot(uk)
  x=phik.dot(x)+Gk.dot(uk)
  noise=np.array([[sigmaGammanoise,0],[0,sigmaqnoise]]).dot(
      np.random.randn(2,1))
  yk=Cr.dot(x)+Dr.dot(uk)+noise
  ykr=yk-noise
  ykhat=Cr.dot(xhatk)+Dr.dot(uk)
  Pk=(phik.dot(Pk)).dot(np.transpose(phik))+Qk
  Sk=(Cr.dot(Pk)).dot(np.transpose(Cr))+Rk
  Kk=(Pk.dot(np.transpose(Cr))).dot(np.linalg.inv(Sk))
  xhatk=xhatk+Kk.dot(yk-ykhat)
  Pk=(eye(4)-Kk.dot(Cr)).dot(Pk)
```

```
# Save the results
T=np.append(T,[t])
Uk=np.append(Uk,[uk])
Yk1=np.append(Yk1,[yk[0]])
Yk2=np.append(Yk2,[yk[1]])
Ykr1=np.append(Ykr1,[ykr[0]])
Ykr2=np.append(Ykr2,[ykr[1]])
Ykhat1=np.append(Ykhat1,[ykhat[0]])
Ykhat2=np.append(Ykhat2,[ykhat[1]])
Xhat0=np.append(Xhat0,[xhatk[0]])
Xhat1=np.append(Xhat1,[xhatk[1]])
Xhat2=np.append(Xhat2,[xhatk[2]])
Xhat3=np.append(Xhat3,[xhatk[3]])
X0=np.append(X0,[x[0]])
X1=np.append(X1,[x[1]])
X2=np.append(X2,[x[2]])
X3=np.append(X3,[x[3]])
srP00=np.append(srP00,[sqrt(Pk[0,0])])
srP11=np.append(srP11,[sqrt(Pk[1,1])])
srP22=np.append(srP22,[sqrt(Pk[2,2])])
srP33=np.append(srP33,[sqrt(Pk[3,3])])
```

```
err0=X0-Xhat0
err1=X1-Xhat1
err2=X2-Xhat2
err3=X3-Xhat3
```


x(1)=V
$\cdot 10^{-5}$
$x(1) - \hat{x}(1)$
$\pm\sqrt{P_{11}}$
x(1) ()
t (s)
x(2)=γ
$\cdot 10^{-4}$
$x(2) - \hat{x}(2)$
$\pm\sqrt{P22}$
x(2) (deg)
t (s)
x(3)=α
$x(3) - \hat{x}(3)$
$\pm\sqrt{P33}$
x(3) (deg)
t (s)
x(4)=q
$x(4) - \hat{x}(4)$
$\pm\sqrt{P44}$
x(4) (rad/s)
t (s)


$\Gamma(m/s^2)$
Time (s)
$\hat{\Gamma}$
Γ
$\cdot 10^{-2}$
$q(rad/s)$
$\hat{q}$
q
$\bar{\Gamma}$
$\bar{q}$


x(1)=V
$\cdot 10^{-6}$
x(1) ()
t (s)
x(1)
xhat(1)
x(2)=γ
$\cdot 10^{-2}$
x(2) (deg)
t (s)
x(2)
xhat(2)
x(3)=α
x(3) (deg)
t (s)
x(3)
xhat(3)
x(4)=q
$\cdot 10^{-2}$
x(4) (rad/s)
t (s)
x(4)
xhat(4)

TUNING A KALMAN FILTER

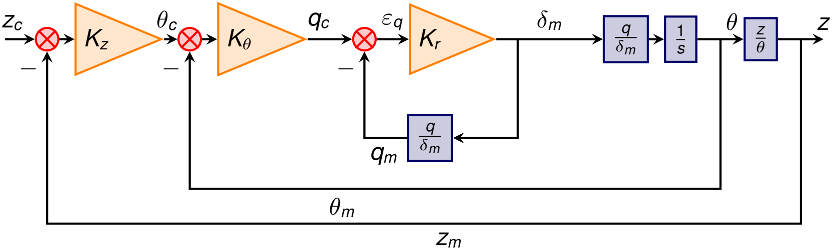
On the previous example, there was no controller. The red graphs around the error correspond to 1 σ errors given by the square root of diagonal terms of the covariance matrix. The errors are expected to lie within $\pm 3\sigma$ for 99.7% of the time, and to lie between $\pm 1\sigma$ for 68.27% of the time.

The way to tune the estimator are:

- P_0, the initial error covariance matrix
- Q the noise process covariance matrix, the higher it is and the lower is the confidence of the filter in the measurements, but it gives robustness to model errors with respect to true system.
- X_0 the initial system state

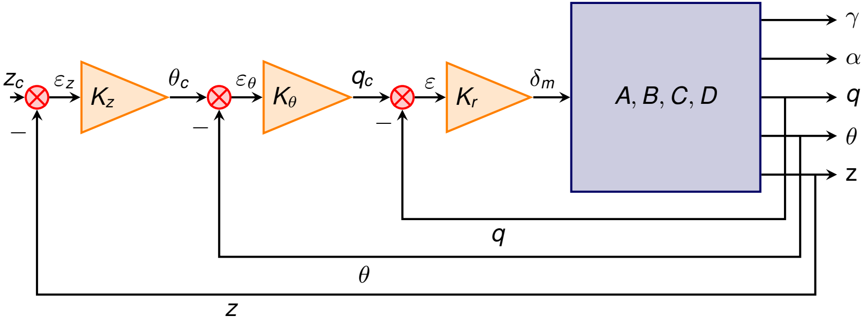
The Kalman filter is able to estimate unmeasured system states, that is very useful with controllers which use full state feedback such as the LQ controller. It can also be useful to manage constraints on variables, such as the incidence α.

GUIDANCE: ALTITUDE HOLD MODE



GUIDANCE: ALTITUDE HOLD MODE

The state space representation can also be used.



For the altitude hold mode, we consider that only the phugoid (slow mode) is preponderant (the incidence oscillation is seen as a fast transient mode). We suppose that the aircraft speed is constant, because there is thrust controller which maintains the speed constant), and that $q \approx 0$.

$$\frac{z}{\theta} = \frac{z}{\gamma}\frac{\gamma}{\theta}$$

$$\dot{z} = V_{éq}\gamma \;\Rightarrow\; \frac{z}{\gamma}(s) = \frac{V_{éq}}{s}$$

$$\dot{\gamma} = Z_\alpha \alpha + Z_{\delta_m}\delta_m = Z_\alpha(\theta - \gamma) + Z_{\delta_m}(K_\theta \theta - K_\theta \theta_c + K_q q)$$

As the pitch angle hold mode is active, $\theta \approx \theta_c$ and in this case q=0. Moreover, the hypothesis allow us to write:

$$\dot{\gamma} \approx Z_\alpha \theta - Z_\alpha \gamma \;\Rightarrow\; \frac{\gamma}{\theta}(s) = \frac{Z_\alpha}{s + Z_\alpha}$$

$$\Rightarrow\; \frac{z}{\theta}(s) = \frac{V_{éq} Z_\alpha}{s(s + Z_\alpha)}$$

Transfer function

$$Tz_{\theta_{BO}}(s) = \frac{z}{z - z_c}(s) = \frac{K_z V_{\acute{e}q} Z_\alpha}{s(s + Z_\alpha)} \times T_{theta_{BF}}$$

$$Tz_{\theta_{BO}}(s) = \frac{K_z V_{\acute{e}q} Z_\alpha}{s(s + Z_\alpha)} \times$$

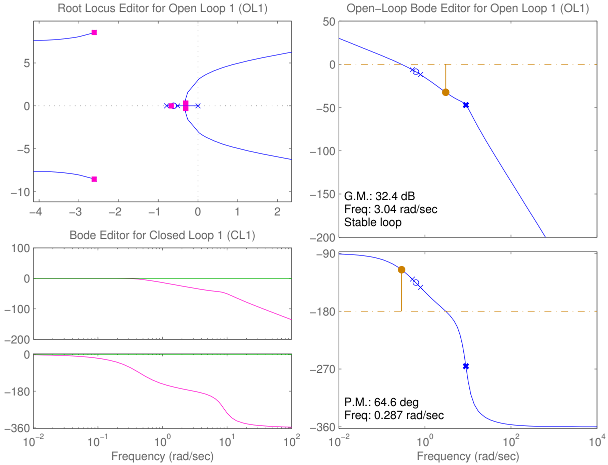
$$\frac{K_\theta K_r K_3 \left(T_\alpha s + 1\right)}{\frac{s^3}{\omega_{AF}} + \left(\frac{2\xi_{AF}}{\omega_{AF}} + K_r K_3 T_\alpha\right) s^2 + \left(1 + K_r K_3 + K_\theta K_r K_3 T_\alpha\right) s + K_\theta K_r K_3}$$

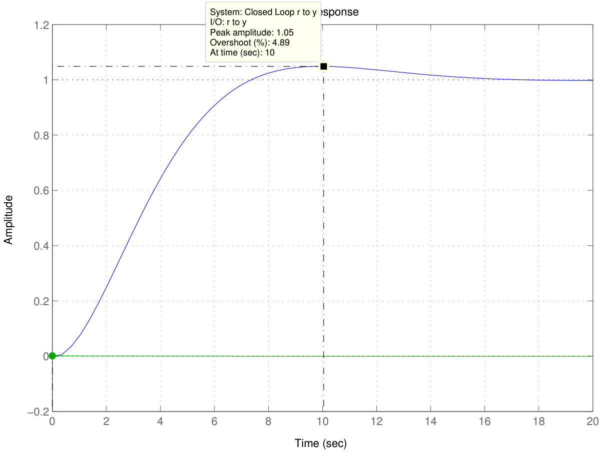
The open loop transfer function has 5 poles and a zero. Then, the Evans root locus will have 5 start points and 1 end point.
Directions of asymptotes:

$$\theta_a = \frac{2\lambda + 1}{4}\pi = \frac{\pi}{4} + \lambda\frac{\pi}{2} \quad avec \quad \lambda \in 0, 1, 2, 3$$

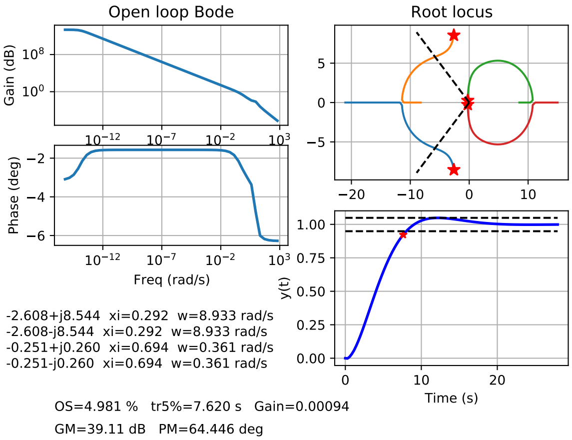
The asymptotes reaches at the point of abscissa $\sigma_a = \frac{\sum poles - \sum zeros}{4}$

```
kz=0.00117;
Tz_theta_BO=Ttheta_bf(5,1)*tf(V_eq*Zalpha,[1 Zalpha 0])
T = sisoinit(1);           % single-loop configuration with
                           % C in the forward path
T.G.Value = Tz_theta_BO;        % model for plant G
T.C.Value = tf(kz,1);  % initial compensator value
T.OL1.View = {'rlocus','bode'};  % views for tuning Open Loop OL1
sisotool(T)

disp('Tz_theta_BO=')
Tz_theta_BO=kz*Tz_theta_BO
disp('Tz_theta_BF=')
Tz_theta_BF=feedback(Tz_theta_BO,1)
damp(Tz_theta_BF)

figure(80)
clf
step(Tz_theta_BF,40)
grid on
title('Step response for altitude hold mode kz=0.00117')
```

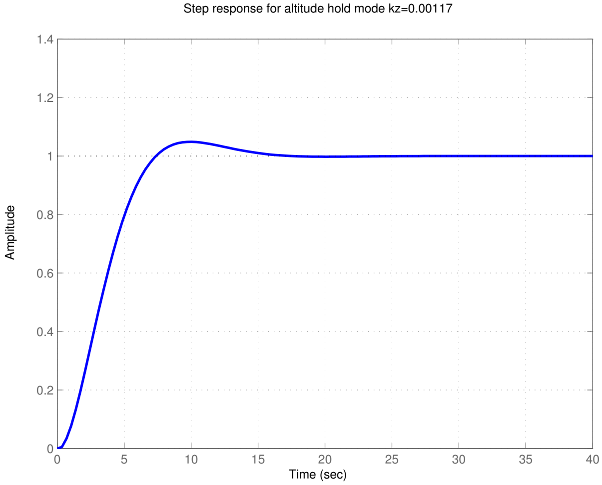

Root Locus Editor for Open Loop 1 (OL1)
Open−Loop Bode Editor for Open Loop 1 (OL1)
G.M.: 32.4 dB
Freq: 3.04 rad/sec
Stable loop
Bode Editor for Closed Loop 1 (CL1)
P.M.: 64.6 deg
Freq: 0.287 rad/sec
Frequency (rad/sec)
Frequency (rad/sec)


System: Closed Loop r to y
I/O: r to y
Peak amplitude: 1.05
Overshoot (%): 4.89
At time (sec): 10
esponse
Amplitude
Time (sec)
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
0
2
4
6
8
10
12
14
16
18
20

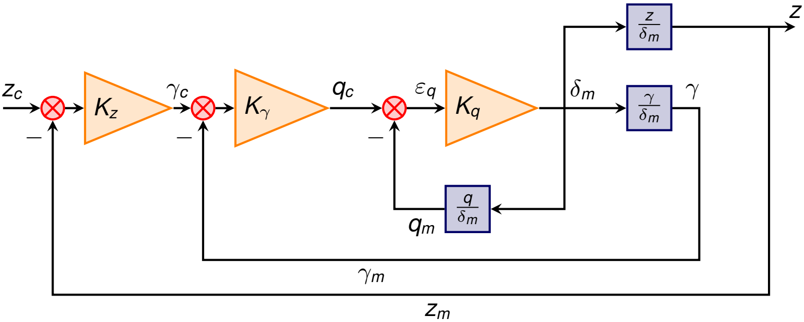
With sisotool under Python



```
z_theta_BO=

Transfer function:
                 16.58 s + 10.2
-------------------------------------------------
s^5 + 6.507 s^4 + 87.29 s^3 + 106.1 s^2 + 32.21 s

Tz_theta_BF=

Transfer function:
                     16.58 s + 10.2
--------------------------------------------------------
s^5 + 6.507 s^4 + 87.29 s^3 + 106.1 s^2 + 48.78 s + 10.2


       Eigenvalue            Damping     Freq. (rad/s)

 -3.06e-01 + 3.09e-01i      7.03e-01        4.35e-01
 -3.06e-01 - 3.09e-01i      7.03e-01        4.35e-01
 -6.74e-01                  1.00e+00        6.74e-01
 -2.61e+00 + 8.55e+00i      2.92e-01        8.94e+00
 -2.61e+00 - 8.55e+00i      2.92e-01        8.94e+00
```

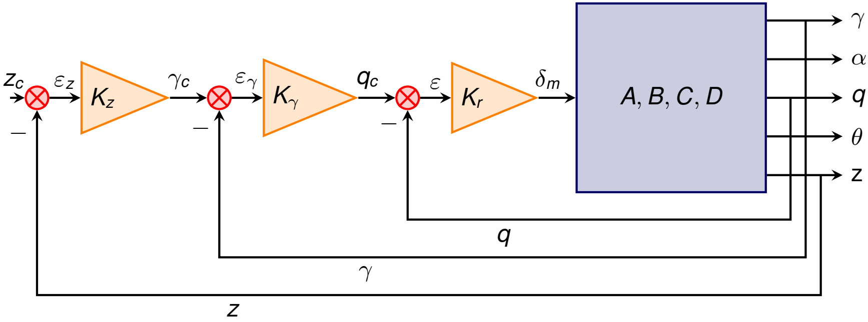

Step response for altitude hold mode kz=0.00117
Amplitude
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
5
10
15
20
25
30
35
40

ALTITUDE HOLD MODE WITH Q, γ AND Z FEEDBACK



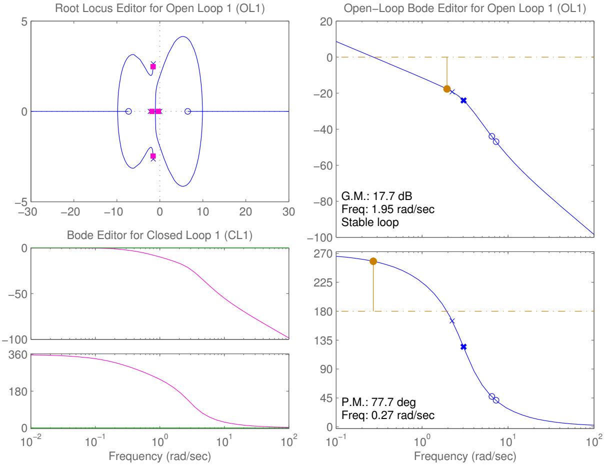
ALTITUDE HOLD MODE WITH Q, γ AND Z FEEDBACK

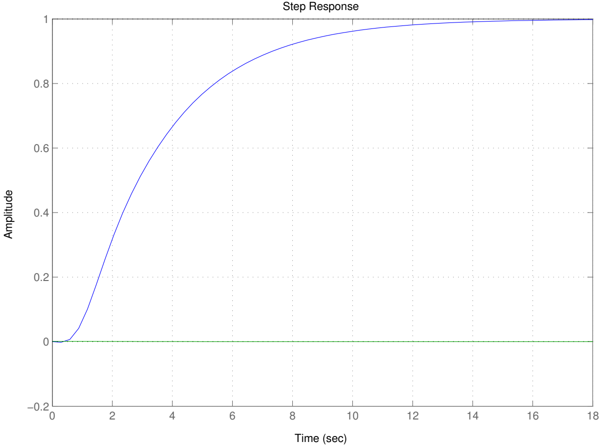
Equivalently, a state space representation can be used.



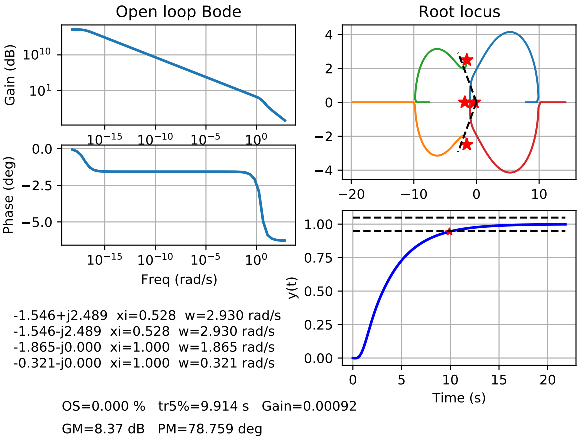
```
Kz=0.001
Tz_gam_bo=Tgam_bfb(5,1)

T = sisoinit(1);          % single-loop configuration with
                          % C in the forward path
T.G.Value = Tz_gam_bo;       % model for plant G
T.C.Value = tf(Kz,1);  % initial compensator value
T.OL1.View = {'rlocus','bode'};  % views for tuning Open Loop OL1
sisotool(T)

disp('Tz_gamma_BO=')
Tz_gam_BO=Kz*Tz_gam_bo
disp('Tz_theta_BF=')
Tz_gam_BF=feedback(Tz_gam_BO,1)
damp(Tz_gam_BF)

figure(90)
clf
step(Tz_gam_BF,20)
grid on
title('Step response for altitude hold mode kz=0.001')
```

```
Tz_gamma_BO=

Transfer function:
   -0.1192 s^2 - 0.09307 s + 5.613
-------------------------------------
s^4 + 5.284 s^3 + 16.08 s^2 + 20.74 s

Tz_theta_BF=

Transfer function:
       -0.1192 s^2 - 0.09307 s + 5.613
---------------------------------------------
s^4 + 5.284 s^3 + 15.96 s^2 + 20.64 s + 5.613


       Eigenvalue           Damping     Freq. (rad/s)

 -3.62e-01                  1.00e+00       3.62e-01
 -1.82e+00                  1.00e+00       1.82e+00
 -1.55e+00 + 2.47e+00i      5.31e-01       2.92e+00
 -1.55e+00 - 2.47e+00i      5.31e-01       2.92e+00
```

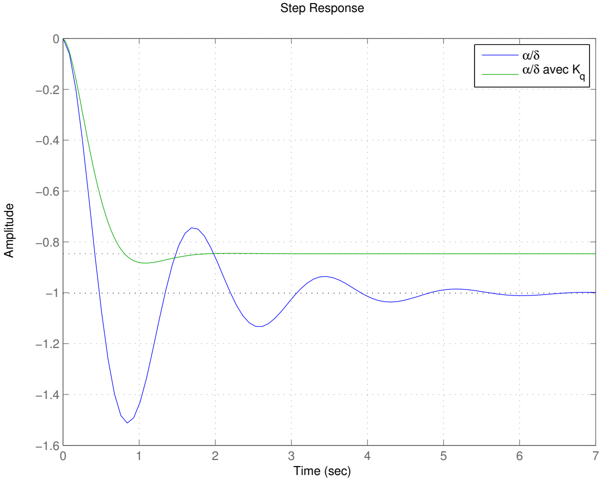

Root Locus Editor for Open Loop 1 (OL1)
Open−Loop Bode Editor for Open Loop 1 (OL1)
G.M.: 17.7 dB
Freq: 1.95 rad/sec
Stable loop
Bode Editor for Closed Loop 1 (CL1)
P.M.: 77.7 deg
Freq: 0.27 rad/sec
Frequency (rad/sec)
Frequency (rad/sec)


Step Response
1
0.8
0.6
0.4
0.2
0
−0.2
Amplitude
0
2
4
6
8
10
12
14
16
18
Time (sec)

With sisotool under Python



FILTERED DAMPER WITH WASHOUT FILTER



We suppose that we use the q feedback loop as an assistance for a human pilot.
In case of damper failure (q loop controller), the human pilot must:

- act smoothly to counter the incidence oscillations
- pilot the aircraft with modified efficiency commands (different open loop and closed loop static gain).

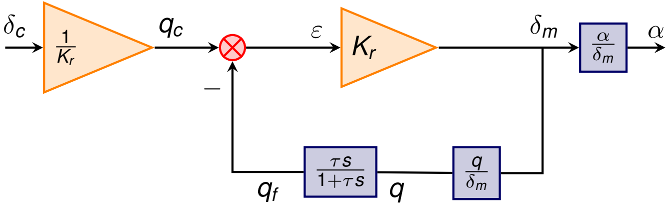
We need to cancel the effects of the damper at low frequency by adding high-pass filter:

$$F_{ph}(s) = \frac{\tau s}{1 + \tau s}$$

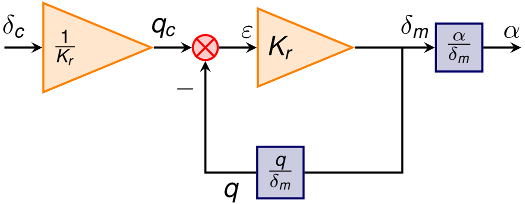
We will choose $\frac{1}{\tau} < \frac{\omega_{AF}}{2}$ ω_{af} is the open loop proper pulsation of the transfer function $\frac{q}{\delta_m}$ and $\frac{\alpha}{\delta_m}$.

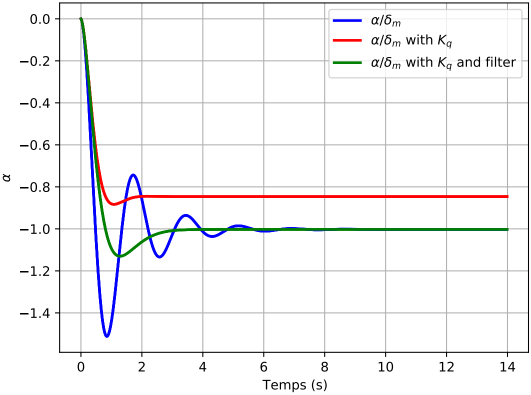
The structure of the control loop with the washout filter takes the following form.

- When $s \gg 1$, at high frequencies (meaning in transient phase), $\frac{\tau s}{1+\tau s} \approx 1$ the control loop is closed and the feedback is active.
- and when $s \ll 1$, at low frequencies (meaning in steady state), $\frac{\tau s}{1+\tau s} \approx 0$ the control loop is open and the feedback is inactive.



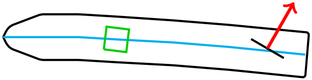
We are going to compare the step response with the washout to the case without it, and also to the open loop case $\frac{\alpha}{\delta}$.





With the washout filter, the steady state response is the same as the open loop response.

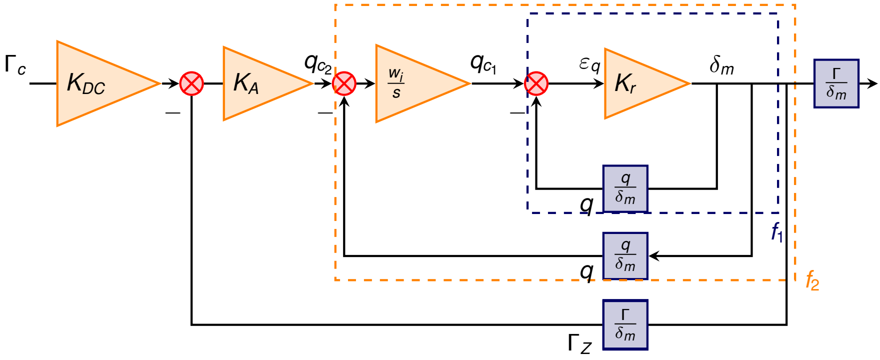
FLEXIBLE MODES



Flexible modes model
The structure of the aircraft deformation under the effects of aerodynamic forces applied on the fins and the lifting surfaces ...
Effects

- possible modification of the aerodynamic coefficients
- disturbance of the accelerometric and gyrometric measures
  By design: chose preferentially the gyrometers location at a vibration node. For the controller design, if the flexible modes are outside the autopilot bandwidth, their effects are attenuated by the use of a band filter (which has a low gain in a narrow frequency band) at fin input.

3 LOOP AUTOPILOT



$$f_1 = \frac{k_r}{1 + k_r \frac{q}{\delta_m}} = \frac{k_r \left(\frac{s^2}{\omega_{af}^2} + \frac{2\xi_{af}}{\omega_{af}} + 1 \right)}{\frac{s^2}{\omega_{af}^2} + \left(T_\alpha k_r K_3 + \frac{2\xi_{af}}{\omega_{af}} \right) s + 1 + k_r K_3}$$

$$f_2 = \frac{\frac{\omega_I}{s} f_1}{1 + \frac{\omega_I}{s} f_1 \frac{q}{\delta_m}}$$

$$f_2 = \frac{k_r \omega_I \left(\frac{s^2}{\omega_{af}^2} + \frac{2\xi_{af}}{\omega_{af}} + 1 \right)}{\frac{s^3}{\omega_{af}^2} + \left(T_\alpha k_r K_3 + \frac{2\xi_{af}}{\omega_{af}} \right) s^2 + \left(k_r(\omega_I T_\alpha + 1) K_3 + 1 \right) s + k_r K_3 \omega_I}$$

$$f_3 = \frac{K_{DC} K_A f_2}{\left(1 + K_A f_2 \frac{\Gamma_Z}{\delta_m} \right)} \frac{\Gamma_Z}{\delta_m}$$

Closed loop transfer function of the controlled aircraft

$$\frac{\Gamma_z}{\Gamma_{z_c}} = \frac{\frac{K_1 K_A K_{DC}}{K_3}\omega_I\left(-\frac{s^2}{\omega_z^2}+1\right)}{\frac{s^3}{\omega_{af}^2 k_r \omega_I K_3} + \left(-\frac{K_1 K_A}{\omega_z^2 K_3} + \frac{T_\alpha}{\omega_I} + \frac{2\xi_{af}}{\omega_{af} k_r \omega_I K_3}\right)s^2 + \left(\frac{k_r K_3(\omega_I T_\alpha + 1) + 1}{k_r \omega_I K_3}\right)s + 1}$$

Which can be written

$$\frac{\Gamma_z}{\Gamma_{zc}} = \frac{\frac{K_1 K_{DC} K_A \omega_I}{K_3}\left(1-\frac{s^2}{\omega_z^2}\right)}{(1+\tau s)\left(\frac{s^2}{\omega^2}+2\xi\frac{s}{\omega}+1\right)}$$

$$\frac{\Gamma_z}{\Gamma_{zc}} = \frac{\frac{K_1 K_{DC} K_A \omega_I}{K_3}\left(1 - \frac{s^2}{\omega_z^2}\right)}{\frac{\tau s^3}{\omega^2} + \left(\frac{2\tau\xi\omega+1}{\omega^2}\right) s^2 + \frac{2\xi+\tau\omega}{\omega} s + 1}$$

It is possible to find the gains of the autopilot k_r, ω_I, K_A by fixing τ, ω and ξ

$$\frac{\Gamma_z}{\Gamma_{zc}} = \frac{\frac{K_1 K_{DC} K_A \omega_I}{K_3}\left(1 - \frac{s^2}{\omega_z^2}\right)}{as^3 + bs^2 + cs + 1}$$

with $a = \frac{\tau}{\omega}$, $b = \left(\frac{2\tau\xi\omega+1}{\omega^2}\right)$, $c = \frac{2\xi+\tau\omega}{\omega}$.

DIRECT 3 LOOP AUTOPILOT SYNTHESIS

A first approach is to determine algebraically the gains of the corrector.
Desired characteristics of the closed loop controlled aircraft

- time constant τ
- pulsation ω
- damping ratio ξ

$$k_r = -\frac{2\,c\,T_\alpha\,\omega_{af}\,\omega_z{}^2\,\xi_{af} - b\,T_\alpha\,\omega_z{}^2 - c\,\omega_{af}{}^2 - T_\alpha + a}{\left(c\,T_\alpha{}^2\,\omega_{af}{}^2\,\omega_z{}^2 - c\,\omega_{af}{}^2\right)\,K_3}$$

$$\omega_I = -\frac{2\,c\,\omega_{af}\,\omega_z{}^2\,\xi_{af} + \left(-c\,T_\alpha\,\omega_{af}{}^2 + a\,T_\alpha - b\right)\,\omega_z{}^2 - 1}{2\,c\,T_\alpha\,\omega_{af}\,\omega_z{}^2\,\xi_{af} - b\,T_\alpha\,\omega_z{}^2 - c\,\omega_{af}{}^2 - T_\alpha + a}$$

$$K_A = -\frac{\left(2\,c\,\omega_{af}\,\omega_z{}^2\,\xi_{af} + \left(-c\,T_\alpha\,\omega_{af}{}^2 - T_\alpha{}^2 + a\,T_\alpha - b\right)\,\omega_z{}^2\right)\,K_3}{\left(2\,c\,\omega_{af}\,\omega_z{}^2\,\xi_{af} + \left(-c\,T_\alpha\,\omega_{af}{}^2 + a\,T_\alpha - b\right)\,\omega_z{}^2 - 1\right)\,K_1}$$

$$K_{DC} = \frac{K_3 + K_A K_1}{K_A K_1}$$

$$\xi = 0.7$$

$$\omega = 7.54\ rad/s$$

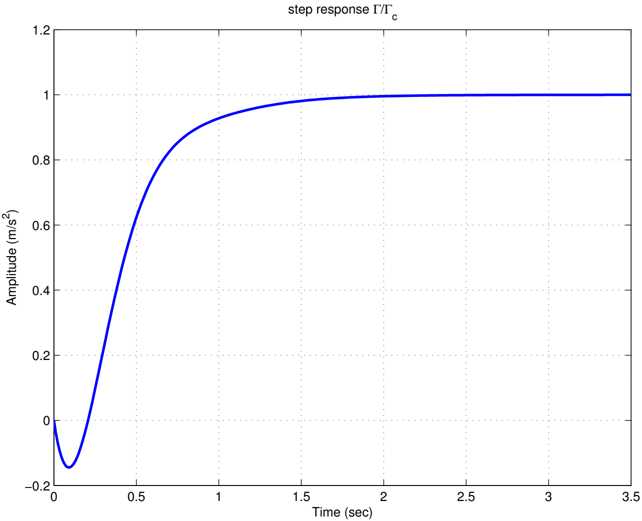
$$\tau = 0.35$$

$$k_r = -1.0505$$

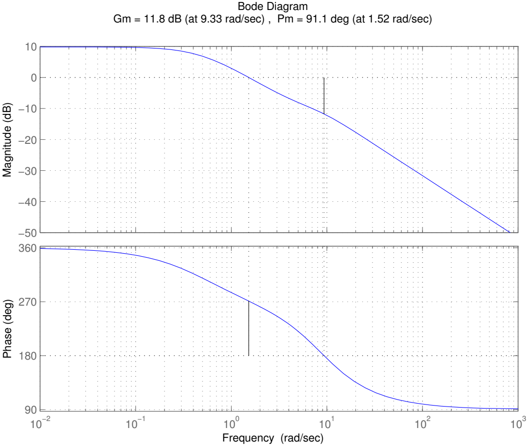
$$\omega_I = 4.4453$$

$$K_A = 0.0115$$

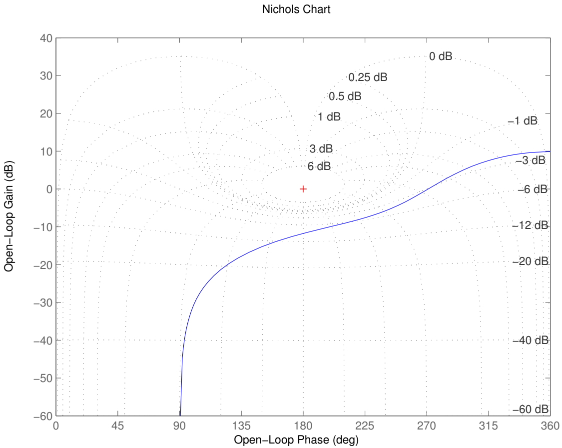
$$K_{DC} = 1.3217$$



Bode diagram and margin



Black-Nichols Diagram and robustness margin for the 3 loop autopilot



The choice of ξ, ω, τ depends on

- the flight point (Mach, altitude)
- researched compromise between performance, precision and robustness
  - if the response time is the priority, a small τ and a large ω are needed (and if needed, by modifying aircraft architecture, meaning aerodynamics, center of gravity position, actuators...);
  - If robustness is the priority, we choose τ and ω allowing minimum open loop gain and phase margin (at actuator input or on extern loop for example). Generally, a gain margin between 5 and 10 and a phase margin around 40° is chosen;
  - If precision is the priority, a large ξ is needed to prevent overshoot.
- Constraints are imposed en terms of
  - maximum transverse acceleration Γ_Z
  - maximum incidence α
  - maximum rotation speed q

  Concerning these last constraints, saturations are introduced at each loop.

Roles of gyrometric and accelerometric feedback

- The gyrometric feedback role is to increase a generally low damping ratio of the open loop aircraft transfer function.
  The gyrometric feedack associated with an integrator in the structure of the 3 loop autopilot allows the controller to cancel the static error (which is the difference between commanded value and achieved value after a transient phase)
- The accelerometric feedback main role is to adjust the response time of the closed loop controlled aircraft

INDEX

BIBLIOGRAPHY

M. Cougnon.
Cours de pilotage automatique
IPSA, 2010.

Paul. Zarchan.
Tactical and Strategic Missile Guidance.
AIAA Progress in Astronautics and Aeronautics Volume 176, 1997.

P. Garnell. D.J. East.
Guided Weapon Control Systems.
Pergamon Press 1977.

BIBLIOGRAPHY

- J-Ch. Gille. P. Decaulne. M.Pélegrin
  Théorie et calcul des asservissements linéaires.
  Dunod 1967.
- R. Carpentier.
  Guidage des avions et des missiles aérodynamiques
  SupAéro 1989.
- P.Borne — G.Dauphin-Tanguy — J-P.Richard — F.Rotella — I.Zambettakis
  Automatique Commande et optimisation des processus
  Technip 1998

BIBLIOGRAPHY

Prof. Jonathan P. How
MIT opencourseware: Aircraft Stability and Control
http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-333-aircraft-stability-and-control-fall-2004/

John H. Blakelock
Automatic Control of Aircraft and missiles
Wiley & sons 1965