Department of Electrical Engineering

EEE3094 Control Engineering

HELICOPTER CONTROL LABORATORIES 2 & 3, 2022

Client requirements

Given a helicopter model with commanded blade angle of attack as an input and height as an output:

Lab 2: Find the system model using both step response and frequency response methods. Compare the simulated and measured responses to arbitrary (i.e. non-training data sets). Lab report (individual or in pairs) to include model, measured data, system identification and Matlab/Simulink simulations. [5% of final coursemark]

Lab 3: (i) Design a suitable controller to control the altitude according to client requirements from Mr Dominic de Maar. The design should consider alternatives and should be systematic, moving from specifications to design to simulation to implementation to testing. (ii) Build, test and demonstrate your controller and quantify the performance against specifications. (iii) Implement a digital version of your controller using fast sampling and the bilinear transform. (iv) Report (individual or in pairs) on your design and its performance. [8% of final course mark]

Deadline dates & mark allocation

23 Aug	Lab 1 bookings 23 Aug-3 Sep
27 Sep	Lab 2 bookings 27 Sep-8 Oct
13 Sep	Lab 1 report 12h00
18 Oct	Lab 2 report 12h00

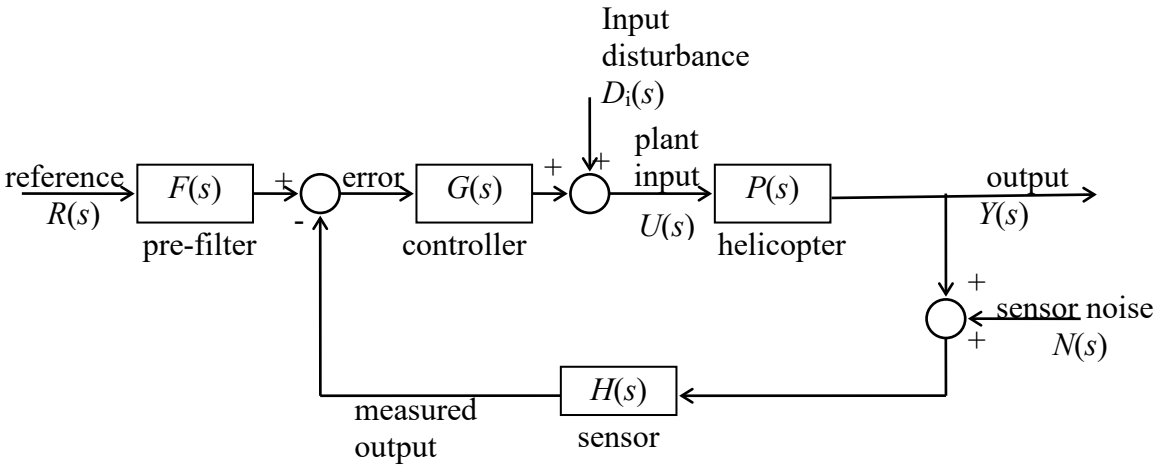


Figure 1 – Helicopter feedback system components

Additional material:

Lab 2: Please see system identification notes below. You should formally derive the equations of motion, but we will not calculate the forces – we hope that we can get the parameters by system identification!

As an example of the trouble we want to avoid, a blade produces lift relative to the angle of attack and velocity squared. The former depends on the blade pitch angle and the movement of the blade through the air (so decreases with increasing vertical speed). The rotor speed is affected by aerodynamic drag loading the motor. Many details are quite poorly understood even with very sophisticated 3D finite element analysis. Gareth D. Padfield, (2007), *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modelling 2nd Edition* Blackwell.

SYSTEM IDENTIFICATION NOTES

Method 1: Step Response

Apply a step of known amplitude and measure the response (and the input step amplitude). The input step should not be so big that it saturates the plant and not so small that you cannot detect the response properly.

Think of a motor car response to throttle input for designing a speed controller: You would like the response to go from 100 km/h to 120 km/h and would use a step in the throttle from say 60% to 70%. You should also make a step down because the response is likely to be different due to the system being nonlinear. If you try a 100% step or a 1% step, the response will not represent the realistic operating range for a speed controller. In practical systems you may use a bounded ramp instead of a step but this makes the analysis more complicated.

First order step response

Can be used if the step response resembles a first order response as shown in Figure 2.

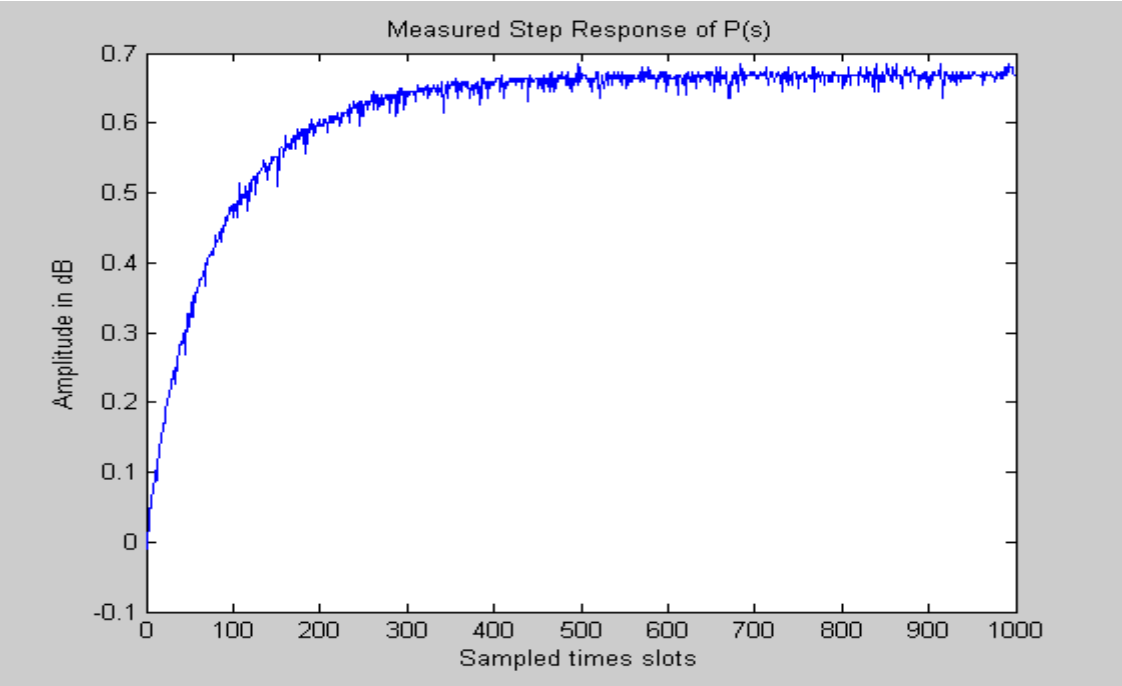


Figure 2: First order step response

The general first order equation is given by: $P(s) = \frac{k}{s\tau+1}$, where k is the steady state gain and τ is the time constant. From the unit step (i.e. $u(t) = \sigma(t)$) response, k and τ (time that output is at 63.2% of k) can be obtained by reading their values off the graph.

Second Order low pass system step response

This method finds a second order approximation of the system. This method can be used if the unit step response resembles a second order low pass response (See Figure 3). The general equation for a second order underdamped low pass system is:

$$P(s) = \frac{k\omega_n^2}{s^2 + 2\zeta\omega_{ns} + \omega_n^2}$$

The values from the step response (e.g. Figure 3) and the above equations can be used to obtain the parameters of $P(s)$.

Maximum overshoot: $M_p = \frac{y_p - y_\infty}{y_\infty} = e^{-\pi\zeta/\sqrt{1-\zeta^2}}$, $y_\infty = k$

Damped natural frequency $\omega_d = \omega_n\sqrt{1-\zeta^2}$

Time to peak $t_p = \frac{\pi}{\omega_d}$

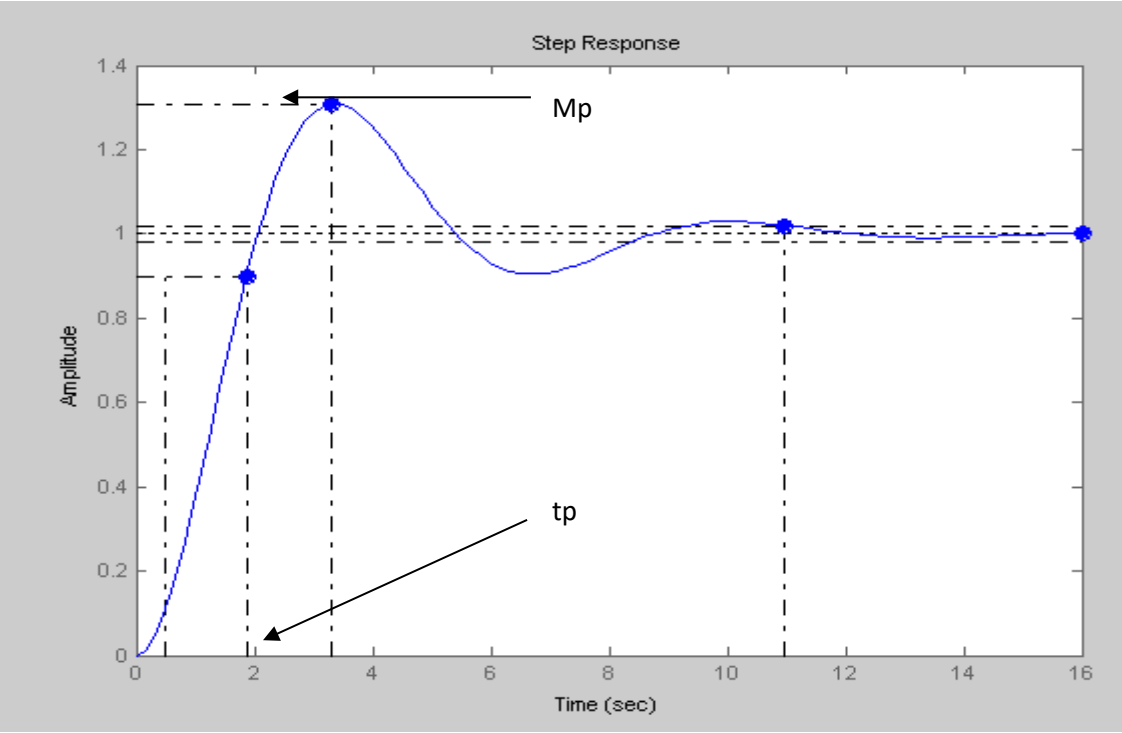
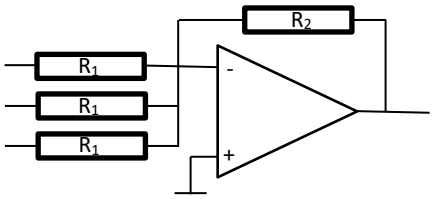


Figure 3: Second Order Step Response

Method 2: Closed loop sinusoidal testing

For stable systems, we can supply a sinusoid of known frequency and amplitude at the input and measure the steady state response amplitude and relative phase. After sweeping through a range of frequencies, we can draw a Bode plot and then derive the transfer function from the Bode plot. In the helicopter problem, because of the integrator between velocity and position, we need feedback to get the system to have a steady state position, and we therefore do the measurement in closed loop. To keep things simple, choose $G(s) = g$, a simple gain.

Before the laboratory: Build an op-amp circuit on a breadboard to act as a summing junction with gain. (We don't mind that the reference is inverted and have added a third input to provide offset.)



Adjust the offset to get the system to a sensible equilibrium position and then apply a sinusoidal reference signal. Measure the PLANT input and output to get the magnitude and phase of the response over about 0.1 to 10 rad/s. Plot the Bode plot and hence find a transfer function that matches this.

Method 3: System Identification Toolbox

You can import the input and output data from your step test directly into the Matlab System ID toolbox and obtain transfer functions. The risk is that as a professional, you must ensure that the results are correct, not just a handle crank. This method is optional in 2022 but worth trying.

Before the System ID laboratory, you should check that you can identify the parameters of two simple systems,

$$P(s) = \frac{2}{0.5s + 1}$$

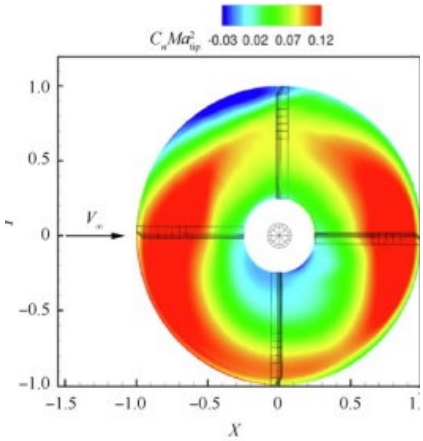
t_i	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4
u_i	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
y_i	0.00	0.00	-0.04	0.80	0.98	1.31	1.47	1.62	1.73	1.80	1.86	1.91	1.92	1.95	1.98

$$P(s) = \frac{4}{s^2 + 2s + 3}$$

Excited with a sinusoid $u(t) = \sin \omega t$

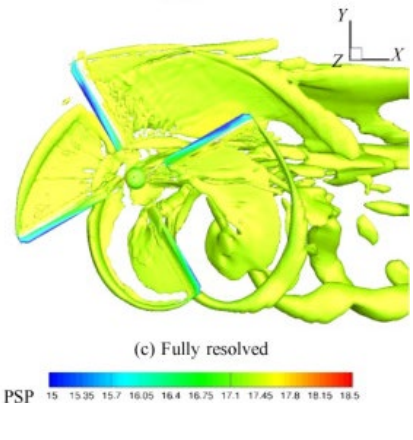
ω[rad/s]	0.1	0.2	0.4	0.8	1	2	4	8	10
$\|y\|$	1.33	1.34	1.35	1.36	1.41	0.97	0.26	0.06	0.04
$\arg y_i$ [°]	-4	-8	-12	-16	-45	-104	-148	-165	-168

Nice pictures to highlight complexity:



Rotor disk normal force coefficient in forward flight at advance ratio μ = 0.35, C_T = 0.016

G.N. Barakos, T. Fitzgibbon, A.N. Kusyumov, S.A. Kusyumov, S.A. Mikhailov, (2020) "CFD simulation of helicopter rotor flow based on unsteady actuator disk model", Chinese Journal of Aeronautics, 33-9, https://doi.org/10.1016/j.cja.2020.03.021.



Wake-visualisation of PSP rotor in forward flight at advance ratio μ = 0.35 and CT = 0.016

Massimo Biava, Walid Khier, Luigi Vigevano,(2012) "CFD prediction of air flow past a full helicopter configuration," *Aerospace Science and Technology*, 19-1, pp 3-18, https://doi.org/10.1016/j.ast.2011.08.007.

Notes for Lab 3: Control Design. Analogue and digital implementations

Please work individually or in pairs on the following. (Select one of the student numbers for all tests.)

1) Design a suitable controller to control the altitude according to client requirements from Mr Dominic de Maar. These will include steady state performance, damping (or overshoot) and reference-output and disturbance-output behaviour. Note that the actual disturbances due to wind gusts are force disturbances and are therefore input disturbances. The design should consider alternatives and input signal levels. It should move systematically from specifications → design → simulation → implementation → testing.
2) Build, test and demonstrate your controller using analogue components (op-amps, resistors, capacitors) and quantify the performance against specifications.
3) Implement a digital version of your controller using fast sampling and the bilinear transform (notes below)
4) Report (individual or in pairs) on your design and its performance.

Discrete time implementation with fast sampling

Later in the course, you will learn how to design discrete time controllers properly but if you sample fast enough, you can get a digital implementation of an analogue controller without too much error. To put "fast" into context, your sampling rate ($1/T$) should be at least 20 times the gain cross over frequency in Hz to avoid having to do a new design. (With proper digital controller design, you can do better than this.)

We will then implement the controller via numerical integration using the trapezoidal rule. To integrate a function, u(t), $y(t) = \int_0^t u(\tau)d\tau$, we calculate (recursively), $y_{i+1} \approx y_i + \frac{T}{2}(u_i + u_{i+1})$, where $\frac{T}{2}(u_i + u_{i+1})$ is the area under the trapezoid between $u(t_i) = u_i$ and $u(t_{i+1})$.

We will use the z-transform to write the following. (For now, the z represents a forward shift in time, more details later.)

$$y_{i+1} \approx y_i + \frac{T}{2}(u_i + u_{i+1}) \quad \leftrightarrows \quad zY(s) \approx Y(z) + \frac{T}{2}\big(zU(z) + U(z)\big)$$

Time domain	**Transform domain**	**Transform domain**
$y(t) = \int_0^t u(\tau)d\tau$	$Y(s) = \frac{1}{s}U(s)$	$s\,Y(s) = U(s)$
$y_{i+1} \approx y_i + \frac{T}{2}(u_i + u_{i+1})$	$Y(z) = \frac{T}{2}\frac{z+1}{z-1}\,U(z)$	$\frac{2}{T}\frac{z-1}{z+1}Y(z) = U(z)$

If we observe that the differentiating operator in the s-domain maps into an equivalent in the z-domain, we discretize the controller by replacing $G(s)$ with $G\left(\frac{2}{T}\frac{z-1}{z+1}\right)$ and go back to the difference equation to get the algorithm.

All this is a bit early in the course but please follow the example below!

Example

If we have a plant, $P(s) = \frac{.1}{s(s+1)}$ and a continuous time lead controller,

$$G(s) = \frac{5(s/1.5 + 1)}{s/5 + 1}$$

The open loop gain cross-over frequency is 3.02 rad/s = 0.064 Hz. Sampling at $T = 0.1$ s meets the requirement for 20x. We then approximate the controller as

$$G(s) = \frac{5\left(\frac{2}{T}\frac{z-1}{z+1}/1.5 + 1\right)}{\left(\frac{2}{T}\frac{z-1}{z+1}/5 + 1\right)}$$
$$= \frac{14.33z - 12.33}{z - 0.600} = \frac{U(z)}{E(z)}$$

$$(z - 0.600)U(z) = (14.33z - 12.33)E(z)$$

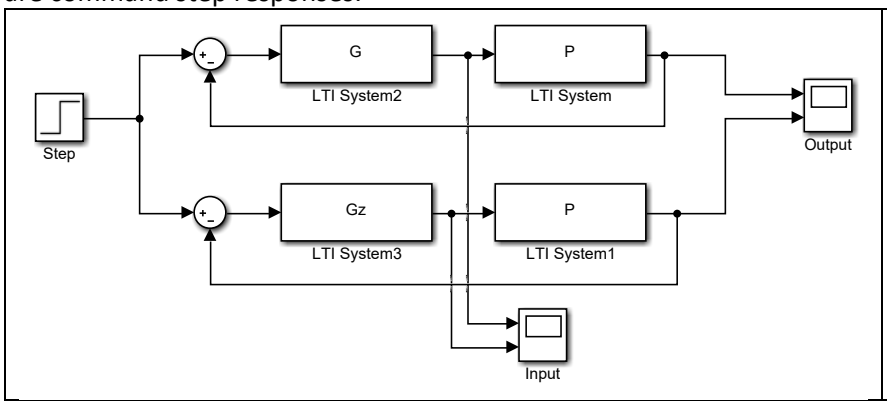
Giving an algorithm

$$u_{i+1} = 0.600u_i + 14.33\, e_{i+1} - 12.33e_i$$

Pseudo-code:

```
...
reference=(float) ADResult(channel1) * ScaleFactorR;
measured=(float) ADResult(channel2) * ScaleFactorY;
error_now=Reference-measured;
input=0.600*Input+14.33*error_now-12.33*error_last;      // input is the controller output (plant input)
error_last = error_now;                                  //save result for next sample
```

Because of the sampling, the continuous and digital controllers do not have identical performance. Here are command step responses:




```
P=1/s/(s+1);
G=5*(s/1.5+1)/(s/5+1);
Gz=c2d(G,0.1,'tustin')
```