S2: Phase Lead Compensation of an Inverted Pendulum

1 Introduction

In this lab we will investigate analogue control of an inverted pendulum. We will use the root-locus tools in MATLAB to complete initial designs, and then investigate the effect of the dynamics which we have neglected in the design.

Electrical and electromechanical engineers will be familiar with the pendulum rig from 1st year laboratories. In that lab, you carefully calibrated the rig, experimentally determined the transfer functions and briefly saw the effects of lead-compensation. In this lab, you will consider the same lead compensator, and with the help of MATLAB, get to fully understand how it works. You will also see why detailed calibration can be avoided. **Electronics and Computer engineers** will be new to the pendulum control system.

The plant consists of an inverted pendulum, see figure 1:

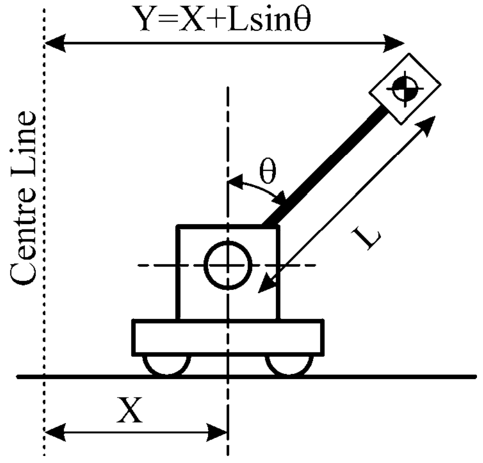

Figure 1

whose *X* position is controlled via a servo, our task is to design the compensator in order that the mass position *Y* moves to the reference position, see figure 2.

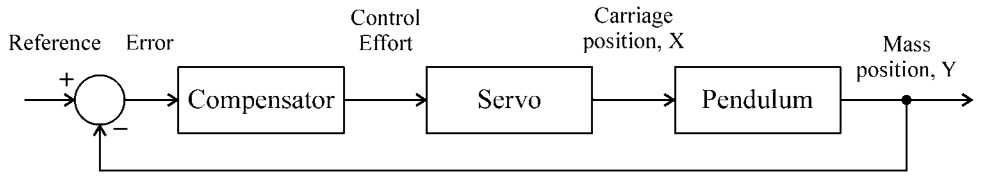

Figure 2

2 Preparation

1. Before the lab you should be familiar with the use of the control systems toolbox in MATLAB. Work through the MATLAB tutorial given on the lab. web-pages (completion of lab S1 also gives sufficient preparation).
2. Make sure you are know what the root-locus *represents* (if the root locus has not yet been covered in lectures note that you do not need to know how to draw it, MATLAB will do that for you in this lab!).

3. Read through the details of this lab carefully, making sure you understand how the transfer function *$H_1(s)$* relates to the compensator circuit (figure 4). If at any stage you are unsure how a MATLAB command works, use

   ```
   help < command >
   ```

   to find out.

3 Plant model

By modeling the pendulum from first principles, we can relate the angle Y to the position X of the pendulum base.

Hence, we consider the transfer function

$$P(s) = \frac{Y(s)}{X(s)}$$

to be given by:

$$P(s) = \frac{-\varpi^2}{s^2 - \varpi^2}$$

where

$$\varpi = \sqrt{\frac{g}{l}}$$

(l is the length of the pendulum, and $g = 9.81\text{ms}^{-2}$ is the acceleration due to gravity). For the Bytronic pendulum, with the mass at the top of the pendulum rod, the effective pendulum length is $l = 0.20$m. Compute the numerical values for $P(s)$. In MATLAB, set up a transfer function pendulum, corresponding to $P(s)$ (have a look at S1 to remind yourself how to enter transfer functions). Use `step` to verify the step response is unstable.

4 Linear position control

To familiarize yourself with the control console, we first consider the position control of the pendulum carriage.

Ensure the power to the pendulum unit is off. Unscrew the pendulum from the mount and place to one side. By referring to figure 3, position the set-point potentiometer (P1) to 0v. Connect the set point (A) to the servo amplifier input terminal (H). Position the servo gain (P3) at maximum, and the velocity gain (P4) at approximately the halfway point. The P3 and P4 settings should remain fixed for the remainder of the lab.

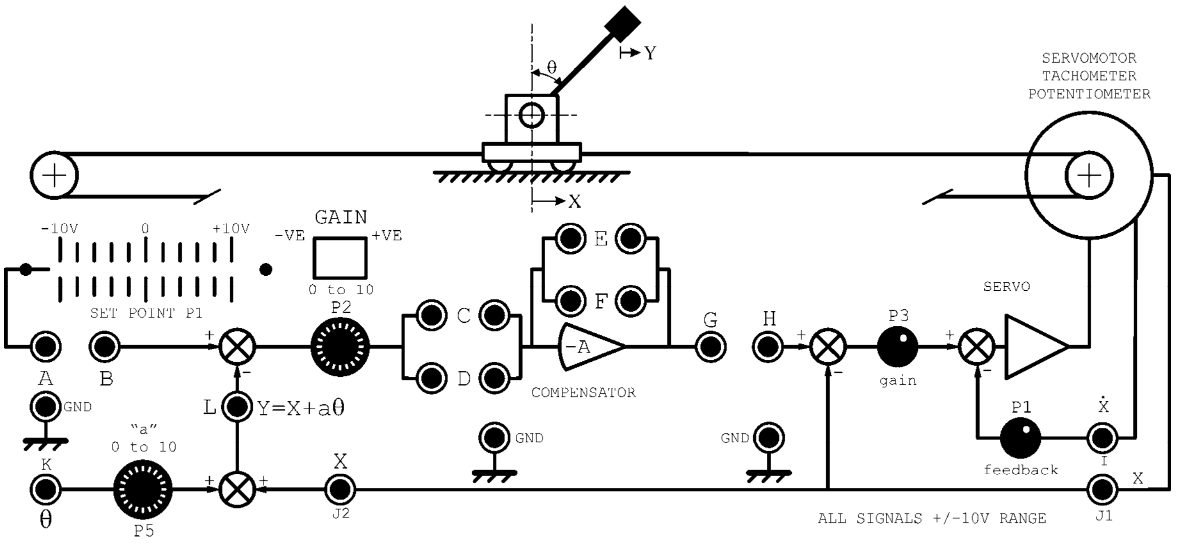

Figure 3

Turn on the power, and verify that the position of the set-point potentiometer determines the position of the carriage (*X*).

5 The role of potentiometer P5

The goal of the lab is to control the inverted pendulum, ie. to control the position of the pendulum mass *Y*. However, *Y* is not directly measured, but can only be inferred from measurements of *X* and *θ*, see figure 1. So since for small angles $\theta \approx sin(\theta)$, we can assume that:

$$Y = X + l\sin\theta = X + l\theta.$$

The *X* position is represented by a voltage *Vx* from the carriage potentiometer, which is added via an operational amplifier circuit to the voltage from the carriage potentiometer, which can be scaled by a factor *a* controlled by potentiometer P5. Thus we have:

$$V_Y = V_X + aV_\theta.$$

A detailed calibration of P5 is not needed in this lab, set P5 to 2.7 and ensure the pendulum weight is at the end of the rod.

6 Proportional Control

Firstly we investigate proportional control in simulation in MATLAB. With pendulum defined to be the transfer function of the pendulum, use the commands

```
k = 1.0
sysclp1 = feedback(pendulum,k)
step(sysclp1)
```

to investigate the response of the system under negative feedback (remember feedback defaults to negative feedback) for several values of *k*, eg. *k* = 0, 1, 2, 10. Comment on the stability of the system. Similarly investigate positive feedback eg. *k* = 0, 1, 2, 10:

```
sysclp2 = feedback(pendulum, k,+1)
step(sysclp2)
```

Now either sketch the root locus under positive and negative feedback, or plot it using MATLAB:

```
rlocus(pendulum),        negative feedback
rlocus(–pendulum),       positive feedback
```

Again, comment on the stability in both cases: can proportional control stabilize the system?

Bearing in mind your conclusions from above, we will now carefully implement positive feedback proportional control on the hardware pendulum.

Switch off the power. Attach the pendulum rod into the mounting in the carriage. Ensure the potentiometer P5 is set to 2.7, and the pendulum weight is securely fixed at the end of the rod. Connect the set point potentiometer to act as a reference input to the control system by linking A and B. Set up the compensator to act as a unity gain inverting amplifier by placing a 100k resistor across terminals *D* and another 100K resistor across terminals *F.* Connect the output of the operational amplifier directly to the servo reference input (link *G* and *H).* Ensure that the gain switch is set to negative.

Set the gain potentiometer (P2) to zero. Support the pendulum with your hand (don't fully let go – why?) and switch on the power. Adjust the set-point potentiometer to bring the carriage into the centre of the rig. Slowly increase the gain P2. Observe the carriage action as you move the pendulum to one side. Compare the behaviour qualitatively for different values of the gain. Is it consistent with the root-locus predictions?

7 Lead Compensation

In this section we design and implement a stabilising controller. We will consider lead compensation, ie. a compensator of the form:

$$H(s) = \frac{1 + c\tau s}{1 + \tau s}$$

By thinking about the root-locus, it is clear that we have to place the zero of the compensator in the LH plane (to hopefully attract the RH pole of the plant). The simplest choice is to place the zero of the compensator to directly cancel the *stable* pole of the plant, ie. choose

$$\frac{1}{c\tau} = \varpi$$

and then to use a 'high' value of *c*, say *c* = 10.

Build a transfer function compensator corresponding in *H(s),* and connect it in series with the plant and a negative gain:

```
sys2 = series(–compensator, pendulum)
```

Examine the root-locus, ie.

```
rlocus(sys2)
```

In reality, we cannot exactly choose $\frac{1}{c\tau} = \varpi$, in fact we will use a compensator of the form:

$$H_1(s) = \frac{1 + 0.1s}{1 + 0.01s}$$

Build a transfer function **`compensator2`** corresponding to $H_1(s)$ and plot the root-locus under positive feedback:

```
sys3 = series(–compensator2, pendulum)
rlocus(sys3)
```

Discuss the result.

Note that this 'approximate' pole cancellation can only be effective around a stable pole: trying to cancel unstable poles has a very different effect – it doesn't work at all – explain!

We now implement the transfer function $H_1(s)$ by means of an op-amp compensator circuit: see figure 4.

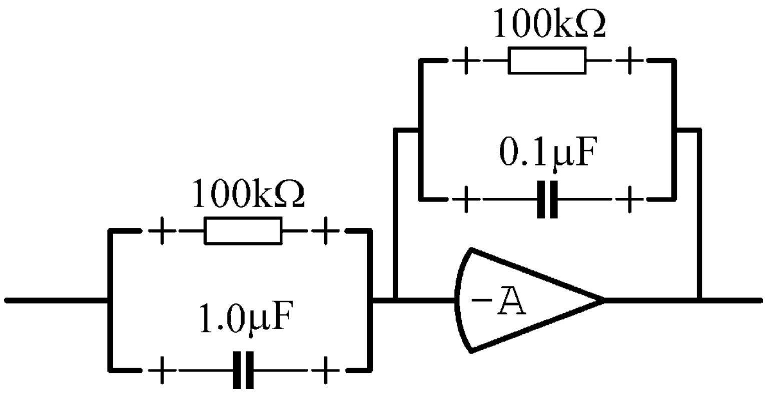

Figure 4

In general we have:

$$\frac{V_0}{V_1} = -\frac{Z_f}{Z_1}$$

where V_0 is the output voltage, V_1 is the input voltage, Z_f is the feedback impedance around the op-amp, and Z_1 is the impedance between the input voltage and the summing junction. For the parallel circuit consisting of a resistor R and a capacitance C_1 we have:

$$\frac{1}{Z_1} = \frac{1}{R} + C_1 s = \frac{1 + RC_1 s}{R}$$

Similarly,

$$\frac{1}{Z_f} = \frac{1}{R} + C_f s = \frac{1 + RC_f s}{R}$$

Thus

$$\frac{V_0}{V_1} = \frac{1 + RC_1 s}{1 + RC_f s}$$

Now verify that the compensator circuit shown in figure 4 corresponds to $H(s)$. We are now ready to implement the control scheme:

1. Ensure that the pendulum mass is fixed in the maximum position on the rod.
2. Set a to 2.7.
3. Ensure that the servo gain (P3) is still at maximum, and the velocity gain (P4) at approximately the half way point.
4. Set the gain (P2) to 0.

5. Ensure that the gain switch is set to negative.
6. Place the compensator components across terminals E and C as shown in figure 4.
7. Hold the pendulum upright in the centre of the track.
8. Connect the controller output to the servo input (ie. connect G and H).
9. Loosely supporting the pendulum, gradually increase the gain (P2) until stability is achieved.
10. You can now fully release the pendulum.

If you have difficulty in balancing the pendulum, try adjusting the potentiometer (a) slightly. Explain using the root-locus plot why small values of the gain P2 do not lead to a stable closed loop. Gently move the set-point potentiometer to non-zero positions. Does the system move to the set-point you expect? Explain the result.

8 What about the neglected servo dynamics?

The servo has significant, if fast dynamics (electrical and electromechanical engineers determined the transfer function in the first year lab – it was not unity!). Yet we have neglected the dynamics of the servo in our design procedure. The final part of this lab investigates the effect of these dynamics, and shows that they can indeed be neglected in design (up to a point...).

With the servo gain (P3) and the servo velocity feedback (P4) fixed in their original positions, the servo dynamics are approximately given by the second order model:

$$G(s) = \frac{1}{1 + 0.02s + 0.00025s^2}$$

This corresponds to a second order system with natural frequency ω and damping ratio ξ given by $\omega = 10Hz$, $\xi = 0.7$. In our design we neglected these servo dynamics, assuming they were fast enough that they could be ignored.

Let us see their effect. In MATLAB, add in the servo dynamics:

```
servo = tf([1], [0.00025  0.02  1])
actual = series(servo, pendulum)
sys4 = series(-compensator2, actual)
```

and plot the root locus

```
rlocus(sys4)
```

By using the command **`rlocfind`** determine the upper and lower values for the range of gains for which the system remains stable. Validate the upper value qualitatively on the hardware by increasing the gain P2 until the onset of stability.

Note that the root-locus for the system without the servo dynamics (**`rlocus(sys4)`**) did not predict this onset of instability.

9 Robustness of the controller

When you have made good choices for the gain P2, say by setting it to around 1.2, you will notice that the pendulum is extremely robust. Tapping the pendulum does not cause it to loose stability. Using

```
k = rlocfind(sys4)
```

to determine a stable value of the gain *k,* and define

```
syscpl3 = feedback(sys4, k)
```

Draw the bode plot:

```
bode(sys4)
```

and estimate the gain and the phase margin.

Now move the pendulum mass down the rod to the 50% position. Comment qualitatively on the relative stability by again tapping the weight. Then move the weight down to the bottom of the rod, and repeat the test and comment on the response. With the pendulum weight at the base of the rod, re-tune potentiometer P5 to get a good response. Now move the pendulum weight to the top of the rod, and assess the relative stability.

If you have time, define new transfer functions **`pendulum2`**, **`pendulum3`**, and redraw the relevant root-loci. Using **`rlocfind`**, investigate the range of stable gains, and compare this to the hardware response. How could these plots be used for calibration? You can also investigate the gain and phase margins via the bode plots.