Nonlinear Dynamic inversion applied to helicopter control
ISAE

Ph. MOUYON
ONERA/DCSD

We consider an helicopter motion in the vertical plan. Figure (1) shows the forces that are considered. Aerodynamic forces are neglected. Our objective is to control the helicopter position.

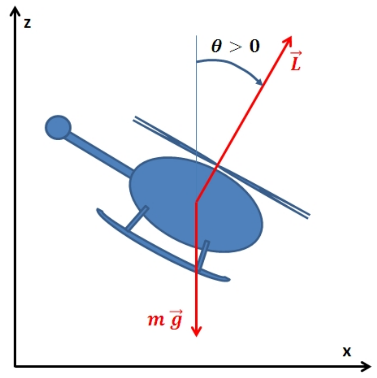

Figure 1: Helicopter

We will first consider the control of the accelerations γ_x and γ_z. A feedback linearization approach will be used for that purpose. Then loops will be added to control the velocities v_x and v_z, and finally the positions x and z.

1 Modeling

The dynamic model is made of three parts

Kinematic

$$\left\{\begin{array}{lcl} \dot{x} & = & v_x \\ \dot{z} & = & v_z \end{array}\right.$$

Forces equation

$$\left\{\begin{array}{lcl} m\,\dot{v}_x & = & L\,\sin\theta \\ m\,\dot{v}_z & = & L\,\cos\theta - m\,g \end{array}\right.$$

Torques equation

$$\left\{\begin{array}{lcl} \dot{\theta} & = & q \\ J\dot{q} & = & -f\,q + T \end{array}\right.$$

Numerical values are as follows:

m	mass	10	$[kg]$
J	inertia	0.2	$[kg\,m^2]$
f	damping	0.1	$[N\,m\,(rad/s)^{-1}]$
g	gravity	9.81	$[m\,s^{-2}]$

Inputs are the thrust (or lift force when hovering) L and the torque T. The lift is limited to $L_{max} = 200\,[N]$. The torque satisfies $|T| \leq T_{max}$ with $T_{max} = 0.035[N\,m]$. Both actuators may be considered as first order systems with transfer function $1/(1 + \tau s)$, where $\tau = 0.1$.

Simulation tool: Two files are provided

1. A script file called *simulation_script.m*. All the simulation parameters are defined in this file. The file also run the simulation and plots chosen signals.
2. A simulink model called *simulation_model_1.slx*, with three blocks: a reference generator, an *actuators* block that models the dynamic of the actuators, and a block *helicopter* that contains the helicopter dynamics.
   The configuration parameters of have been set to: *stop time* = T_{sim}, *solver option type* = $Fixed - step$, and *Fixed–step size* = DT_{sim}.
   - The inputs of the *helicopter* block are the thrust L and the torque T. Outputs are the positon, velocity and acceleration $(x, v_x, \gamma_x, z, v_z, \gamma_z)$ and the pitch angle and the rotation rate (θ, q).
   - The *actuators* block simulates the actuators dynamic taking into account for position saturations.
   - The *references* block computes the demands. For this openloop model the demands d_1 and d_2 are applied on the inputs L and T. These signals are steps with a starting time, a stop time, and an amplitude.

Select and run the section of the script that correspond to the open-loop model in order to check the open loop behavior. It will plot the demands (L_d,T_d), the controls (L,T), and the accelerations (γ_x,γ_z).

2 Feedback linearization

The outputs to be controlled are the accelerations γ_x and γ_z. A feedback linearization approach will be used for that purpose. The architecture of this control law is depicted on figure 2.

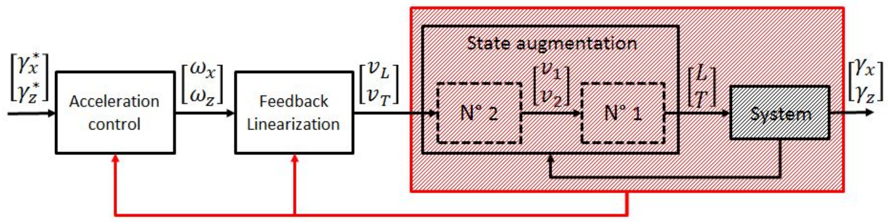
$\begin{bmatrix}\gamma_x^* \\ \gamma_z^*\end{bmatrix}$
Acceleration control
$\begin{bmatrix}\omega_x \\ \omega_z\end{bmatrix}$
Feedback Linearization
$\begin{bmatrix}v_L \\ v_T\end{bmatrix}$
State augmentation
N° 2
$\begin{bmatrix}v_1 \\ v_2\end{bmatrix}$
N° 1
$\begin{bmatrix}L \\ T\end{bmatrix}$
System
$\begin{bmatrix}\gamma_x \\ \gamma_z\end{bmatrix}$

Figure 2: Acceleration control architecture

Your job: is to define, implement and test this control law.

1. Remark that the characteristic orders of the accelerations are equal to zero (i.e. there exists a direct feedthrough), and that the decoupling matrix Δ is not inversible.

2. Show that thanks to (two) state augmentations, it is possible to get a non singular decoupling matrix Δ. The new inputs will be denoted (v_L, v_T).

3. Implement this first part of the control law in a block called *state augmentation.* It is better to do this in a copy *simulation_model_2.slx* of the initial model.

   - Be careful to initialize the integrator generating L to $m * g$ so that the simulation starts from the equilibrium.
   - Simulink may believe that there is an algebraic loop and generate an error. To avoid this, you can insert a "transport delay" block on each output of the *state aumentation* block.

4. Show that product $\Delta^{-1}\,\Delta_0$ has a fairly simple expression. Then give the expression of the control law that yields to an input/output behavior that is linear and decoupled, with all poles fixed at 0. The new inputs will be denoted (w_x, w_z).

5. Implement this second part of the control law in a block called *feedback linearization.*

6. Run the simulation with the control law. Find a simple mean to check that the feedback linearization behave as expected.

7. Then design a stabilizing state feedback to control γ_x and γ_z. and ensures their convergence towards references γ_x^* and γ_z^*. The expected response time is one second on both axis.

8. Implement this third part of the control law in a block called *acceleration control.* It is better to do this in a copy *simulation_model_3.slx* of the previous model.
   Complete the script *simulation_script.m* by the computation of the feedback gains to be used.

9. Run the simulation script. Be happy... or not!

3 Velocity and position control

Now it is possible for you to implement a speed and position control loop. First make a copy of the previous simulink model and call it *simulation_model_4.mdl.* Then do the job!

Some guidelines:

1. The required response time for the speed control is $3\,s$. Thus you can assume than the acceleration control loop is very fast (i.e. the acceleration is equal to its demand, and $\dot{v}_x \approx \gamma_x^*$). Then a simple P controller can be used to regulate the speed.

2. The required response time for the position control is $10\,s$. Thus you can assume than the speed control loop is very fast (i.e. the speed is equal to its demand, and $\dot{x} \approx v_x^*$). Then a simple PI controller can be used to regulate the position. The integral control is mandatory if wind is present (but there are no in our simulation).

3. Show than if the expected response times (for speed and position) are decreased too short then oscillations appear. Could you find any way to reduce them by maintaining a fast response?