ENGINEERING TRIPOS PART IIA

ELECTRICAL AND INFORMATION ENGINEERING
TEACHING LABORATORY

MODULE EXPERIMENT 3F1

FLIGHT CONTROL

Objectives:

- Simulation of various aircraft models on the computer.
- Study real-time (manual) control and the limitations imposed by time delays.
- Design of a simple autopilot.
- Illustrate frequency response concepts in analogue and digital control systems, conditions for oscillation in feedback systems and stability.
- Gain familiarity with MATLAB.

1 Introduction

The setting for the experiment is shown in Figure 1. The **plant** is an aeroplane, simulated by the computer. It to be controlled about its pitch axis and the measured output $y(t)$ is the pitch angle.

The **controller** processes the error signal $e(t)$ and outputs the joystick position $u(t)$. The effect of disturbances is represented at the plant input by $d(t)$. **Manual control** is achieved by means of a 'head-up display', shown in MATLAB's first graphics window. The output, $y(t)$, is the distance of the horizon line from the middle of the scale. If $|y(t)|$ ever exceeds 10, the aeroplane is deemed to have crashed and the simulation stops. The cursor on the right is the joystick position, again limited to the range ± 10; this is controlled by the mouse, which should be positioned in the graphics window over the joystick cursor.

A plot of $e(t)$ and $u(t)$ is displayed automatically in the second MATLAB graphics window after each simulation run. The third graphics window is used for Bode plots.

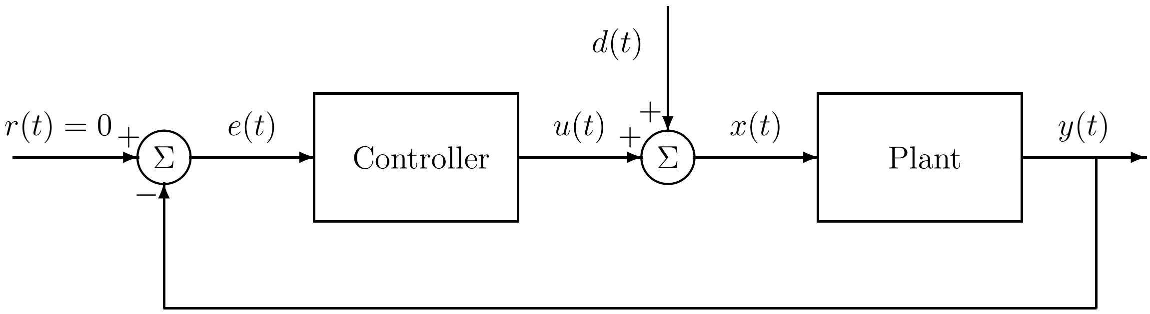

Figure 1: Block diagram

1.1 Background control theory

The theory for this experiment involves a mixture of discrete-time and continuous-time concepts. This is quite typical in digital control applications. Indeed, in the actual experiment, the plant is discrete-time while the manual controller acts in continuous-time. The aircraft models themselves are all specified initially in continuous-time and can be represented there by a Laplace transfer function $G(s)$. For simulation, they are discretised at a specified sampling rate, after which they can be represented by a z-transfer function $\hat{G}(z)$. Similarly, the controller can be represented in continuous or discrete time.

The objectives of the controller in Figure 1 are:

P1. To stabilise the system.

P2. To reduce the effect of disturbances $d(t)$ on the error $e(t)$ especially at low frequencies.

P3. To achieve a fast speed of response, e.g. to bring $e(t)$ back to zero quickly in response to an impulse at $d(t)$.

P4. To ensure that the above features are insensitive to moderate changes in the system.

Properties P2–4 tend to be improved if we can make the gain of the controller large. The natural question is: 'What limits our ability to increase the gain and hence improve performance?' There are two main limitations:

L1. Amplitude constraints on the actuator signal $u(t)$. For our flight control experiment there is a maximum deflection for the elevators. Other examples are: maximum power cannot be exceeded in a motor drive; a flow control valve can only vary between 'off' and 'fully open'.

L2. The closed loop system may become unstable (or poorly damped) if the controller gain is too large.

The question of closed-loop stability (and the preservation of adequate stability margins) is of great concern in control design. One of the principal tools for such analysis is the *Nyquist Stability Criterion* which can be applied to a continuous or discrete time feedback loop. This can be summarised in the following procedure.

N1. Plot the Nyquist diagram of $K(s)G(s)$, i.e. the locus of $K(j\omega)G(j\omega)$ as ω varies from $-\infty$ through 0 to $+\infty$. (In discrete time, the Nyquist diagram of $\hat{K}(z)\hat{G}(z)$ is the locus of $\hat{K}(e^{j\theta})\hat{G}(e^{j\theta})$ as θ varies from $-\pi$ to $+\pi$.)

N2. Let N be the number of anti-clockwise encirclements of the -1 point by the Nyquist diagram.

N3. Apply the stability test: the closed loop is stable if and only if N equals the number of unstable poles of $K(s)G(s)$. (In discrete time, $\hat{K}(z)\hat{G}(z)$).

1.2 Getting started

To get started you will need to download the following files to a directory on your computer:

```
bodedisp.m  fmodels.m  grphc2.m
flysim.m    grphc1.m   startup.m
```

If you haven't already installed MATLAB the instructions on how to do so are given here: `https://help.eng.cam.ac.uk/software/matlab/`
You should then carry out the following steps.

1. Launch MATLAB and set the Current Folder to the one where you have placed the files.

2. In the Command Window you should see the MATLAB prompt `>>` where you should type "startup" to run the startup.m file.

3. You will be prompted for your userid to be used to label your graphs. Type in your crsid.

4. Now type "flysim" at the prompt. This opens Figure 1 which is a simple head-up display showing the horizon (broad red line) and joystick forward-back position (yellow diamond).

5. Place the cursor over the joystick position and hit carriage return. You will then be flying a plane with pitch dynamics described by the the plant $2/(s+1)$. Moving your cursor up and down moves the joystick.

6. You can use the up-arrow at the prompt to find the previous line(s) and type carriage return to apply the command again. After each run you will find printed the minimum, maximum and average sampling period. The latter can be checked that it is not too much above the target of 30 milliseconds (indicating slow response or interrupts).

1.3 Plotting and printing

Labelling and printing

Before saving/printing any graphs you should label them using the `title` command (type '`help title`' for instructions) and annotate them as necessary. It is recommended that you save as a pdf, or print to a file, for inclusion in your lab report. (A screendump or, if you have access to a printer and wish to physically print out your graphs and annotate, is also acceptable if the plots are clear enough.) This applies to any subsequent places in the guide where you are requested to do print-outs.

Plotting Bode diagrams

You will need to use the command `bodedisp` for plotting Bode diagrams. First, the variables `Kgain` and `Dtime` need to be set at the command prompt. Then, typing the command `bodedisp` plots the discrete-time Bode diagram of the current plant in series with a proportional gain `Kgain` and a time delay of `Dtime` seconds. The plant is specified in continuous time by a Laplace transfer function with numerator and denominator `num` and `den`. The sampling period is taken to be `srate`. You can alter the minimum and maximum frequencies and the number of frequency points by typing `bodedisp(fmin,fmax,fnum)` for your choice of `fmin`, `fmax` and `fnum`. The default `bodedisp` is the same as `bodedisp(0.1,pi/srate,100)`. Note that the maximum frequency should not exceed `pi/srate`.

2 Manual control

You begin with a simplified aircraft model:

$$\ddot{y}(t) + M\dot{y}(t) = Nx(t)$$

where M is a coefficient of aerodynamic damping and N is a coefficient of aerodynamic effectiveness of the elevators. What is the Laplace transfer function of this plane? Set $M = N = 10$ and try flying this plane. To do this it is recommended that you copy `flysim.m` to a new file `flysim1.m` (say) which you will then edit. You will need to modify `num` and `den` in the new file. (Enter these and other data requested into the worksheet as you proceed. Questions underlined in the text are for the report—see Section 4.)

Now change the runtime to 5 seconds and for a disturbance impulse of weight `5` bring the plane back to level flight—you will need to edit the parameters `runtime` and `wght` in the `flysim1.m` file. Place the mouse cursor on the joystick indicator before the run. After a few trials *print out* a typical response. A simple model of you, the controller, is a proportional gain k in series with a pure time delay D. What is the Laplace transfer function of this controller? From your plot estimate your own values of k and D (annotate the plot as necessary).

Plot the Bode diagram for the open loop including the controller—see section 1.3 for instructions—and *print out*. Estimate the phase margin (you may need to adjust the frequency scale to get a good estimate). Calculate how much extra time delay this control loop would tolerate before going unstable. <u>Use your Bode diagram to make a sketch of the Nyquist diagram in your report</u>.

Now take a *step* disturbance of magnitude 5, a run time of 10 seconds, and try to keep the plane level. *Print out* the response. Are you using any integral action? Give a brief explanation.

2.1 Pilot induced oscillation

Next you are asked to fly a different aircraft whose control system is designed by a fresh graduate—not from Cambridge! It can be modelled by the transfer function

$$G_1(s) = \frac{c}{(Ts+1)^3}$$

Choose the parameters as follows: $T = 4D/\pi$ and $c = \sqrt{8}/k$ for your own values of k and D. Take an impulse of weight $5kD$ and attempt to bring the plane back to level flight. In `flysim1.m` this can be achieved with the commands:

```
T=4*Dtime/pi;
num=sqrt(8)/Kgain;
den=[T^3,3*T^2,3*T,1];
wght=[5*Kgain*Dtime,0,0,0];
```

What you should have observed is a phenomenon traditionally known as 'pilot-induced oscillation' or 'PIO'—though, to be fair, it is not all the pilot's fault. *Print out* a suitable oscillatory response. For comparison, *print out* the response with a zero input. *Print out* the Bode diagram of the open loop, over a suitable frequency interval, and use it to explain the oscillation of the feedback loop. How does your observed period of oscillation compare to the theoretical prediction? Can you give a rough guideline to the control designer to make PIO less likely?

2.2 Sinusoidal disturbances

Now consider some models for the F4E fighter aircraft, the data for which is stored in the file `fmodels.m`. These models represent the open loop dynamics, linearised about 4 different operating points (see comments in the file), with the control input being a fixed linear combination of the elevator and canard rudder deflections. A control system needs to be designed to make the aircraft flyable (we won't do this here). Indeed, the first three operating points have unstable dynamics, only the third of which can be manually controlled to any extent. The fourth operating point is stable with a lightly damped mode.

The data can be loaded by typing `fmodels` at the matlab prompt. The transfer function data is stored in the variables `num1`, `den1`, `num2`, etc. To fly the fourth operating point, for example, set:

```
num=num4;den=den4;
```

in your `flysim1.m` file. (The first three operating points are not used in the experiment but you are welcome to try flying them, if you have plenty of time).

Consider the 4th operating point and try the following experiment. Put in a sinusoidal disturbance of frequency 0.66 Hz and magnitude 1 and try to keep the aircraft as close to level flight as possible over a 10 second run. Repeat the experiment with no control input. *Print out* the response in each case. Are you able to reduce the error?

Print out a Bode diagram of the plant with `Kgain = 1` and your own estimated time delay, over an appropriate frequency interval, and use it to find: (1) the maximum proportional gain for which the closed loop system is stable, and (2) the gain and phase at 0.66 Hz. For a plant transfer function of G and a controller transfer function of K find the open and closed loop transfer functions from d to y. (You will need to make use of the above for the theoretical analysis in the FTR. Some hints: You should have found that the maximum stabilising proportional gain is less than one, i.e. negative in dBs. Did you notice that you were more delicate in moving the joystick than before?)

2.3 An unstable aircraft

Now consider the control of an unstable aircraft. Take a simplified model:

$$G_2(s) = \frac{2}{-1 + sT}$$

which has an unstable pole at $1/T$. Take a runtime of 5 secs and `wght=[0.1,0,0,0];` Starting with $T = 1$ gradually decrease T and, each time, try to stabilise the plane for the full 5 seconds. What is the fastest pole you can stabilise? *Print out* a copy of your response for this case which should be quite oscillatory.

Sketch the Nyquist diagram (continuous time) of G_2 (hint: it is exactly circular). Explain using the Nyquist criterion why the feedback system is stable with a proportional gain greater than 0.5. How is the Nyquist diagram modified if a small time delay D is introduced into the system?

2.4 Broom balancing*

(full technical report only)

The problem of stabilizing an unstable aircraft is similar in many respects to the problem of balancing a broom.

Consider the question: "What is the shortest upside down broom I can balance on my hand?" The linearised equations of motion (assuming negligible handle weight, length L, horizontal position of your hand x, angle θ to vertical, and considering one dimension) are

$$\ddot{x} + L\ddot{\theta} = g\theta$$

Suppose we measure the horizontal position of the top of the broom, $y = x + L\theta$ and then use the feedback signal, $z = y + T\dot{y}$ where $T^2 = L/g$. Calculate the transfer function from x to z under this arrangement.

This assumes the particular proportional-derivative action controller given, but this is a reasonable choice. Do you notice a similarity with the dynamics of the unstable aircraft?

Hence provide estimates of the minimum value of L based on your above results. Compare this with reality, and comment.

3 Autopilot

Now consider the following model for a transport aircraft on approach to landing:

$$G_3(s) = \frac{6.3s^2 + 4.3s + 0.28}{s^5 + 11.2s^4 + 19.6s^3 + 16.2s^2 + 0.91s + 0.27}.$$

Fly this sluggish plane a bit. (With an impulse of weight 10, a runtime of 60 seconds and no control input from the joystick you can observe a very slow, lightly damped oscillation. This is called the *phugoid mode* in aircraft dynamics.)

Your next task is to design an autopilot. Start by implementing a simple proportional negative feedback controller with gain 5. To do this, first remove the small dead-zone in the joystick by changing 0.05 to 0.0 in the following line of the 'for' loop in the `flysim1.m` file:

```
pp=sign(pp)*min(max(0,abs(pp)-0.05),10);
```

Now comment out the line

```
pp=p(1,2);
```

by inserting a `%`, and add at the same place the line

```
pp=-5*y;
```

Test this controller with a disturbance impulse of weight 2 and a runtime of 15 seconds. Notice that the phugoid mode has been stabilised. Now increase the proportional gain until the closed loop system just oscillates. *Print out* the response. What is the value of gain K_c at which this occurs and what is the period of oscillation T_c? What is the gain margin of the loop when a proportional controller with gain 5 is used? (Hint: you do *not* need any Bode diagrams to do this!)

3.1 A PID controller

The autopilot to be implemented is a PID (proportional-integral-derivative) controller of the form

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de}{dt} \right). \tag{3.1}$$

What is the transfer function of this controller? The constants should be selected as follows:

$$K_p = 0.6K_c, \quad T_i = 0.5T_c, \quad T_d = 0.125T_c.$$

The above settings are known as Ziegler-Nichols rules in process control. You will need to approximate the integral and derivative terms in order to implement them digitally. To do this, set the following variables before the start of the 'for' loop:

```
integ=0;deriv=0;yprev=0;
```

as well as `Kp`, `Ti` and `Td`. Now replace the line `pp=-5*y;` by the following lines:

```
integ=???;
deriv=(y-yprev)/srate;
pp=-Kp*(y+integ/Ti+deriv*Td);
yprev=y;
```

where the `???` in the first equation should be a simple update law for `integ` (the estimate of $\int_0^t y(\tau)d\tau$).

Test the closed loop system with a disturbance `wght=[2,2,0,0];` and a runtime of 10 seconds. Observe the effect of the integral action. Now increase the derivative gain by 40% and observe the reduced oscillation. For these gain settings, *print out* the response.

3.2 Integrator wind-up

Now change the disturbance to

```
wght=[20,2,0,0];
```

You should now observe a phenomena called integrator wind-up. This can cause unwanted overshoot/undershoot after a period of input saturation due to the integrator being 'wound up' and having to unload again. One simple corrective remedy is to prevent the integrator state from ever getting too large. Do this by adding the line:

```
integ=sign(integ)*min(abs(integ),???);
```

after the update equation for `integ`. Replace `???` with the smallest number which guarantees a zero steady-state error in response to a step disturbance of weight 2 (<u>and explain your reasoning with reference to</u> (3.1)). *Print out* the response both with and without your modification. *Print out* your final `flysim1.m` file.

4 Lab report

You are *not* required to write a full report (with introduction, aims and objectives etc). *All* that is required is:

1. The requested plots and print-outs.
2. A completed worksheet.
3. Answers to items underlined in the text.

Handwritten answers for 3. are acceptable, which may be written on the graphs (or on extra sheets if required). *Your lab report should be a single pdf file comprising: cover sheet, plots, answers to underlined items, and worksheet.*

5 Full Technical Report

Guidance on the preparation of Full Technical Reports is provided both in Appendix I of the General Instructions document and in the CUED booklet A Guide to Report Writing, with which you were issued in the first year. If you are offering a Full Technical Report on this experiment, you should include a discussion section with seven subsections addressing the following points. Include your Laboratory Report as an appendix and refer to it where appropriate.

- Section 2.2: Use the results of your calculations to indicate $K(j\omega_1)G(j\omega_1)$, where $\omega_1 = 0.66 \times 2\pi$ rad/s, on an Argand diagram for suitable stabilising proportional gains and hence estimate $|(1 + K(j\omega_1)G(j\omega_1))|$. Does the theory predict that your feedback will help attenuate the sinusoidal disturbance for stabilizing gains?
- Section 2.3: It turns out that if $D > T$ then no proportional gain exists to stabilise the system. Verify this claim analytically in your report.

- Carry out the exercises in section 2.4.

- Section 3.1: There are two principal sets of Ziegler-Nichols rules for tuning PID controllers. Give a brief description of the two sets of rules. Briefly discuss the advantages and disadvantages of these tuning methods. Give references for any books consulted.

- Section 3.1: Which method was used to approximate the derivative of the PID controller? Calculate the transfer function of the discretised controller (z-domain).

- Consider the discretisation of blocks with time-delays. Consider the continuous-time plant $G(s) = e^{-D_1 s}/s$ placed in the arrangement illustrated in Figure 2. Find the discrete-time transfer function from $u(k)$ to $y(k)$. The DAC is a simple zero order hold, and you may assume that the ADC and DAC operate synchronously with sampling period T (Hint: You may find it useful to write $D_1 = nT + D_0$ where n is an integer and $0 \leq D_0 < T$).

  The routine `bodedisp.m` obtains a time-delayed discrete-time frequency response by first calculating the discrete-time transfer function of the loop excluding the time-delay, then approximating the time-delay by an integer product of the sampling time. Discuss the differences between these two schemes, considering accuracy under different time-delays and sampling times and ease of calculation.

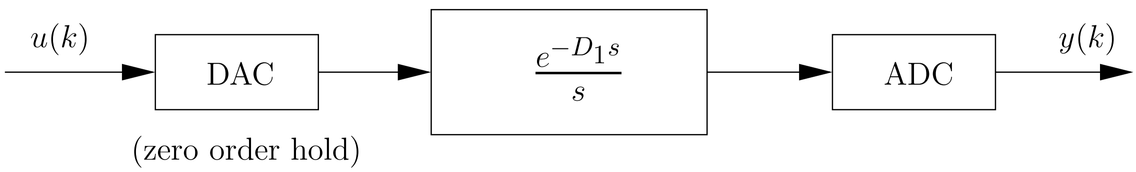

Figure 2: ADC/DAC arrangement for discretization

- There are a number of options available to analyse the delayed loop including the PID controller. One is to find the product of the plant and controller transfer functions, and then apply `bodedisp.m`. Is this a reasonable approach? Can you suggest some alternatives?

M. C. Smith
4 October 2020

Worksheet

2. Simplified aircraft model. Transfer function =

`num =` `den =`

Controller transfer function =

$k =$ $D =$

Phase margin =

Amount of extra time delay which can be tolerated =

2.1. PIO. Period of oscillation (observed) =

Period of oscillation (theoretical) =

2.2. Sinusoidal disturbances.

Maximum stabilising gain =

Gain at 0.66 Hz = Phase at 0.66 Hz =

Open loop T.F. $(y \to d) =$ Closed loop T.F. $(y \to d) =$

2.3. Fastest pole. $T =$

3. Autopilot. Proportional gain $K_c =$

Period of oscillation $T_c =$

3.1 Transfer function of PID controller =

PID constants: $K_p =$ $T_i =$ $T_d =$

Final value of $T_d =$