CONTROL OF AIRCRAFT

COMPLEMENT : LQ CONTROLLER

October 16, 2020

Control aircraft

OPTIMAL CONTROLLER

The linear Quadratic Regulator (LQR) is a state feedback controller, $u = -Kx$ whose gain value is obtained by minimizing a criteria J.
Minimize

$$J = \frac{1}{2}\mathbf{x}^{\mathrm{T}}(t_f)\mathbf{S}_f\mathbf{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}\left(\mathbf{x}^{\mathrm{T}}(t)\mathbf{Q}(t)\mathbf{x}(t) + \mathbf{u}^{\mathrm{T}}(t)\mathbf{R}(t)\mathbf{u}(t)\right)\,\mathrm{d}\,t$$

$Q(t)$ positive semi-definite ($Q \geq 0$) is the state weight matrix and $R(t)$ positive definite ($R > 0$) is the command weight matrix.
Process to control

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Solve via maximum principle

$$\mathbf{H} = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + \lambda^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})$$

$$\dot{\mathbf{x}} = \left(\frac{\partial \mathbf{H}}{\partial \boldsymbol{\lambda}}\right) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad \mathbf{x}(0) = \mathbf{x}_0$$

$$-\dot{\boldsymbol{\lambda}} = \left(\frac{\partial \mathbf{H}}{\partial \mathbf{x}}\right) = \mathbf{Q}\mathbf{x} + \mathbf{A}^T\lambda \quad \lambda(T) = \mathbf{P}_1 x(T)$$

$$0 = \frac{\partial \mathbf{H}}{\partial \mathbf{u}} = \mathbf{R}\mathbf{u} + \lambda^T\mathbf{B} \implies \mathbf{u} = -\mathbf{R}^{-1}\mathbf{B}^T\boldsymbol{\lambda}$$

The solution is obtained by solving the two point boundary value problem (hard to solve in general) (X known for time $t = 0$ and λ known for final time $t = T$).

As an alternative, we could suppose the solution is under the form

$$\lambda(t) = \mathbf{P}(t)\mathbf{x}(t)$$

then we have

$$\dot{\lambda} = \dot{\mathbf{P}} + \mathbf{P}\dot{\mathbf{x}} = \dot{\mathbf{P}} + \mathbf{P}(\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{T}\mathbf{P})\mathbf{x}$$

$$-\dot{\mathbf{P}}\mathbf{x} - \mathbf{P}\mathbf{A}\mathbf{x} + \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{P}\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{A}^{\mathsf{T}}\mathbf{P}\mathbf{x}$$

Satisfied if we can found $\mathbf{P}(t)$ such that

$$\mathbf{0} = -\mathbf{P}\mathbf{A} - \mathbf{A}^{\mathsf{T}}\mathbf{P} + \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{P} - \mathbf{Q}$$

by taking $T = \infty$ and by eliminating terminal constraint, we have when A(t), B(t), Q(t) and R(t) are constants and when $t_f \to \infty$

$$J = \frac{1}{2} \int_0^{\infty} (\mathbf{x}^{\mathrm{T}}(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^{\mathrm{T}}(t)\mathbf{R}\mathbf{u}(t)) \, \mathrm{d}\, t$$

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

The optimal command is

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$$

With **K**

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^{T}\mathbf{P}$$

and **P** is the solution of the algebraic Riccati's equation

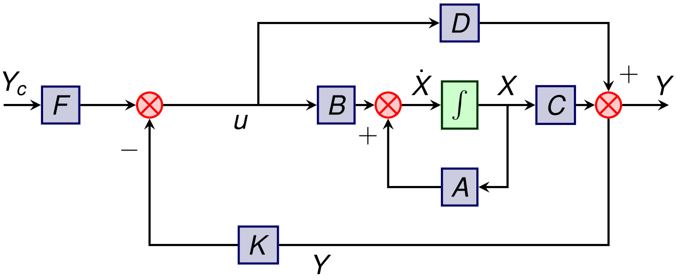
$$\mathbf{0} = -\mathbf{P}\mathbf{A} - \mathbf{A}^{T}\mathbf{P} + \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{T}\mathbf{P} - \mathbf{Q}$$

The controller **K** can be provided by the Matlab command

```
[G, X, L] = lqr (sys, Q, R)
```

LQR AND TRACKING CONTROLLER

The linear quadratic regulator is designed with null reference input. If we want this controller to follow a reference input Y_c, several approaches are possible. A first solution is to use a feedforward of the reference signal.



We want the closed loop to follow a refence signal $Y_c = CX_c$ for time $t \in [0, T]$.
The error signal is

$$e = Y_c - Y = C(X - X_c)$$

The criteria to minimize is

$$J = \min_u \left(\frac{1}{2} e(T)^T Q_f e(T) + \frac{1}{2} \int_0^T (e^T Q e + u^T R u) dt \right)$$

subject to

$$\dot{X} = AX + Bu \quad X(0) = X_0$$

The solution is

$$u = -R^{-1}B^T(PX + F)$$

with

$$\dot{P} = -PA - A^TP + PBR^{-1}B^TP - \tilde{Q} \qquad P(T) = \tilde{Q}_f \qquad \tilde{Q} = C^TQC \geq 0$$

$$\dot{F} = -(A - BR^{-1}B^TP)F + \tilde{Q}X_c \qquad F(T) = -\tilde{Q}_fX_c(T) \qquad \tilde{Q}_f = C^TQ_fC$$

This finite time-horizon optimal control problem can be solved by using the differential Riccati Equation (DRE).

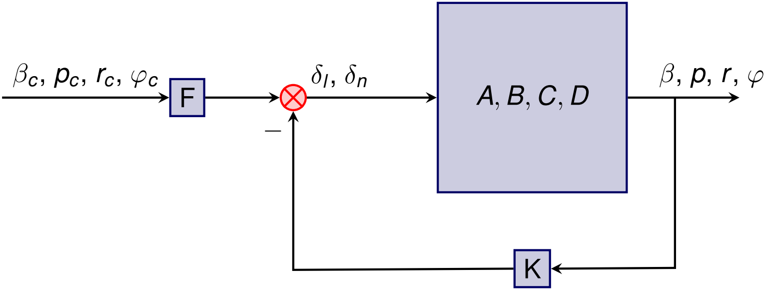
An approximation of the solution can be found for Linear Time Invariant systems, by taking the steady state solutions P and F.

$$K = R^{-1} B^T P_\infty$$

$$F = -R^{-1} B^T \left((A - BK)^T \right)^{-1} C^T Q$$

Another approach is to use a LQR with a augmented system which include the integral of the error. This controller will ensure null steady state error for a step input.

CONTROL AIRCRAFT

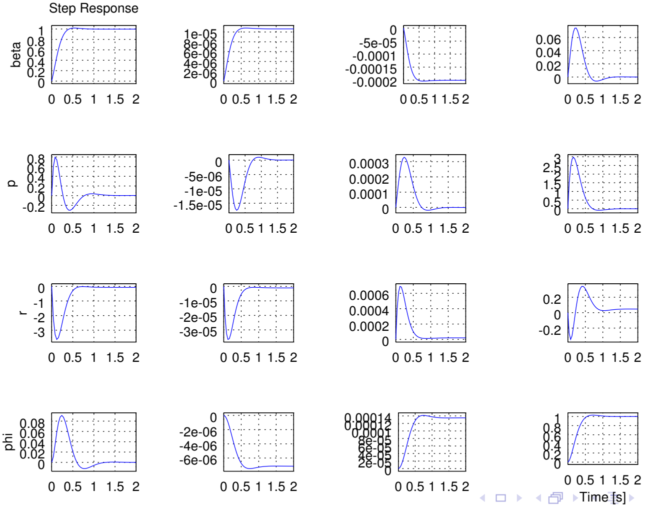

$\beta_c, p_c, r_c, \varphi_c$
F
δ_l, δ_n
$-$
A, B, C, D
β, p, r, φ
K

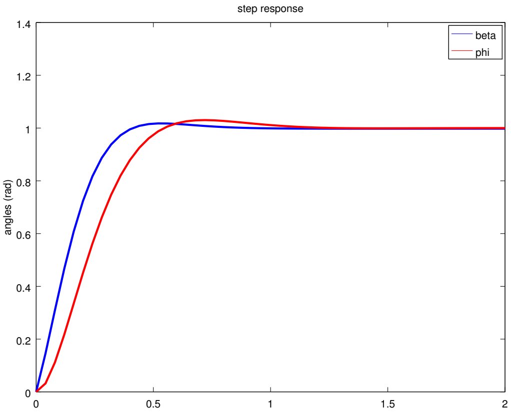
```
% X=[beta p r phi]
% u=[dl dn]
% Y=[ny p r phi]
a=[ -0.140  0.053 -0.999 0.047
 -2.461 -0.992 0.262 0
   1.585 -0.041 -0.267 0
   0 1.000 0.053 0];

b=[0 0.030
 0.404 0.260
    0. -0.680
  0     0];
c=eye(4);
d=zeros(4,2);
```

```
sys=ss(a,b,c,d,'statename',{'beta','p','r','phi'},...
                          'inputname',{'dl','dn'},...
                          'outputname',{'beta','p','r','phi'}
% optimization criterion
deg=pi/180;
Q=diag([10/(5*deg) 1/3 1/3 10/(5*deg)]);
R=0.01*diag([1,1]);
```

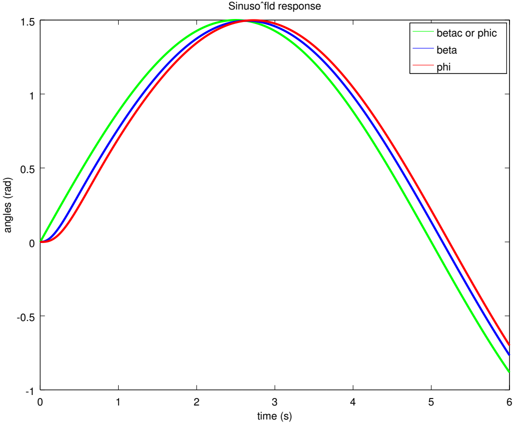
```
% LQR : constant gain matrix K1
[K1, P1, L1] = lqr (sys, c'*Q*c, R);

F=-inv(R)*b'*inv((a-b*K1)')*c'*Q;
syscl1=ss(a-b*K1,b*F,c,zeros(4,4),'statename',
                         {'beta','p','r','phi'},...
                         'inputname',
                         {'betac','pc','rc','phic'},...
                         'outputname',
                         {'beta','p','r','phi'});
>>K1
K1 =
   -18.7844    21.1014     6.3598   105.8427
   102.8559     1.4806   -13.9861    15.1695
```

```
figure(1)
step(syscl1);

figure(2)
H1beta=syscl1(1,1);
H1phi=syscl1(4,4);
[Y11,T]=step(syscl1(1,1));
[Y44,T]=step(syscl1(4,4));
plot(T,Y11,'b','linewidth',2,T,Y44,'r','linewidth',2)

ylabel('angles (rad)')
legend('beta','phi')
title('step response')
```

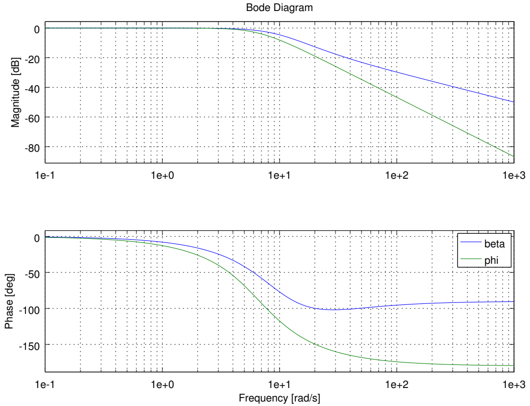

Step Response
beta
p
r
phi
Time [s]


step response
beta
phi
angles (rad)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2

```
figure(3)
T=0:0.01:6;
U=1.5*sin(0.2*pi*T);
y1beta=lsim(H1beta,U,T);
y1phi=lsim(H1phi,U,T);
plot(T,U,'g',T,y1beta,'b',T,y1phi,'r','linewidth',2)
xlabel('time (s)')
ylabel('angles (rad)')
legend('betac or phic','beta','phi')
title('Sinusod response')
```

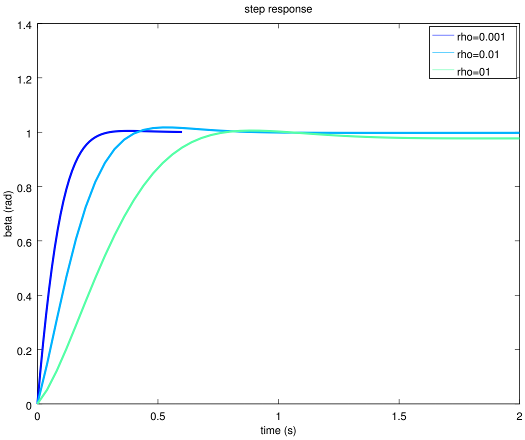

Sinusoïd response
betac or phic
beta
phi
angles (rad)
time (s)
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6

```
figure(4)
bode(H1beta,H1phi)
legend('beta','phi')
```


Bode Diagram
Magnitude [dB]
0
-20
-40
-60
-80
1e-1
1e+0
1e+1
1e+2
1e+3
Phase [deg]
0
-50
-100
-150
beta
phi
Frequency [rad/s]

```
% sensitivity to rho parameter in weight on command
figure(5)
clf
hold on
i=1;
col=colormap;
for rho=[0.001 0.01 0.1]
  R=rho*diag([1,1]);
  [K1, P1, L1] = lqr (sys, c'*Q*c, R);
  F=-inv(R)*b'*inv((a-b*K1)')*c'*Q;
  syscl1=ss(a-b*K1,b*F,c,zeros(4,4),
                                    'statename',
                                    {'beta','p','r','phi'},...
                                    'inputname',
                                    {'betac','pc','rc','phic'},...
                                    'outputname',
                                    {'beta','p','r','phi'});
```

```
    H1beta=syscl1(1,1);

    [Y11,T]=step(syscl1(1,1));

    h=plot(T,Y11,'linewidth',2)
    set(h,'color',col(i*10,:))
    xlabel('time (s)')
    ylabel('beta (rad)')

    title('step response')
    i=i+1;
end
legend('rho=0.001','rho=0.01','rho=01')
```


step response
rho=0.001
rho=0.01
rho=01
beta (rad)
time (s)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.5
1.5
2

PROS AND CONS OF LQR

Pros

- can handle MIMO (multiple input - multiple output) systems
- the weight matrix influence the energy consumed by actuator and the performance
- robustness

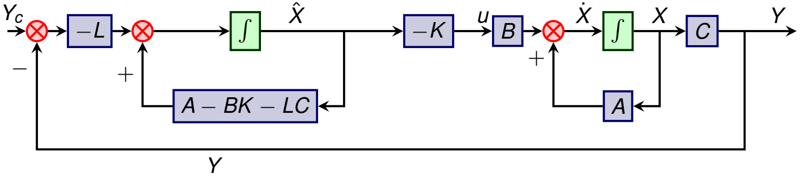
Cons

- for linear systems
- may loose robustness when used with an estimator (e.g. Kalman filter)

CONTROL AIRCRAFT

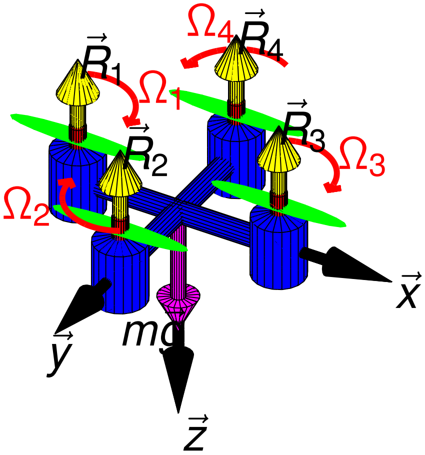
LQG CONTROLLER

The LQ regulator can use the state estimated by a Kalman filter. This is the LQG controller.
The example below is an output feedback controller. The controller gain and the estimator gain are computed separately.



CONTROL AIRCRAFT

QUADROTOR MODELLING



The 4 rotors create a force on z axis

$$\vec{F}_z = -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)\vec{z}$$

With Ω_i being the rotation speeds of each rotor and b the lift aerodynamic coefficient of each couple of blades.
Note that the rotation spins are clockwise and counterclockwise by pair of rotors in order compensate the torques 2 by 2 and to prevent the quadrotor to rotate around yaw axis while in hovering flight.

The pitch moment is created by differential thrust between rotors 1 and 3.

$$\vec{M}_y = \frac{\ell b(-\Omega_1^2 + \Omega_3^2)}{I_{yy}} \vec{y}$$

ℓ is the distance of each rotor to the center of gravity of the quadrotor in xy plane.
The roll moment is created by differential thrust between rotors 2 and 4.

$$\vec{M}_x = \frac{\ell b(-\Omega_2^2 + \Omega_4^2)}{I_{xx}} \vec{x}$$

The yaw moment is created by differential drag between all the rotors.

$$\vec{M}_z = \frac{d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)}{I_{zz}}\vec{z}$$

with d the torque constant.

The non linear equations of motion of the quadrotor are :

$$\dot{u} = rv - qw - g\sin\theta$$

$$\dot{v} = pw - ru + g\cos(\theta)\sin(\varphi)$$

$$\dot{w} = qu - pv + g\cos(\varphi)\cos(\theta) - \frac{b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)}{m}$$

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} qr + \frac{\ell b(-\Omega_2^2 + \Omega_4^2)}{I_{xx}}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{\ell b(-\Omega_1^2 + \Omega_3^2)}{I_{yy}}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)}{I_{zz}}$$

$$\dot{\varphi} = p + \sin\phi \tan\theta q + \cos\varphi \tan\theta r$$

$$\dot{\theta} = \cos\varphi q - \sin\varphi r$$

$$\dot{\psi} = \frac{\sin\varphi}{\cos\theta} q + \frac{\cos\varphi}{\cos\theta} r$$

$$\dot{x} = \cos\theta\cos\psi u + (-\cos\varphi\sin\psi + \sin\varphi\sin\theta\cos\psi)v$$
$$+(\sin\varphi\sin\psi + \cos\varphi\sin\theta\cos\psi)w$$

$$\dot{y} = \cos\theta\sin\psi u + (\cos\varphi\cos\psi + \sin\varphi\sin\theta\sin\psi)v$$
$$+(-\sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi)w$$

$$\dot{z} = -\sin\theta u + \sin\varphi\cos\theta v + \cos\varphi\cos\theta w$$

In order to obtain a linearized model, we need first to set the equilibrium point. At the equilibrium point, the quadrotor will be hovering at a constant altitude and will have no linear or rotational speed. Without loss of generality, the quadrotor is supposed to be at 10 m above the ground. ψ is fixed to a arbitrary value.

$$u = v = w = 0$$
$$p = q = r = 0$$
$$x = y = 0$$
$$z = -10\ m$$
$$\varphi = \theta = 0$$
$$\psi = \psi_{eq} = 0$$

In order to equilibrate the weight, all rotors must have the same rotation speed, which is

$$\Omega_H = \sqrt{\frac{mg}{4b}}$$

The linearized state space model uses the following state vector

$$X = \begin{pmatrix} u & v & w & p & q & r & \varphi & \theta & \psi & x & y & z \end{pmatrix}^T$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cos\psi_{eq} & -\sin\psi_{eq} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sin\psi_{eq} & \cos\psi_{eq} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

With the command vector $U = \begin{pmatrix}\Omega_1 & \Omega_2 & \Omega_3 & \Omega_4\end{pmatrix}^T$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{2b\Omega_H}{m} & -\frac{2b\Omega_H}{m} & -\frac{2b\Omega_H}{m} & -\frac{2b\Omega_H}{m} \\ 0 & -\frac{2\ell b\Omega_H}{I_{xx}} & 0 & \frac{2\ell b\Omega_H}{I_{xx}} \\ -\frac{2\ell b\Omega_H}{I_{yy}} & 0 & \frac{2\ell b\Omega_H}{I_{yy}} & 0 \\ -\frac{2d\Omega_H}{I_{zz}} & \frac{2d\Omega_H}{I_{zz}} & -\frac{2d\Omega_H}{I_{zz}} & \frac{2d\Omega_H}{I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

CONTROLLER SYNTHESIS

We will suppose that the autopilot has full access to all the states (the use of a perfect estimator filter is needed).
So

$$C = I_{12\times 12}$$

The command u has no direct influence on measurements.

$$D = 0_{12\times 4}$$

The controller will be generated via LQR method. The optimal controller will take as a constraint on the states the matrix Q, and as a constraint on the commands the matrix R.
For the choice of Q coefficients, an emphasis is made on yaw angle ψ and on altitude z.

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \end{pmatrix}$$

$$R = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix}$$

The corrector gain is obtained using the Matlab command lqr.

```
% Quadrotor mass (kg)
m=0.589;

% Body moment of inertia (kg.m^2)
Ixx=6.532e-3;
Iyy=6.6944e-3;
Izz=1.2742e-2;

% thrust (lift) factor (N.s^2)
b=4.3248e-5;
```

```
% torque constant
% drag factor (N.m.s^2)
d=5.96927e-8;
% distance between center of quadrotor
% and center of propeller (m)
l=0.2319;

% gravity acceleration (m/s^2)
g=9.81;

% propeller speed in hovering (rad/s)
OmegaH=sqrt(m*9.81/4/b)

psi0eq=0;
```

```
A  =[0  0  0  0  0  0  0  -g  0  0  0  0
     0  0  0  0  0  0  g   0  0  0  0  0
     0  0  0  0  0  0  0   0  0  0  0  0
     0  0  0  0  0  0  0   0  0  0  0  0
     0  0  0  0  0  0  0   0  0  0  0  0
     0  0  0  0  0  0  0   0  0  0  0  0
     0  0  0  1  0  0  0   0  0  0  0  0
     0  0  0  0  1  0  0   0  0  0  0  0
     0  0  0  0  0  1  0   0  0  0  0  0
     cos(psi0eq)  -sin(psi0eq)  0  0  0  0  0  0  0  0  0  0
     sin(psi0eq)   cos(psi0eq)  0  0  0  0  0  0  0  0  0  0
     0             0            1  0  0  0  0  0  0  0  0  0];
```

```
B =[0  0  0  0
    0  0  0  0
    -2*b*OmegaH*[1  1  1  1]/m
    [0  -1  0  1]*2*l*b*OmegaH/Ixx
    [-1  0  1  0]*2*l*b*OmegaH/Iyy
    [-1  1  -1  1]*2*d*OmegaH/Izz
    0  0  0  0
    0  0  0  0
    0  0  0  0
    0  0  0  0
    0  0  0  0
    0  0  0  0];
C =eye(12);
D =zeros(12,4);
% open loop state space model
sys=ss(A,B,C,D);
```

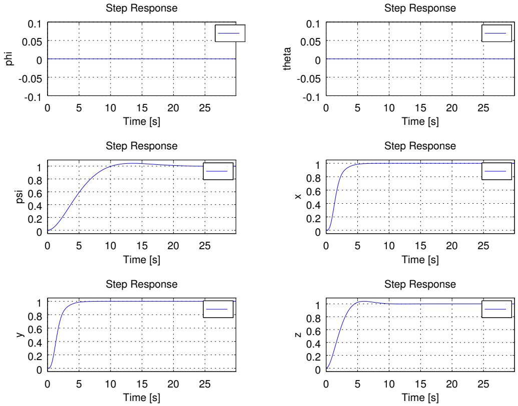
CONTROLLER SYNTHESIS

```
Q=diag([1 1 1 1 1 1 1 1 100 1 1 1 1]);
R=0.1*eye(4);
% gain calculus
[K, P, L] = lqr (sys, Q, R);
F=-inv(R)*B'*inv((A-B*K)')*C'*Q;
% closed loop system
syscl=ss(A-B*K,B*F,C,zeros(12,12));
```

We can see the step response with

```
figure(1)
subplot(321)
step(syscl(7,7))
subplot(322)
step(syscl(8,8))
subplot(323)
step(syscl(9,9))
subplot(324)
step(syscl(10,10))
subplot(325)
step(syscl(11,11))
subplot(326)
step(syscl(12,12))
```

CLOSED LOOP STEP RESPONSE (LINEAR MODEL)

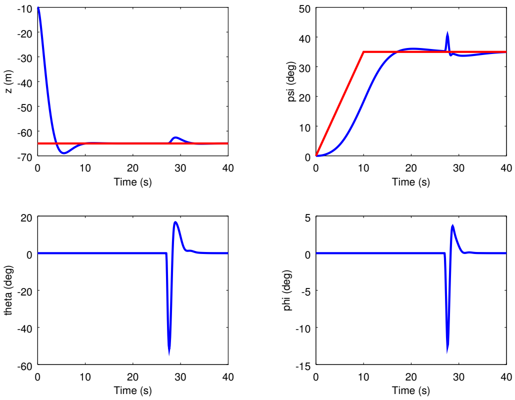


The behavior of the controller is checked with a non linear model of the quadrotor. The quadrotor command is first to climb to 65 *m* while turning slowly around yaw axis (35° in 10 *s*), and then at 27 *s*, to go to $x = 10$ *m* and $y = 5$ *m*, while keeping the same altitude.

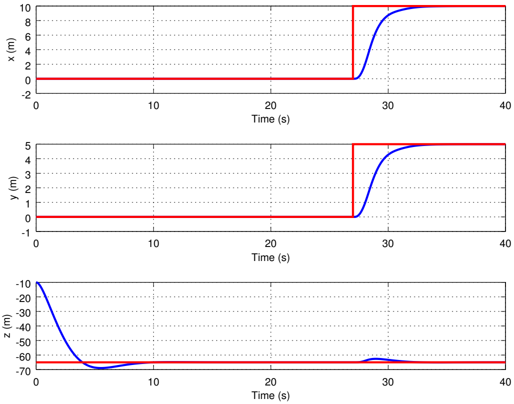
The controller certainly can be improved, as it is sensitive to cross coupling between the different axis. But it is provided here as an example of the use of a linear quadratic regulator applied to an aircraft with a behavior different from the one of an airplane.

Note that the useful commands here are x, y, z and ψ, as the quadrotor is maneuvering around hovering conditions (the other desired state vector components are set to 0). But we could imagine other quidance modes (constant horizontal speed cruise flight, constant flight path angle climb or descent trajectory for example).

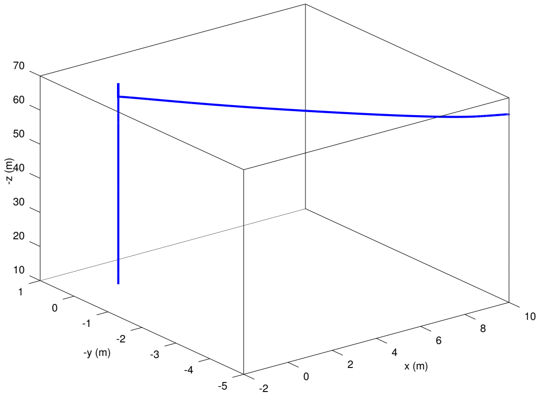
Closed loop response (non linear model)



Closed loop response (non linear model)



Closed loop response (non linear model)



Quadrotor LQR traker with integrators

The model is augmented with 4 states which are the integrals of ψ, x, y, z.
The linearized state space model uses the following state vector

$$X_{aug} = \begin{pmatrix} X^T & \int \psi \, dt & \int x \, dt & \int y \, dt & \int z \, dt \end{pmatrix}^T$$

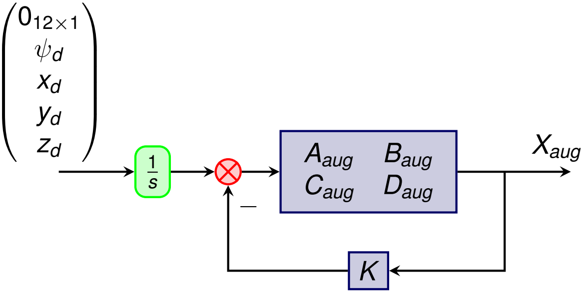
$$A_{aug} = \begin{pmatrix} \multicolumn{2}{c}{A} & 0_{12\times 4} \\ 0_{4\times 8} & I_{4\times 4} & 0_{4\times 4} \end{pmatrix}$$

With the command vector $U = \begin{pmatrix} \Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 \end{pmatrix}^T$

$$B_{aug} = \begin{pmatrix} B \\ 0_{4\times 4} \end{pmatrix}$$

$$C_{aug} = I_{16\times 16} \qquad D_{aug} = 0_{16\times 4}$$

The following architecture will allow to follow the desired inputs $\begin{pmatrix} 0_{12\times 1} & \psi_d & x_d & y_d & z_d \end{pmatrix}^T$



MATLAB CODE

```
% Xaug=[u v w p q r phi theta psi x y z
%         int_psi_err int_x_err int_y_err int_z_err]'
% U=[w1 w2 w3 w4]'
% Yaug=Xaug
Aaug=[A zeros(12,4);
      zeros(4,8) eye(4) zeros(4,4)];
Baug=[B;
      zeros(4,4)];
Caug=eye(16);
Daug=zeros(16,4);
sysaug=ss(Aaug,Baug,Caug,Daug)
% emphasis on integral of psi and integral of z
Qaug=diag([1 1 1 1 1 1 1 1 100 1 1 11 400 1 1 10]);
```

```
[K1, P1, L1] = lqr (sysaug, Qaug, R);
sysaugcl=ss(Aaug-Baug*K1,Baug*K1,Caug,zeros(16,16));
Aint=zeros(16,16);
Bint=zeros(16,16);
Bint(13:16,13:16)=eye(4,4);
Cint=Bint;
Dint=Aint;
ssint=ss(Aint,Bint,Cint,Dint);
syscl=series(ssint,syscl);
```

```
figure(1)
subplot(221)
step(syscl(9,13))
subplot(222)
step(syscl(10,14))
subplot(223)
step(syscl(11,15))
subplot(224)
step(syscl(12,16))
```

Python code

```
#mass (kg)
dxdt.m=0.589

# Body moment of inertia (kg.m^2)
dxdt.Ixx=6.532e-3
dxdt.Iyy=6.6944e-3
dxdt.Izz=1.2742e-2

# thrust (lift) factor (N.s^2)
dxdt.b=4.3248e-5

# torque constant
# drag factor (N.m.s^2)
dxdt.d=5.96927e-8
# distance between center of quadrotor and center of propeller (m)
# = LP
dxdt.l=0.2319
# another drag factor (N.m.s^2)
#d=1.1e-6

# acclration de la pesanteur (m/s^2)
dxdt.g=9.81


# propeller speed in hovering (rad/s)
#OmegaH=215
dxdt.OmegaH=sqrt(dxdt.m*9.81/4/dxdt.b)


psi0eq=pi/4*0
```

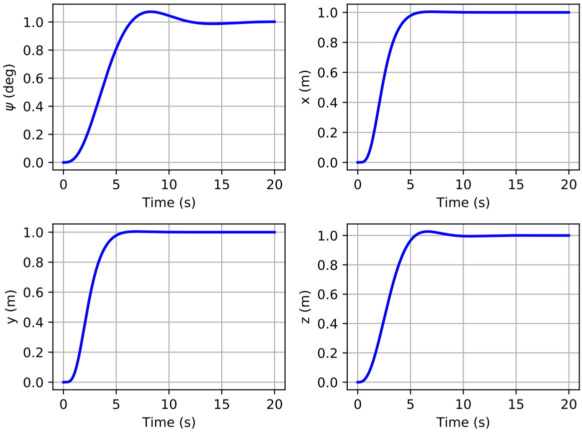
```
# augmented system with the integral of the errors
# int(psid-psi) int(xd-x) int(yd-y) int(zd-z)]  (between 0 and t)
#
#      0 1 2 3 4 5 6   7     8   9  10 11 12             13         14         15
# X = [u v w p q r phi theta psi x  y  z  int(psid-psi) int(xd-x) int(yd-y) int(zd-z)]
A1=np.matrix([\
   [0,0,0,0,0,0,0,-dxdt.g,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,dxdt.g, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,1,0,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,1,0,0, 0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,1,0, 0,0,0,0,0,0,0,0,0],
   [cos(psi0eq),-sin(psi0eq),0,0,0,0,0,0,0,0,0,0,0,0,0,0],
   [sin(psi0eq), cos(psi0eq),0,0,0,0,0,0,0,0,0,0,0,0,0,0],
   [0,          0,          1,0,0,0,0,0,0,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,1,0,0,0,0,0],
   [0,0,0,0,0,0,0, 0,0,0,0,1,0,0,0,0]])
```

```
B1=np.matrix([\
    [0,0,0,0],
    [0,0,0,0],
    -2*dxdt.b*dxdt.OmegaH/dxdt.m*ones((4)),
    2*dxdt.l*dxdt.b*dxdt.OmegaH/dxdt.Ixx*np.array([0,-1,0,1]),
    2*dxdt.l*dxdt.b*dxdt.OmegaH/dxdt.Iyy*np.array([-1,0,1,0]),
    2*dxdt.d*dxdt.OmegaH/dxdt.Izz*np.array([-1,1,-1,1]),
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0],
    [0,0,0,0]])
```

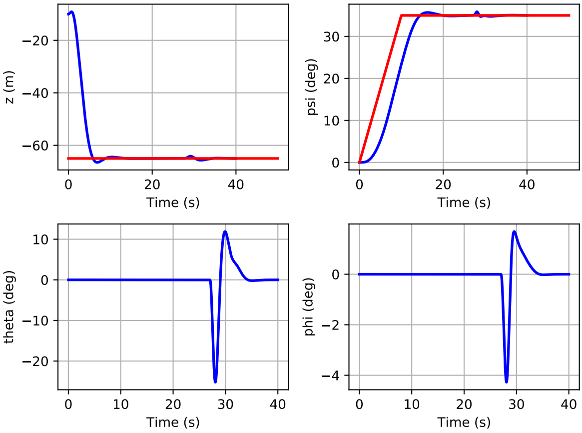
```
C1=np.eye(16)
D1=np.zeros((16,4))
Q1=np.diag([1,1,1,1,1,1,1,1,100,1,1,11,400,1,1,10])
sys1=ss(A1,B1,C1,D1)
K1, P1, L1 = lqr (sys1, C1.T*Q1*C1, R)
syscl=control.matlab.ss(A1-B1*K1,B1*K1,C1,np.zeros((16,16)))
Aint=zeros((16,16))
# calculate the integral of psi, x, y, z (between 0 and t)
Bint=zeros((16,16))
Bint[12:16,12:16]=eye(4,4)
Cint=Bint
Dint=Aint
ssint=ss(Aint,Bint,Cint,Dint)
syscl=series(ssint,syscl)
```

```
fig=figure(1)
fig.set_tight_layout(True)
fig.canvas.set_window_title('Linear step response')
subplot(221)
y,t=control.matlab.step(syscl[8,12],arange(0,20,0.01))
plot(t,y,'b',lw=2)
xlabel('Time (s)')
ylabel(r'$\psi$ (deg)')
grid(True)
subplot(222)
y,t=control.matlab.step(syscl[9,13],arange(0,20,0.01))
plot(t,y,'b',lw=2)
xlabel('Time (s)')
ylabel('x (m)')
grid(True)
subplot(223)
y,t=control.matlab.step(syscl[10,14],arange(0,20,0.01))
plot(t,y,'b',lw=2)
xlabel('Time (s)')
ylabel('y (m)')
grid(True)
subplot(224)
y,t=control.matlab.step(syscl[11,15],arange(0,20,0.01))
plot(t,y,'b',lw=2)
xlabel('Time (s)')
ylabel('z (m)')
grid(True)
```

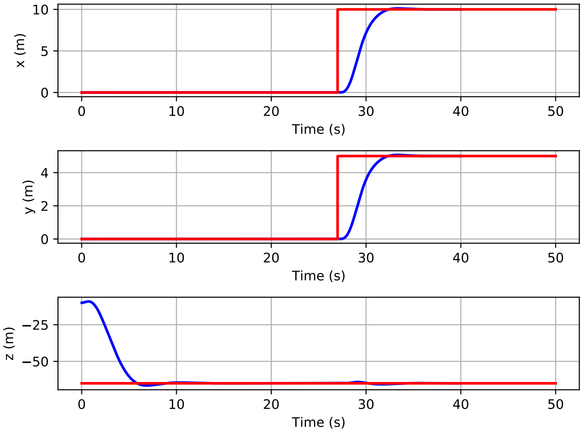
CLOSED LOOP STEP RESPONSE (LINEAR MODEL)



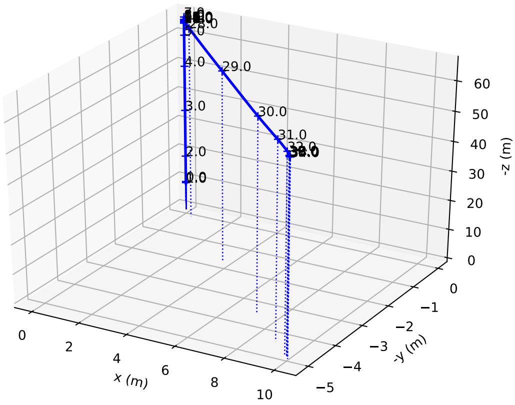
Closed loop response (non linear model)



Closed loop response (non linear model)



CLOSED LOOP RESPONSE (NON LINEAR MODEL)



Behavior of the quadrotor controller

- The controller behaves correctly around hovering conditions.
- We observe coupling during transient phases.
- Yaw (θ) and roll (φ) angle are commanded by the autopilot when an x or y displacement are commanded (which also depend on the current value of the yaw angle ψ).
- PID controller could have also be used. LQR controller advantage is to take into account directly coupling between the different axes.