AE4610

Welcome

Welcome to AE4610 - Dynamics and Control Laboratory!

Here you will find the lab manuals for all experiments that will take place this semester. Navigate to the appropriate experiment on the left-hand menu. You can export each experiment as a PDF if you want a copy for your records, or to print as a hard copy. Also, bookmark this page on your smartphone or tablet as well as your PC/laptop as a handy guide during your lab session.

Experiments

DC Motor

Objective

This laboratory experiment is designed to give the students a clear understanding of the DC motor control system. DC motors are used in typical flight control systems to actuate various devices (ailerons, flaps, elevator, rudder, etc). The experiment is divided into three parts.

Part A deals with modeling of a DC motor.

Part B includes the design of a position control system for the motor.

Part C involves implementation and evaluation of the controller designed in Part B.

Part A: Modeling

Any control problem consists, in general, of three main phases: The first phase is modeling/system identification, the second phase is controller design, and the third phase is controller implementation, testing and evaluation. This part of the experiment involves the first phase, namely, system identification and modeling of a DC motor. Identification of a complex system can be very challenging; however, for some systems it may be possible to find a simple model. The objective of Part A of this experiment is to find the parameters

of the reduced model for the DC motor used in the lab. This process yields a transfer function description of the DC motor system.

DC Motor

A picture of the DC motor set in the lab is shown in Fig. 1.1. Figure 1.2 is a schematic of a DC motor. The motor is made up of a rotor and a stator. The stator windings create a magnetic field of intensity Φ_f . In a permanent magnet DC motor (of the same type used in the lab) the stator is made of magnetic material and there is no field winding.

The rotor is made up of a set of armature windings and a commutator. The commutator is used to energize an individual winding when that winding is perpendicular to the stator field. That causes the magnetic field of the armature Φ_a to be always perpendicular to the field of the stator. This results in maximum torque for a given armature current. In general, the motor torque is proportional to the product of field intensities

$$T_m = k_1 \Phi_f \Phi_a$$

The armature field intensity is proportional to the armature current i_a . Hence $\Phi_a = k_2 i_a$, and since the stator field intensity Φ_f is constant, it follows that

$$T_m = K_m i_a$$

(1)

where $K_m = k_1 \Phi_f k_2$.

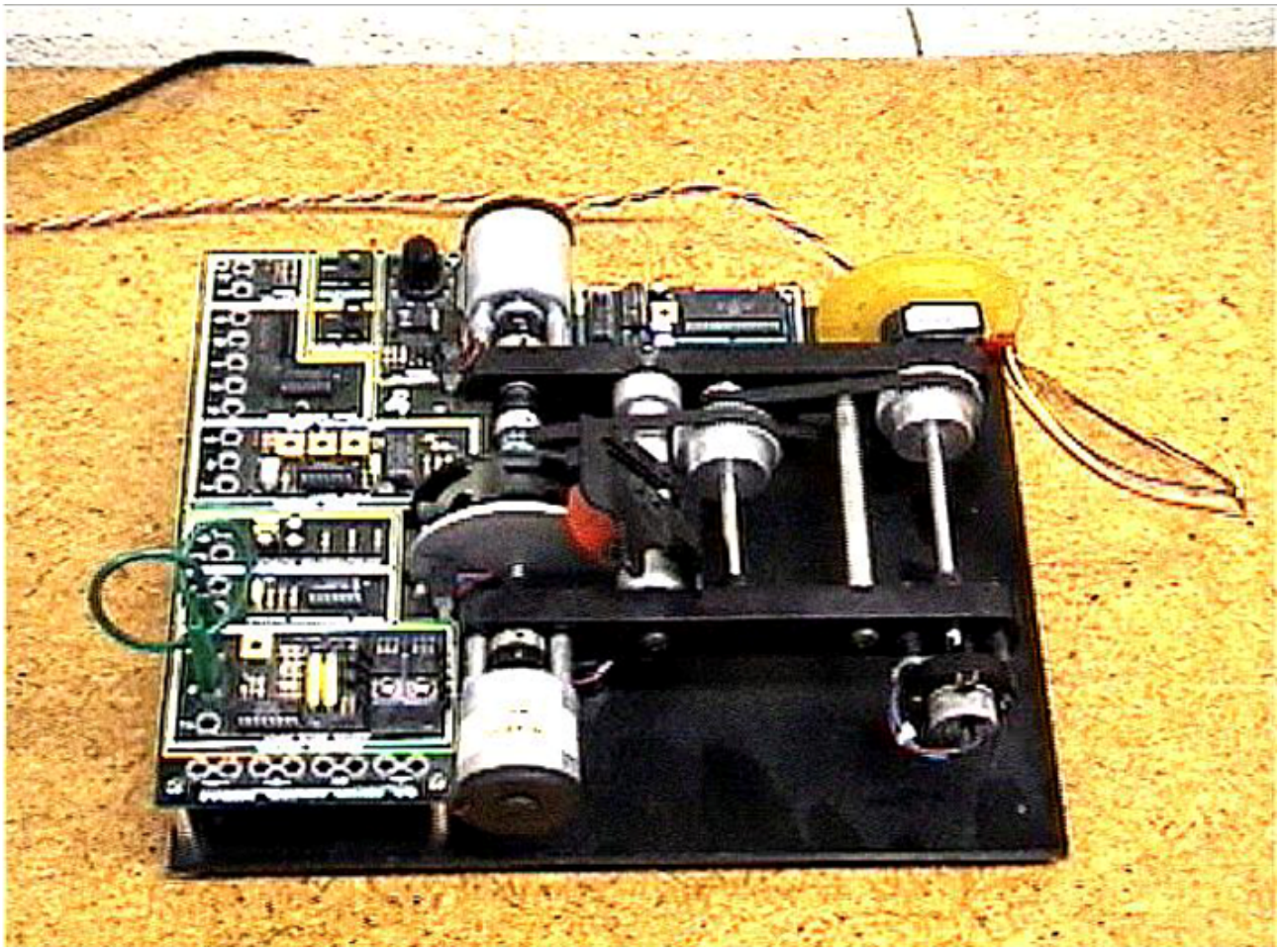


Figure 1.1: DC motor used in the lab.

When the armature winding rotates in a magnetic field, a back voltage, V_b (back electromotive-force (EMF) voltage) is generated that is proportional to the rate of change of the flux passing through the winding, and hence, it is also proportional to the rotor speed. It therefore follows that

$$V_b = K_b \omega$$

A schematic of the electrical and mechanical parts of the motor are shown in Fig. 1.2.

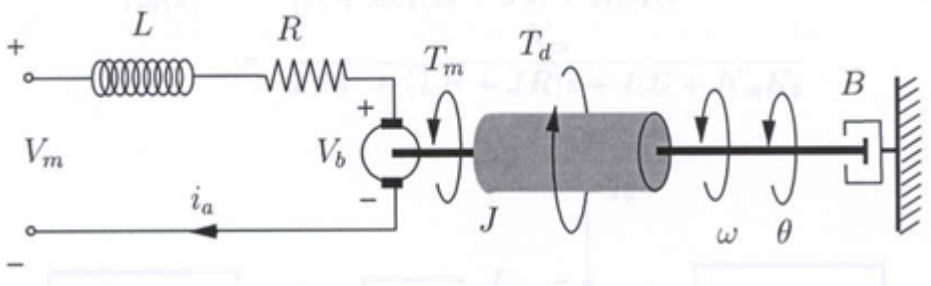


Figure 1.2: Schematic of a typical DC motor.

From the electric circuit of the armature we have

$$V_m = L\frac{di_a}{dt} + Ri_a + V_b$$

(1.2)

For the mechanical part of the motor we have

$$T_m - T_d - B\omega = J\frac{d\omega}{dt}$$

(1.3)

where

- L : armature inductance
- i_a : armature current
- R : armature resistance
- V_b : back EMF voltage
- V_m : voltage applied to motor
- J : total inertial (rotor and load)
- T_m : motor torque
- T_d : disturbance torque
- B : damping coefficient due to friction
- ω : angular velocity of the motor

The Laplace transform of (1.2) and (1.3) yields

$$V_m(s) = (sL + R)i_a(s) + V_b(s)$$

$$= (sL + R)i_a(s) + K_b\omega(s)$$

(1.4)

and

$$T_m(s) - B\omega(s) - T_d(s) = sJ\omega(s)$$

(1.5)

and using (1.1) we get

$$K_m i_a(s) - T_d(s) = (sJ + B)\omega(s)$$

Combining (1.4) and (1.5) and *assuming negligible disturbance torque*, one obtains that

$$\frac{\omega(s)}{V_m(s)} = \frac{K_m}{(R+Ls)(B+Js)+K_mK_b}$$

$$= \frac{K_m}{LJs^2+(LB+JR)s+RB+K_mK_b}$$

(1.6)

Often, L ≈ 0 and B ≈ 0. In this case, the transfer function in (1.6) simplifies to

$$\frac{\omega(s)}{V_m(s)} = \frac{K_m}{JRs+K_mK_b} = \frac{1/K_b}{\tau_m s+1}$$

(1.7)

where

$\tau_m = \frac{JR}{K_mK_b}$ is the **time constant** of this first-order system.

Figure 1.3 below shows a block diagram representation of the DC motor system.

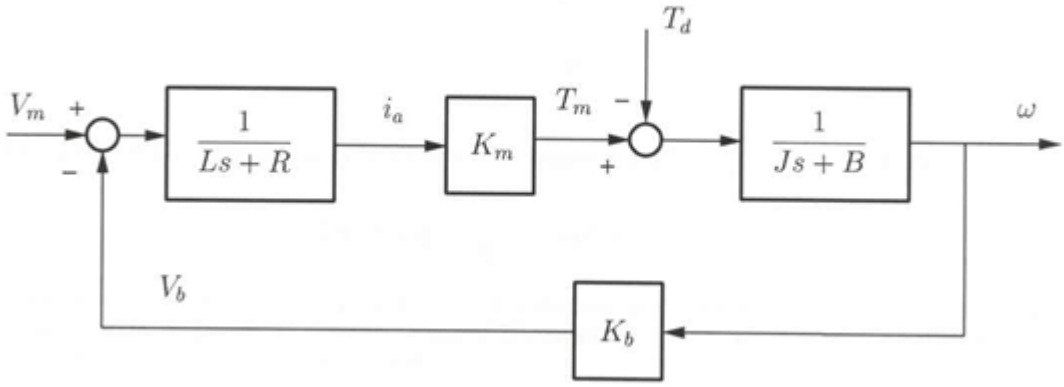


Figure 1.3: Block diagram of the DC motor

It is of interest to note that K_m is related to K_b . This can be shown by considering the power balance in steady state. The power input to the motor is

$$P_{in} = V_b i_a = K_b \omega i_a$$

The power delivered to the shaft is

$$P_{out} = T_m \omega = K_m \omega i_a$$

Equating the previous two expressions yields that $K_m = K_b$. (Note that the above relationship is based on the assumption that same units are used for both P_{in} and P_{out})

The transfer function for the DC motor as derived in (1.7) and as shown in Fig 1.3 is

$$\frac{\omega(s)}{V_m(s)} = \frac{K_m}{JRs + K_m K_b} = \frac{1/K_b}{\tau_m s + 1}$$

(1.8)

where

$$\tau_m = \frac{JR}{K_b^2} \qquad (K_m = K_b)$$

(1.9)

The DC gain of the system is calculated by evaluating (1.8) at s = 0. Hence

$$DC\ Gain = \frac{1}{K_b}\ [(rad/sec)/volt]$$

(1.10)

Thus, we have reduced the DC motor to a first-order system as shown in Figure 1.4. The theoretical values of the DC gain and time constant (τ_m) are given by (1.9) and (1.10). The DC gain and time constant values can also be determined experimentally, as shown in the next section.

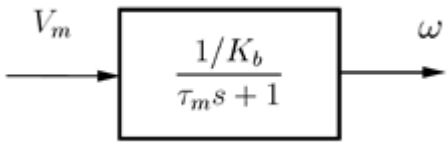


Figure 1.4: Simplified block diagram of a DC motor.

First Order System Response to a Step Input

The DC gain and the time constant value of the DC motor can be determined by subjecting the motor to a step voltage input. A typical first order system response to a step input is illustrated in Fig. 1.5. **The DC gain is obtained by dividing the steady state value of the output by the magnitude of the input step.** Recall that from the final value theorem

$$y_{ss} = lim_{t\to\infty} y(t) = lim_{s\to 0}\, sY(s)$$

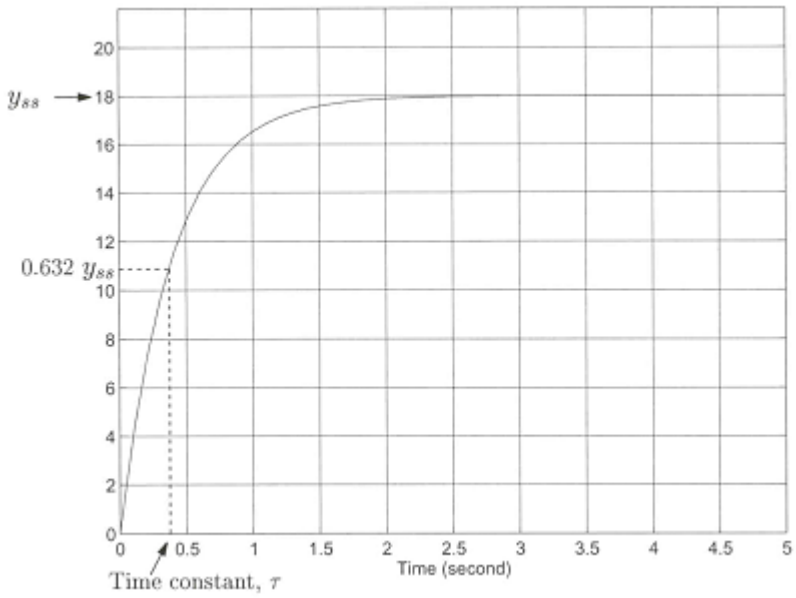


Figure 1.5: Typical First Order System Response for a Step Input

The time constant τ_m is the time at which the output reaches 63.2% of its steady state value.

Procedure

This part is skipped and instead a set of measured data will be provided for completion of the analysis described below.

1. Open the MATLAB and locate the file **DCMotor_OL.mdl** under the path **C:\AE4610_Controls_Lab\DCMotor** and open it. This is the block diagram for this part of the experiment.

2. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button). This generates the controller code in the MATLAB window.

3. Open the scope **Theta_dot**.

4. Turn on the power supply.

5. For the braked position of the magnetic brake (1 position), choose 3 different input voltages (< 5 volts).

6. Press **Connect to Target**  button in the menu bar and then press **Start**  . Run the motor with a step input of the selected input voltage for 10 seconds.

7. Save the motor angular velocity data with different names, for example type in the command line: `save DCMotor_OL_Data_3V.`

8. Do steps 6, 7 for all three step inputs.

9. Run the motor for 10 seconds for a sinusoidal voltage input. Select the amplitude (< 5 volts) and frequency (< 5 rad/s).

10. Email the data to yourself and delete the data files. **DO NOT DELETE THE SIMULINK MODEL**. Close Simulink and **DO NOT SAVE**.

Analysis

1. Plot the measured angular velocity of the DC motor vs. time for each step input. From the plot, determine the time constant and the DC gain of the DC motor as described in the above section.

2. Average the DC gain and time constant for the three different input voltages to get a single value of the DC gain and time constant.

3. Build a SIMULINK model that represents the DC motor model obtained experimentally (see Figure 1.4). Use the time constant and DC gain from Step 2.

4. Simulate the sinusoidal input used in Step #9 in your SIMULINK model and compare the output with the experimental angular velocity obtained in the experiment. Plot the simulated and experimental angular velocity responses on the same plot. Make sure to label each response. Explain any differences in the responses (very briefly, in a couple of sentences).

Part B: Controller Design

Objectives

The objective of Part B of the DC motor experiment is to design a position control system for the motor to meet a given set of specifications.

Equipment Required

- [ ] PC
- [ ] MATLAB and SIMULINK software

Experiment Notes

Now that we have identified the time constant and the DC gain for the system in the modeling part of this experiment, the next step is to design a position control system in order to convert the given DC motor into a position servo. A classical PID controller will be used to control the position of the motor subject to a given set of specifications, for example, bandwidth, steady state error requirements, etc.

Recall that the transfer function of the motor from V_m to ω in (1.8) is approximated as

$$\frac{\omega(s)}{V_m(s)} = \frac{1/K_b}{\tau_m s + 1}$$

Proportional (P) Control

Since $\theta = \omega$, where θ is the angular position of the motor, the transfer function from the applied voltage to the motor angular position is

$$G(s) = \frac{\theta(s)}{V_m(s)} = \frac{1/K_b}{s(\tau_m s+1)} = \frac{A}{s(\tau_m s+1)}$$

(1.11)

where $A = 1/K_b = DC$ gain of the motor.

The block diagram of a DC motor with a proportional controller is shown in Fig. 1.6. In this case the controller transfer function (H) is a simple gain K_P . From Fig. 1.6, the closed-loop transfer function from the commanded angular position θ_c to actual angular position θ is given by

$$\frac{\theta(s)}{\theta_c(s)} = \frac{AK_p}{\tau_m s^2 + s + AK_p} = \frac{AK_P / \tau_m}{s^2 + \frac{1}{\tau_m} s + \frac{AK_P}{\tau_m}}$$

Comparing the above transfer function to the standard form for a second-order system, i.e.,

$$\frac{\theta(s)}{\theta_c(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

we notice that the proportional gain K_p affects the natural frequency (and hence the bandwidth) of the closed-loop system.

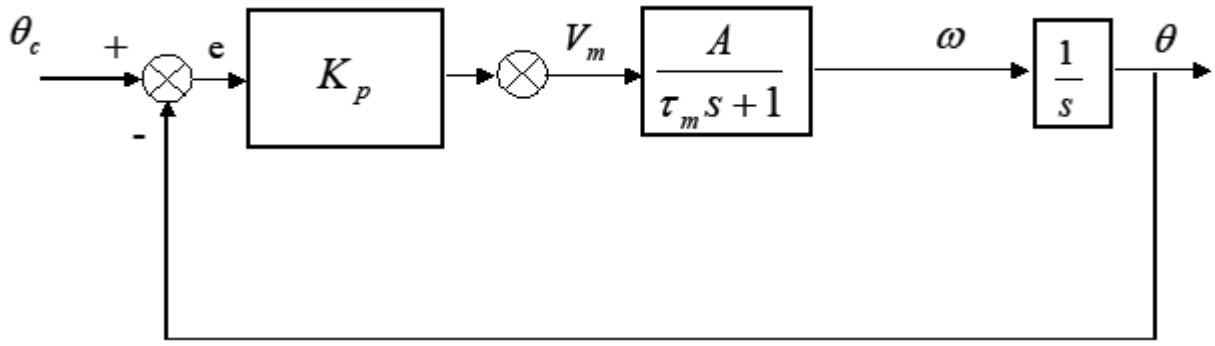


Figure 2.6: Closed-loop block diagram for the position control of a DC motor using a proportional controller

Proportional-plus-Derivative (PD) Control

The block diagram of a DC motor with an inner loop angular rate (i.e., derivative) feedback and an outer loop angular position error (proportional) feedback is shown in Fig. 1.7. From Fig. 1.7, the closed-loop transfer function from the commanded angular position θ_c to actual angular position θ is given by

$$\frac{\theta(s)}{\theta_c(s)} = \frac{AK_p}{\tau_m s^2 + (1 + AK_d)s + AK_p} = \frac{AK_P / \tau_m}{s^2 + \frac{(1 + AK_d)}{\tau_m}s + \frac{AK_P}{\tau_m}}$$

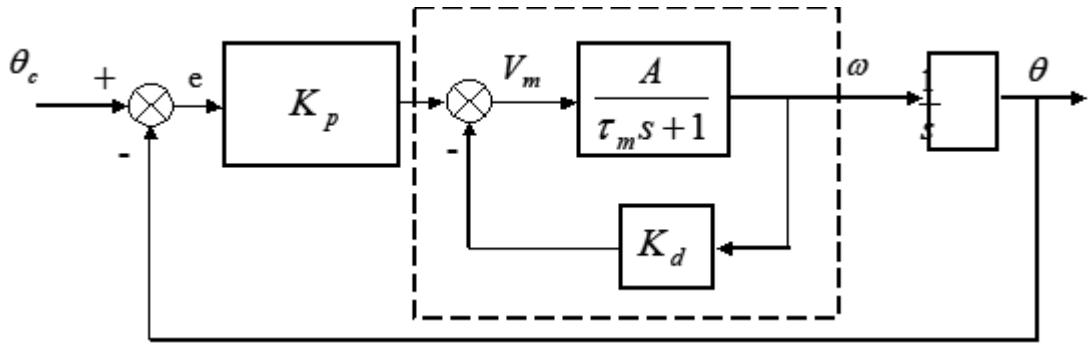


Figure 1.7: Closed-loop block diagram for the position control of a DC motor using a proportional-plus-derivative (PD) controller.

Comparing the above transfer function with the standard form for a second-order system, we notice that the derivative gain K_d affects the damping while the proportional gain K_p affects the natural frequency of the closed-loop system.

Proportional-plus-Integral-plus-Derivative (PID) Control

The block diagram of a DC motor with an inner loop angular rate (i.e., derivative) feedback and an outer loop angular position error plus integral of angular position error (i.e., proportional plus integral) feedback is shown in Fig. 1.8.

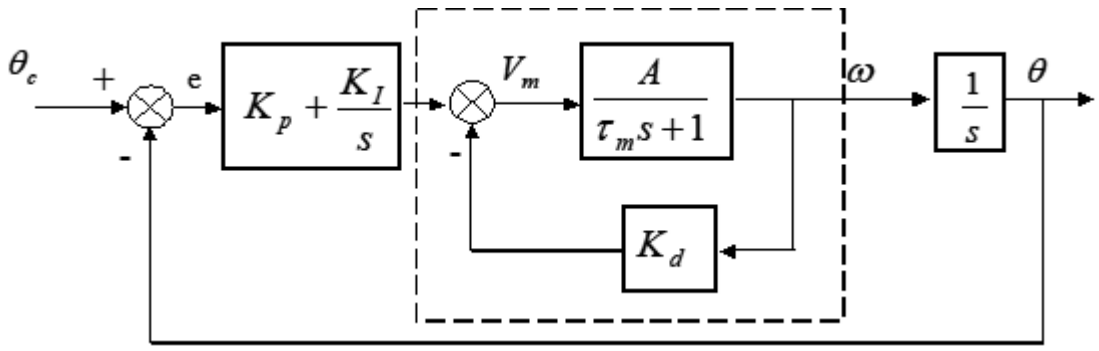


Figure 1.8: Closed-loop block diagram for the position control of the DC motor using a proportional-plus-Integral-plus-derivative (PID) controller

From Fig. 1.8, the open-loop transfer function GH becomes

$$GH = \frac{(K_p s + K_I)}{s} \frac{A}{(\tau_m s + 1 + AK_d)} \frac{1}{s}$$

indicating that with integral feedback, the type of the system increases from 1 to 2 and hence, results in zero steady state error to both ramp and step command inputs.

Sampling Time

The most important difference between analog and digital control is that digital systems operate using a clock. The timing of this clock and the amount of operations necessary to implement a controller place a limit on how frequently the controller can access sensors, make calculations, and modify the control inputs. (In addition, other elements of the control system themselves may introduce additional time delay. For this reason, the sampling time T_s of the digital system is an important parameter.) Digital controllers, in general, will have a maximum T_s . Increasing the T_s further will make the closed loop system unstable.

Design Specifications

The closed-loop DC motor system must meet the following specifications.

1. **Bandwidth frequency of at least 10 rad/sec** (Bandwidth is an important measure of the frequency range over which the system output follows well the command signal. **It is defined as the frequency at which the magnitude ratio is -3 dB in the closed-loop system frequency response plot**. Sinusoidal inputs with frequencies less than the bandwidth frequency are tracked 'reasonably well' by the system.)

2. **Phase margin of at least 60 deg**. (Gain and phase margins are stability margins to accommodate model variations. A flight control system actuator is typically subject to varying load, resulting in model variations. Hence, it is important to design a controller with sufficient stability margins to accommodate such variations and other effects such as wear with usage.)

3. **Zero steady state error to both step and ramp commands**.

Procedure

1. Using the model you have obtained from the prelab, input the transfer function G(s) (equation (1.11)) into MATLAB.

2. First consider a proportional controller, i.e., `C = KP` .Input `C = 1` into MATLAB.

3. Run `rltool(G,C)` . Select 'New Plot -> New Step' and 'Tuning Methods -> Closed Loop Bode Editor.' In the time response plot, right-click, select 'Systems' and uncheck the 'closed loop: r to u (green)' item (only blue should remain). Always do this in all the labs. In the closed loop Bode response window, select the magnitude plot only and readjust the magnitude plot limits to -5 dB and 0 dB under properties and limits. The bandwidth frequency (defined above) can be read from the closed loop Bode plot. It may be useful to add a grid to the bode plot.

4. In **rltool**, select 'Open-loop Bode' under 'Tuning Methods -> Bode Editor'. The phase margin (PM) of the system can be read off the open-loop bode plot.

5. Change the controller gain by double clicking 'C' under 'Controllers and Fixed Blocks' or by using your cursor on the root locus and check if all the design specifications can be met using a proportional controller, which would be optimal from a design standpoint. If not, at least select a value of KP such that bandwidth is more than 10 rad/sec. Save

the root locus, open and closed loop Bode, and time response plot. Save the gain K_p that you used.

6. Consider a proportional and derivative controller. In the compensator editor window under 'C', add a zero around –10. Using the cursor or manually input values, adjust the zero location of the controller and readjust the proportional gain (if necessary) such that, at least, the bandwidth and phase margin specifications are met. (Note that with a PD controller for this system, the open-loop transfer function GC is still of Type 1. Hence, the zero steady state error specification for ramp input cannot be met with a PD controller.) Save the root locus, open and closed loop Bode and time response plot. The compensator transfer function in **rltool** in this step is of the form shown below. Use it to determine the gains K_p and K_d . Save these gains.

   IMPORTANT: What is the difference between Figure 1.7 and the control structure that you now have in **rltool**? They are not identical. K_d is multiplied by the derivative of the Θ error signal in **rltool**, yet it is multiplied directly by ω in Figure 1.7. Why is this acceptable when dealing with a step input? Is there a way to mimic this structure exactly in **rltool**? (If you attempt to change it, make certain you modify the rest of these instructions carefully to achieve the correct results.)

7. Consider a PID controller. Using K_P and K_d values you have obtained from the previous steps, edit the compensator (i.e. add poles and zeros to match the format below) to include integral feedback with the integral gain set to a small value (i.e., K_I is roughly 0.05). The controller transfer function for this step is

$$C = K_P + K_d(s) + \frac{K_I}{s} = \frac{K_I(\frac{K_d}{K_I}s^2 + \frac{K_P}{K_I}s + 1)}{s}$$

8. Re-adjust the proportional and derivative gains (if necessary, by modifying the zeros, not the overall gain, K_I) such that bandwidth and phase margin specs are met.

9. Save the root locus plot, open and closed loop system Bode plots, and time response plot as graphs. Also, save your gains (you must convert from the **rltool** compensator format, back to our conventional gain representation – K_p , K_d , K_I).

10. Using the controller you have designed, obtain the closed loop transfer function and save your result. (Do this in MATLAB using the transfer function variables of G and C, or export the closed-loop transfer function from **rltool(T_r2y)**, using the 'Export'.) (May be listed as 'IOTransfer_r2y'). In MATLAB, change the exported state space into a transfer function using `tf(IOTransfer_r2y)`

11. Obtain poles of the closed loop system. Compute the natural frequency and damping of the dominant poles of the closed loop system. You can do all this from your exported transfer function using the function `damp()`. Save your results.

12. Construct a SIMULINK diagram using Fig. 1.8 as a guide. However, do not use a transfer function block for anything but the plant. (This allows you to modify your gains easily when necessary.) Include the controller gains from your design. Save your Simulink model for further use in Part C of the experiment.

13. Run the responses to a unit ramp input first with a PD controller (by setting `KI = 0`) and then with a PID controller (`KI = 0.05`). Run the simulation for about 30 sec.

14. Make comparison plots of ramp responses with PD and PID controllers (on the same plot) to show any differences between the two responses, especially in steady state response (i.e., for t>>0). The difference can be more easily spotted by plotting the error variable (i.e. error between the commanded position and the actual position) for each controller. Save the comparison plot and error plot.

Part C: Controller Implementation & Evaluation

Objectives

The objective of this part of the DC motor experiment is to implement the controller designed in part B and evaluate its performance.

Equipment Required

The following is a list of the required equipment to perform this experiment:

- [ ] Q8-USB or Q2-USB interface board
- [ ] MATLAB and SIMULINK software
- [ ] DC Motor

Controller Implementation

1. Open the MATLAB and locate the file **DCMotor_CL.mdl** under the path **C:\AE4610_Controls_Lab\DCMotor** and open it.
2. ***P Controller Test***: To understand the behavior of a Proportional controller, implement a P controller by typing the values for K_p and K_i to 0 in the command window. Set K_p as 10.
3. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.
4. Open the ***Theta Command*** block and make sure that the Final value is equal to **5/Kp**. This will limit the input to the motor to be less than a preset limit of 5 volts.
5. Open the scope *Theta*.
6. Turn on the power supply.
7. Press **Connect to Target**  button in the menu bar and then press **Start**  . Run the motor with a step input of the selected input voltage for 10 seconds.
8. Save your data.
9. ***PD Controller Test***: By introducing a derivative gain (K_d) notice how the system characteristics change. Set K_p to 1 and K_d to 2. Observe the behavior and save the data.

10. ***PID Controller***: Open the controller block and change the current values of K_p , K_d and K_i to match with your controller from Part B.

Controller Evaluation

1. Make sure that the magnetic brake position of the DC motor is in braked position (Position 1). Run the motor with a step input of the selected input voltage for 10 seconds and save the data.

2. Repeat Step 1 with the magnetic brake set in other positions (position 0 or 2). Save the data (i.e. filename “dccloopoffdesign”).

3. Repeat Step 1 with the magnetic brake in the original position (position 0) and the sampling time set to 0.1 sec. To set the sampling time, you need to go to the block diagram file, choose “Simulation” -> “Model Configuration Parameters” -> “Solver”. Set the fixed step to 0.1 sec. (It should have been 0.01 previously.) Save the data (i.e. filename “dccloopslowsampling”).

4. Email the data to yourself and delete the data files. **DO NOT DELETE THE SIMULINK MODEL**. Close Simulink and **DO NOT SAVE.**

Analysis

1. Run a nominal case simulation using your SIMULINK model from Part B, setting the command input to the value you used during the experiment. Compare the simulation results with corresponding experimental results for the nominal case and explain any differences between those results.

2. Compare the simulation results for the nominal case with the experimental results for the off-design case and explain any differences.

3. Modify the SIMULINK model to include a zero-order hold block at the input (labeled as V_m in Fig. 1.8) to the DC motor in order to account for the effect of the sampling time used in the experiment. (Note this is different from the quantization error introduced by

the digital encoder in measuring the angular position). Save this block diagram for inclusion in your lab report.

4. Run simulations with the zero-order hold set to 0.01 sec and 0.1 sec and save simulation outputs.

5. Compare the simulation output with a zero-order hold of 0.01 sec to the nominal case experimental results and explain any differences between this comparison and your initial nominal case simulation/experimental comparison.

6. Compare the simulation output with a zero-order hold of 0.1 sec to the slow sampling experimental results and explain any differences.

Lab Report

- Include a brief synopsis of what you did in Part A, all the controller design work from Part B, and all your work from Part C, namely all of the plots and comparisons asked for. We do not need a repeat of the lab manual, so don't spend time entering equations unless you directly used them at some point.

- Include the analysis questions from Part A

- Include the following in your analysis section:
  - Effect of proportional controller gain on closed loop system behavior
  - Effect of derivative controller gain on closed loop system behavior
  - Effect of integral controller gain on closed loop system behavior
  - Effect of sampling time on closed loop system behavior
  - Controller performance in off-design conditions

Gyroscope

Objective

The purpose of this experiment is to design a controller that maintains the direction of a gyroscope under base excitation. The controller can also be used to rotate the gyro platform to a desired orientation using the gyroscopic principle.

Image credit: NASA

Equipment Required

- [ ] Gyroscope (Electromechanical plant)
- [ ] Input/Output Electronic ECP Model 750 box
- [ ] DSP based Controller/Data Acquisition Board which is already installed in a PC
- [ ] MATLAB, SIMULINK and ECP software

Part A: Modeling

The Gyroscope Model

This experiment will be performed using the Model 750 Control Gyroscope. The system, shown in Fig. 3.1, consists of an electromechanical plant and a full complement of control hardware and software. The user interface to the system is via an easy-to-use PC-based environment that supports a broad range of controller specification, trajectory generation, data acquisition and plotting features. A picture of the setup including the DSP card and input/output electronics is shown in Fig. 3.2.

Figure 3.1: The Model 750 Control Moment Gyroscope

Description of the Gyroscope

The gyroscope consists of a high inertia brass rotor suspended in an assembly with four angular degrees of freedom, as seen in Fig. 3.3. The rotor spin torque is provided by a rare earth magnet type DC motor (Motor #1) whose angular position is measured by an optical encoder (Encoder #1) with a resolution of 2000 counts per revolution. The motor drives the rotor through a 3.33:1 gear reduction ratio, which amplifies both the torque and encoder resolution by this factor. The first transverse gimbal assembly (body C) is driven by another rare earth motor (Motor #2) to effect motion about Axis #2. The motor drives a 6.1:1 capstan to amplify the torque between the adjoining bodies C and B. A 1000 line encoder (Encoder #2) with 4 x interpolation is mounted on the motor to provide feedback of the relative position between bodies C and B with resolution of 24,400 counts per revolution.

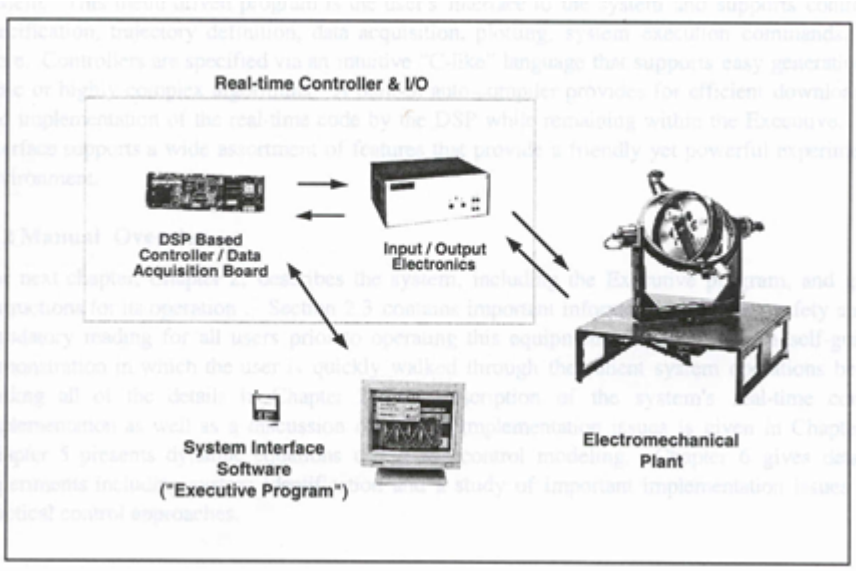


Figure 3.2: The Gyroscope Control System Setup

The subsequent gimbal assembly, body B, rotates with respect to body A about Axis #3. There is no active torque applied about this axis. A brake, which is actuated via a toggle switch on the Controller box, may be used to lock the relative position between bodies A and B and hence reduce the system degrees of freedom. The relative angle between A and B is measured by Encoder #3 with a resolution of 16,000 counts per revolution. Finally, body A rotates without actively applied torque relative to the base frame (inertial ground) along Axis #4. The Axis #4 brake is controlled similarly to the Axis #3 brake and the

relative angle between body A and the base frame is measured by an optical encoder (Encoder #4) with a resolution of 16,000 counts per revolution.

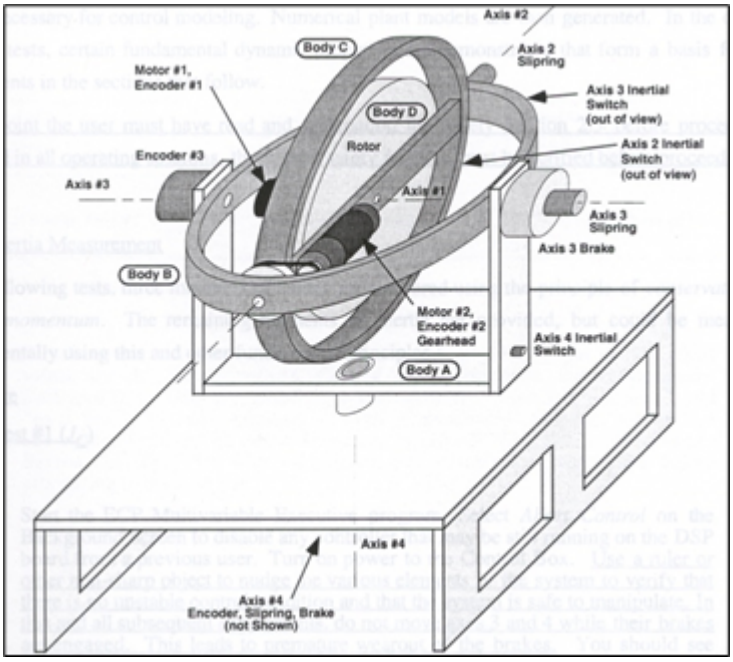


Figure 3.3: Control Moment Gyroscope Apparatus

Inertial switches or “g-switches” are installed on bodies A, B, and C to sense any overspeed condition in the gimbal assemblies. The switches are set to actuate at 2.1 g’s. For Axis #2, limit switches and mechanical stops are provided at the safe limit of travel. When any of these normally closed switches sense a high angular rate condition, they open and thereby cause a relay to turn off power to the controller box. When this power is lost, the fail-safe brakes (power-on-to-release type) at Axes #3 and #4 engage. Also, upon loss of power, the windings of Motor #1 and #2 have shorted, thereby effecting electromechanical damping. Thus all axes are actively slowed and stopped whenever an over-speed or over-travel condition is detected.

Metal slip rings are included at each gimbal axis to provide for continuous angular motion. These low noise, low friction, slip rings pass all electrical signals including those of the motors, encoder, g-switches, limit switches, and brakes to the control box.

Mathematical Model of the Gyroscope

In the configuration used in the experiment, gimbal Axis #3 is locked (ω_3 = 0), so that bodies A and B become one and the same (see Fig. 3.4). The resulting plant is useful for

demonstrations of gyroscopic torque action where the position and rate, q_4 and ω_4, may be controlled by rotating gimbal #2 while the rotor is spinning. Figure 3.5 shows the coordinate system used for this model.

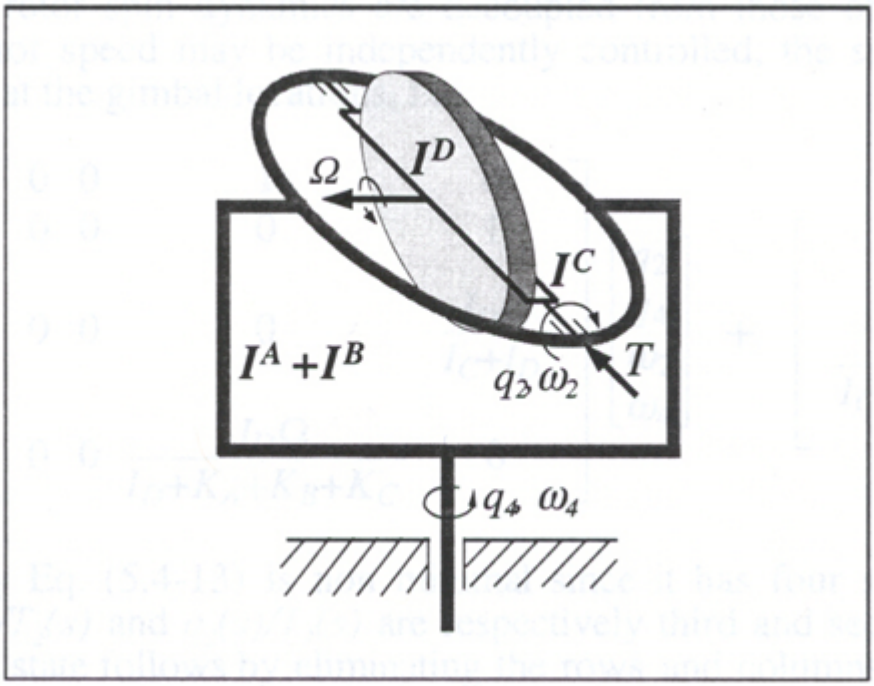


Figure 3.4: Gimbal # 3 Locked, All Others Free

Basic Equations

Figure 3.5 shows the definitions of the coordinate axes used for the model development. Let I_x, J_x, K_x (x = A, B, C and D) denote the scalar moments of inertia respectively in the bodies A, B, C, and D. Referring to Fig 3.5, the inertia matrices I^A, I^B, I^C, I^D are given as follows:

$$I^A = \begin{bmatrix} I_A & 0 & 0 \\ 0 & J_A & 0 \\ 0 & 0 & K_A \end{bmatrix}, I^B = \begin{bmatrix} I_B & 0 & 0 \\ 0 & J_B & 0 \\ 0 & 0 & K_B \end{bmatrix}, I^C = \begin{bmatrix} I_C & 0 & 0 \\ 0 & J_C & 0 \\ 0 & 0 & K_C \end{bmatrix}, I^D = \begin{bmatrix} I_D & 0 & 0 \\ 0 & J_D & 0 \\ 0 & 0 & K_D \end{bmatrix}$$

The basic gyroscope equation can be written as:

$$\vec{T} = \frac{d}{dt}\vec{H} = \dot{\vec{H}} + \vec{\omega}_F \times \vec{H}$$

(3.1)

where $\vec{T}$ is the vector of applied torque. Referring to Fig. 3.5, the components of applied torque can be written as

$$\vec{T} = \begin{bmatrix} T_2 \\ T_1 \\ 0 \end{bmatrix}$$

(3.2)

where T_1 and T_2 are applied torques about axes 1 and 2, respectively (see Fig. 3.5).

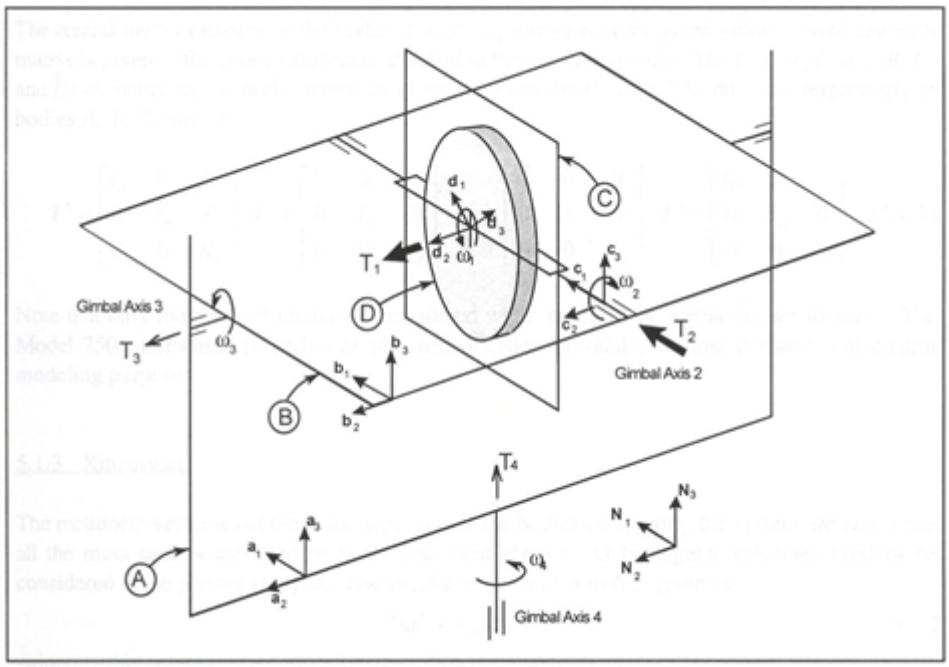


Figure 3.5: Coordinate Frame Definitions

The components of the angular momentum vector $\vec{H}$ and the angular velocity vector $\vec{\omega}_F$ can be written as

$$\vec{H} = \begin{bmatrix} (I_D + I_C)\omega_2 \\ J_D\omega_1 \\ (I_D + K_A + K_B + K_C)\omega_4 \end{bmatrix}$$

(3.3)

$$\vec{\omega}_F = \begin{bmatrix} \omega_2 \\ 0 \\ \omega_4 \end{bmatrix}.$$

(3.4)

Substituting equations 3.2, 3.3 and 3.4 into Eq. 3.1 results in the following set of nonlinear equations:

$$\begin{aligned} T_1 &= J_D\dot{\omega}_1 + (I_C + I_D)\omega_2\omega_4 - (I_D + K_A + K_B + K_C)\omega_2\omega_4 \\ T_2 &= (I_C + I_D)\dot{\omega}_2 - J_D\omega_1\omega_4 \\ 0 &= (I_D + K_A + K_B + K_C)\dot{\omega}_4 + J_D\omega_1\omega_2 \end{aligned}$$

(3.5)

Linearized Model

In this section, we proceed to linearize the nonlinear gyroscopic equations (Eq. 3.5) by imposing certain assumptions.

Assumptions

1. The angle of rotation of the rotor disk, D, about the gimbal axis 2 is small.

2. $\frac{\omega_2}{\Omega}, \frac{\omega_4}{\Omega}$ are small where Ω is the spin speed of the rotor disk D (rotational speed of the rotor) which is given as 400 RPM or 41.89 rad/sec.

Upon linearization ($\omega_2\omega_4 \approx 0, \omega_1\omega_4 \approx \Omega\omega_4, \omega_1\omega_2 \approx \Omega\omega_2$), we further obtain the following linearized equations:

$$T_1 - J_D \dot{\omega}_1 = 0$$

$$T_2 + J_D\Omega\omega_4 - (I_C + I_D)\dot{\omega}_2 = 0$$

$$J_D\Omega\omega_2 + (I_D + K_A + K_B + K_C)\dot{\omega}_4 = 0$$

(3.6), (3.7), (3.8).

In these equations, the rotor spin dynamics (Eq. 3.6) are decoupled from those of the second and fourth gimbals (Eqs. 3.7 and 3.8). Since the rotor speed may be independently controlled, the salient dynamics become those involving motion at the gimbal locations. The gimbals' angular positions q_2 and q_4 are related to ω_2 and ω_4 as

$$\dot{q_2} = \omega_2 \quad and \quad \dot{q_4} = \omega_4$$

(3.9)

Taking Laplace transform of Eqs. 3.7, 3.8 and 3.9 and eliminating □2 and □4 from the resulting equations results in the following transfer functions for $q_4(s)/T_2(s)$ and $q_2(s)/T_2(s)$:

$$G_4(s) = \frac{q_4(s)}{T_2(s)} = -\frac{J_D\Omega}{(I_C+I_D)(I_D+K_A+K_B+K_C)s^3+\Omega^2 J_D^2 s}$$

(3.10)

$$G_2(s) = \frac{q_2(s)}{T_2(s)} = \frac{(I_D+K_A+K_B+K_C)s}{(I_C+I_D)(I_D+K_A+K_B+K_C)s^3+\Omega^2 J_D^2 s}$$

(3.11)

Part B: Controller Design

Control System Overview

The objective is to design a controller for regulation and control of the angular position q4 using the gimbal torque T_2 . This mimics the following problem: Given a spacecraft with a momentum gyro, we would like to regulate and control the spacecraft attitude (q_4) using the gimbal torque (T_2) of the gyroscope. In this lab, the position of Axis #4, q_4 , is controlled by torqueing the Axis #2 motor. This is accomplished here with the use of a technique known as successive loop closure. First, a rate feedback loop around ω_2 is closed to damp the nutation mode of the gyroscope. Then an outer loop is closed to control q_4 . The block diagram for this process can be seen in Figure 3.6.

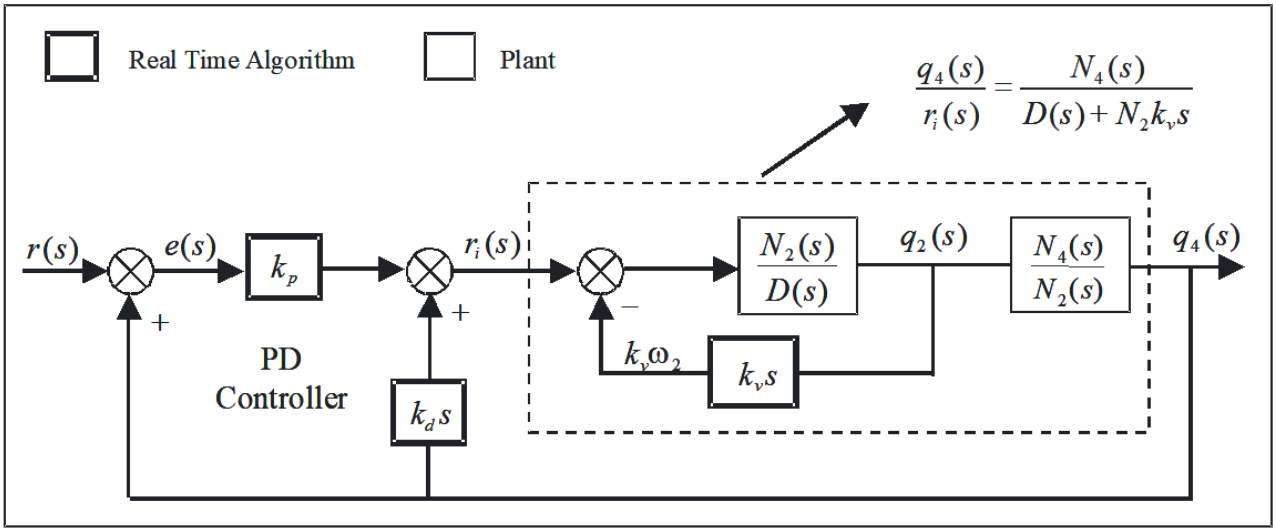


Figure 3.6: Successive Loop Closure Control Scheme

The equations defining the numerators and denominator in the block diagram are as follows.

$$D(s) = (I_C + I_D)(I_D + K_A + K_B + K_C)s^3 + \Omega^2 J_D^2 s$$

(3.12)

$$N_2 = k_{e2}k_{u2}(I_D + K_A + K_B + K_C)s$$

(3.13)

$$N_4 = -k_{e4} k_{u2} \Omega J_D$$

(3.14)

Note that the q_2 and q_4 shown in Fig. 3.6 represent measurements using encoders at Axes #2 and #4, respectively. The values k_{e2} and k_{e4} represent the encoder gains for Encoders #2 and #4, respectively. Finally, k_{u2} is a control effort gain and Ω is the rotational speed of the rotor in rad/sec (given above). The numerical values of various parameters are given in Tables 1 through 3 in the next page.

System Parameters

Table 1: Gyroscope Inertia Values

Body	Inertia Element	Value (kg - m^2)
A	K_A	0.067
B	I_B	0.012
	J_B	0.012
	K_B	0.03
C	I_C	0.0092
	J_C	0.023
	K_C	0.022
D	I_D	0.015
	J_D	0.027

Table 2: Encoder Gains

Axis i	Output/Rev $\bar{k}$ (counts/rev)	Gain $\bar{k_{ei}} = \bar{k}/(2\pi)$ (counts/rad)

1	6667	1061
2	24400	3883
3	16000	2547
4	16000	2547

Table 3: Control Effort Gains

Gain	Value (N/count)
k_{u1}	1.28 E-05
k_{u2}	9.07 E-05

> $k_{ei} = \bar{k_{ei}} * 32$ where the number 32 is called the **firmware gain.** The controller firmware multiplies the encoder and commanded position signals internally by 32. This is done for increasing the numerical precision. This multiplication is not performed on the plotted data. The constants used in the transfer functions N_2, N_4 are k_{ei} and not $\bar{k_{ei}}$.

Specifications

1. All closed-loop poles must be in the left half of the complex plane
2. The peak time in response to a unit step command input must be less than 0.2 sec.
3. The 2% settling time to a unit step command input must be less than 0.5 sec.
4. For the outer loop, k_p must be less than 6.0 and k_d must be less than 0.4.

Procedure

1. Select the inner loop derivative gain k_v to be equal to 0.08.

2. Open a new m-file and compute the transfer function $q_4(s)/r_i(s)$ using equations (3.12) through (3.14) and Fig. 3.6. Save your m-file.

3. Use **rltool** in MATLAB and select appropriate values for k_p and k_d to meet the given set of specifications. If needed, readjust the k_v value. You need to meet the requirements in **rltool**, however, you may have to change them later in SIMULINK to ensure they work. Save the **rltool** figures and your gains. Make sure to use the consistent controller architecture in the **rltool**.

4. Develop a SIMULINK diagram of the block diagram shown in Fig. 3.6 and save it for use in part C of the experiment. Use two separate transfer functions as shown in Fig 3.6.

5. Obtain the SIMULINK response to a unit step command using the set of gains you have selected. If the response does not meet the design requirements, then modify the gains until it does. Save the response, and make sure it proves you meet the requirements.

6. Individually change each of the three gains – k_p, k_d , and k_v – in three separate runs so that only one gain is ever altered at a time. Reduce k_p and k_d each by 50% and reduce k_v by 25%. Obtain simulation response to the same unit step command used in the previous step and compare it with the simulation response for the nominal case by graphing them on the same plot. Observe the effect of k_p, k_d , and k_v on the response behavior of the closed loop system.

7. The TA's will need to see: Gains, **rltool** plots, SIMULINK diagram, SIMULINK response to a step command (proving peak and settling time requirements are met), and the 3 modified responses.

> You must have your work checked out by one of the TAs before leaving the lab to get credit for your work.

Part C: Controller Implementation & Evaluation

The plant gimbals 2 and 3 must be initially oriented as shown in Figure 3.7 in order for the dynamic equations to be correct. Care must be taken to match the particular orientation of the gimbal members to achieve correct polarity. This may be checked by rotating the respective gimbals in a positive direction as identified by Fig. 3.4 and verifying that the encoder counts are increasing.

After releasing Axis #3 and Axis #4 brakes, some residual friction may exist. It is therefore advised to slowly rotate the corresponding assembly, using a ruler or any other long slender object, several revolutions to free any excessive friction prior to performing tests.

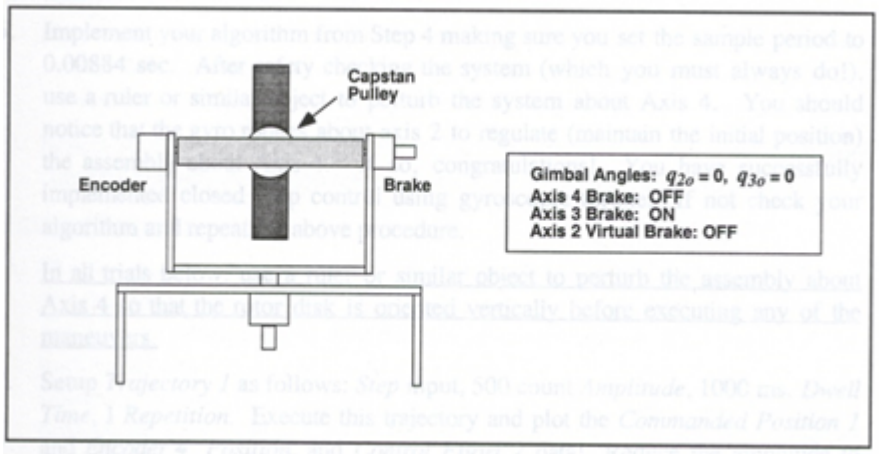


Figure 3.7: Configuration for experiments

Safety Instructions

Safety Note 1: The system's safety functions must be verified before each operational session. The system must be checked visually to verify that the rotor support structure, protective clear rotor cover, brakes and the inertial switches all appear to be undamaged and securely fastened.

Safety Note 2: In the event of an emergency, control effort should be immediately discontinued by pressing the red “OFF” button on the front of the control box.

Warnings

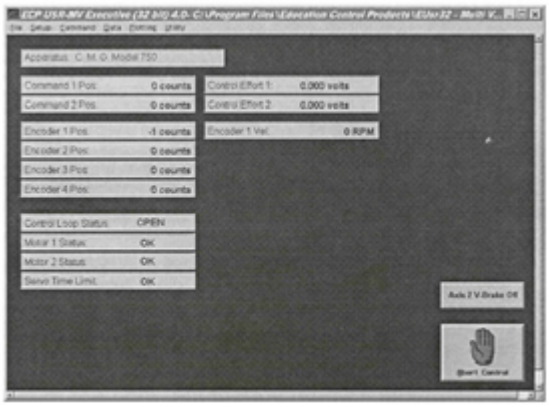
Warning 1: Stay clear of and do not touch any part of the mechanism while it is moving, while a trajectory is being executed or before the active controller has been safety checked.

Warning 2: The rotor should never be operated at speeds above 825 RPM. The user should take precautions to assure that this limit is not exceeded.

Warning 3: Never leave the system unattended while the Control box is powered on.

Controller Implementation

1. Enter the program by clicking on shortcut to **Gyroscope** on the desktop and turn on the gyroscope. Setup the apparatus as in Fig. 3.7. On the screen of the PC you should see the following picture:



2. Enter **File** menu, choose **Load Setting** and select the file **C:\Program Files (x86)\ECP Systems_MV\mv\default.cfg.**

3. Initialize the rotor speed to 400 RPM by clicking on the **Command menu** and then on **Initialize Rotor Speed.**

4. Open the control program **GyroControl_Gimbal4_PID.alg** by going to the **Setup** menu and clicking on **Control Algorithm**. Then click the **Load From Disk** button and select

C:\AE4525_Controls_Lab\Gyro_Controller\GyroControl_Gimbal4_PID.alg. Be sure that the sample period, T_s , is **0.00884** seconds and set the k_v, k_p and k_d values to the values from your controller design in Part B. **Check the values with a TA before entering them**. **Do not input magnitude of k_p > 6.0 nor k_d > 0.4.** T_s is located both on the control algorithm screen and in the program itself, so be sure that both values are correct. If any of the values need to be changed, do this by clicking the **Edit Algorithm** button. Once any necessary changes have been made, exit the edit algorithm section by selecting **Save Changes and Quit** under the **File** menu. The algorithm can now be implemented by selecting **Implement Algorithm**.

5. Use a ruler or similar object to perturb the system about Axis #4. You should notice that the gyro rotates about Axis #2 to regulate the assembly about Axis #4. In all the trials below, perturb the system about Axis #4 so that the rotor disk is oriented vertically before executing any maneuver.

Controller Evaluation

1. Now, prepare the input for the system by selecting the **Command Menu** and then **Trajectory 1**. Select the **Step Input** and then click the **Setup** button. Enter the following parameters for this case: step size, 200 counts; dwell time, 1000 ms.; 1 repetition. When finished, hit **OK**, then **OK** again to leave the setup trajectory menus.

2. Now, go to the **Data** menu and click on **Setup Data Acquisition**. Be sure that the Commanded Position 1, Encoder 4 Position and Control Effort 2 are all located in the **Selected Items** box. If they are not, add them to that list by selecting them in the **Possible Choices** box and then clicking on the **Add Item** button. When finished, hit **OK** to exit this menu.

3. Go to the **Command Menu** and click **Execute**. Make sure that both **Normal Data Sampling** and **Execute Trajectory 1 Only** are checked. Then click the **Run** button. The input trajectory will be run on the gyroscope now. When the box on the screen says **Upload Complete**, click on the **OK** button.

4. After the data collecting has finished, go to the **Plotting** menu and select **Setup Plot**. Set Command Position 1 and Encoder 4 Position on the left axis and Control Effort 2

on the right axis. Click on the **Ok** button when finished. This will generate a plot of the data from the executed trajectory. If desired, zooming the plot can be accomplished by going to the **Plotting** menu and selecting **Axis Scaling**.

5. Save the data by clicking on the **Data** menu and going to **Export Raw Data**.

6. Repeat step 6 and setup another step input. Enter the following parameters for this case: step size, 1000 counts; dwell time, 1000 ms; 1 repetition. Repeat steps 8 to 10 for this new input and save the data.

7. The step response is useful for system characterization but is seldom used for an actual in-service trajectory because it is excessively harsh (high acceleration and jerk (rate of change of acceleration)). A more common trajectory used for tracking applications is a ramp. Setup a ramp input as follows: Ramp input with Unidirectional Moves not checked, Distance = 6000 counts, Velocity = 2000 counts/sec, Dwell Time = 1000 ms, 2 repetitions. Execute this maneuver for your design and plot the Commanded Position 1 and Encoder 4 Position data. You may also want to view the Control Effort 2 and Encoder 2 Velocity data. (This should be done in separate plots since the scaling of these two variables is greatly different.) Note the relatively close tracking and rapid accelerations at each end of the constant velocity sections. This would not be possible for the small actuator of Axis #2 acting on the massive assembly in a conventional fashion. By using gyroscopic control actuation and the associated transfer of momentum stored in the rotor, the high authority control is made possible. Save the data.

8. Remember to switch off the system when you are done with your experiments.

Analysis & Lab Report

- Include an overview of the controller design from Part B.

- Using the SIMULINK block diagram you have developed of Fig. 3.6 in Part B, simulate the system response for the same step and ramp commands you have used in Part C. Compare your simulated responses with the corresponding experimental results. What

is the rise time and % overshoot in the experiment and simulation responses to the step command used?

Torsional Pendulum

Objective

The objectives of this laboratory experiment are as follows:

1. Identify the parameters of a multi-degree-of-freedom system.

2. Demonstrate some key concepts associated with proportional plus derivative control for a two degree-of-freedom torsional mechanism.

3. Implement a PD controller on a 2-disk (2 DOF) system where the feedback is based on θ1 of the lower disk. Such a scheme is referred to as **collocated** since the sensor output is directly coupled to the actuator input.

Implementing a suitable controller on a 2-disk (2 DOF) system, where the feedback is based on θ_3 of the upper disk is referred to as non-collocated since the sensor output and the actuator input are at different locations. The addition of the spring and the second inertia to the rigid body single DOF increases the plant order by two and adds an oscillatory mode to the plant dynamics. This may be thought of, in a sense, as a dynamic disturbance to the rigid body plant (single DOF).

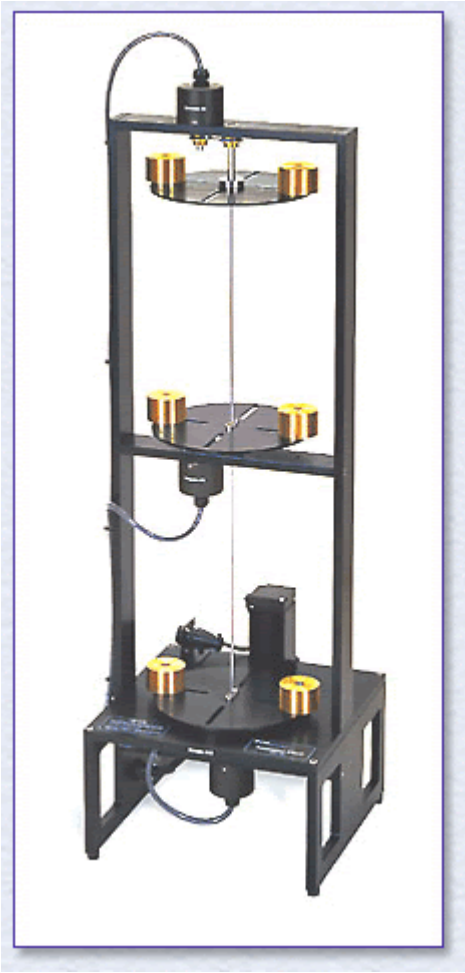


Image source: https://www.phywe.com/en/torsional-vibrations-and-torsion-modulus.html

Equipment Required

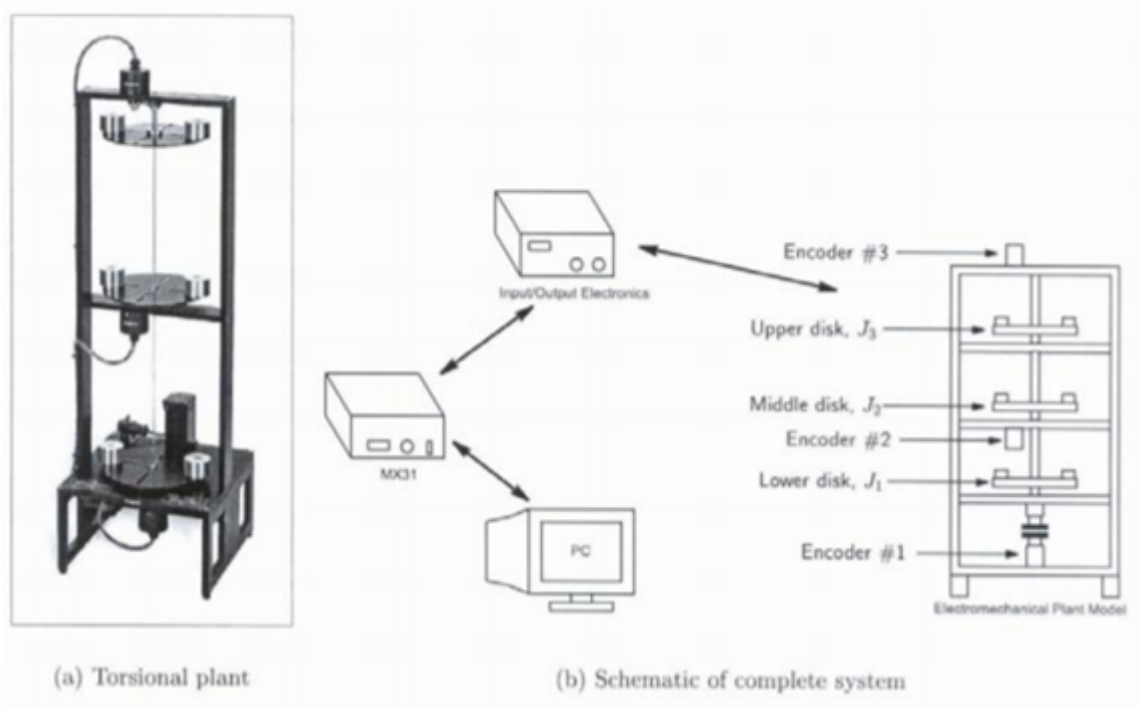
- [ ] Torsional mechanism (electromechanical plant Model 205a)
- [ ] Input / Output Electronic ECP Model 250 box
- [ ] DSP based Controller/Data Acquisition Board which is already installed in a PC
- [ ] MATLAB, SIMULINK and ECP software

Part A: Modeling

This part of the experiment involves the first phase – system identification and modeling of the torsional mechanism. This is performed by representing the mechanism as a combination of spring-mass systems and determining the model parameters from the system response. The approach will be to indirectly measure the inertia, spring, and damping parameters by making measurements of the system response while set up in a pair of classical spring-mass configurations.

Electromechanical Plant

A schematic of the plant interconnection with the control computer is shown in Fig. 4.1-(b). The plant is shown in Fig. 4.1-(a). It consists of three disks supported by a torsionally flexible shaft which is suspended vertically on anti-friction ball bearings. The shaft is driven by a brushless servo motor connected via a rigid belt and pulley system with a 3:1 speed reduction ratio. An encoder located on the motor is used for commutation. This is the process by which the current is distributed to the motor coils. In order to commutate the motor, a sensor connected to the motor shaft is used to feedback the instantaneous rotor position. The encoder on the base of the shaft measures the angular displacement (in counts) which is converted to an angle θ_1 of the lower disk, J_1 . The second disk, J_2 , is connected to its encoder which measures θ_2 , by a belt/pulley with a 1:1 pulley ratio and similarly the third disk, J_3 , is connected to its encoder (which measures θ_3) by a rigid belt/pulley with a 1:1 pulley ratio.



(a) Torsional plant (b) Schematic of complete system

Figure 4.1. Experimental control systemv

The plant may be placed in a variety of free and clamped configurations with 1, 2, and 3 degrees of freedom. For 1 and 2 DOF plants, the torsional spring constant k1 may be halved by the choice of disk location. Changing configuration often requires removing or replacing inertia disks. Although these operations are straightforward, it is recommended that they are performed with care. The user may change inertia values by changing the number of masses and/or their location on a given disk.

In the following experimental procedure, we will identify the plant parameters J_1, J_3, k_1, c_1 and c_3 , where J is the mass moment of inertia, k is the torsional spring constant, and c is the damping coefficient.

System Identification

Consider a one-DOF underdamped system model given by the second-order scalar differential equation

$$J\ddot{\theta} + c\dot{\theta} + k\theta = 0$$

(4.1)

We will use the logarithmic **decrement** to estimate the damping ratio of the system. The solution of (4.1) is given by

$$\theta(t) = \Theta e^{-\zeta\omega_n t} cos(\omega_d t + \phi)$$

where

$$2\zeta\omega_n = \frac{c}{J}, \quad \omega_n = \sqrt{\frac{k}{J}}, \quad \omega_d = \omega_n\sqrt{1-\zeta^2}$$

(4.2)

The value of θ during the two complete successive cycles is given by

$$\theta(t_1) = \Theta e^{-\zeta\omega_n t_1} cos(\omega_d t_1 + \phi)$$

$$\theta(t_2) = \Theta e^{-}\zeta\omega_n t_2 cos(\omega_d t_2 + \phi)$$

where

$$t_2 = T_d + t_1,\ with\ T_d = \frac{2\pi}{\omega_d}$$

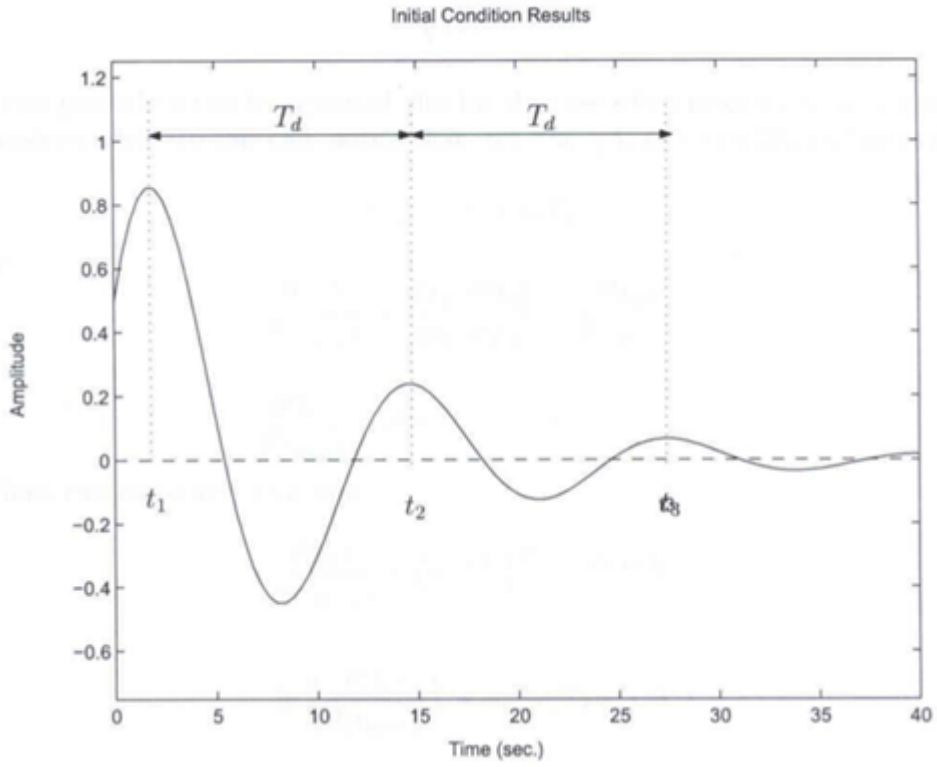


Figure 4.2. Calculation of the logarithmic decrement

Since

$$cos(\omega_d t_2 + \phi) = cos(2\pi + \omega_d t_1 + \phi) = cos(\omega_d t_1 + \phi)$$

it follows that

$$\frac{\theta(t_1)}{\theta(t_2)} = \frac{e^{-\zeta\omega_n t_1}}{e^{-}\zeta\omega_n(t_1+T_d)} = e^{\zeta\omega_n T_d}$$

(4.3)

Let the logarithmic decrement δ be given by

$$\delta = \ln \frac{\theta(t_1)}{\theta(t_2)}$$

It follows then easily from (4.3) and (4.2) that

$$\delta = \zeta\omega_n T_d = \zeta\omega_n \frac{2\pi}{\omega_n\sqrt{1-\zeta^2}}$$

or that

$$\delta = \frac{2\pi\zeta}{\sqrt{1-\zeta^2}} = (\frac{2\pi}{\omega_d})(\frac{c}{J})$$

(4.4)

Finally, if δ is known, an estimate for ζ is given by

$$\zeta = \frac{\delta}{\sqrt{(2\pi)^2+\delta^2}}$$

The previous procedure can be repeated also for the case when measurements are taken over non-successive cycles. To this end, notice that (see Fig. 4.2) $t_3 = t_1 + 2T_d$ and more generally, that

$$t_m = t_1 + mT_d$$

Therefore

$$\frac{\theta(t_1)}{\theta(t_{m+1}} = \frac{\theta(t_1)(\theta(t_2)}{\theta(t_2)\theta(t_3)} \dots \dots \frac{\theta(t_m)}{\theta(t_{m+1}}$$

where

$$\frac{\theta(t_i)}{\theta(t_{i+1})} = e^{\zeta\omega_n T_d}, i = 1, \dots \dots, m$$

The previous two equations give that

$$\frac{\theta(t_1)}{\theta(t_{m+1})} = (e^{\zeta\omega_n T_d})^m = e^{m\zeta\omega_n T_d}$$

or that

$$\ln(\frac{\theta(t_1)}{\theta(t_2)} = m\zeta\omega_n T_d m\delta$$

Finally,

$$\delta = \frac{1}{m} \ln(\frac{\theta(t_1)}{\theta(t_{m+1})})$$

Loaded vs. Unloaded Disk

Assume that the natural frequency of the loaded disk is ω_{nl} and the corresponding damping ratio is ζ_l . Therefore,

$$\omega_{nl} = \frac{\omega_{d_l}}{\sqrt{1-\zeta_l^2}} \qquad\qquad \zeta_l = \frac{\delta_l}{\sqrt{(2\pi)^2+\delta_l^2}}$$

(4.5)

where ω_{dl} is the damped natural frequency for the loaded disk case (denoted by subscript 'l'). Similarly, for the unloaded disk case (subscript 'u') one has

$$\omega_{n_u} = \frac{\omega_{du}}{\sqrt{1-\zeta_u^2}}, \qquad \zeta_u = \frac{\delta_u}{\sqrt{(2\pi)^2+\delta_u^2}}$$

(4.6)

Recall that

$$\omega_{n_l}^2 = \frac{k}{J_m + J}, \quad \omega_{n_u}^2 = \frac{k}{J}$$

for the loaded and unloaded cases, respectively. Solving for k and equating the resulting expressions yields

$$w_{n_l}^2 \left(J_m + J\right) = \omega_{n_u}^2 J$$

or that

$$J = \frac{J_m \omega_{n_l}^2}{\omega_{n_u}^2 - \omega_{n_l}^2}$$

(4.7)

From (4.7) and (4.2), the damping coefficient can be computed from

$$c = 2\zeta_u J \omega_{n_u}$$

(4.8)

Experimental Procedures

1. For model 205a, clamp the center disk using the 1/4” bolt, square nut, and clamp spacer as shown in Fig. 4.3. Only light torqueing on the bolt is necessary.

2. Secure four 500g masses on the upper and lower disks. Verify that the masses are secured properly and that each is at a center distance of 9.0 cm from the shaft centerline.

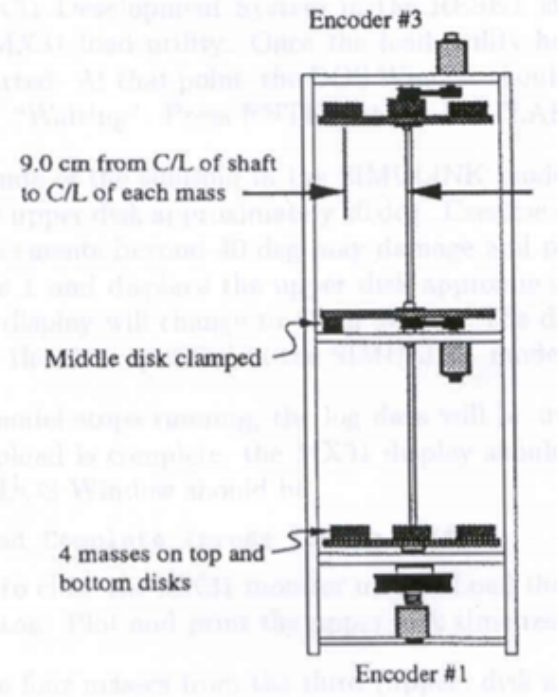


Figure 4.3. Electromechanical plant model 205a

3. Enter the program by clicking on shortcut to 3D Torsion on the desktop and turn on the torsional mechanism.

4. Enter **File** menu, choose **Load Setting** and select the file **C:\Program Files (x86)\ECP Systems\cn\default.cfg**.

5. Choose the correct personality file by going to **Utility** menu and clicking on **Download Controller Personality File**. Download the file **C:\Program Files (x86)\ECPSystems\cn\M205di6.pmc**. This will implement ‘CLOSED’ Loop by default. Click on **Abort Control** at the lower right of the screen to ‘OPEN’ the loop.

6. Enter the **Control Algorithm** box via the **Setup** menu and set `Ts = 0.00442` second, then OK. Enter the **Command** menu, go to **Trajectory** and select **Step, Setup**. Select

Open Loop Step and input a step size of 0 (zero), a duration of 4000 ms and 1 repetition. Exit to the Background Screen by consecutively selecting **OK**. This puts the controller board in a mode for acquiring 8 sec of data on command but without driving the actuator.

7. Under **Data** menu, go to **Setup Data Acquisition** and select Encoder #1 and Encoder #3 as data to acquire and specify data sampling every 2 (two) servo cycles, i.e. every 2 Ts's. Select **OK** to exit. Select **Zero Position** from the **Utility** menu to zero the encoder positions.

8. From the **Command** menu, select **Execute**. Prepare to manually displace the upper disk by approximately 20 deg. Exercise caution in displacing the inertia disk; displacements beyond 40 deg may damage and possibly break the flexible drive shaft. (Displacements beyond 25 deg will trip a software limit which disables the controller indicated by "Limit Exceeded" in the Controller Status box in the Background Screen. To reset, simply reselect Execute from the Command menu.) With the upper disk displaced approximately 20 deg (≤ 1000 encoder counts as read on the Background Screen display) in either direction, select **Run** from the **Execute** box and release the disk approximately 1 second later. The disk will oscillate and slowly attenuate while encoder data is collected to record this response. Select **OK** after data is uploaded.

9. Under the **Plotting** menu, select **Setup Plot** and choose Encoder #3 position and then select **Plot Data** from the **Plotting** menu. You will see the upper disk time response.

10. Save the data by clicking on the **Data** menu and going to **Export Raw Data**.

11. Remove the four masses from the third (upper) disk and repeat steps 8 through 10 for the unloaded disk. If necessary, repeat step 6 to reduce the execution (data sampling) duration.

12. Repeat steps 8 through 11 for the lower disk, disk # 1. Here in step 9, you will need to remove Encoder #3 position and add Encoder #1 position to the plot set-up.

13. Remember to switch off the system when you are done with your experiments.

Analysis

1. Determine the damped natural frequency (in rad/sec) of the upper disk from the plot obtained for the upper disk when it is loaded with the 4 masses by choosing several consecutive cycles (say 5 to 10) in the amplitude range between 100 and 1000 counts (much smaller amplitude responses become dominated by nonlinear friction effects and do not reflect the salient system dynamics). This damped frequency for the loaded upper disk, ω_{dUl} , is related to the natural frequency for the loaded upper disk, ω_{nUl} , according to Eqn. 4.6, where the subscript 'Ul' denotes upper loaded disk.

2. Measure the reduction from the initial cycle amplitude X_0 to the last cycle amplitude X_n for the n cycles measured above. Using relationships associated with the logarithmic decrement and damping ratio, find the damping ratio ζ_{Uu} and the natural frequency ω_{nUl} .

3. Determine the damped frequency ωdUu for the unloaded upper disk from the corresponding unloaded upper disk response of the Experimental Procedure. Repeat steps 1 and 2 of the Analysis to calculate the damping ratio ζ_{Uu} and the natural frequency ω_{nUl} of the unloaded upper disk, where the 'Uu' subscript denotes unloaded upper disk.

4. Obtain $\omega_{nLl}, \omega_{nLu}, \zeta_{Ll}$ and ζ_{Lu} from the plots of the lower disk response for the loaded and unloaded cases respectively, where the 'Ll' and 'Lu' subscripts denote loaded and unloaded disk cases respectively for the lower disk. How do these damping ratios compare with that for the upper disk?

5. Use the following information pertaining to each mass piece to calculate the portion of each disk's inertia attributable to the masses for the 'Ul' (upper disk - loaded) and 'Ll' (lower disk - loaded) cases.
   - Mass (including bolt and nut): 500g (± 5g)
   - Diameter: 5.00 cm (± 0.02 cm)
   - Calling the inertia from these masses about the shaft centerline as J_m , use the following relationships to solve for the unloaded upper disk inertia J_U , and upper torsional shaft spring k_U .
   - $\omega_{n_{Ul}}^2 = \frac{k_U}{J_m + J_U}$ and
   - $\omega_{n_{Uu}}^2 = \frac{k_U}{J_U}$. Using ζ_{Uu} and ω_{nUu} , find the damping coefficient c_U with Eqn. 4.8.

6. Repeat the procedure to find the lower unloaded disk inertia (J_L), spring constant of the lower torsional shaft (k_L) and the damping of the lower disk (c_L). (Take the inertia contribution of the motor, belt, and pulleys to be $J_{motor-belt-pulleys}$ = 0.0005 kg.m2).

7. Compare the damping ratios of the loaded and unloaded cases of upper and lower disks and report your observations.

Part B: Controller Design

This part of the experiment involves the second phase – development of PD collocated control on the torsional mechanism. This is done by using the parameters determined from Part A to model the system and determining proportional and derivative gains for the controller using MATLAB based on the given design specifications.

The configuration of the torsional mechanism that will be used for controller design and evaluation is shown in Fig. 4.4.

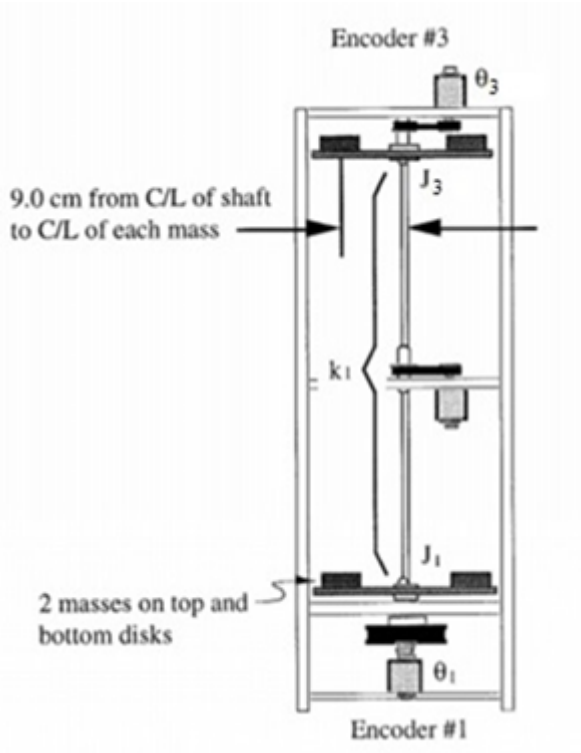


Figure 4.4. Configuration of the torsional mechanism used for controller design and evaluation

In this configuration, the center disk is removed and two masses are added to both top and bottom disks.

Theoretical Background

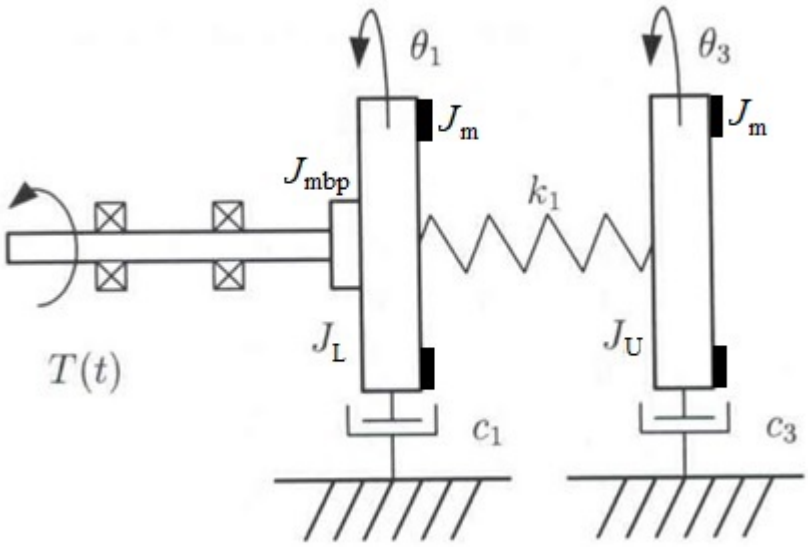


Figure 4.5. Free-free two DOF torsional plant

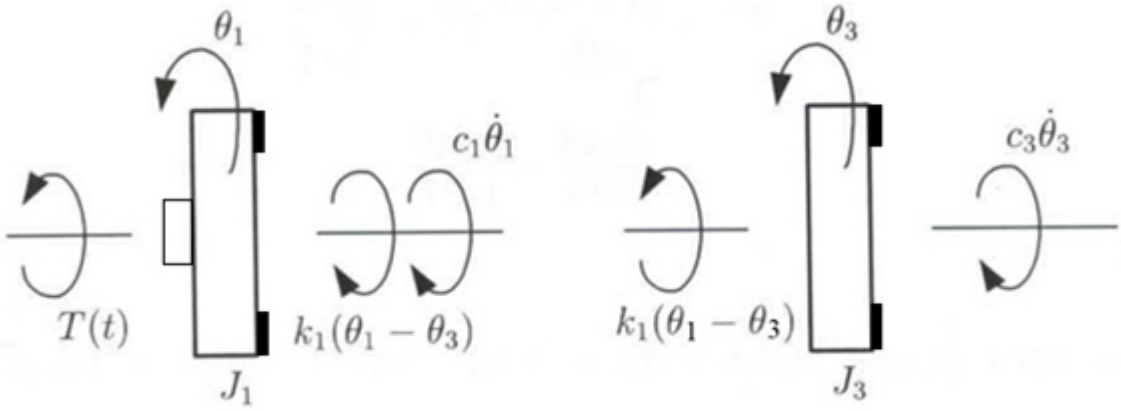


Figure 4.6. Free body diagram of the lower disk (left) and upper disk (right)

Equations of Motion

This section provides time and Laplace domain expressions which are useful for linear control implementation and are used in the experiments described later. The most general form of the two degree of freedom torsional system is shown in Fig. 4.5, where friction is idealized as being viscous.

It is important to note that the setup includes two small masses on each disk and the controller design will be done for this case. Hence, J_1 includes the mass moment of inertia of the lower disk (J_L), the mass moment of inertia contribution from two masses (J_m) and the mass moment of inertia of the motor-belt-pulley system (J_{mbp}), i.e., $J_1 = J_L + J_m + J_{mbp}$. Similarly, J3 includes the mass moment of inertia of the upper disk (J_U) and the mass moment of inertia contribution from two masses (j_m), i.e., $J_3 = J_U + J_m$. Additionally, c_1 is taken as c_{Lu} while c_3 is taken as c_{Uu} .

> In Part A, J_m represented the total mass moment of inertia contribution from four masses (added to each disk). In Part B and Part C, there is a contribution of mass moment of inertia from only two masses, so in this case, J_m (as calculated in Part A) should be suitably modified and will be lower than earlier. ai

Using the free body diagram of the 2 DOF free-free case, shown in Fig. 4.6, and summing torques acting on J_1 , one obtains

$$J_1\ddot{\theta}_1 + c_1\dot{\theta}_1 + k_1\theta_1 - k_1\theta_3 = T(t)$$

(4.9)

where:

- $T(t) = k_{VT}$ V
- V = applied voltage
- k_{VT} , gain that converts volts to torque in N-m, is calculated as follows (see Fig. 4.7):

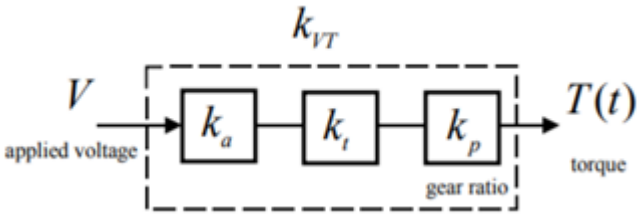


Figure 4.7. Computing the gain kVT

where:

- k_a , the Servo Amp gain = 2 amp/V
- k_t , the Servo Motor Torque constant = 0.1 N-m/amp
- k_p , the Drive Pulley ratio = 3 (N-m @ disk/N-m @ Motor)

Thus, substituting these values in (4.10), we obtain k_{VT} = 0.6 N-m/V. Similarly, summing torques acting on J_3

$$J_3\ddot{\theta}_3 + c_3\dot{\theta}_3 + k_1\theta_3 - k_1\theta_1 = 0$$

(4.11)

These may be expressed in a state space realization as

$$\dot{x} = AxBT(t)$$

where

$$x = \begin{bmatrix} \theta_1 \\ \theta_3 \\ \dot{\theta}_1 \\ \dot{\theta}_3 \end{bmatrix}, \qquad A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{J_1} & \frac{k_1}{J_1} & -\frac{c_1}{J_1} & 0 \\ \frac{k_1}{J_3} & -\frac{k_1}{J_3} & 0 & -\frac{c_3}{J_3} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{J_1} \\ 0 \end{bmatrix}$$

(4.12)

Taking the Laplace transfer of the above equations and assuming zero initial conditions, we may solve for the transfer functions

$$\frac{\theta_1(s)}{T(s)} = \frac{(J_3 s^2 + c_3 s + k_1)}{D(s)}$$

(4.13)

$$\frac{\theta)3(s)}{T(s)} = \frac{k_1}{D(s)}$$

(4.14)

where

$$D(s) = J_1 J_3 s^4 (c_1 J_3 + c_3 J_1) s^3 + (J_1 k_1 + J_3 k_1 + c_1 c_3) s^2 + (c_1 k_1 + c_3 k_1) s$$

(4.15)

In Figs. 4.5 and 4.6, k_1 is the resultant of the stiffness of the lower (k_L) and the upper (k_U) shaft in series. The values for the stiffness of the lower and the upper shaft are the stiffness parameters that you identified in Part A. Then knowing that these shafts are in series, use the formula

$$k_1 = \frac{k_L k_U}{k_L + k_U}$$

to obtain the total stiffness. The block diagram of the closed-loop system with the PD controller is shown in Fig. 4.8. The gain k_{hw} is the hardware gain and is given by

$$k_{hw} = k_c k_a k_t k_p k_e k_s$$

where

- k_c , the DAC gain = 10V / 32,768 DAC counts
- k_a , the Servo Amp gain = approx. 2 (amp/V)

- k_t , the Servo Motor Torque constant = approx. 0.1 (N-m/amp)
- k_p , the Drive Pulley ratio = 3 (N-m @ Disc / N-m @ Motor)
- k_e , the Encoder gain = 16,000 pulses / 2π radians
- k_s , the Controller Software gain = 32 (controller counts / encoder or ref input counts)

The gains of the PD controller K_p and K_d are the free parameters that must be chosen to achieve specific performance objectives (rise time, overshoot, etc.). The plant transfer function

$$G_p(s) = \frac{n(s)}{d(s)} = \frac{\theta_1(s)}{T(s)}$$

where θ_1 is the lower disc displacement and T is the input torque at the motor.

In this experiment, we consider PD control of a 2-disk system where the controlled output, θ_1 , is of the lower disk. Such a scheme is referred to as collocated since the sensor output is rigidly coupled to the actuator input.

The addition of the spring and second inertia increases the plant order by two and adds an oscillatory mode to the plant dynamics. This may be thought of, in a sense, as a dynamic disturbance to the rigid body plant. The collocated PD control implemented here is the approach most commonly used in industry. It may be practically employed when there is flexibility between the actuator and some inertia, and the location of objective control being near the actuator. If the location of objective control is at the distant inertia, however, this method has its limitations.

The approach in this experiment will be to design the controller by interactively changing the PD gains and observing their effect on the physical system.

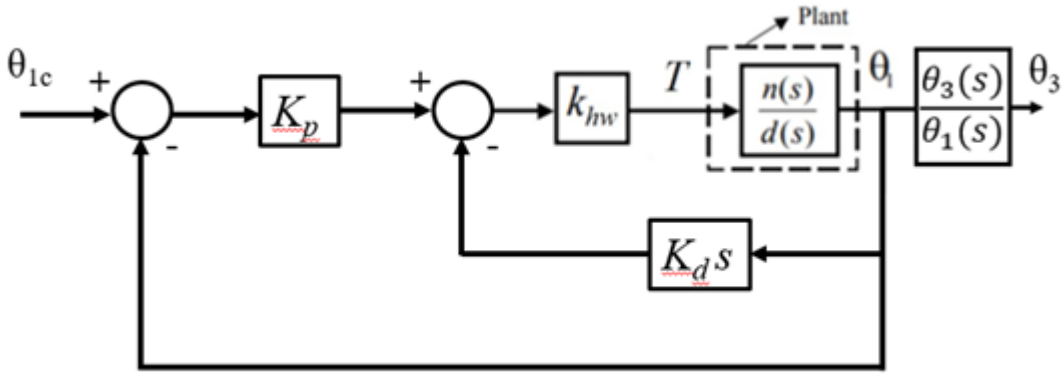


Figure 4.8. Closed-loop block diagram of the torsional plant with a PD controller - collocated control

Specifications & Procedure

Specifications

1. All closed-loop poles must be in the left half of the complex plane.
2. The rise time to a unit step command input must be less than 0.4 sec (for θ_1).
3. The overshoot in response (for θ_1) to a unit step command input must be less than 10% without excessive oscillation.
4. K_p must be less than 1.0 and K_d must be greater than 0.02 but less than 0.2.

Procedure

1. Open a new m-file and compute the transfer function $\theta_1(s)T/(s)$ using Eqn. 4.13. Save your m-file.
2. Use **rltool** in MATLAB and select appropriate values for Kp and Kd to meet the given set of specifications. Start with an initial guess for the gains and adjust. You need to meet the requirements in **rltool**. Save the **rltool** figures and your gains. Make sure to use the consistent controller architecture in the **rltool**. This will be known as high gain controller.

3. Compute the transfer function $\theta_3(s)/\theta_1(s)$ using Eqns. 4.13 and 4.14.

4. Develop a SIMULINK diagram using the block diagram shown in Fig. 4.8 as reference and save it.

5. Obtain the SIMULINK response of θ_1 to a unit step command using the set of high gains (from Step 2) you have selected. Save the response, and make sure it proves you meet the requirements.

6. Using the gains from the high gain controller as starting points in SIMULINK, iteratively reduce gains, until you obtain a well-behaved step response of θ_3 with ≤ 10% overshoot (without excessive oscillation) and as fast a rise time as possible. Save these gains as a different set, which will correspond to a low gain controller and will be tested in Part C.

7. Obtain the SIMULINK response of θ_1 to a unit step command using the set of low gains (from Step 6) and compare it with the response from Step 5. Are the design specifications still satisfied?

8. The TAs will need to see: Gains, rltool plots, Simulink diagram, and Simulink response to a step command (proving peak and settling time requirements are met).

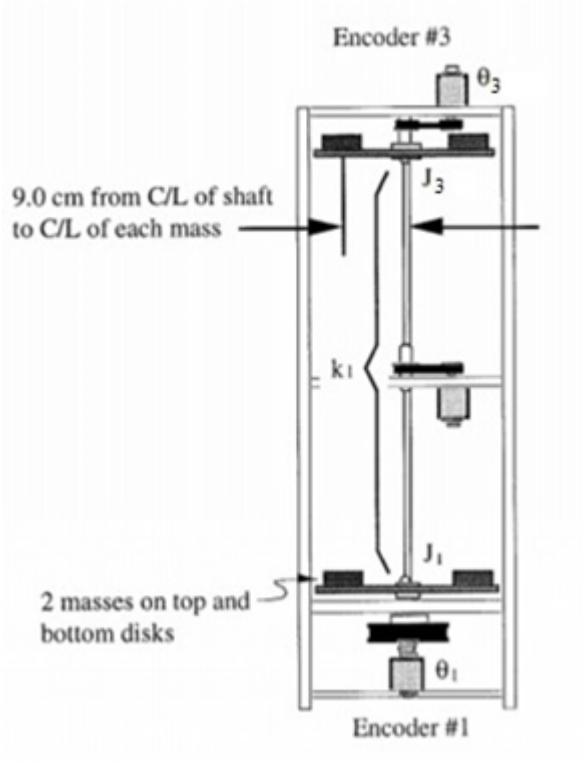
> ⓘ You must have your work checked out by one of the TAs before leaving the lab to get credit for your work.

Part C: Controller Implementation & Evaluation

Implementation

1. Set up the system with two masses on the upper and lower disk as shown in Fig. 4.9. Ensure that the two masses are located along the hub split line of the disks. Observe that the middle disk has been removed.

2. Enter the program by clicking on shortcut to 3D Torsion on the desktop and turn on the torsional mechanism.



3. Enter **File** menu, choose **Load Setting** and select the file **C:\Program Files (x86)\ECP Systems\cn\default.cfg**.

4. Choose the correct personality file by going to **Utility** menu and clicking on **Download Controller Personality File**. Download the file **C:\Program Files (x86)\ECPSystems\cn\M205di6.pmc**.

5. Enter the **Control Algorithm** box under **Setup** and ensure that the sample period, Ts, is 0.00442 seconds and select **Continuous Time Control**. Select **PI + Velocity Feedback** (this is the return path derivative form) and **Setup Algorithm**. Enter the high gain values Kp and Kd determined earlier (Ki = 0) and select OK. Check the values with a TA before entering them. Do not input magnitude of Kp > 1.0 nor Kd > 0.2 and Kd < 0.02. Ts is located both on the control algorithm screen and in the program itself, so be sure that both values are correct. The algorithm can now be implemented by selecting **Implement Algorithm,** then OK.

6. First displace the lower disk with a light, non-sharp object (e.g., a plastic ruler) to verify stability prior to touching plant. Similarly displace the upper disk and observe the response. Note the difference in their stiffness.

Evaluation

1. Go to the **Data** menu and click on **Setup Data Acquisition**. Check that data is gathered every 5 servo cycles. Be sure that the Commanded Position, Encoder 1 Position and Encoder 3 Position are all located in the **Selected Items** box. If they are not, add them to that list by selecting them in the **Possible Choices** box and then clicking on the **Add Item** button. When finished, hit **OK** to exit this menu.

2. Prepare the input for the system by selecting the **Command** menu and then **Trajectory**. Select the **Step -> Setup -> Closed loop Step Input**. Enter the following parameters for this case: step size, 1000 counts; dwell time, 5000 ms; 1 repetition. When finished, hit **OK**, then **OK** again to leave the setup trajectory menus.

3. Go to the **Command** menu and click **Execute**. Then click the **Run** button. The input trajectory will be run on the torsional mechanism now. When the box on the screen says **Upload Complete**, click on the **OK** button.

4. After data collection has finished, go to the **Plotting** menu and select **Setup Plot**. Set Command Position and Encoder 1 Position on the left axis. Click on the **Plot Data** button when finished. This will generate a plot of the data from the executed trajectory. If desired, zooming the plot can be accomplished by going to the **Plotting** menu and selecting **Axis Scaling**. Similarly obtain the plot of Encoder 3 position.

5. Save the data by clicking on the **Data** menu and going to **Export Raw Data**.

6. Repeat Steps 5-11 using the low gain controller values determined in Part B. How does the physical stiffness of the setup compare with the high gain controller?

7. Remember to SWITCH OFF the system when you are done with your experiments.

Analysis

1. Compare the steady state error values of the high gain controller as well as low gain controller response of θ_1 obtained in the experiment and explain any difference.

2. Simulate the system response for the same step command you have used in the lab experiment. Compare your simulated responses with the corresponding experimental results for both θ_1 and θ_3 . What is the rise time and % overshoot in the experiment and simulation responses to the step command used (only for θ_1)? Explain the cause for any difference.

3. Using rltool, determine the gain margin (GM) and phase margin (PM) of the closed loop system of the controller for the high gain controller as well as the low gain controller.

2DOF Helicopter

Introduction

The Quanser Aero Experiment can be configured as a conventional dual-rotor helicopter, as shown in Figure 1. The front rotor that is horizontal to the ground predominantly affects the motion about the pitch axis while the back or tail rotor mainly affects the motion about the yaw axis (about the shaft).

Fig. 1: Quanser Aero Experiment

The tail rotor in helicopters is also known as the anti-torque rotor because it is used to reduce the torque that the main rotor generates about the yaw. Without this, the helicopter would be difficult to stabilize about the yaw axis. Because the rotors on the Quanser Aero Experiment are the same size and equidistant from each other, the tail rotor also generates a torque about the pitch axis. As a result, both the front and back/tail rotors generate a torque on each other.

Background

Equations of Motion

The free-body diagram of the Quanser Aero Experiment is illustrated in Figure 2.

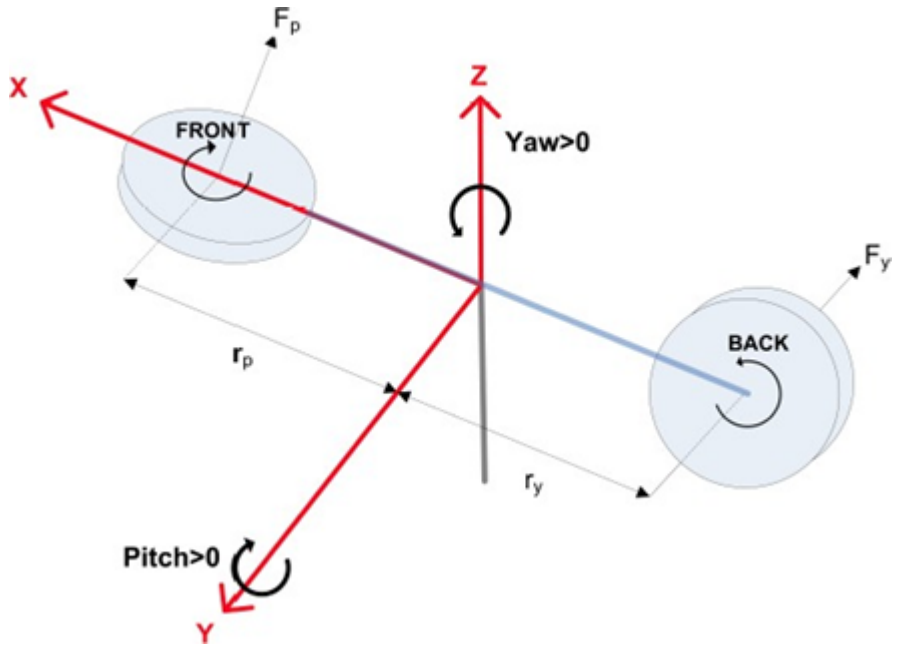


Fig. 2: Simple free-body diagram of Quanser Aero Experiment

The following conventions are used for the modeling:

- The helicopter is horizontal and parallel with the ground when the pitch angle is zero, i.e. $\theta = 0$.
- The pitch angle increases positively, $\dot{\theta}(t) > 0$, when the front rotor is moved upwards and the body rotates counter-clockwise (CCW) about the Y axis.
- The yaw angle increases positively, $\dot{\psi}(t) > 0$, when the body rotates counter-clockwise (CCW) about
  the Z axis.
- Pitch increases, $\dot{\theta} > 0$, when the front rotor voltage is positive $V_\theta > 0$.
- Yaw increases, $\dot{\psi} > 0$, when the back (or tail) rotor voltage is positive, $V_\psi > 0$.

When voltage is applied to the pitch motor, V_θ , the speed of rotation results in a force, F_θ , that acts normal to the body at a distance r_θ from the pitch axis. The rotation of the propeller generates a torque about the pitch rotor motor shaft which is in turn seen about the yaw axis. Thus rotating the pitch propeller does not only cause motion about the pitch axis but also about the yaw axis. As described earlier, that is why conventional helicopters include a tail, or anti-torque, rotor to compensate for the torque generated about the yaw axis by the large, main rotor.

Similarly, the yaw motor causes a force F_ψ that acts on the body at a distance r_ψ from the yaw axis as well as a torque about the pitch axis.

The equations of motion can be approximated after neglecting reaction torques from the rotors:

$$J_\theta \ddot{\theta} + D_\theta \dot{\theta} + K_\theta \theta = \tau_\theta$$

(1)

$$J_\psi \ddot{\psi} + D_\psi \dot{\psi} = \tau_\psi$$

(2)

where the torques acting on the pitch and yaw axes are

$$\begin{aligned} \tau_\theta &= K_{\theta\theta} V_\theta + K_{\theta\psi} V_\psi \\ \tau_\psi &= K_{\psi\theta} V_\theta V_\theta + K_{\psi\psi} V_\psi \end{aligned}$$

(3)

The parameters used in the EOMs above are:

- J_θ : the total moment of inertia about the pitch axis
- D_θ : the damping about the pitch axis
- K_θ : the stiffness about the pitch axis
- K_m : torque thrust gain from the pitch rotor
- V_θ : voltage applied to the pitch rotor
- V_ψ : voltage applied to the yaw rotor motor

Similarly, the total moment of inertia and damping about the yaw axis is J_ψ and D_ψ , respectively.

The total moment of inertia acting about the pitch and yaw axes are

$$J_\theta = J_{body} + 2J_{prop}$$

(4)

$$J_\psi = J_{body} + 2J_{prop} + J_{yoke}$$

(5)

Expressing the rotor as a single-point mass, the inertia acting about the pitch or yaw axis from a single
rotor is $J_{prop} = m_{prop} r^2$. Modeling the helicopter body as a cylinder rotating about its center, the inertia
is $J = m_{body} \frac{L_{body}^2}{12}$. Finally the forked yoke that rotates about the yaw axis can be approximated as
cylinder rotating about its center as well and expressed as $J_{yoke} = m_{yoke} \frac{r_{yoke}^2}{2}$.
Evaluating the moment of inertia using the parameters listed in the Quanser Aero Experiment User Manual gives:

$$J_\theta = 0.0219 \; kg \cdot m^2$$
$$J_\psi = 0.0220 \; kg \cdot m^2$$

First-Order Response

The step response of a first-order transfer function

$$Y(s) = \frac{K}{\tau s + 1} U(s)$$

(6)

where K is the DC or steady-state gain and τ is the time constant is illustrated in Figure 3. This is for a
system with K = 1 and τ = 0.05.

To obtain the time constant from the response, find the time it takes to reach $1 - e^{-1}$ or 63% of its
final steady-state value:

$$y(t_1) = y_1 = (1 - e^{-1})(y_{ss} - y_0)$$

The time constant is $\tau = t_1 - t_0$, where t_0 is the start time of the step and t_1 is the time it takes to reach

63% of the final value, as illustrated in Figure 3.

Second-Order Response

The free-oscillatory equation of motion of a second-order system described by

$$J\ddot{\alpha} + D\dot{\alpha} + K\alpha = 0$$

(7)

is shown in Figure 4. Assuming the initial conditions $\alpha(0-) = \alpha_0$, the Laplace transform of Equation 7
is

$$\alpha(s) = \frac{\alpha_0/J}{s^2+D/Js+K/J}$$

(8)

The prototype second-order equation is defined

$$s^2 + 2\zeta\omega_n s + \omega_n^2$$

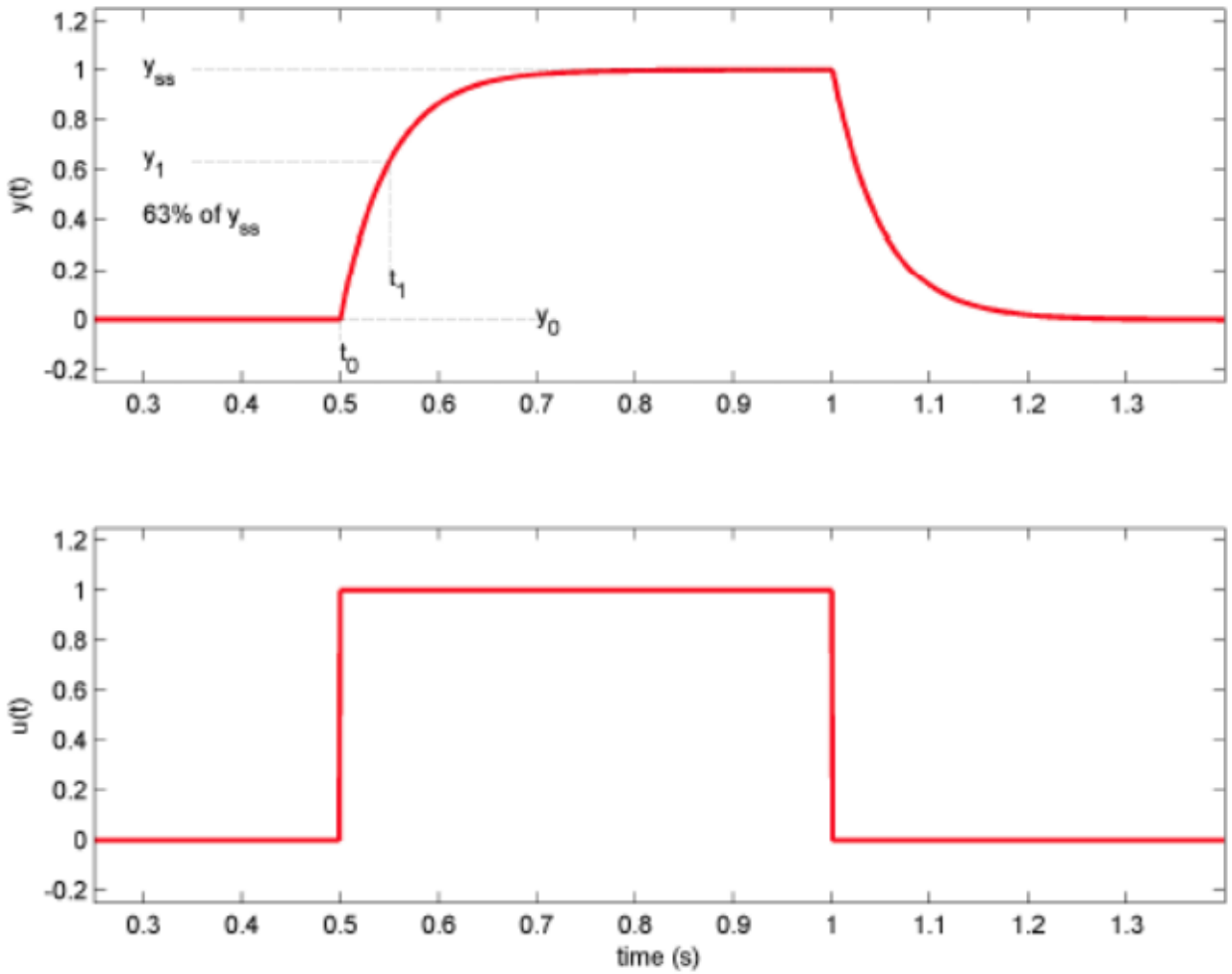


Fig. 3: Free Oscillation Response

where ζ is the damping ratio and ω_n is the natural frequency. Equating the characteristic equation in

Equation (8) to this gives

$$\omega_n^2 = \frac{K}{J}$$
$$2\zeta\omega_n = \frac{D}{J}$$

Finding the Natural Frequency

The period of the oscillations in a system response can be found using the equation

$$T_{osc} = \frac{t_n - t_1}{n-1}$$

(9)

where t_n is the time of the n^{th} oscillation, t_1 is the time of the first peak, and n is the number of
oscillations considered. From this, the damped natural frequency (in radians per second) is

$$\omega_d = \frac{2\pi}{T_{osc}}$$

(10)

and the undamped natural frequency is

$$\omega_n = \frac{\omega_d}{\sqrt{1-\zeta^2}}$$

(11)

Finding the Damping Ratio

The damping ratio of a second-order system can be found from its response. For a typical second-order
underdamped system, the subsidence ratio (i.e. decrement ratio) is defined as

$$\delta = \frac{1}{n-1} \ln \frac{O_1}{O_n} \ ,$$

(12)

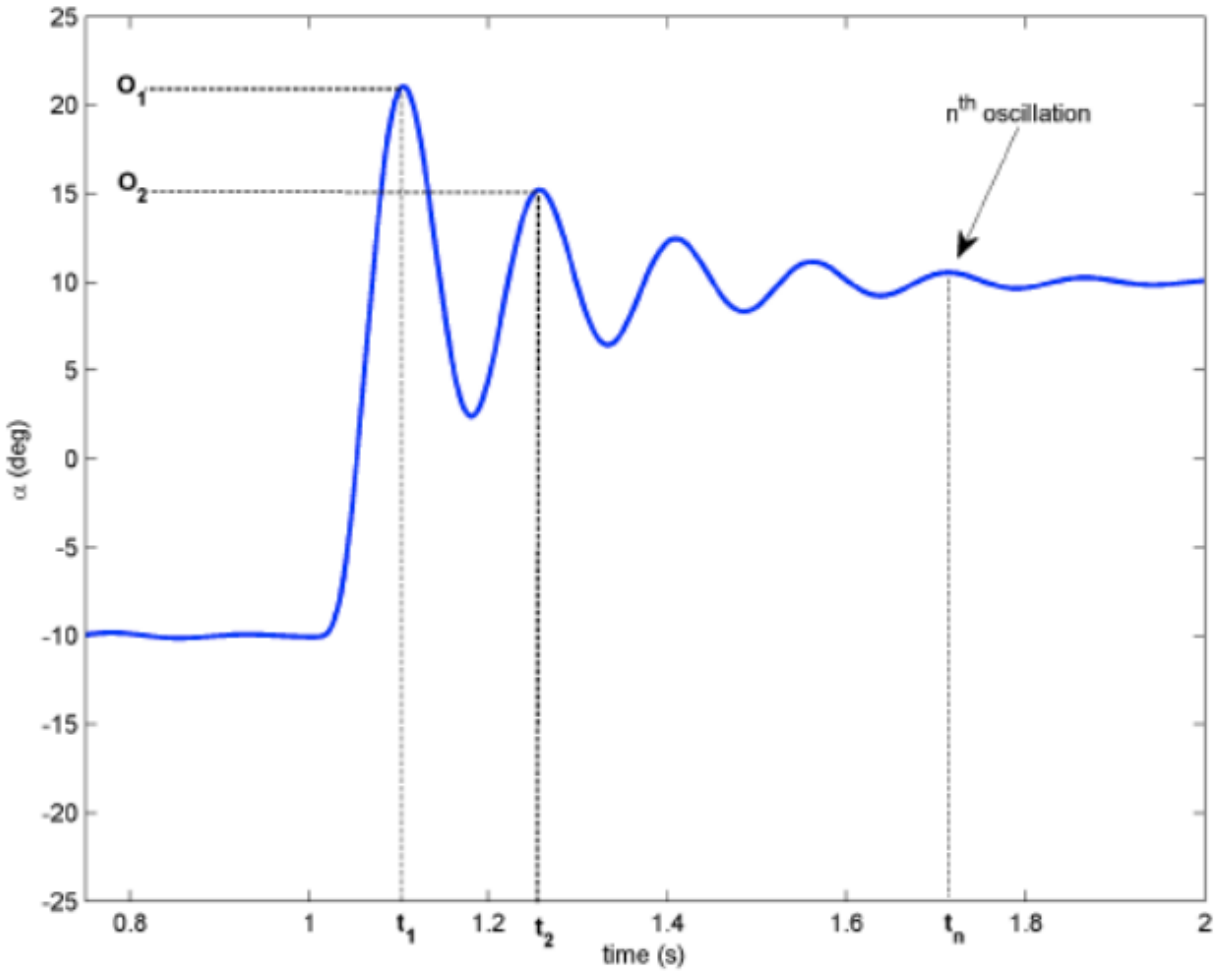


Fig. 4: Free Oscillation Response

where O_1 is the peak of the first oscillation and O_n is the peak of the n^{th} oscillation. Note that $O_1 > O_n$,
as this is a decaying response. The damping ratio can then be found using

$$\zeta = \frac{\delta}{\sqrt{4\pi^2+\delta^2}} \tag{13}$$

Estimating the Viscous Damping Coefficients

The viscous damping coefficients acting about the pitch and yaw axes, D_θ and D_ψ in Equation (1)
and Equation (2), can be found from the free-oscillation response. The free-oscillation response about the

pitch and about the yaw are different, however.

Pitch Axis: By locking the yaw axis (using the Allen key supplied), this allows us to focus on the
1 DOF pitch-only system. Apply a short step voltage to mimic an impulse and get the free-oscillation
response of the pitch. Remark that the impulse response is second-order free-oscillation response. The
resulting 1 DOF pitch-only equations of motion is

$$J_\theta \ddot{\theta}(t) + D_\theta \dot{\theta} + K_\theta \theta = 0$$

(14)

Taking its Laplace transform gives

$$J_\theta \left(\Theta(s) s^2 - \theta(0^-) - \dot{\theta}(0^-) \right) + D_\theta \left(\Theta(s) - \theta(0^-) \right) + K_\theta \Theta(s) = 0$$

(15)

Assuming the initial velocity is zero, $\dot{\theta}(0^-) = 0$, and solving for position we get

$$\Theta(s) = \frac{J_\theta}{J_\theta s^2 + D_\theta s + K_\theta} \theta(0^-) = \frac{J_\theta D_\theta}{s^2 + D_\theta / J_\theta + K_\theta / J_\theta} \theta(0^-)$$

(16)

The pitch free-oscillation transfer function matches the prototype second-order transfer function in Equation (8) Based on the measured damping ratio and natural frequency of the response, the friction (or
stiffness) of the system is

$$K_\theta = J_\theta \omega_n^2$$

(17)

and the viscous damping is

$$D_\theta = 2\zeta \omega_n J_\theta$$

(18)

Yaw Axis: The 1 DOF yaw-only equations of motion is

$$J_\psi \ddot{\psi} + D_\psi \ddot{\psi} = 0$$

In terms of angular rate, the equation becomes

$$J_\psi \dot{\omega}_\psi(t) + D_\psi \omega_\psi(t) = 0 \tag{19}$$

where $\omega_\psi(t) = \dot{\psi}(t)$. Taking its Laplace transform

$$J_\theta(\Omega_\psi(s)s - \omega_\psi(0^-)) + D_\psi \Omega_\psi(s) = 0$$

and solving for the speed we get

$$\Omega_\psi(s) = \frac{J_\psi}{J_\psi s + D_\psi}\omega_\psi(0^-) = \frac{J_\psi/D_\psi}{J_\psi/D_\psi s + 1}\omega_\psi(0^-)$$

The yaw free-oscillation transfer function matches the prototype first-order transfer function in Equation (6). Based on the measured time constant of the response, its damping can be found with

$$D_\psi = \frac{J_\psi}{\tau} \tag{20}$$

Estimating the Thrust Parameters

By locking the yaw axis, this allows us to focus on the 1 DOF pitch-only system, i.e., eliminating any
motions introduced in the yaw axis when applying a voltage to the pitch rotor. The equations of motion
for the 1 DOF actuated system is

$$J_\theta \ddot{\theta} + D_\theta \dot{\theta} + K_\theta \theta = K_{\theta\theta} V_\theta$$

(21)

Solving for the thrust gain we get

$$K_{\theta\theta} = \frac{J_\theta \ddot{\theta} + D_\theta \dot{\theta} + K_\theta \theta}{V_\theta}$$

(22)

Remark that this is the thrust torque gain parameter. To force thrust gain would be $K_{\theta\theta}/r_p$, where r_p is the distance between the helicopter pivot and the center of the pitch rotor.

Similarly, to find the thrust gain acting on the yaw axis only system, lock the pitch axis, and apply a voltage to the tail rotor. This system is represented by

$$J_\psi \ddot{\psi} + D_\psi \dot{\psi} = K_{\psi\psi} V_\psi$$

(23)

or,

$$J_\psi \dot{\omega}_\psi + D_\psi \omega_\psi = K_{\psi\psi} V_\psi$$

(24)

where $\omega_\psi = \dot{\psi}$ is the angular rate of the yaw axis. The yaw torque thrust gain is

$$K_{\psi\psi} = \frac{J_\psi \dot{\omega}_\psi + D_\psi \omega_\psi}{V_\psi}$$

(25)

The cross-torque thrust parameters, $K_{\theta\psi}$ and $K_{\psi\theta}$ in Equation (3), represent the coupling between the axes. To find the cross-torque acting on the pitch axis from a torque applied to the tail rotor, unlock both pitch and yaw axes such that it is free to move in 2 DOF, apply a voltage to the tail rotor, and examine the response of the pitch. The equations representing these dynamic, when $V_\theta = 0$, are

$$J_\theta \ddot{\theta} + D_\theta \dot{\theta} + K_\theta \theta = K_{\theta\psi} V_\psi$$

(26)

Putting this in terms of angular rate, $\omega_p = \dot{\theta}$, and solving for the gain we get

$$K_{\theta\psi} = \frac{J_\theta \dot{\omega}_\theta + D_\theta \omega_p}{V_\psi}$$

(27)

Similarly, to identify the cross-torque gain parameter that is generated about the yaw axis from a pitch
torque (i.e. voltage applied to the front rotor), we have the equation

$$J_\psi \ddot{\psi} + D_\psi \dot{\psi} = K_{\psi\theta} V_\theta$$

(28)

and the gain can be found using

$$K_{\psi\theta} = \frac{J_\psi \dot{\omega}_\psi + D_\psi \omega_p si}{V_\theta}$$

(29)

Part A: System Identification

Experimental Steps for Finding System Parameters

1) Lock the yaw axis to enable motions about the pitch axis only

2) Open the *C : \AE4610_Controls_Lab \ Quanser \ q_aero_free_osc_response_pitch*

3) Select a -20 V impulse

4) Select simulation time 30 sec.

5) Build the model

6) Press **Connect to Target** and **Run**

7) Copy *aero_pitch_free_osc_rsp.mat* to your folder

8) Close the Simulink model. DO NOT SAVE THE CHANGE

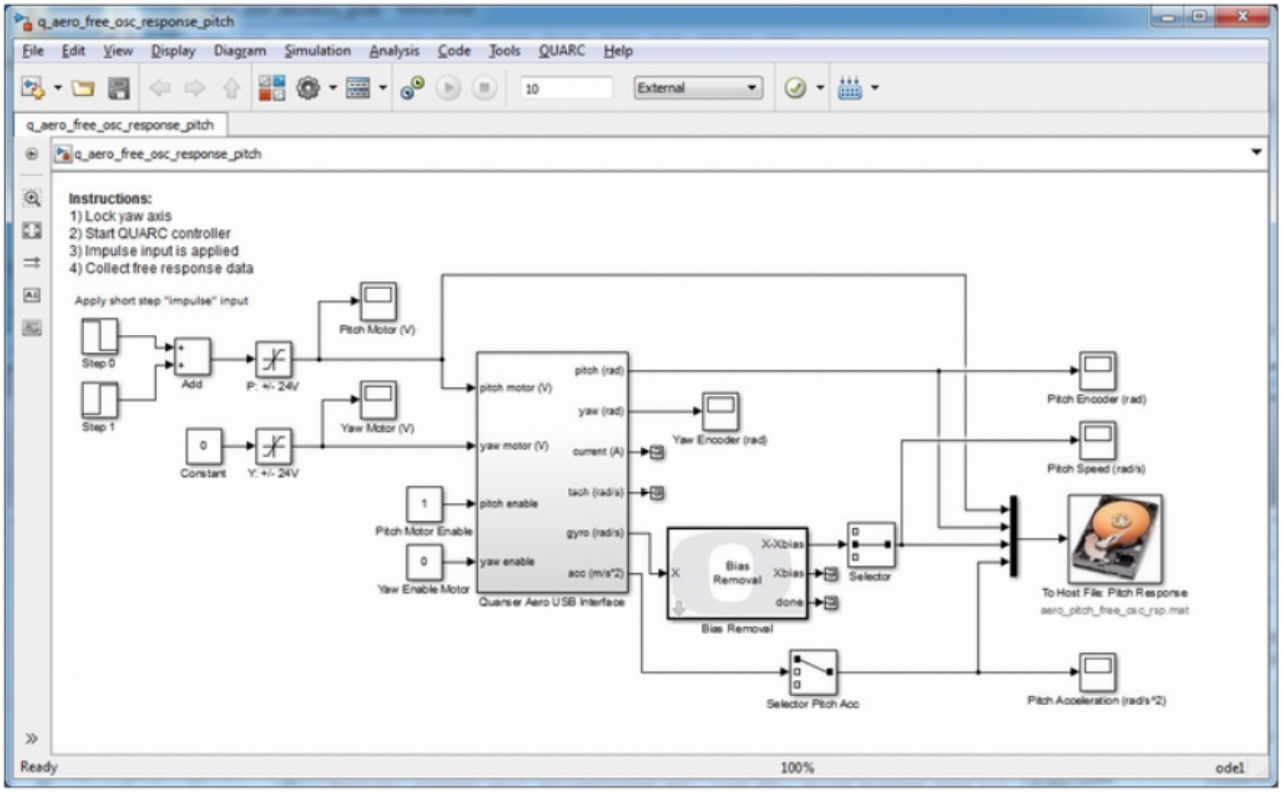


Fig. 5: Model used to acquire free-oscillation response about pitch

9) Open the *C : \AE4610_Controls_Lab \ Quanser \ q_aero_step_response_pitch*

10) Select a step voltage 10V

11) Select simulation time 30 sec.

12) Build the model

13) Press **Connect to Target** and **Run**

14) Copy *aero_pitch_step_rsp.mat* to your folder

15) Close the Simulink model. DO NOT SAVE THE CHANGE.

16) Unlock the yaw axis

17) Lock the pitch axis

18) Open the *C : \AE4610_Controls_Lab \ q_aero_free_osc_response_yaw*

19) Apply an impulse of 20V to the tail rotor

20) Select simulation time 5 sec.

21) Build the model

22) Press **Connect to Target** and **Run**

23) Copy *aero_yaw_free_osc_rsp.mat* to your folder

24) Close the Simulink model. DO NOT SAVE THE CHANGE.

25) Open the *C : \AE4610_Controls_Lab \ Quanser \ q_aero_step_response_yaw*

26) Select a step voltage 15V

27) Select simulation time 60 sec.

28) Build the model

29) Press **Connect to Target** and **Run**

30) Copy *aero_yaw_step_rsp.mat* to your folder

31) Close the Simulink model. DO NOT SAVE THE CHANGE.

32) Unlock both the pitch and yaw axes to enable the full 2 DOF motion

33) Open the *C : \AE4610_Controls_Lab \ Quanser \ q_aero_step_response_pitch_from_yaw*

34) Select a step voltage 12V

35) Select simulation time 40 sec.

36) Build the model

37) Press **Connect to Target** and **Run**

38) Copy *aero_pitch_from_yaw_step_rsp.mat* to your folder

39) Close the Simulink model. DO NOT SAVE THE CHANGE.

40) Open the *C : \AE4610_Controls_Lab \ Quanser \ q_aero_step_response_yaw_from_pitch*

41) Select a step voltage 12V

42) Select simulation time 60 sec.

43) Build the model

44) Press **Connect to Target** and **Run**

45) Copy *aero_yaw_from_pitch_step_rsp.mat* to your folder

46) Close the Simulink model. DO NOT SAVE THE CHANGE.

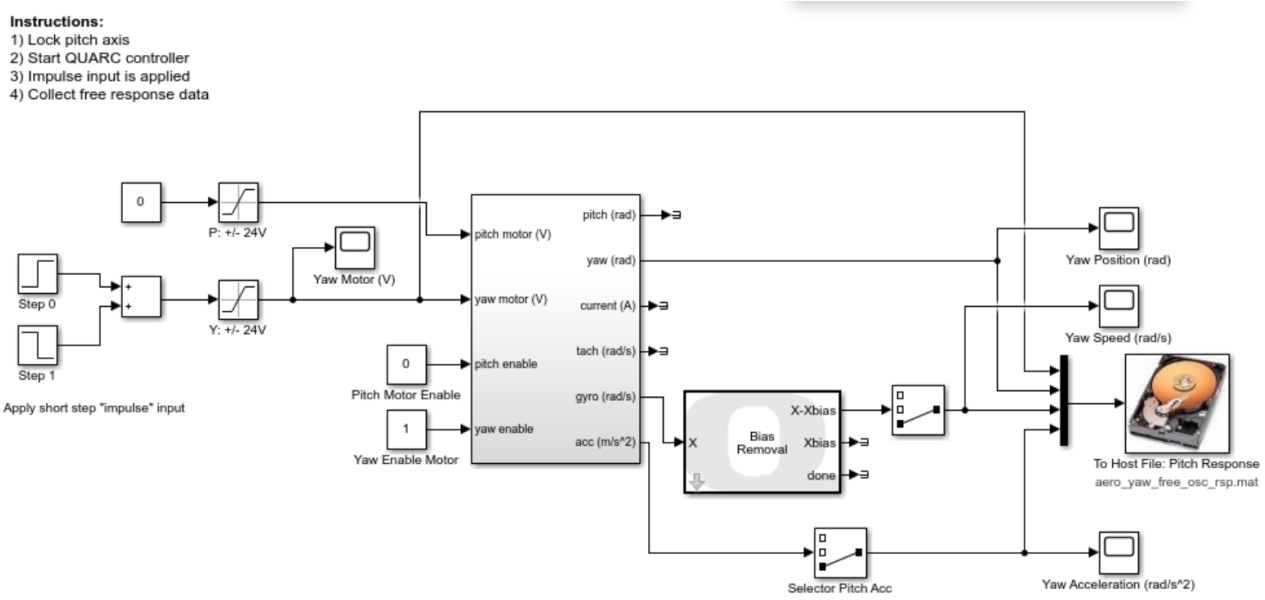


Fig. 6: Yaw Pulse Response

Analysis

1. Find the natural frequency and damping ratio from the impulse response of pitch axis. Use Equations (9) - (13). Then find the stiffness and viscous damping coefficient about the pitch axis by using Equation (17) and (18).

2. Find the time constant from yaw pulse response and obtain the daming D_ψ by using Equation (20).

3. Find the thrust gain parameters $K_{\theta\theta}, K_{\psi\psi}, K_{\theta\psi}, K_{\psi\theta}$ from steady state of step response by using Equations (22), (25), (27), and (29).

4. Build your own simulink model and mimic the experiment. Compare the simulation results with the experiment results.

5. How can we improve the estimation procedure?

Part B: Controller Design

Background

Linear State-Space Representation

Given the linear state-space equations: $\dot{x} = Ax + Bu$ and $y = Cx + Du$, we define the state for the
Quanser Aero Experiment as

$$x^T = \begin{bmatrix} \theta(t), & \psi(t), & \dot{\theta}(t), & \dot{\psi}(t) \end{bmatrix}$$

(30)

the output vector as

$$y^T = \begin{bmatrix} \theta(t), & \psi(t) \end{bmatrix}$$

(31)

and the control variables as

$$u^T = \begin{bmatrix} \theta(t), & \psi(\theta) \end{bmatrix}$$

(32)

where θ and ψ are the pitch and yaw angles, respectively, and V_θ and V_ψ are the motor voltages applied to the pitch and yaw rotors (i.e. the main and tail rotors). Using the equations of motion in Equation (1) and (2), the state-space matrices are

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -K_{\psi\theta}/J_\theta & 0 & -D_\theta/J_\theta & 0 \\ 0 & 0 & 0 & D_\psi/J_\psi \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_{\theta\theta}/J_\theta & K_{\theta\psi}/J_\theta \\ K_{\psi\theta} & K_{\psi\psi}/J_\psi \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

(33)

In this section a state-feedback controller is designed to regulate the pitch and yaw angles of the Quanser Aero Experiment to desired angles. By using the state-space model, we can find a control gain K based on the coupled dynamics to stabilize this system. The control gains are computed using the Linear-Quadratic Regulator (LQR) algorithm. The general state-feedback control is illustrated in Figure 7. The state-feedback controller is defined

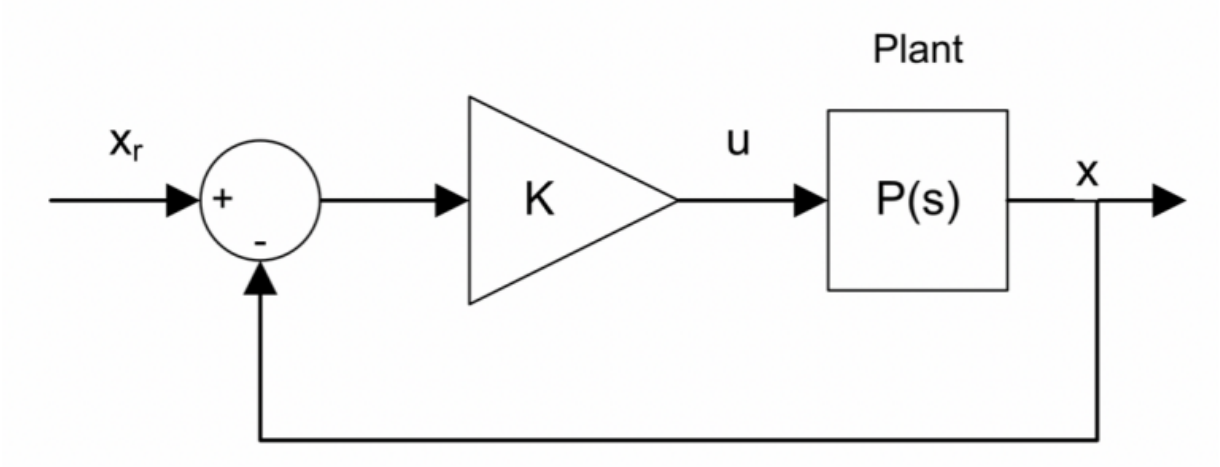


Fig. 7: Model used to acquire free-oscillation response about pitch

$$u = K(x_d - x),$$

(34)

where x is the state defined in Equation (30)

$$x_d^T = \begin{bmatrix} \theta_d, & \psi_d, & 0, & 0 \end{bmatrix}$$

(35)

is the reference command (or setpoint) state with the desired pitch and yaw angles, θ_d and ψ_d , and

$$u^T = \begin{bmatrix} V_\theta & V_\psi \end{bmatrix}$$

(36)

is the control input where Vθ is the front/pitch motor voltage and Vψ is the tail/yaw motor voltage.

Linear Quadratic Regulator (LQR) optimization can be used for finding the control gain parameters

of the Quanser Aero Experiment flight control. Given the state-space representation, the LQR algorithm

computes a control law u to minimize the performance criterion or cost function

$$J = \int_0^\infty (x_{ref} - x(t))^T Q(x_{ref} - x(t)) + u(t)^T R u(t) dt.$$

(37)

The design matrices Q and R hold the penalties on the deviations of state variables from their setpoint and

the control actions, respectively. When an element of Q is increased, therefore, the cost function increases

the penalty associated with any deviations from the desired setpoint of that state variable, and thus the

specific control gain will be larger. When the values of the R matrix are increased, a larger penalty is

applied to the aggressiveness of the control action, and the control gains are uniformly decreased.

Since there are four states , $Q \in \mathbb{R}^{4\times 4}$, and two control variables, $R \in \mathbb{R}^{2\times 2}$. The setpoint, x_d is given

above the control strategy used to minimize cost function J is thus given by

$$u = K(x_d - x) = k_{p,\theta}(\theta_d - \theta) + k_{p,\psi}(\psi_d - \psi) - k_{d,\theta}\dot{\theta} - k_{d,\psi}\dot{\psi}$$

(38)

Designing an LQR Controller

Various control software already have LQR optimization routines that can be used to generate the state feedback control gain K . In order for the closed-loop response to satisfy certain time-domain specifications,

the closed-loop system is typically simulated using its dynamic model, in software, first. This is an iterative

process. By adjusting the weighting matrices Q and R and then running the simulation, we can find a control that satisfies the user?s requirements. Further, we must ensure that the control signal u is smooth
(i.e. does not chatter) and does not surpass the limits of the actuator.

LQR Control Design and Simulation

LQR is used to find the state-feedback control gain K that will stabilize the Quanser Aero Experiment to the user's desired pitch and yaw angles. Our desired closed-loop response should match the following specifications.

Desired closed-loop response specifications for pitch

1. Steady-state error: pitch e_{ss} ≤ 2 deg, yaw e_{ss} ≤ 2 deg.
2. Peak time: t_p ≤ 2 s.
3. Percent Overshoot: PO ≤ 7.5%.
4. No actuator saturation: $|V_\psi|$ ≤ 24V and $|V_\theta|$ ≤ 24V .

The state-space matrices derived in Equation (33) are entered in the State-Space block in Simulink and the control gain is set to the Matlab variable K.

Running the closed-loop state-feedback LQR simulation

1. Using parameters from Part A, obtain state-space matrices of the system neglecting coupling effect.
2. Design Q, R , and K .
3. Build the Simulink model of the system
4. Simulate the closed-loop response of the system with pitch only command: $\theta_d = 10\ deg$.

5. Simulate the closed-loop response of the system with yaw only command: $\psi_d = 45\ deg$.
6. Simulate the closed-loop response of the system with pitch and yaw command: $\theta_d = 10\ deg$ and $\psi_d = 45\ deg$.
7. Check whether your controller meets the desired specifications.
8. If it fails to meet the specifications, tune Q to meet the specifications
9. Save the data and plot the closed-loop system response for pitch and yaw
10. Now, use system matrices with coupling effect and use the controller without coupling
11. Repeat step 4-6
12. The closed-loop response in this step does not have to meet the desired specifications
13. Now, design Q, R , and K with coupling effect
14. Build the Simulink model of the system
15. Repeat step 4-6
16. Check whether your controller meets the desired specifications.
17. If it fails to meet the specifications, tune Q to meet the specifications
18. Save the data and plot the closed-loop system response for pitch and yaw

Checkoff List

- [ ] Results from part A
- [ ] System matrices neglecting coupling effect
- [ ] System response of the system neglecting coupling effect (pitch & yaw)
- [ ] System matrices with coupling effect

- [ ] System response of the system with coupling effect (pitch & yaw)

Analysis

1. What happened when you neglect the coupling effect?
2. Is there any systematic way to design Q and R ?

Part C: Controller Implementation

In this section the state-feedback control is implemented on the Quanser Aero Experiment using theq_aero_2dof_lqr_control Simulink diagram shown in Figure 8 with QUARC.

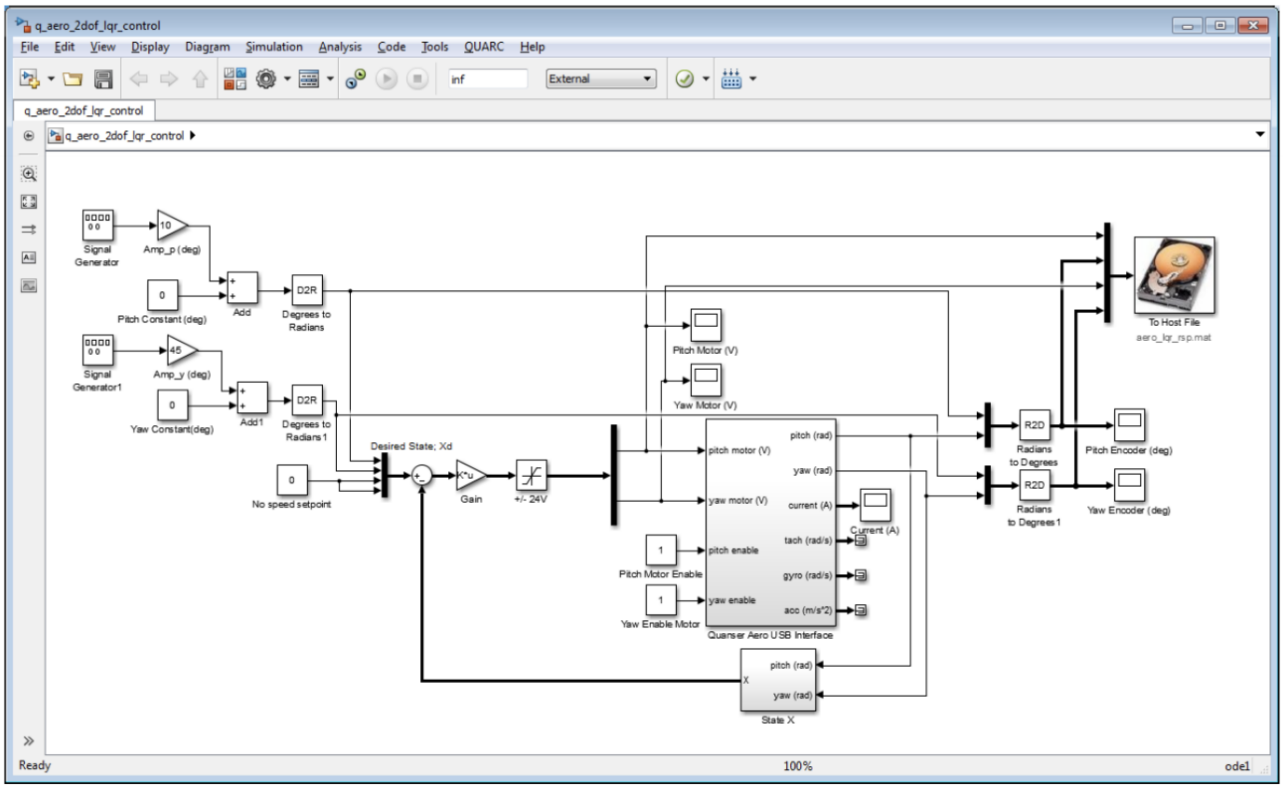


Fig. 8: Simulink model used to run LQR controller

Desired Closed-Loop Response Specifications

1. Steady-state error: pitch e_{ss} ≤ 2 deg, yaw e_{ss} ≤ 2 deg.
2. Peak time: t_p ≤ 2 s.

3. Percent Overshoot: PO ≤ 7.5%.
4. No actuator saturation: $|V_\psi|$ ≤ 24V and $|V_\theta|$ ≤ 24V .

Running the PD on the Quanser Aero Experiment

1. Unlock both pitch and yaw axes to enable the full 2 DOF motion.
2. Open *q_aero_2dof_lqr_control*
3. Use K obtained in Part B. neglecting the coupling effect.
4. Build
5. Use pitch command (10 deg) only by changing Amp_y to zero.
6. Run Simulink
7. Copy *aero_lqr_rsp.mat* to your own folder
8. Examine the obtained closed-loop response and see if it matches the desired specifications
9. Use yaw command (45 deg) only by changing Amp_p to zero.
10. Run Simulink
11. Copy *aero_lqr_rsp.mat* to your own folder
12. Examine the obtained closed-loop response and see if it matches the desired specifications
13. Use both pitch and yaw command
14. Run Simulink
15. Copy *aero_lqr_rsp.mat* to your own folder
16. Examine the obtained closed-loop response and see if it matches the desired specifications
17. Use K obtained in Part B. with the coupling effect.
18. Repeat step 4-16
19. Close the Simulink. DO NOT SAVE THE CHANGE!

Analysis Question

1. Did your controller successfully meet the specs? If not, why?
2. Compare the result of Part C to the simulation result of Part B. Discuss what caused the difference.
3. How does the coupling effect affects the performance of the controller?
4. How can we improve the controller?

3DOF Helicopter

Objective

The purpose of this experiment is to design an attitude controller for the 3-DOF helicopter mechanism and to test the controller using the real time experimental set up.

Equipment Required

- [ ] 3-DOF helicopter mechanism
- [ ] Q8-USB interface board
- [ ] MATLAB, SIMULINK and QUARC software
- [ ] Power module for the PCI Multi-Q board and the helicopter mechanism

Image credit: Lehigh University

Part A: Modeling

Description of a 3-D Helicopter Mechanism

The 3-DOF helicopter mechanism used in this experiment is shown in Fig. 5.1. The 3-DOF helicopter consists of a base upon which an arm is mounted as shown in Fig. 5.2. The arm carries the helicopter body at one end and a counterweight at the other. It can pitch about a

longitudinal axis as well as yaw about a vertical axis. Encoders mounted on these axes allow for measuring the pitch and the yaw of the arm. The helicopter body is mounted at the end of the arm as shown in Fig. 5.3. The helicopter body is free to roll about the arm. The roll angle is measured via a third encoder. Two motors with propellers mounted on the helicopter body can generate a force proportional to the voltage applied to the motors.

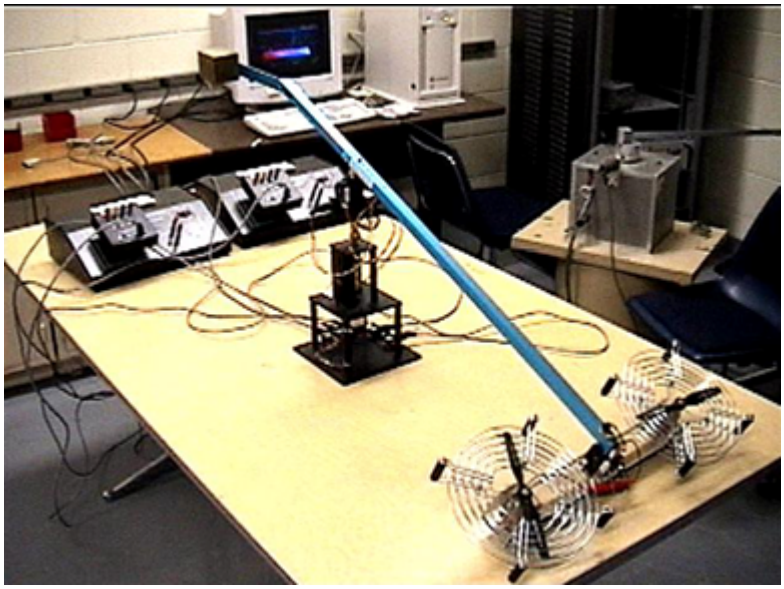


Figure 5.1: 3-D helicopter mechanism in the lab

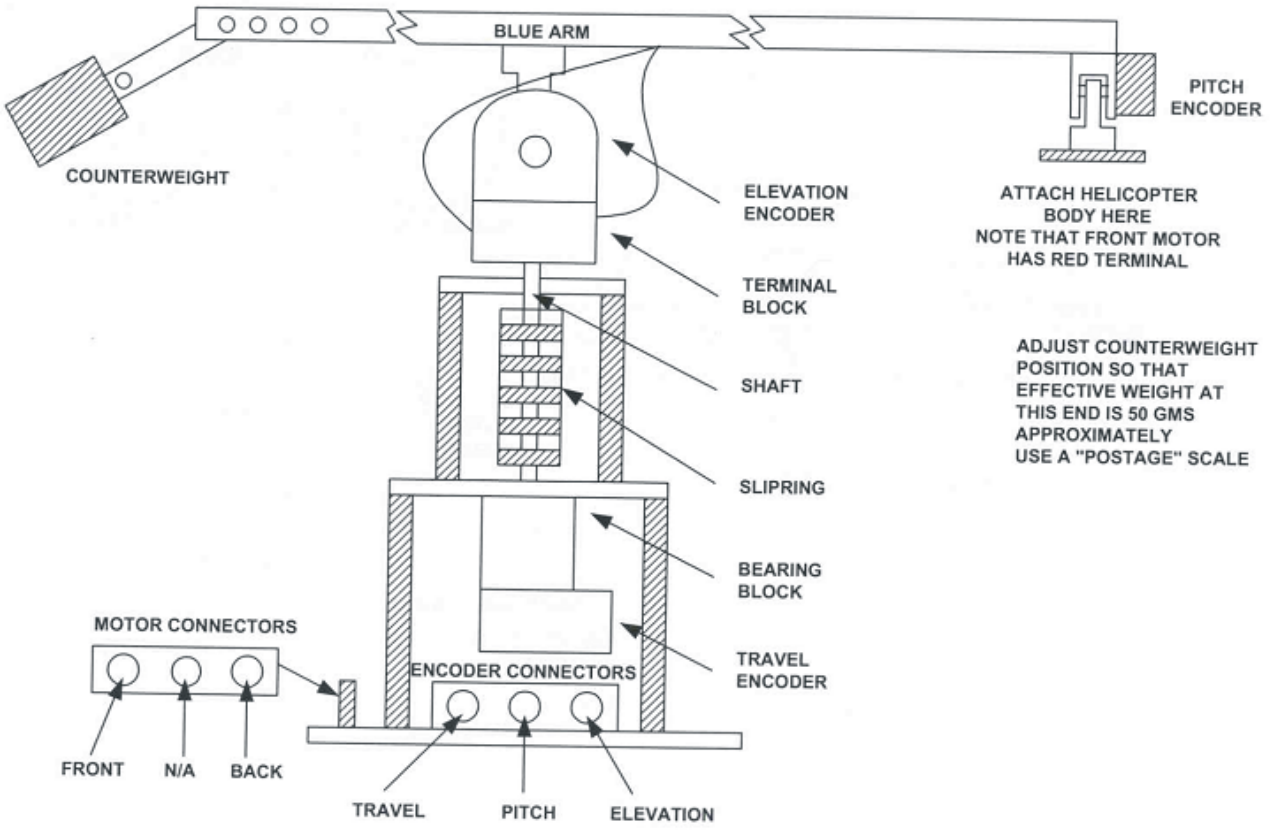


Figure 5.2: Schematic of 3-DOF helicopter mechanism

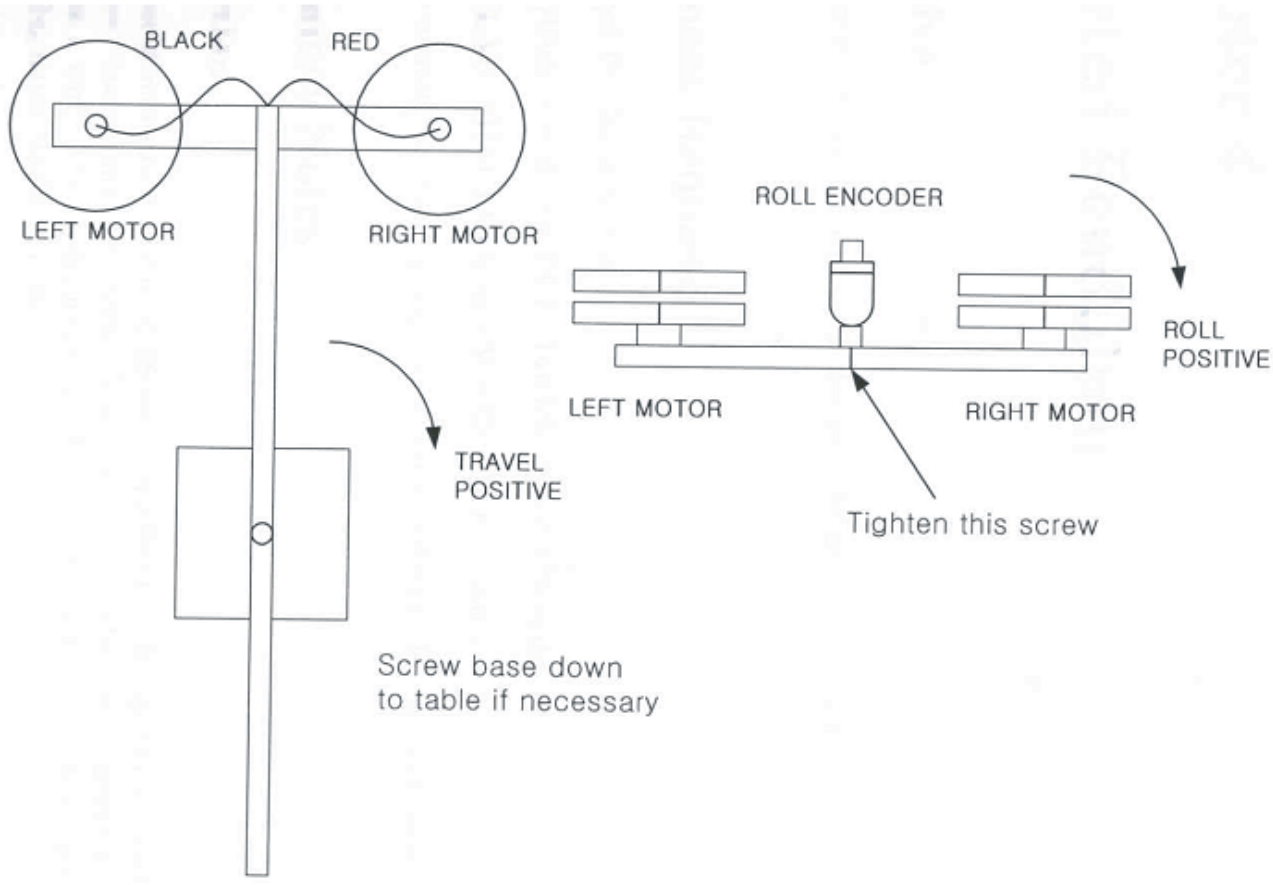


Figure 5.3: Main parts of the 3-DOF helicopter mechanism (looking from the back and top)

The force generated by the propellers can cause the helicopter body to lift off the ground. The purpose of the counterweight is to reduce the power requirements on the motors. All electrical signals to and from the arm are transmitted via a slip-ring with 8 contacts thus eliminating the possibility of tangled wires and reducing the amount of friction and loading about the moving axes.

Mathematical Model

Pitch

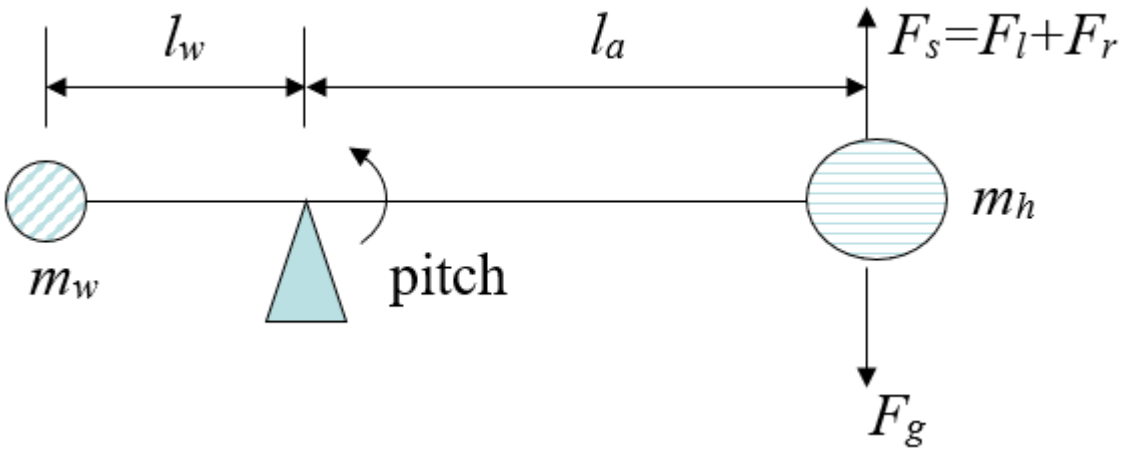


Figure 5.4: Pitching moment calculation

Consider the diagram in Fig. 5.4. For a zero roll angle, the pitching torque is controlled by the forces generated by the two propellers (i.e. $F_l + F_r$). The body will not rise until the total force $F_s = F_l + F_r$ is greater than the mass differential force F_g

$$F_g = \frac{l_a m_h g - l_w m_w g}{l_a}$$

(5.1)

A sketch of the configuration in 3-D motion is shown in Fig. 5.5. Once the body is in the air with a roll angle (ϕ), the equation for pitch (θ) is given by

$$
\begin{aligned}
\mathrm{J}_{\theta}\ddot{\theta} &= l_a \mathrm{F_s} - \mathrm{l_a F_g} = l_a\left(F_l + F_r\right)\cos\phi - l_a \mathrm{F_g} \\
\Rightarrow \mathrm{J}_{\theta}\ddot{\theta} &= K_f l_a\left(V_r + V_l\right)\cos\phi - T_g \\
&= K_f l_a V_s \cos\phi - T_g
\end{aligned}
$$

(5.2)

where,

- J_θ : moment of inertia of the body about the pitching axis
- K_f : force constant of the motor/propeller combination
- l_a : distance from the pivot point (longitudinal axis) to the propellers
- l_w : distance from the pivot point (longitudinal axis) to the counterweight
- θ : pitch angle measured in radians (positive in the direction indicated in Fig. 5.4)
- ϕ : roll angle (see Fig. 5.5)
- V_r : voltage applied to the right motor resulting in force F_r
- V_l : voltage applied to the left motor resulting in force F_l
- T_g : Mass differential torque = $l_a F_g$

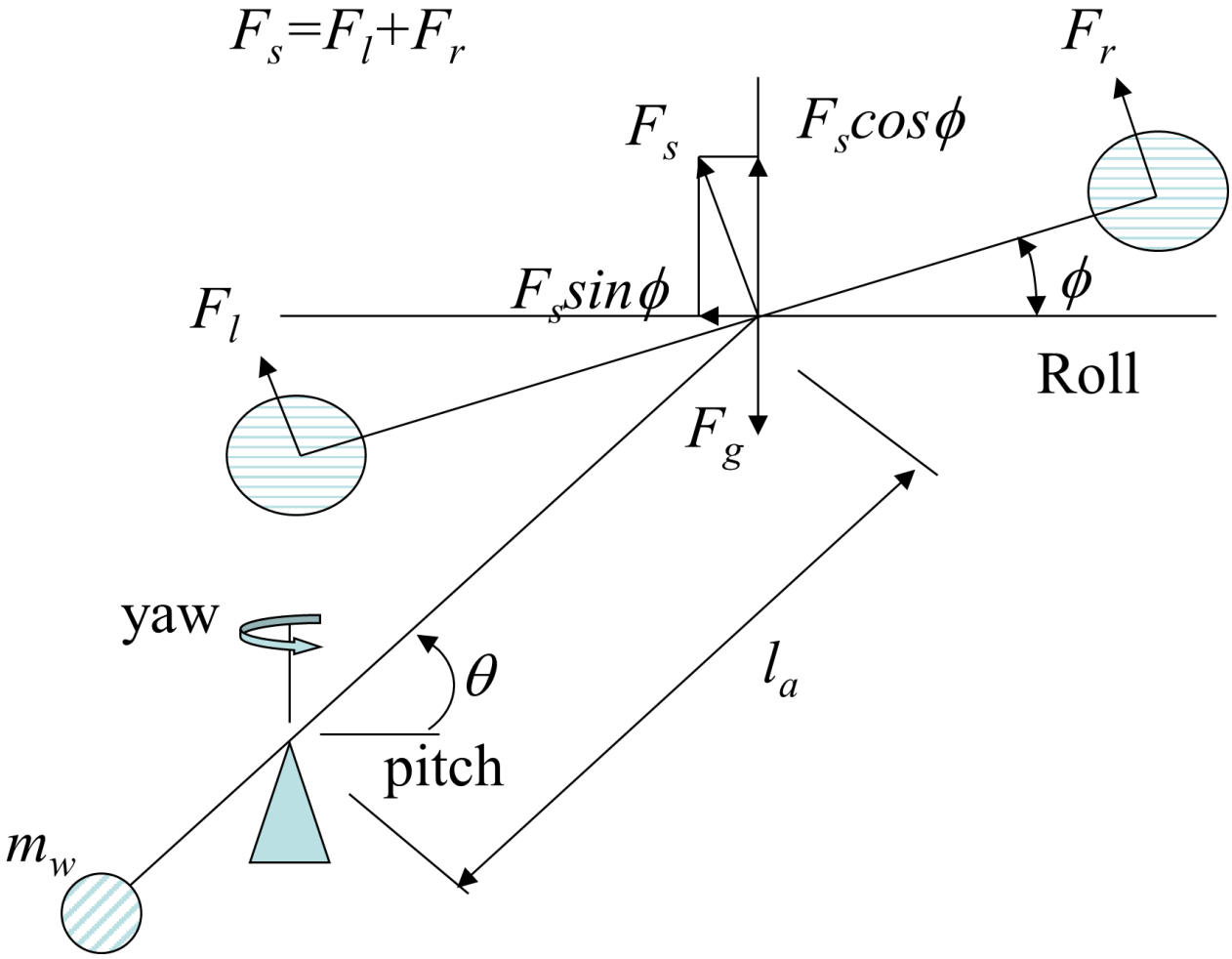


Figure 5.5. Sketch of the configuration in 3-D motion.

Roll

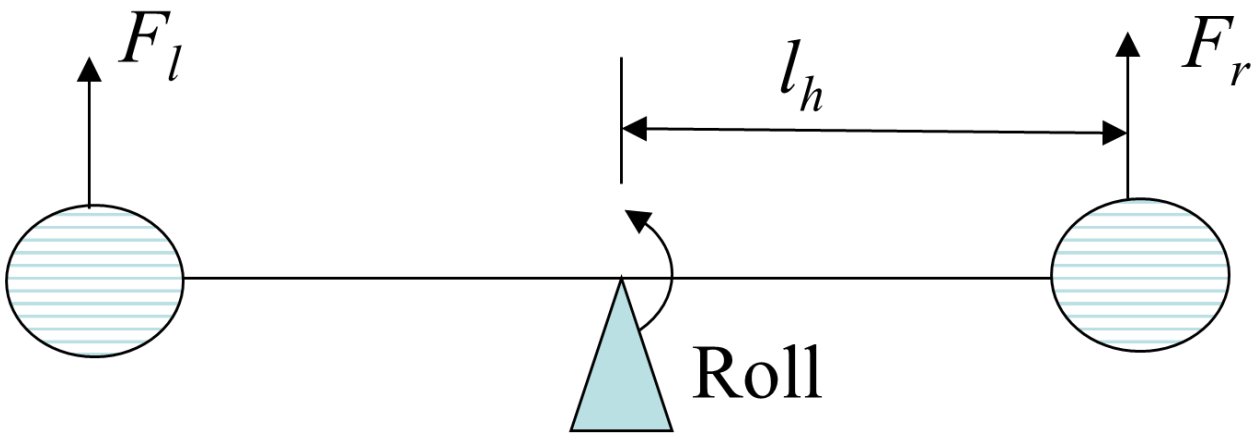


Figure 5.6: Rolling moment calculation (as seen looking from the front)

Consider the diagram in Fig. 5.6. The rolling moment is controlled by the difference of the forces generated by the propellers. If the force generated by the right motor is higher than the force generated by the left motor, the helicopter body will roll to left (positive).

$$J_\phi \ddot{\phi} = F_r l_h - F_l l_h$$

(5.3)

$$= K_f l_h (V_r - V_l) = K_f l_h V_d$$

where

- J_ϕ : mass moment of inertia of the body about the roll axis.
- l_h : distance from the roll axis to either motor.
- ϕ : roll angle measured in radians.

Yaw

The only way to apply a force in the lateral direction is to roll the helicopter body. Assume that the body rolled to left (positive) by an angle ϕ (see Fig. 5.5). The horizontal component of the sum of the rotor forces will cause a torque about the yaw axis which results in a positive yaw acceleration.

$$\begin{aligned} J_\psi \dot{r} &= (F_r + F_l)\sin(\phi) l_a \\ &= K_f l_a (V_l + V_r)\sin(\phi) \\ &= K_f l_a V_s \sin(\phi) \end{aligned}$$

(5.4)

where,

- J_ψ : moment of inertia of the body about the yaw axis.
- $\dot{r}$: yaw rate in rad/sec.

Equations (5.2), (5.3) and (5.4) mean that the pitching acceleration is a function of the sum of the voltages applied to the motors, the rolling acceleration is a function of the difference in the voltages applied to the motors, and the yaw acceleration is a function of the roll angle.

Part B: Controller Design

Pitch Controller

The purpose is to design a controller to control the pitch angle of the helicopter by supplying the appropriate sum of voltage to the motors. We will assume the mass differential torque has been balanced by a constant thrust from the propellers. Assuming roll angle to be small, we obtain the following linear system

$$J_\theta \ddot{\theta} = K_f l_a (V_r + V_l) = K_f l_a V_s$$

(5.5)

where $V_s = (V_r + V_l)$ is the sum of the voltages applied to the motors. First, design a PD controller of the form

$$V_s = -K_{\theta p}(\theta - \theta_c) - K_{\theta d}\dot{\theta}$$

(5.6)

where θ_c is the commanded pitch angle. With this controller, the closed-loop transfer function for pitch is

$$\frac{\theta(s)}{\theta_c(s)} = \frac{K_f l_a K_{\theta p}}{s^2 J_\theta + K_f l_a K_{\theta d} s + K_f l_a k_{\theta p}} = \frac{(K_f l_a K_{\theta p}/J_\theta)}{s^2 + (K_f l_a K_{\theta d}/J_\theta)s + (K_f l_a K_{\theta p}/J_\theta)}$$

(5.7)

The standard form for the transfer function of a second order system is

$$\text{Standard 2nd order T.F. form} = \frac{\omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

(5.8)

Comparing Eq. (5.7) with Eq. (5.8), and for desired values of closed loop pitching motion natural frequency ($\omega_{n\theta}$) and damping ratio (ς_θ), we obtain

$$K_f l_a K_{\theta p}/J_\theta = \omega_{n\theta}^2 \quad \rightarrow \quad K_{\theta p} = \frac{J_\theta \omega_{n\theta}^2}{K_f l_a}$$

(5.9)

$$K_f l_a K_{\theta d}/J_\theta = 2\varsigma_\theta \omega_{n\theta} \quad \rightarrow \quad K_{\theta d} = \frac{J_\theta (2\varsigma_\theta \omega_{n\theta})}{K_f l_a}$$

(5.10)

We can also include an integral term in the controller (PID controller) to compensate for the mass differential torque T_g . (Above we assumed that the helicopter was perfectly balanced i.e. T_g = 0, but we need K_I because in the experiment it is not. Why don't we balance it to make this easier? There is a very good reason for it.)

The controller is now

$$V_s = -K_{\theta p}(\theta - \theta_c) - K_{\theta d}\dot{\theta} - K_{\theta i}\int(\theta - \theta_c)dt$$

(5.11)

We will tune the integral gain $K_{\theta i}$ using simulation of the nonlinear system with an initial guess of $K_{\theta i} = 0.01K_{\theta p}$.

Roll Controller

The purpose is to design a controller to control the roll angle of the helicopter by supplying an appropriate difference of voltage to the motors. The open-loop axis equation of motion for roll is

$$J_\phi \ddot{\phi} = K_f l_h (V_r - V_l) = K_f l_h V_d$$

(5.12)

where

$$V_d = (V_r - V_l)$$

We implement a PD controller of the form

$$V_d = -K_{\phi p}(\phi - \phi_c) - K_{\phi d}\dot{\phi}$$

(5.13)

where ϕ_c is the commanded roll angle. The resulting closed-loop transfer function is given by

$$\frac{\phi(s)}{\phi_c(s)} = \frac{K_f l_h K_{\phi p}}{s^2 J_\phi + K_f l_h K_{\phi d} s + K_f l_h K_{\phi p}}$$

(5.14)

Similar to the pitch controller, the roll controller gains are determined by comparing the closed loop system transfer function Eq. (5.14) with the standard form of the transfer function of a second order system of Eq. (5.8) to obtain

$$K_f l_h K_{\phi p}/J_\phi = \omega_{n\phi}^2 \quad \rightarrow \quad K_{\phi p} = \frac{J_\phi \omega_{n\phi}^2}{K_f l_h}$$

(5.15)

$$K_f l_h K_{\phi d}/J_\phi = 2\varsigma_\phi \omega_{n\phi} \quad \rightarrow \quad K_{\phi d} = \frac{J_\phi (2\varsigma_\phi \omega_{n\phi})}{K_f l_h}$$

(5.16)

where $\omega_{n\phi}$ and ς_ϕ are respectively the desired natural frequency and damping ratio of the closed loop rolling motion.

Yaw Controller

The purpose is to design a controller to control the **yaw rate** of the helicopter by commanding a roll angle. We will first assume that the configuration is in equilibrium, i.e., the sum of the rotor forces $F_l + F_r$ is nearly equal to the mass differential force F_g. Note that we will later relax this assumption in the nonlinear simulation. With this assumption, the **linearized** equation for the yaw dynamics is given by

$$J_\psi \dot{r} = F_g l_a \phi$$

(5.17)

In order to achieve a desired yaw attitude rate r_c , we design a PI controller that commands a desired roll angle ϕ_c based on a given *yaw rate* command, as follows:

$$\phi_c = -K_{rp}(r - r_c) - K_{ri}\int(r - r_c)dt$$

(5.18)

We assume that roll response (ϕ) follows the roll command (ϕ_c) almost instantaneously (note that this assumption has certain implication on how we design roll and yaw control loops). Assuming K_{ri} = 0, the resulting closed loop transfer function is given by

$$\frac{r(s)}{r_c(s)} = \frac{F_g l_a K_{rp}}{J\psi s + F_g l_a K_{rp}}$$

(5.19)

The standard form for the transfer function of a first order system is given by

$$\text{Standard TF of 1st Order System} = \frac{1}{\tau s + 1}$$

(5.20)

Comparing Eq. (5.19) with Eq. (5.20), and for a desired value of closed loop yaw rate response time constant, we obtain

$$\tau = \frac{J_\psi}{F_g l_a K_{rp}} \quad \rightarrow \quad K_{rp} = \frac{J_\psi}{F_g l_a \tau}$$

(5.21)

Once the proportional gain is determined, we can add the integral term ($K_{ri} = 0.01K_{rp}$) and tune the integral gain K_{ri} through nonlinear simulation of the system.

System Parameters

The following are the estimated parameter values for the 3-DOF Helicopter Mechanism.

$$J_\theta = 0.997\ kg-m^2$$

$$J_\phi = 0.046\ kg-m^2$$

$$J_\psi = 1.097\ kg-m^2$$

$$l_a = 0.66\ m$$

$$l_h = 0.18m$$

$$K_f = 0.4\ N/volt$$

$$F_g = 1.91N$$

Procedure

1. Design a PID controller for the control of pitch attitude θ and a PD controller for the control of roll attitude using the system parameters given above. The closed loop system must meet the following specifications for a step response:
    1. In pitch: 3 sec < peak time < 3.5 sec, 5% settling time < 5 sec
    2. In roll: 1.2 sec < peak time < 1.8 sec, 5% settling time < 2 sec

    Note: For a second-order system:

    $$Peak\ time = \frac{\pi}{(\sqrt{1-\varsigma^2})\omega_n} \qquad 5\%\ Settling\ time = \frac{3}{\varsigma\omega_n}$$

    Use the following steps to design the pitch and roll controller:
    First, pick a value for the peak time and settling time that falls within the above specifications (**Note:** It is probably a good idea to pick a peak time less than the settling time). Next, using the above equations, calculate the values of the natural frequency ωn and damping ς. Use Eqs. (5.9) and (5.10) for determining the PD controller gains for the pitch controller. Similarly, use Eqs. (5.16) and (5.17) to

determine the PD controller gains for the roll controller. Initially set the integral gain of the pitch controller to a default value of. We will adjust the integral gain later using nonlinear simulation of the system. At the end of Step 1, you should have five gains.

Note: The roll controller is a PD controller, and does not have an integral gain.

2. Assuming that the roll angle tracks instantaneously, design a PI controller for control of yaw rate r. The closed loop system must meet the following specification:

   In yaw rate response: 4 sec < rise time < 5 sec

   Note: For a first order system, the rise time (tr) is related to the time constant (t) by the following relation:

   $$t_r = \tau \ln 9$$

   (5.22)
   Pay attention to the assumption above. The yaw controller works through commanding a roll angle, but for this initial design, we assume that roll angle matches this command instantaneously.

   Select a value of rise time in the specified range and use Eq. (5.22) to determine the value of time constant (τ). Then use eq. (5.21) to determine the proportional gain K_{rp} . Use a default value of for the integral gain. We will adjust the integral gain later using nonlinear simulation of the system.

3. Using Eqs. (5.5) and (5.6), build a SIMULINK model of the closed loop system for the pitch PID controller and enter the gains you found. The design that you performed gave you values that should work well, but they are only an initial guess. You must implement them and test the response to ensure it meets the requirements, and modify the gains if necessary. Run a closed loop response to a pitch attitude step command of 20 deg. Plot the response and verify that the response meets the given specifications. (**Note:** You can test the requirements with this command as well. You don't have to use a unit step for testing, because this is a linear system.)

4. Repeat the above process for roll (i.e., build a PD controller model in SIMULINK using Eqs. (5.12) and (5.13) and enter the gains you found. Test the roll controller using a roll

attitude step command of 10 deg. Plot the response and verify that the response meets the given specifications.

5. Develop the SIMULINK model for the yaw rate controller. This one is a little different, because it does not command motor voltages directly, as the others did. Instead, for a given yaw rate, the controller commands a roll angle. Assume $\phi = \phi_c$ and use Eqs. (5.17) and (5.18) to build a SIMULINK model. Plot yaw rate and yaw attitude (this is integral of yaw rate) responses to a 10 second pulse yaw rate command of 10 deg/sec and verify that the response meets the given specification. If needed adjust the gains.

6. We will now add non-linear terms into the SIMULINK model that were not included as part of the initial system. The system was linearized because a linear system is easier to analyze and design. Now, you need to include two non-linear terms, $\sin(\phi)$ and $\cos(\phi)$, in your model to make it more realistic. **Look specifically at pitch and yaw dynamics and consider how roll angle affects them. (Figure 5.5 should help you in this)**. Also, you need to include the mass differential torque as a constant input to the pitch dynamics. Further, you need to replace F_g by F_s in the yaw dynamics. The commanded roll attitude from the yaw controller becomes input to the roll controller, thus coupling the yaw and roll loops. The propeller force F_s from the pitch controller is multiplied by , and therefore $F_s \sin(\phi)$ becomes the input to the yaw dynamics. Likewise, the propeller force F_s is multiplied by $\cos(\phi)$, and therefore $F_s \cos(\phi)$ becomes the input to the pitch dynamics.

7. You need to use a unit step input for yaw rate and verify that your yaw controller meets the rise time requirement. Then, design a pulse input of 10 deg/sec to simulate a yaw rate command given by the pilot. Integrate yaw rate to obtain yaw attitude. Determine the duration of the pulse input that is needed to achieve a 180 deg turn. This can be done through trial and error, if you like. Plot the yaw rate and yaw attitude responses.

   Note: Since the pitch loop includes the constant mass differential torque, include several seconds (say 50-100 seconds) of simulation with zero commands initially so that the system reaches an equilibrium by balancing the mass differential force by the propeller force.

8. Repeat step 7 while simultaneously commanding 20 degrees of pitch attitude and a 10 deg/sec yaw rate pulse of required duration to achieve a 180 deg turn.

9. You need to complete the controller design prior to your lab. Also, you need to complete all simulations at home for inclusion in the group report. Make sure you have: all your gains listed, there should be 7 of them, the pitch angle response, the roll angle response, the yaw rate and angle response, and that all of the requirements are met for all of them, and what pulse duration was needed for the 180 degree turn for the two cases of with zero pitch attitude and 20 deg pitch attitude.

> ⓘ **Each of you must complete the controller design at home and have the gain values from your design with you when you go to the lab in order to receive lab participation credit. Also, each of you must include your controller designs as part of the lab report to get credit for your work. Each of you must complete SIMULINK responses at home and include them in the group report to get credit for your work.**

Part C: Controller Implementation and Evaluation

Running the Experiment

> ⚠ Always stay clear of the helicopter while running and catch it while stopping.

Pitch and Roll Control

1. Turn the power on.

2. Open the MATLAB and locate the file **Helicopter_PitchRoll_PID.mdl** under the path **C:\AE4610_Controls_Lab\Helicopter** and open it. This is the block diagram for this part of the experiment.

3. To build the model, choose **QUARC** and then **Build** (or press the **Build Model**  button). This generates the controller code.

4. Set the gains in the Pitch Controller and Roll Controller blocks in the Simulink diagram.

5. Double click the Pitch Command block to open it and change the gain after the Pitch(step) input to 0.

6. Double click the Roll Command block to open it and change the gain after the Roll(step) input to 0.

7. Hold the helicopter in level position before starting the experiment until the fans are in full power.

8. Press **Connect to Target**  button in the menu bar and then press **Start**  . Check if the controller can maintain the attitude of helicopter.

9. Double click on the scopes and open then.

10. Gently disturb the helicopter to see the effect of the controller.

11. Double click the Pitch Command block to open it and change the gain after the Pitch(step) input to 1 and run the experiment from level position.

12. Save the data with a descriptive name.

13. Double click the Pitch Command block to open it and change the gain after the Pitch(step) input to 0.

14. Double click the Roll Command block to open it and change the gain after the Pitch(step) input to 1 and run the experiment from level position.

15. Save the data with a descriptive name and close the Simulink model. **DO NOT SAVE THE MODEL**.

Pitch, Roll, and Yaw Control

1. Open the MATLAB and locate the file **Helicopter_PitchRollYaw_PID.mdl** under the path **C:\AE4610_Controls_Lab\Helicopter** and open it. This is the block diagram for this part of the experiment.

2. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.

3. Set the gains in the Pitch Controller, Roll Controller, and Yaw Rate Controller blocks in the Simulink diagram.

4. Double click the Pitch Command block to open it and change the gain after the Pitch(step) input to 0.

5. Double click the Roll Command block to open it and change the gain after the Roll(step) input to 0.

6. Double click the Yaw Rate Command block to open it and change the gain after Summer to 1.

7. Hold the helicopter in level position before starting the experiment until the fans are in full power.

8. Press **Connect to Target** button  in the menu bar and then press **Start**  .

9. Save the data with a descriptive name.

10. Turn the POWER OFF.

Analysis

1. Compare the actual responses with the simulated responses from Part B. Comment on the differences in the responses.

2. Give a reason why it is better to design a yaw *rate* controller instead of a *yaw attitude* controller.

3. Comment on what one would expect if one were to design the yaw controller rise time to be of the same or less than that of the roll controller.

Group Lab Report

Include the controller designs, the SIMULINK block diagrams and the responses from each member of the group clearly marked by the student's name. Include a controller evaluation section and answer the analysis questions. Make certain that for the analysis, you compare the differences between experimental and simulation responses. There should be a lot to talk about, as this experiment has a lot of variables and non-linear influences. Discuss any discrepancies you see, and any logical reasons for why they exist.

Flexible Beam

Objective

The purpose of this experiment is to understand the existence of vibrations in a rotary flexible link and the arising mode shapes. Any beam-like structure exhibits vibrations either due to external changing loads or due to reorientation via actuators. The experiment deals with the modelling and identification of modal frequencies and mode shapes of free vibrations in the rotary flexible link. This analysis is particularly useful to understand structural vibrations and modes and how to contain them in real-world applications like aircraft and spacecraft structures as well as robotic links/manipulators.

Image credit: Quanser

Equipment Required

- [ ] Flexible link plant (Quanser FLEXGAGE module)
- [ ] Power amplifier (Quanser VoltPaq-X2)
- [ ] Data acquisition board (Quanser Q2-USB)
- [ ]

- [ ] Rotary servo plant (Quanser SRV02)
- [ ] MATLAB and SIMULINK

Modelling and Vibration Analysis

This experiment involves system identification and modeling of the flexible link. The objective is to find the stiffness of the flexible link and conduct a frequency sweep across a range to determine the structural frequencies and mode shapes of the link.

Rotary Flexible Link Model

This experiment is performed using the Quanser Rotary Flexible Link mounted on SRV-02 servo motor. This system, shown in Fig. 6.1, consists of an electromechanical plant, where a flexible link is rotated using a servo motor. The base of the flexible link is mounted on the load gear of the servo motor system. The servo angle, θ, increases positively when it rotates counter-clockwise (CCW). The servo (and thus the link) turn in the CCW direction when the control voltage is positive, i.e., $V_m > 0$.

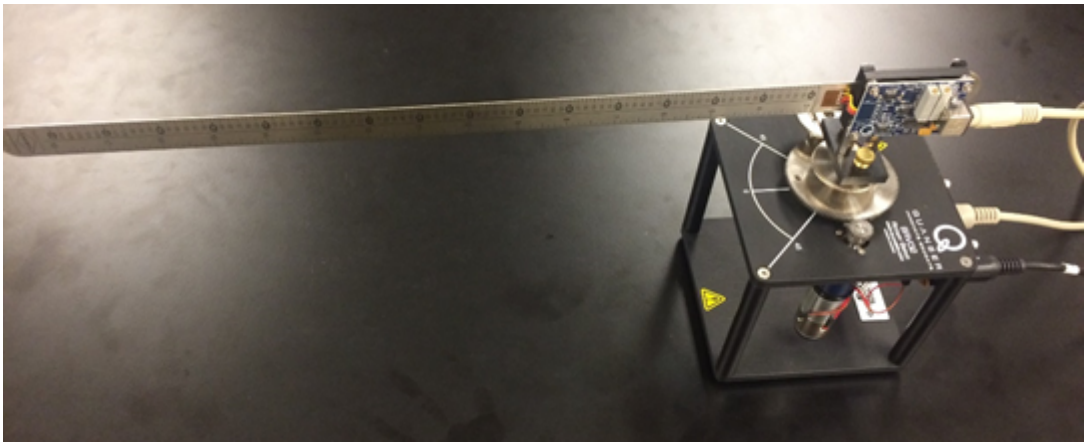


Figure 6.1. Rotary Flexible Link Setup

The main components of the setup are labelled in Figs. 6.2 and 6.3 and are listed in Table 6.1.

Table 6.1. Setup Components

No.	Component
1	SRV02 Plant (Servo motor)
2	FLEXGAGE Module
3	FLEXGAGE Link
4	Strain Gauge
5	Strain Gauge Circuit
6	Thumbscrews
7	Sensor Connector
8	OFFSET Potentiometer
9	GAIN Potentiometer

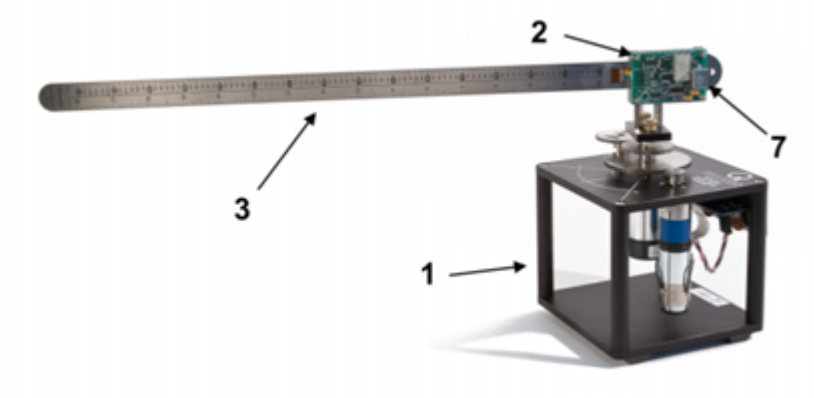


Figure 6.2. FLEXGAGE coupled to SRV02

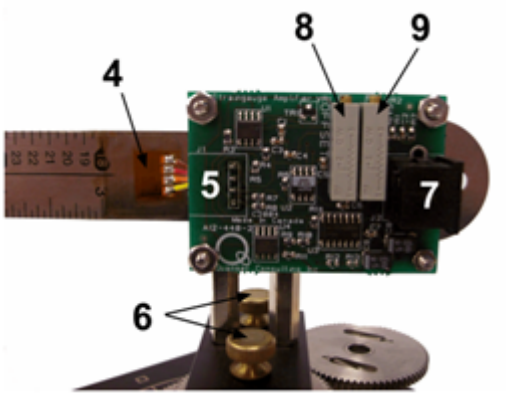


Figure 6.3. Strain Gauge Closeup

The FLEXGAGE module consists of the strain gauge, the strain gauge circuitry, and a sensor connector. The flexible link is attached to this module and the strain gauge is fixed at the root of the link. The module is mounted onto the servo motor which is the actuator for this system. The strain gauge sensor produces an analog signal proportional to the deflection of the link tip.

The link can be schematically represented as shown in Fig. 6.4. The flexible link has a total length of L_1 , a mass of m_1 , and its moment of inertia about the center of mass is $a = J_1$. The deflection angle of the link is denoted as α and increases positively when rotated CCW.

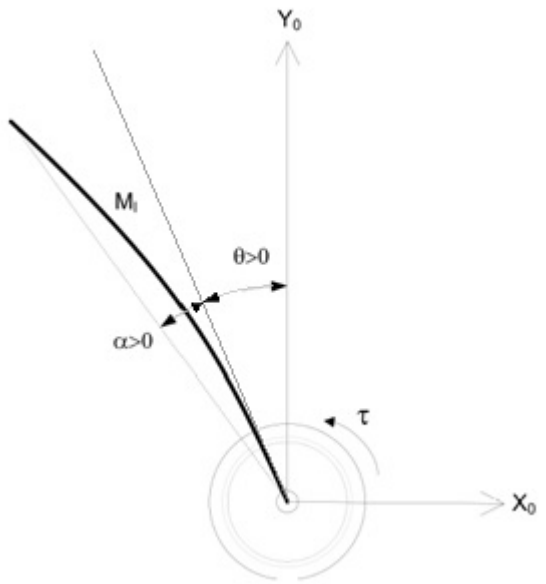


Figure 6.4. Rotary Flexible Link Angles

The complete flexible link system can be represented by the diagram shown in Fig. 6.5. The control variable is the input servo motor voltage, V_m which is proportional to the angular rate of the servo motor. This generates a torque, τ, at the load gear of the servo that rotates the base of the link, which is given by

$$\tau = \frac{\eta_g K_g \eta_m k_t (V_m - K_g k_m \dot{\theta}}{R_m} = C_1 V_m - C_2 \dot{\theta}$$

(6.1)

where the various constants are SRV02 parameters which are mentioned in Table 6.2.

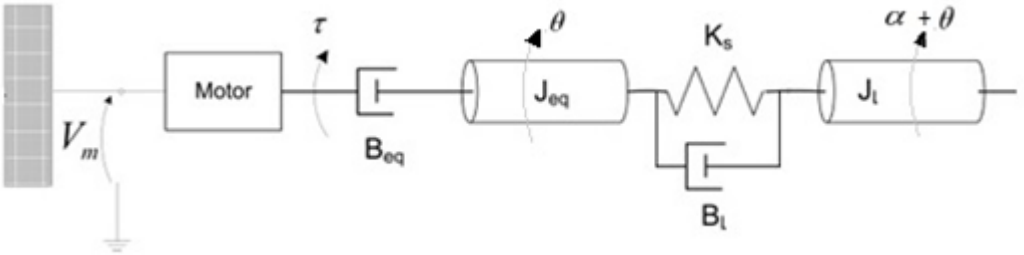


Figure 6.5. Rotary Flexible Link Model

The viscous friction coefficient of the servo is denoted by B_{eq} . This is the friction that opposes the torque being applied at the servo load gear. J_{eq} represents the moment of inertia of the SRV02 when there is no load. The friction acting on the link is represented by the viscous damping coefficient B_l . The flexible link is modeled as a linear spring with the stiffness K_s with moment of inertia J_l .

Table 6.2. Setup Parameters

Symbol	Parameter	Value (Units)
m_l	Mass of flexible link	0.065 kg
L_l	Length of flexible link	0.419 m
b_l	Width or breadth of flexible link	2.083 x 10-2 m
h_l	Thickness of flexible link	8.128 x 10–4 m
E	Young's modulus of flexible link	200 GPa
I	Area moment of inertia of link cross-section	9.32 x 10-13 m4
B_{eq}	High-gear viscous damping coefficient of SRV02	0.015 N.m/(rad/s)
J_{eq}	Equivalent high-gear moment of inertia of SRV02 (no load)	0.00208 kg.m2
R_m	Motor armature resistance	2.6 Ω
k_t	Motor torque constant	7.68 x 10-3 N.m

η_m	Motor efficiency	0.69
k_m	Back-emf constant	7.68 x 10-3 V/(rad/s)
K_g	High-gear total gearbox ratio	70
η_g	Gearbox efficiency	0.90
B_l	Viscous damping coefficient of flexible link	To be calculated
J_l	Moment of inertia of flexible link about pivoted end	To be calculated
K_s	Stiffness of flexible link	To be calculated
m	Mass per unit length of flexible link	To be calculated

Equations of Motion

The servo and flexible link can be modelled as a lumped mass system separated by an equivalent spring and damper, which represent the stiffness and damping coefficient of the flexible link respectively. The equations that describe the motions of the servo and the link with respect to the servo motor torque, i.e. the dynamics, can be obtained using free body diagram (FBD) analysis of the lumped mass moments of inertia (J_{eq} and J_l).

The torque balance on J_{eq} yield Eqn. 6.2 and the torque balance on J_l yield Eqn. 6.3.

$$J_{eq}\ddot{\theta} + B_{eq}\dot{\theta} - B_l\dot{\alpha} - K_s\alpha = \tau$$

(6.2)

$$J_l\ddot{\theta} + J_l\ddot{\alpha} + B_1\dot{\alpha} + K_s\alpha = 0$$

(6.3)

On rearranging Eq. 6.3 to obtain an expression for $B_l\dot{a} + K_s\alpha$ and substituting it in Eqn. 6.2, the equations of motion (EOM) for the rotary flexible link system can be obtained as

$$(J_{eq} + J_l)\ddot{\theta} + J_l\ddot{\alpha} + B_{eq}\dot{\theta} = \tau$$

(6.4)

$$J_l\ddot{\alpha} + J_l\ddot{\theta} + B_l\dot{\alpha} + K_s\alpha = 0$$

(6.5)

Stiffness Determination

The stiffness of the flexible link can be determined from the free oscillation of the link using a second order model. The free-oscillatory equation of motion of this second-order system is obtained by setting θ̈ term to zero in Eqn. 6.5, i.e. by holding θ constant, which is shown in Fig. 6.6. The resulting equation will be

$$J_l\ddot{\alpha} + B_l\dot{\alpha} + K_s\alpha = 0$$

(6.6)

Assuming the initial conditions $\alpha(0) = \alpha_0$ and $\dot{\alpha}(0) = 0$, the Laplace transform of Eqn. 6.6

$$A(s) = \frac{\frac{\alpha_0}{J_l}}{s^2 + \frac{B_l}{J_l}s + \frac{K_s}{J_l}}$$

(6.7)

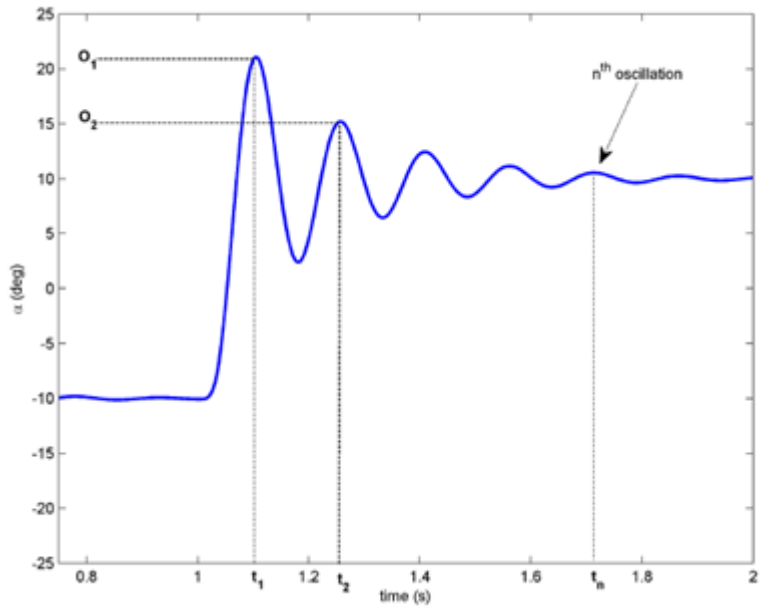


Figure 6.6. Free Oscillation Response

The prototype second-order equation is defined as

$$s^2 + 2\varsigma\omega_n s + \omega_n^2$$

where ζ is the damping ratio and ω_n is the natural frequency. Equating this to the characteristic equation in Eqn. 6.7 gives

$$\omega_n^2 = \frac{K_s}{J_l} \tag{6.8}$$

$$2\zeta\omega_n = \frac{B_l}{J_l} \tag{6.9}$$

J_l represents the moment of inertia of the link about the pivot. This can be calculated approximately by considering the link as a rod rotating about a pivot at one edge ($J_l = \frac{m_l L_l^2}{3}$). Equations 6.8 and 6.9 can be used to determine the stiffness and damping of the flexible link once the natural frequency and damping ratio are known.

The damping ratio of this second-order system can be found from its response (underdamped system) using the subsidence or decrement ratio given by

$$\delta = \frac{1}{n-1} \ln \frac{0_1}{O_n} \tag{6.10}$$

where O_1 is the peak of the first oscillation and O_n is the peak of the nth oscillation. Note that $O_1 > O_n$, as this is a decaying response.

The damping ratio is defined as

$$\zeta = \frac{1}{\sqrt{1+(\frac{2\pi}{\delta})}} \tag{6.11}$$

The period of the oscillations in a system response can be found using the equation

$$T_{osc} = \frac{t_n - t_1}{n-1} \tag{6.12}$$

where t_n is the time of the *nth* oscillation, t1 is the time of the first peak, and *n* is the number of oscillations considered.

From this, the damped natural frequency (in rad/s) is

$$\omega_d = \frac{2\pi}{T_{osc}} \tag{6.13}$$

and the undamped natural frequency is

$$\omega_n = \frac{\omega_d}{\sqrt{1-\zeta^2}} \tag{6.14}$$

Part 1

Experimental Procedure

1. Open MATLAB and locate the file **FlexLink_FreeOsc.mdl** under the path **C:\AE4610_Controls_Lab\Rotary_Flexible_Link** and open it. This is the block diagram for this part of the experiment.

2. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.

3. Open the scope **alpha**.

4. Turn on the power supply.

5. Press **Connect to Target**  button in the menu bar and hold on the SRV02 load to prevent any rotation at the root.

6. Press **Start**  and immediately perturb the flexible link. Keep holding the SRV02 base until the data is collected for the complete run (5 seconds).

7. Save the link deflection angle data for the free oscillation with different names, for example type in the command line:
   save Group No._FreeOsc_1.

8. Repeat the steps 6, 7 two more times for different perturbation locations along the link or different perturbation angles.

9. Email the data to yourself and delete the data files. **DO NOT SAVE OR DELETE THE SIMULINK MODEL**.

Analysis

1. Plot the measured angular deflection of the link vs. time for each case. From the plot, determine the time period of oscillation by considering the first and say, the fourth or fifth oscillation. Use this information to determine the damped frequency, undamped frequency and stiffness.

2. Average the stiffness value of the link for the three sets of data to get a single value of the link stiffness.

3. Compare the damped and undamped frequencies and report your observation. What does this signify with respect to the flexible link damping?

Modal Frequencies and Mode Shapes

The flexible link can be considered as a thin continuous (uniform) cantilever beam anchored at one end and free at the other end. Using the Euler-Bernoulli beam theory, the

equation of motion can be written as

$$EI\frac{\partial^4 Y(x,t)}{\partial x^4} + m\frac{\partial^2 Y(x,t)}{\partial t^2} = q(x,t)$$

(6.15)

where E is the modulus of rigidity of beam material (assumed constant), I is the area moment of inertia of the beam cross-section (assumed constant), $Y(x,t)$ is displacement in y direction at distance x from fixed end at time t, ω is the circular natural frequency, m is the mass per unit length (m = ρA, ρ is the material density, A is the cross section area), x is the distance measured from the fixed end and q is the external applied force per unit length.

Also, the angle of deflection α is related to the displacement as

$$\alpha = \frac{\partial Y}{\partial x}$$

(6.16)

The general solution to Eq. 6.15 can be obtained using separation of variables (Eqn. 6.17).

$$Y(x,t) = v(x)s(t)$$

(6.17)

Substituting (6.17) in (6.15), setting $q(x,t) = 0$ and rearranging gives

$$\frac{d^4 v(x)}{dx^4} s(t) + v(x) \frac{m}{EI} \frac{d^2 s(t)}{dt^2} = 0$$

(6.18)

Equation 6.18 can be rewritten as

$$\frac{\frac{d^4 v(x)}{dx^4}}{v(x)} = -\frac{m}{EI}\left(\frac{\frac{d^2 s(t)}{dt^2}}{s(t)}\right)$$

(6.19)

Since the left side of Eqn. 6.19 is only a function of x and the right side of Eqn. 6.19 is only a function of t, they both must equal a constant. Let this constant be β^4 . Thus, Eqn. 6.19 can be written as the following two equations

$$\frac{d^4v(x)}{dx^4} - \beta^4 v(x) = 0$$

(6.20)

$$\frac{d^2s(t)}{dt^2} + \frac{EI}{m}\beta^4 s(t) = 0$$

(6.21)

The solution of Eq. 6.20 gives the displacement v (as a function of x), which will be of the form

$$v(x) = a\sin(\beta x) + b * \cos(\beta x) + c\sinh(\beta x) + d\cosh(\beta x)$$

(6.22)

where a, b, c, and d are unknown constants. The general solution of Eqn. 6.21 is given by

$$s(t) = g\sin(\omega t) + h\cos(\omega t)$$

(6.23)

where $\omega = \sqrt{\frac{EI}{m}}\beta^2$, g and h are unknown constants.

The constants in Eqn. 6.22 are determined from four boundary conditions while the constants from Eqn. 6.23 are determined from two initial conditions.

For a clamped-free or cantilever beam, the geometric boundary conditions are

$$Y(x=0,t) = 0 \rightarrow v(x=0) = 0$$

$$\frac{\partial Y}{\partial x}|_{(x=0,t)} = 0 \rightarrow \frac{dv}{dx}|_{x-0} = 0$$

and the natural boundary conditions are

$$M(x=L,t)=0 \rightarrow EI\frac{\partial^2 Y}{\partial x^2}|_{x=L,t}=0 \rightarrow \frac{d^2v}{dx^2}|_{x=L}=0$$

$$V(x=L,t)=0 \rightarrow -EI\frac{\partial^3 Y}{\partial x^3}|_{(x=L,t)}=0 \rightarrow \frac{d^3v}{dx^3}|_{(x=L)}=0$$

where M represents the bending moment and V represents the shear force.

On substitution of the geometric boundary conditions at x = 0 in Eqn. 6.22 and its derivative, the following relations can be obtained

$$v(x=0)=0 \rightarrow d=-b$$

$$\frac{dv}{dx}|_{(x=0)}=0 \rightarrow c=-a$$

Hence, Eqn. 6.22 becomes

$$v(x)=a[\sin(\beta x)-\sinh(\beta x)]+b[cos(\beta x)-\cosh(\beta x)]$$

(6.24)

On further substitution of the natural boundary conditions at x = L in the derivatives of Eqn. 6.24 yields

$$\frac{d^2v}{dx^2}|_{(x=L)}=0 \rightarrow a[\sin(\beta L)+\sinh(\beta L)]+b[\cos(\beta L)+\cosh(\beta L)]=0$$

$$\frac{d^3v}{dx^3}|_{(x=L)} = 0 \rightarrow a[cos(\beta L) + \cosh(\beta L)] - b[\sin(\beta L) - \sinh(\beta L)] = 0$$

or

$$\begin{bmatrix} sin(\beta L) + sinh(\beta L) & cos(\beta L) + cosh(\beta L) \\ cos(\beta L) + cosh(\beta L) & -sin(\beta L) + sinh(\beta L) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0$$

For a non-trivial solution, the determinant of the above matrix must be 0, i.e.

$$Det \begin{bmatrix} sin(\beta L) + sinh(\beta L) & cos(\beta L) + cosh(\beta L) \\ cos(\beta L) + cosh(\beta L) & -sin(\beta L) + sinh(\beta L) \end{bmatrix} = 0$$

which gives the following characteristic equation

$$[\sin(\beta L) + \sinh(\beta L)][-\sin(\beta L) + \sinh(\beta L)] - [\cos(\beta L) + \cosh(\beta L)]^2 = 0$$

The above equation simplifies to

$$\cos(\beta L)\cosh(\beta L) = -1$$

There are infinite solutions to this characteristic equation, which are given by

$$\beta L = 1.875, 4.694, 7.855, \ldots \ldots$$

$$\beta_{1,2,3,\ldots} = \frac{1.875}{L}, \frac{4.694}{L}, \frac{7.855}{L}, \ldots \ldots$$

Thus, the mode shapes are

$$v_i(x) = [\cosh(\beta_i x) - \cos(\beta_i x)] + \frac{\cos(\beta_i L)+\cosh(\beta_i L)}{\sin(\beta_i L)+\sinh(\beta_i L)}[\sin(\beta_i x) - \sinh(\beta_i x)]$$

(6.25)

Since $\omega = \sqrt{\frac{EI}{m}}\beta^2$, the modal frequencies ω_i are given by

$$\omega_i = \sqrt{\frac{EI}{m}}\beta_i^2, \quad i = 1, 2, 3, \ldots \ldots$$

(6.26)

or

$$\omega_1 = \sqrt{\frac{EI}{m}}\left(\frac{1.875}{L}\right)^2$$

$$\omega_2 = \sqrt{\frac{EI}{m}}\left(\frac{4.694}{L}\right)^2$$

$$\omega_3 = \sqrt{\frac{EI}{m}}\left(\frac{7.855}{L}\right)^2$$

> m is mass per unit length

The Mode shapes of a uniform cantilever beam are shown in Fig. 6.7.

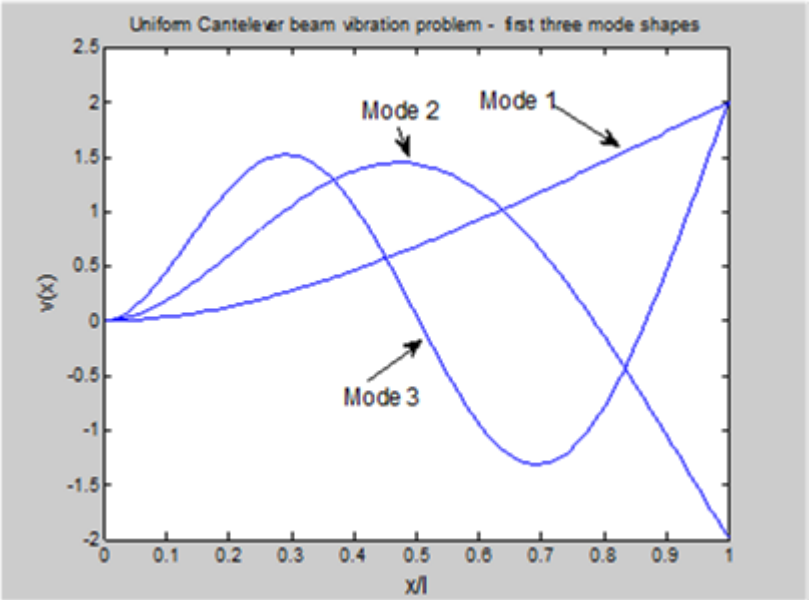


Figure 6.7. First three mode shapes of a uniform cantilever beam.

In order to determine the modal frequencies and mode shapes, the corresponding frequencies can be excited by providing a sinusoidal input to the link via external means. When the frequency of the input coincides with either the fundamental frequency or higher frequency modes, the corresponding modes will be excited due to resonance and their mode shapes can be observed physically. Hence, the following experiment involves a frequency sweep across a range provided as input to the flexible link via the servo motor to identify the frequencies and observe the corresponding mode shapes.

Part 2

Experimental Procedure

1. Open MATLAB and locate the file **FlexLink_ExciteMode.mdl** under the path **C:\AE4610_Controls_Lab\Rotary_Flexible_Link** and open it. This is the block diagram for this part of the experiment.

2. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.

3. Open the scope **alpha**.

4. Turn on the power supply.

5. Ensure that the manual switch is connected to the **Chirp signal** input. This signal will provide a sinusoidal signal of fixed amplitude, with frequency increasing at a linear rate with time.

6. Open the **Chirp signal** command block and make sure that the Initial frequency is **0.1 Hz**, the target time is **0.25 s** and the Frequency at target time is **0.2 Hz**. This will allow the frequency sweep to take place at a reasonable rate and ensure that the relevant frequencies are covered within the span of time.

7. Press **Connect to Target**  button in the menu bar and then press **Start**  . Run the servo motor with the chirp input voltage for 60 seconds.

8. Save the data with proper title.

9. Email the data to yourself and delete the data file. **DO NOT DELETE THE SIMULINK MODEL**.

Analysis

1. Plot the measured angular deflection of the link vs. time. Using the time domain plot, perform frequency analysis using Fast Fourier Transform (FFT) [refer to the Appendix for the code] or a similar technique to identify the number of dominant frequencies and their magnitudes present in the signal.

2. Compare the first dominant frequency with the natural frequency of the flexible link calculated from Analysis (1) and state your observation. Explain any similarities or differences spotted.

Calculations (to be done before lab session)

1. Using the system parameters provided in Table 6.2, determine the mass per unit length of the beam (m).

2. Using Eqn. 6.26 and the values of necessary system parameters, calculate the first and second modal frequencies.

Part 3

Experimental Procedure

1. Open MATLAB and locate the file **FlexLink_ExciteMode.mdl** under the path **C:\AE4610_Controls_Lab\Rotary_Flexible_Link** and open it. This is the block diagram for this part of the experiment.

2. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.

3. Open the scope **alpha**.

4. Turn on the power supply.

5. Connect the manual switch to the **Sine wave** signal input. This signal will provide a sinusoidal signal of fixed amplitude and fixed frequency.

6. Open the **Sine wave** signal command block and make sure that the Amplitude is **3** and Phase is **0 rad**. Enter the first modal frequency determined from the calculations in **rad/sec**.

7. Press **Connect to Target**  button in the menu bar and then press **Start**  . Run the servo motor with the sine wave for at least 20 seconds.

8. Tune the frequency value by increasing or decreasing in steps of 1 rad/sec until the first mode shape is clearly visible.

9. Observe the corresponding mode in the link and note down the number of nodes and their locations.

10. Save the data with proper title.

11. Repeat steps 6 to 10 for the second modal frequency identified.

12. Email the data to yourself and delete the data file. **DO NOT DELETE THE SIMULINK MODEL**.

Analysis

1. Plot the mode shapes observed using Eqn. 6.25 in MATLAB (plot v(x) vs x/L) and record node locations.

2. Record the number of nodes and their locations for each mode. Determine the location as a ratio of the link length and compare with the values obtained from the mode shape plots.

3. Compare the calculated first and second modal frequencies (after tuning) with the corresponding first and second dominant frequencies obtained from Step 1 of Analysis (2) and explain your observation.

Appendix

The following code performs FFT analysis on the flexible link angle response –

```
T = 0.002;
```

```
Fs = 1/T;
```

```
L = 30001;
```

```
f = Fs*(0:(L/2))/L;
```

```
Ya = fft(output_alpha);
```

```
Pa2 = abs(Ya/L);
```

```
Pa1 = Pa2(1:L/2+1);
```

```
Pa1(2:end-1) = 2*Pa1(2:end-1);
```

```
figure
```

```
plot(f,Pa1);
```

```
xlim([0 150]);
```

```
xlabel('Frequency (Hz)');
```

```
ylabel('|Amplitude {\alpha}|');
```

```
title('Single-Sided Amplitude Spectrum of \alpha');
```

```
figure
```

```
plot(time,output_alpha);
```

```
xlabel('Time (s)');
```

```
ylabel('\alpha (deg)');
```

```
title('Flexible Link Angle vs Time');
```

Inverted Pendulum

Introduction

Proportional (/Derivative) Control Determined through LQR Analysis

This experiment will demonstrate the control of a system using **full state feedback**, where the control system is designed using the principles of Linear Quadratic Regulator (LQR) theory. This is an analysis technique that would be seen in more detail during advanced studies. You are not required to completely understand the derivation of the method itself. However, *you should understand how to formulate the required matrices (Q and R) and why the overall goal (to minimize the cost, J) leads to the requirements on those matrices, as well as the reasoning behind the formulation of the cost equation*, which serves as the purpose behind the method's derivation.

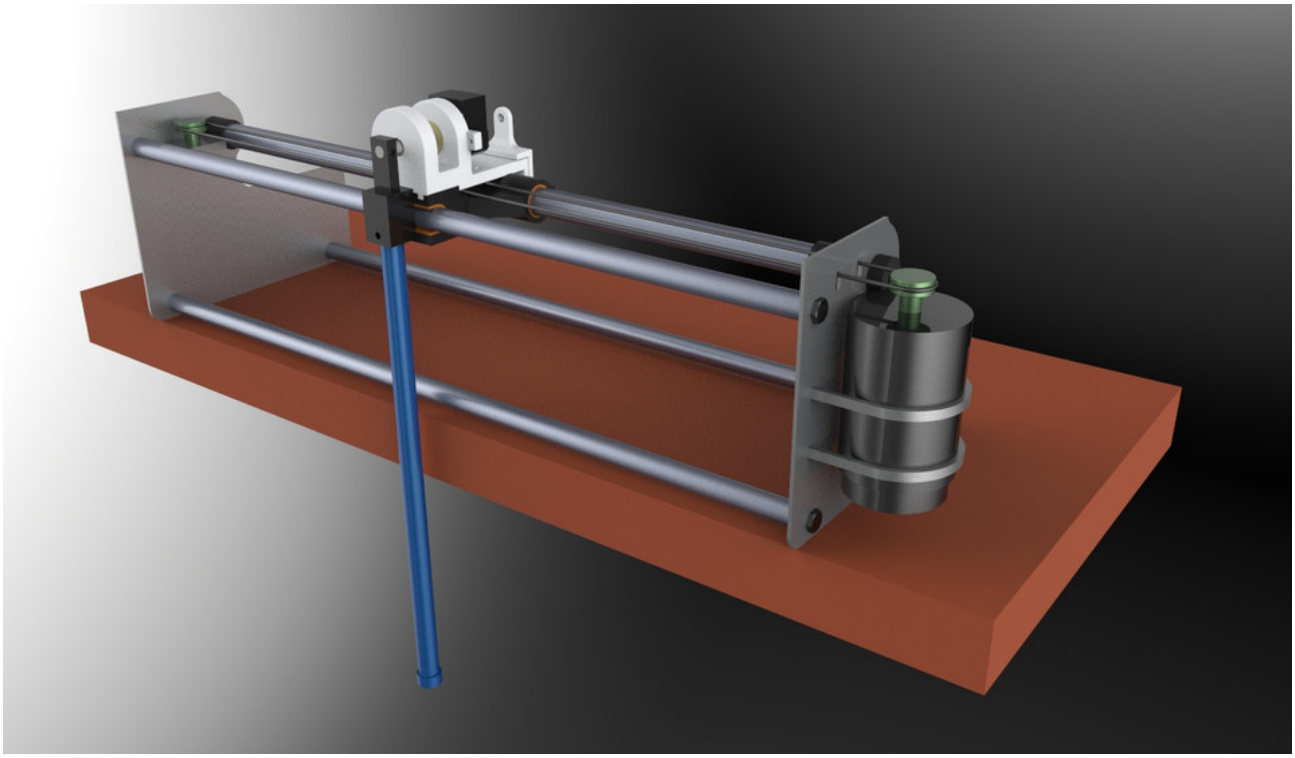


Image source: https://grabcad.com/library/inverted-pendulum-1

Linear Quadratic Theory (LQR)

Consider the system

$$\dot{X} = AX + BU$$

where X is a state vector and U is a $m \times 1$ control input vector. Therefore, A and B are matrices of dimensions $n \times n$ and $n \times m$, respectively. Let us suppose that we wish to design a state-feedback controller

$$U = -KX$$

such that the closed-loop system

$$\dot{X} = (A - BK)X$$

is stable. Here K is a matrix of gains of dimensions $m \times n$. In achieving closed-loop stability, we wish to use as little control effort as possible. We also wish to avoid large overshoot of the trajectories during the transients. One way to achieve both of these objectives is to find a control law which minimizes the following quadratic cost

$$J = \int_0^\infty (X^T QX + U^T RU)dt$$

where Q and R are weighing matrices of appropriate dimensions. In order for the previous problem to make sense we must ensure that the two terms in the integral are positive. The requirement that

$$X^T QX \geq 0$$

for all vectors X is satisfied if the matrix Q is **positive semi-definite** (all its eigenvalues are greater than or equal to zero). The requirement that

$$U^T RU > 0$$

for all vectors U is satisfied if the matrix R is **positive definite** (all its eigenvalues are greater than zero). This optimization problem has a closed-form solution. The optimal gain matrix K is given by

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

where the $n \times n$ matrix P is the positive definite solution of the following **Algebraic Riccati Equation** (ARE):

$$0 = A^T P + PA - PBR^{-1} B^T P + Q$$

Example of a Simple Mass/Spring/Damper System

Consider the system in Fig. 7.1. The equation of motion is

$$m\ddot{x} + c\dot{x} + kx = F$$

or in state-space form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

where $x_1 = x$ and $x_2 = \dot{x}$. Let $m = 1,\ k = 2$ and $c = 1$. The system matrices are

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

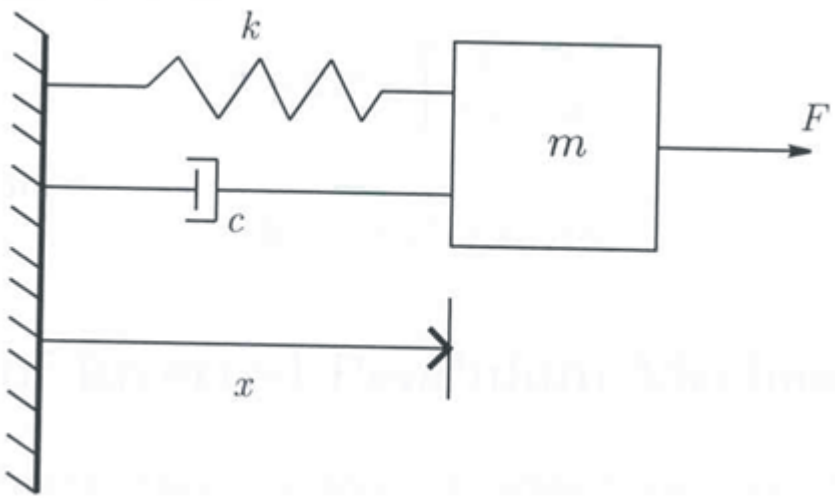


Figure 7.1: A simple mass / spring / damper system

Choose the weighting matrices as

$$Q = \begin{bmatrix} 12 & 0 \\ 0 & 4 \end{bmatrix}, \quad R = 1$$

The Riccati equation gives

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} 12 & 0 \\ 0 & 4 \end{bmatrix}$$

Expanding, we obtain

$$\begin{gathered} p_{12}^2 + 4p_{12} - 12 = 0 \\ p_{11} - 2p_{22} - p_{12} - p_{12}p_{22} = 0 \\ p_{22}^2 - 2p_{12} + 2p_{22} - 4 = 0 \end{gathered}$$

Solving these equations, we get

$$p_{11} = 10,\ p_{12} = 2,\ p_{22} = 2$$

or that

$$P = \begin{bmatrix} 10 & 2 \\ 2 & 2 \end{bmatrix}$$

Notice that the solution matrix P is positive definite. Its eigenvalues are $\lambda_{1,2} = 10.472,\ 1.528$.

The optimal gain matrix is given by

$$K = R^{-1} B^T P$$
$$K = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \end{bmatrix}$$

The closed-loop system matrix is

$$A - BK = \begin{bmatrix} 0 & 1 \\ -4 & -3 \end{bmatrix}$$

and the eigenvalues are $\lambda_{1,2} = -1.5 \pm i1.323.$

The closed loop system is stable.

Control of an Inverted Pendulum

Objective

The purpose of this experiment is to design a balance control using the principles of linear quadratic regulator (LQR) theory for an inverted pendulum mechanism.

Equipment Required

- [ ] Inverted Pendulum mechanism
- [ ] Q8-USB interface board
- [ ] MATLAB, SIMULINK and QUARC software.
- [ ] Power module for the PCI Multi-Q board and the pendulum mechanism.

Introduction

In this lab we demonstrate a control design using modern “state-space” methods. The plant consists of an inverted pendulum on a cart. The plant has two distinct equilibrium points of which one is stable and the other is unstable. In this experiment, we will show how the plant in its unstable configuration can be controlled using a suitable controller. The methodology used to design the control law is based on the linear quadratic regulator (LQR) theory. The inverted pendulum mechanism to be used in this lab is shown in Fig. 7.2.

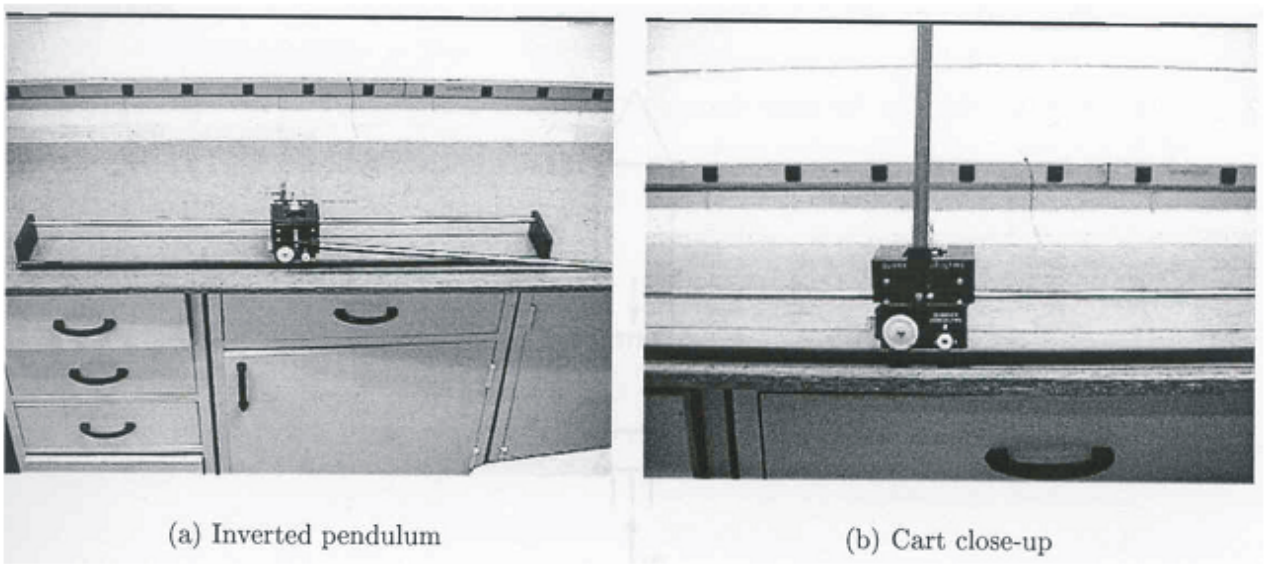


Figure 7.2: Inverted Pendulum mechanism in the lab

Mathematical Model of the Inverted Pendulum Mechanism

Consider an inverted pendulum of mass m_p and length $2l_p$ connected on a sliding cart of mass m_c as shown in Fig. 7.3.

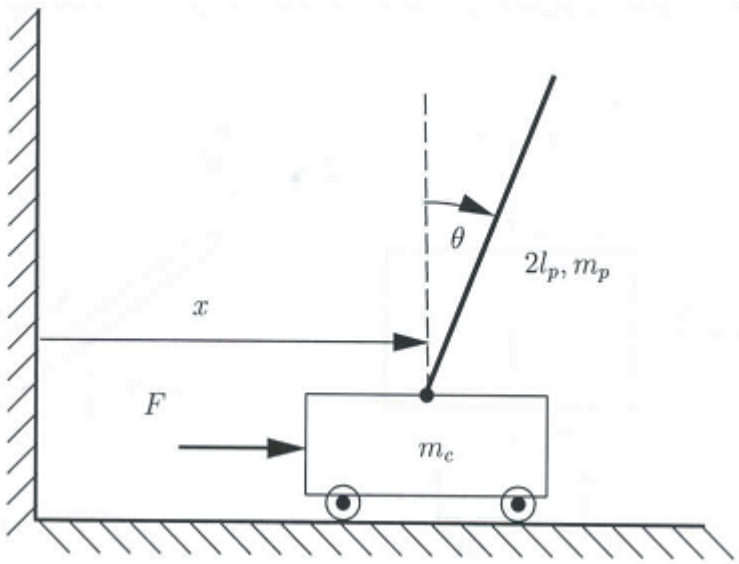


Figure 7.3: Inverted pendulum mechanism

The angle θ is measured from the up-right position. We can apply a force to the cart through the cart wheel motors. We wish to design a controller to balance the pendulum to the up-right position $\theta = 0$. Note that the problem of balancing a pendulum is akin to the problem of stabilizing a rocket during ascent.

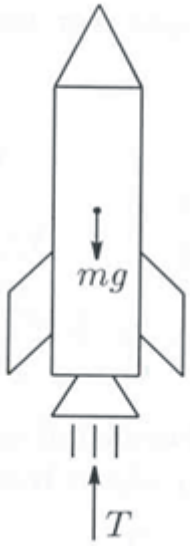


Figure 7.4: Powered rocket during ascent

The equations of motion for the system can be easily derived using the free-body diagrams shown in Fig. 7.5 as follows (here we assume that the track on which the cart moves without sliding):

$$(m_p + m_c)\ddot{x} + m_p\ddot{\theta}l_p\cos(\theta) - m_p\dot{\theta}^2 l_p\sin(\theta) = F$$
$$m_p l_p\cos(\theta)\ddot{x} + I\ddot{\theta} + m_p\ddot{\theta}l_p^2 - m_p g l_p\sin(\theta) = 0$$

where I is the moment of inertia about the pendulum's mass center given by $I = \frac{m_p l_p^2}{3}$.

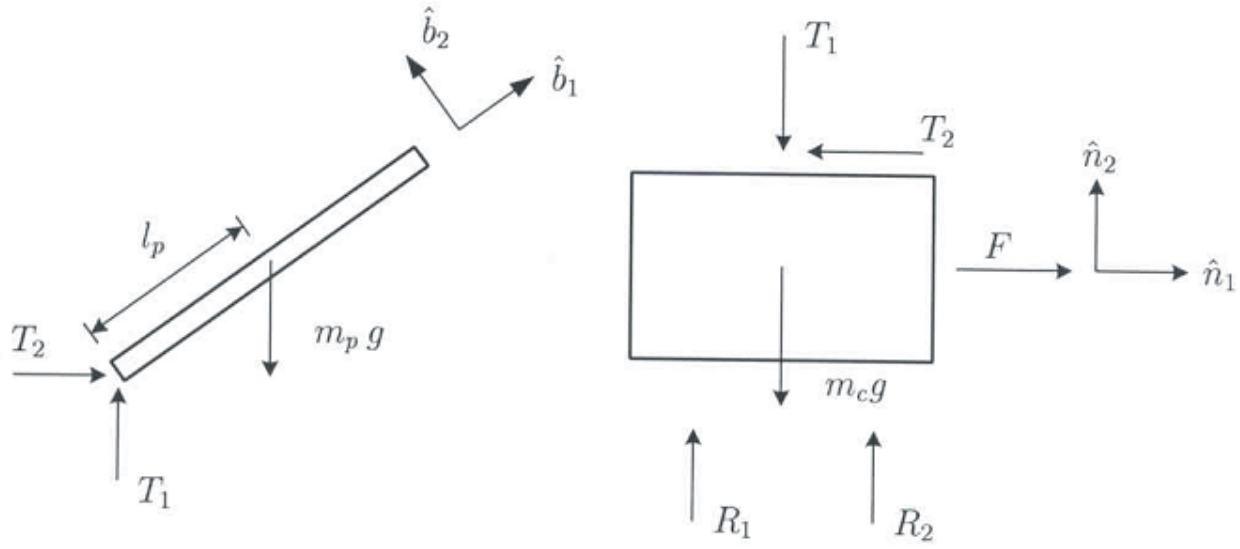


Figure 7.5: Free Body Diagrams

This system has two equilibria $(\dot{x} = \ddot{x} = \dot{\theta} = \ddot{\theta} = 0)$ which are
$(x_{e1}, \theta_{e1} = (0, \pi)\ and\ (x_{e2}, \theta_{e2}) = (0, 0)$

The first equilibrium corresponds to the down position (stable) and the second equilibrium corresponds to the up-right position (unstable) of the pendulum. Linearizing the previous equations about the equilibrium of interest (0, 0) we obtain the following equations:

$$\frac{d}{dt}\begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-3m_p g}{(m_p + 4m_c)} & 0 & 0 \\ 0 & \frac{3(m_p + m_c)g}{(m_p + 4m_c)l_p} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{4}{(m_p + 4m_c)} \\ \frac{-3}{(m_p + 4m_c)l_p} \end{bmatrix} F$$

The input force F in the above equations can be written in terms of the input voltage V applied to the motor of the cart and the cart velocity $\dot{x}$ as shown in the following:

First, we write the motor equation as

$$V = I_m R_m + K_m K_g \omega_g = I_m R_m + K_m K_g \frac{\dot{x}}{r}$$

or

$$I_m = \frac{V}{R_m} - \frac{K_m K_g}{R_m} \frac{\dot{x}}{r}$$

where

- V : voltage applied to motor (Volts)
- I_m : current in motor (Amp)
- K_m : back EMF constant (Volt/(rad/sec))
- K_g : gear ratio in motor gearbox
- R_m : motor armature resistance (Ohms)
- ω_g : angular velocity of the wheel (rad/sec)
- $\dot{x}$: cart velocity (m/sec)
- r : radius of motor pinion that meshes with the track (m)

The torque generated at the output of the motor is given by

$$T_m = K_t K_g I_m$$

where K_m is the motor torque constant given by

$$K_t = kK_m$$

where k is the conversion factor from volt-amp to N-m given by

$$k = 1.3558 N - m/s/(volt - amp)$$

Thus the force transmitted to the cart via the pinion is

$$F = \frac{T_m}{r} = \frac{K_t K_g V}{r R_m} - \frac{K_t K_g K_m K_g}{R_m} \frac{\dot{x}}{r^2} = \frac{k K_m K_g V}{r R_m} - \frac{k (K_m K_g)^2}{R_m} \frac{\dot{x}}{r^2}$$

and on substituting the values of

$$K_m = 0.00767 volt - amp/rad/s,\ K_g = 3.7$$
$$k = 1.3558 N - m/s/(volt - amp),\ r = 0.00635 m,\ R_m = 2.6\ ohms$$

we get

$$F = 2.3305 V - 10.4152 \dot{x}$$

Letting $a_1 = 2.3305$ and $a_2 = 10.4152$, the input force F can be written as

$$F = a_1 V - a_2 \dot{x}$$

Substitution of the above expression for F in the linearized equations results in

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-3m_p g}{(m_p + 4m_c)} & \frac{-4a_2}{(m_p + 4m_c)} & 0 \\ 0 & \frac{3(m_p + m_c)g}{(m_p + 4m_c)l_p} & \frac{3a_2}{(m_p + 4m_c)l_p} & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{4a_1}{(m_p + 4m_c)} \\ \frac{-3a_1}{(m_p + 4m_c)l_p} \end{bmatrix} V$$

The above set of linear differential equations is of the form $\dot{X} = AX + BV$, where $X = (x, \theta, \dot{x}, \dot{\theta})^T$. We seek a linear control law of the form $V = -KX$, where K is a 1x4 vector of gains.

Linear Quadratic Regulator (LQR) Design

The LQR formulation provides a linear control law

$$V = -KX$$

which minimizes the quadratic cost given by

$$J = \int_0^\infty (X^T Q X + V^T R V) dt$$

for some user-defined matrices $Q = Q^T$ and $R = R^T$. It is assumed that Q is positive semi-definite (all the eigenvalues are non-negative) and R is positive definite (all the eigenvalues are positive). Here V is the input voltage, X given by

$$X = \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}$$

is the state vector, and K given by

$$K = \lfloor k_1 \ k_2 \ k_3 \ k_4 \rfloor$$

is the controller gain vector. Note that the input voltage V is scalar and therefore, R is also scalar. In practice, the weighting matrices Q and R are chosen to be diagonal. A simple way for choosing Q and R is the following (Bryson's rule):

Assuming the time-domain design specifications are:

$$|V| \leq \bar{V}, |X_i| \leq \bar{X_i}$$

one can choose

$$R = \frac{1}{\bar{V}^2}, \ Q = diag\left(\frac{1}{\bar{X}_i^2}\right)$$

Further tuning of R and Q may be required for improving the controller performance. Note that increasing R means increasing the penalty on the control input (using less

control effort). Similarly, increasing Q means increasing the penalty on the states (states reaching equilibrium position quickly and with less overshoot).

The block diagram of the experimental set up in the lab is shown in Fig. 7.6

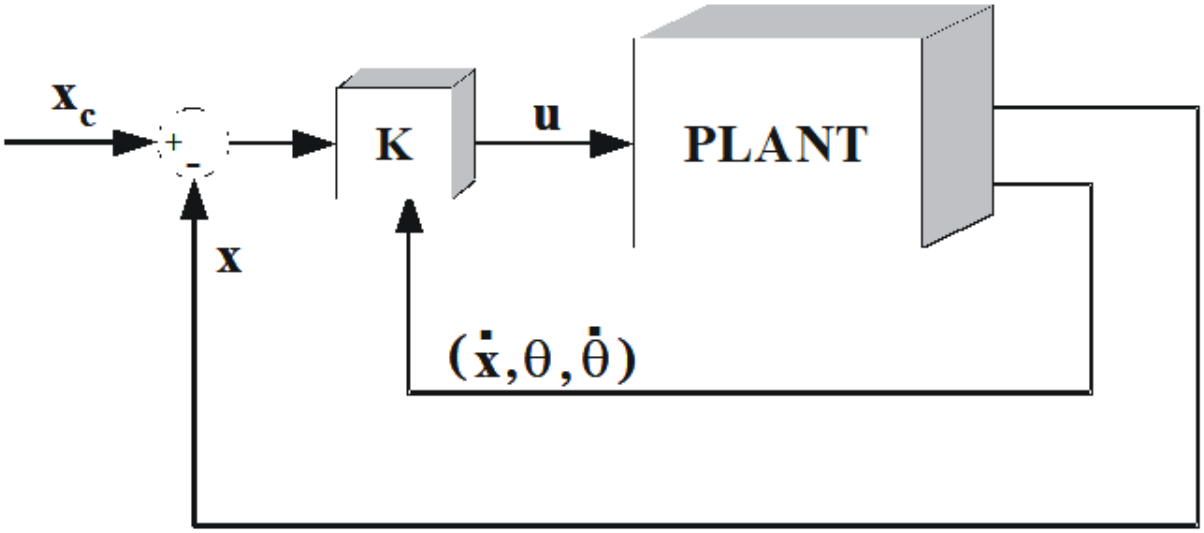


Fig 7.6: Block diagram of the experimental set-up

(Note: Conventional error signal shown, but $u = -K\Delta X, \Delta X = X - X_C$.)

where x is position of the cart, $\dot{x}$ is velocity of the cart, θ is angular position of the rod from the vertical, $\dot{\theta}$ is angular velocity of the rod, and K is a vector of feedback gains $K = \lfloor k_1 \ k_2 \ k_3 \ k_4 \rfloor$.

Controller Design Procedure

1. Obtain the A and B matrices of the system using $m_p = 0.21\ kg,\ m_c = 0.815\ kg$, and $l_p = 0.305\ m$.

2. Choose $Q \geq 0$ and $R > 0$ based on the following:
   Assume that:
   1. $0.4m \leq |x_{max}| \leq 0.5m$
   2. $\frac{\pi}{6} rad \leq |\theta_{max}| \leq \frac{\pi}{4} rad$

3. $6\ volts \leq |V_{max}| \leq 10\ volts$

No constraints on $\dot{x}_{max}$ and $\dot{\theta}_{max}$. (They can go to infinity…)
According to Bryson's rule, this will result in the following Q and R matrices:

$$Q = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 1.6211 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \ and\ R = [0.01]$$

3. Using the `lqr` command in MATLAB, calculate the optimal feedback gain vector K. The `lqr` command basically solves the Algebraic Ricatti Equation that you met earlier. `>> [k,s,e] = lqr(A,B,Q,R);`

4. Compute the open-loop and closed-loop eigenvalues of the system and comment on the stability of the open and closed-loop systems.

5. Compute the natural frequencies and damping ratios of the closed-loop system.

Controller Implementation and Evaluation

> ⚠ CAUTION: Always be ready to stop the experiment if the cart goes unstable. This can be done either by pressing the stop button in the Simulink or turning the power off.

1. Turn the power on.

2. Open the MATLAB and locate the file **InvPendulum_LQR.mdl** under the path **C:\AE4610_Controls_Lab\Inverted_Pendulum** and open it. This is the block diagram for this part of the experiment.

3. To build the model, choose **QUARC** and then **Build** (or press the **Build Model** button ). This generates the controller code.

4. Set the gain after the Command Signal block to 0.

5. Put the cart right in the middle of the track and hold up the pendulum fixed and straight. Do not release it until the controller is on.

6. Double click on the scopes and open then.

7. Press **Connect to Target**  button in the menu bar and then press **Start**  .

8. Gently disturb the pendulum to see the effect of the controller and then stop the simulation.

9. Set the gain after the Command Signal block to 1 and run the simulation. It makes the cart to move 20 cm to the right after 5 seconds.

10. Save the data with a descriptive name. **DO NOT SAVE THE MODEL**.

11. Turn the POWER OFF

Analysis

- Include the results from the controller design procedure Steps 1-5.

- Build a SIMULINK block diagram using Fig. 7.6 and obtain the simulated response to the same command input used in the experiment. Compare the simulated response with the experimental response and explain any differences between them.

> Average the first 10 seconds of the experimental data to obtain initial conditions for your state-space block in SIMULINK

Quadcopter

Introduction

Objective

This experiment will introduce students to the basic principles for controlling the flight of a small quadcopter, the CrazyFlie 2.1. Weighting only 27 grams and having 9.2 cm of length and width, is a “nanoquad” which has rapidly become one of the preferred platforms for quadcopter research.

In Part A, students will adjust the gains of a PID controller to follow autonomously a waypoint maneuver. In Part B, the students will interact with the obstacle avoidance capabilities of the quadcopter.

Equipment Required

- [ ] Crazyflie 2.1 quadcopter with flow deck and multi-ranger deck
- [ ] Crazyradio 2.4 GHz
- [ ] Crazyflie Python Client

Background

We define the body frame of a quadrotor with the X, Y and Z axes shown below in Figure 8.1, with roll angle about the x-axis, pitch angle about the y-axis, and yaw angle about the z-axis. There are different ways of attaching a reference frame to a quadcopter, Figure 8.1 shows the convention which is consistent with Crazyflie quadrotor’s source code.

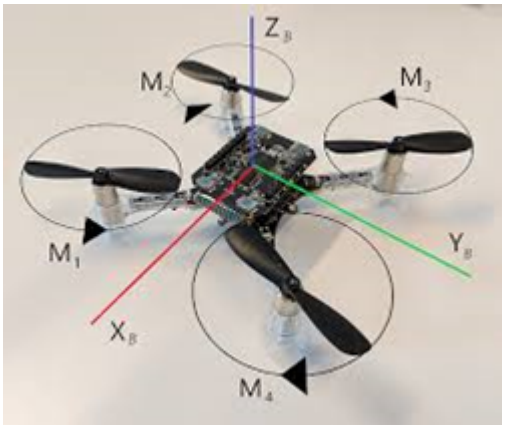


Figure 8.1. Quadrotor Body Frame Axes

A quadrotor has four sources of thrust and torque with its four rotors. As seen below in Figure 8.2, adjacent rotors rotate in opposite directions. In Figure 8.2, if the output of motors 1 & 2 are increased and motors 3 & 4 are decreased, the quadrotor can maintain an equal amount of total thrust while creating a roll moment. If the output of motors 1 & 4 are increased and motors 2 & 3 are decreased, the quadrotor can maintain an equal amount of total thrust while creating a pitch moment. If the output of motors 1 & 3 are increased and the output of motors 2 & 4 are decreased, the quadcopter will develop a yawing moment. This arrangement of the rotors enables full control of a quadrotor in 3D space.

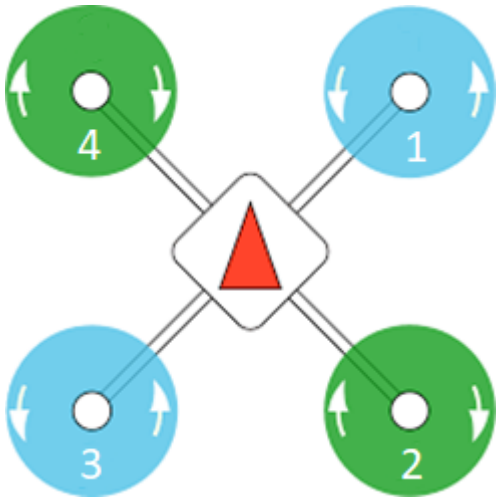


Figure 8.2 Quadrotor Propeller Direction. Note: Does NOT represent actual configuration.

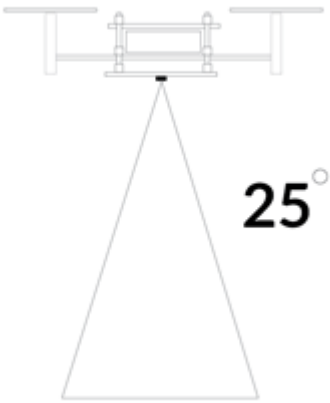


Figure 8.3 Cone Shape of the Crazyflie Height Sensing.

Flow Deck

The Flow Deck consists of a laser sensor that measures distance to the ground and a low-resolution camera called optical flow sensor which measures movement parallel to the ground. Together these sensors allow the CrazyFlie to interpret its movement in horizontal and vertical directions.

Note that the laser distance sensor will return the minimum distance to any object within its detection range as depicted in Figure 8.3. During the quadcopter's operation, it may mistake an object (like a box lying on the ground) as the actual ground, try to maintain a certain distance over the false ground, and therefore suddenly and unexpectedly accelerate upwards. In order to avoid that the flight area must be cleared out before the flight tests.

Multi-ranger Deck

The Multi-ranger Deck uses five laser sensors to measure the distance in the front/back/left/right/up direction. This enables the CrazyFlie to detect proximity to objects up to four meters away. This allows CrazyFlie to avoid obstacles.

Part A: Position PID Controller Tuning

Four PID controllers work together to facilitate position control of the CrazyFlie. The outermost position controller, the controller we will be tuning in this part of the lab, takes the target position as reference, and calculates a target linear velocity to be sent to the ***velocity***

controller. The velocity controller takes target linear velocity as reference, and calculates a target roll/pitch angle to be sent to the ***attitude controller***, which uses the target roll/pitch angle as reference and calculate target roll/pitch rate (angular velocity). Finally, the innermost ***rate controller*** takes the target roll/pitch rate as reference, and directly controls the motors to generate moment in X/Y axis to achieve desired angular rates. In this lab, we will vary the PID gains for the outmost position controller and treat the inner controllers as a black box. We shall use the following PID controller gains as a starting point.

Table 2. Initial PID Gains

Initial PID Gains	X-Axis	Y-Axis	Z-Axis
k_p	2.0	2.0	2.0
k_d	0.2	0.2	0.2

Recall that the current altitude is measured with the laser distance sensor, which reports the distance to the closest obstacle within the cone shaped detection range (Figure 8.3). Therefore, if the CrazyFlie is too close to a wall or a surface, the height reading may not be accurate. Also, note that the area of the cone increases with altitude.

To see the effect the PID gains have on vehicle performance, we will command the quadrotor to follow a set of waypoints, each time with different PID gains.

We shall use lab8_part1_pid.py to control the CrazyFlie quadrotor. The program allows the user to set the PID gains for the position and velocity controllers. Upon execution, the program instructs the quadrotor to take off, follow the selected set of waypoints, and land, while recording the quadrotor's position throughout the experiment.

There are two sections of the program you may want to change:

1) Waypoints

```
# Change the sequence according to your setup
#             x    y    z  YAW
sequence = [
    # make sure the x,y coordinate of the first waypoint is 0,0
    (0, 0, 0.4, 0),
    (0.5, 0, 0.4, 0),
    (0.5, 0.5, 0.4, 0),
    (0.5, 0.5, 0.1, 0),
]
```

The waypoints are given in reference to the world coordinate frame, which is aligned with the vehicle body frame when the program is executed. X is aligned with forward direction, Y with the leftward direction, and Z points upward.

You may change target waypoints to your liking, but make sure:

- [ ] You use the same waypoints for each experiment
- [ ] The waypoints are safe and achievable
- [ ] The first waypoint has (x,y) coordinates of (0,0)

2) PID Gains

```
# set PID gains
def set_gains(scf):
    # Default gain
    # posCtlPid.xKp: 2.0
    # posCtlPid.xKi: 0.0
    # posCtlPid.xKd: 0.0
    # posCtlPid.yKp: 2.0
    # posCtlPid.yKi: 0.0
    # posCtlPid.yKd: 0.0
    # posCtlPid.zKp: 2.0
    # posCtlPid.zKi: 0.5
    # posCtlPid.zKd: 0.0

    # Modify Position Gains
    scf.cf.param.set_value('posCtlPid.xKp', 2.0)
    scf.cf.param.set_value('posCtlPid.xKd', 0.5)
    scf.cf.param.set_value('posCtlPid.yKp', 2.0)
    scf.cf.param.set_value('posCtlPid.yKd', 0.5)
    scf.cf.param.set_value('posCtlPid.zKp', 2.0)
    scf.cf.param.set_value('posCtlPid.zKd', 0.5)

    # Modify Velocity Gains
    scf.cf.param.set_value('velCtlPid.vxKp', 10.0)
    scf.cf.param.set_value('velCtlPid.vxKi', 1.0)
    scf.cf.param.set_value('velCtlPid.vyKp', 10.0)
    scf.cf.param.set_value('velCtlPid.vyKi', 1.0)
    scf.cf.param.set_value('velCtlPid.vzKp', 15.0)
    scf.cf.param.set_value('velCtlPid.vzKi', 15.0)
```

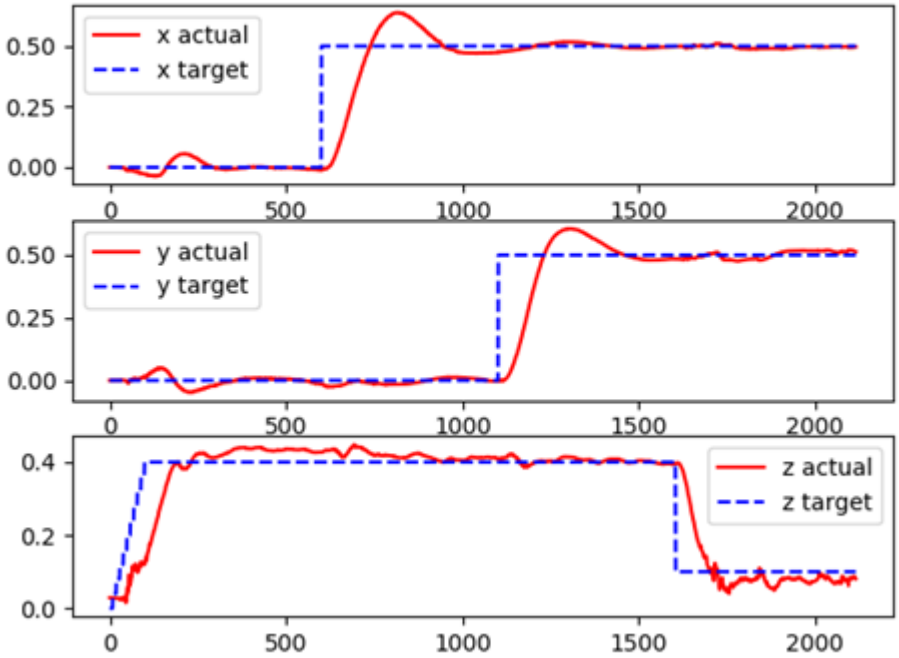
This is the part of the code where you can change the PID gains.

You should read the code to make sure you understand what to expect. After the program is finished, a log would be generated with the name lab8_log.npy. Make sure you change the name of this file before running the next experiment, otherwise it would be overwritten.

To view the log, run lab8_part1_plot.py. If you have changed the name of the log, edit the following line in lab8_part1_plot.py to read the correct log.

```
1.  data = np.loadtxt('lab8_log.npy', delimiter=',')
```

You should see the following figures (3D plot appears as the first figure is closed):



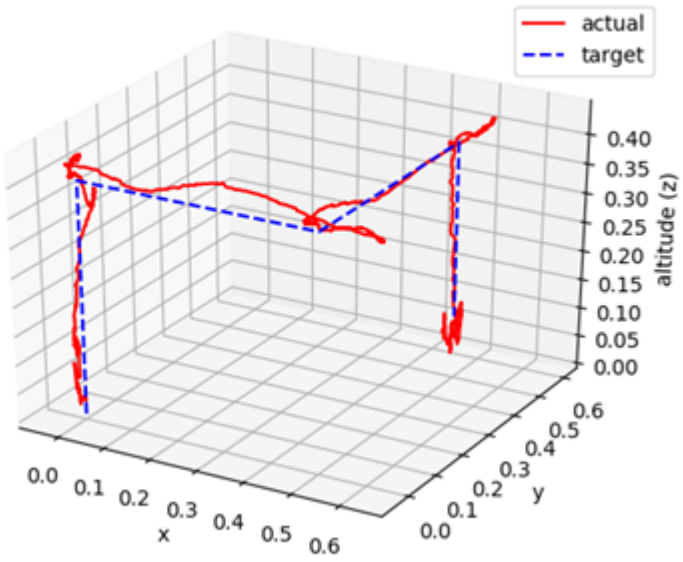


Figure 8.4 Example plots generated by lab8_part1_plot.py

Using these graphs, you will need to compare control performance with different gains in your lab report. Use the following gains for your experiments:

- Gain Set #1: Nominal Gain in Table 2
- Gain Set #2: Higher k_p
- Gain Set #3: Higher k_d

Table 3 Gain Set #2

Initial PID Gains	X-Axis	Y-Axis	Z-Axis
k_p	3.0	3.0	3.0
k_d	0.2	0.2	0.2

Table 4 Gain Set #3

Initial PID Gains	X-Axis	Y-Axis	Z-Axis
k_p	2.0	2.0	2.0

k_d	0.5	0.5	0.5

Procedure

1. Change PID gains to gain set #1
2. Run **python3 lab8_part1_plot.py**
3. Change log name, view log
4. Change PID gains to gain set #2
5. Run **python3 lab8_part1_plot.py**
6. Change log name, view log
7. Change PID gains to gain set #3
8. Run **python3 lab8_part1_plot.py**
9. Change log name, view log
10. Compare three logs, discuss the difference, and explain how each gain affects the outcome

---

Part B: Obstacle Avoidance Using a Distance Sensor

This part of the experiment will examine the obstacle avoidance for a small quadcopter. Obstacle avoidance is becoming increasingly important in small drone applications. This lab will use the quadrotor's proximity sensor to detect objects in front of the drone. The concept can be extended to detect objects on all sides.

The addition of obstacle avoidance capabilities is essential for indoor quadrotor flight. While the piloting of the quadrotor is the inner-loop control, the obstacle avoidance can be

considered the outer-loop control, overriding the pilot's inputs to prevent a collision.

The proximity sensor must be used in conjunction with an appropriate obstacle avoidance algorithm that is relevant to the flight conditions and scenario. In the case of this lab, the quadrotor will be not be allowed to come within 40 cm of the object in front of it. A minimum distance of 40 cm should therefore always be maintained. The quadrotor will set a velocity in the direction opposite to the incoming object. As another example, an algorithm could also be created to maintain proximity to a nearby object, i.e., maintaining a distance of 40 cm away from a target. This type of algorithm would be useful in swarm or teammate following applications.

For this experiment, we will try to mimic static obstacles and incoming obstacles. To mimic static obstacles, we place an obstacle within the safety margin and see how the vehicle reacts to it. For an incoming obstacle, we move the obstacle closer to the vehicle and see how the vehicle reacts.

The file **lab8_part2_push.py.** contains the following two functions:

1) `is_close`

```
1.  def is_close(range):
2.      MIN_DISTANCE = 0.4  # m
3.
4.      if range is None:
5.          return False
6.      else:
7.          return range < MIN_DISTANCE
```

This function determines whether there is an obstacle within the given minimum distance.

2) `main`

```
if __name__ == '__main__':
    # Initialize the low-level drivers (don't list the debug drivers)
    cflib.crtp.init_drivers(enable_debug_driver=False)

    cf = Crazyflie(rw_cache='./cache')
    with SyncCrazyflie(URI, cf=cf) as scf:
        start_position_printing(scf)
        with MotionCommander(scf) as motion_commander:
            with Multiranger(scf) as multiranger:
                keep_flying = True

                while keep_flying:
                    VELOCITY = 0.5
                    velocity_x = 0.0
                    velocity_y = 0.0

                    if is_close(multiranger.front):
                        velocity_x -= VELOCITY
                    if is_close(multiranger.back):
                        velocity_x += VELOCITY

                    if is_close(multiranger.left):
                        velocity_y -= VELOCITY
                    if is_close(multiranger.right):
                        velocity_y += VELOCITY

                    if is_close(multiranger.up):
                        keep_flying = False

                    motion_commander.start_linear_motion(
                        velocity_x, velocity_y, 0)

                    time.sleep(0.1)

            print('Demo terminated!')

    # Process the data before logging
    init_time = logged_data[0][0]
    for i in range(len(logged_data)):
        # Log time is in ms
        logged_data[i][0] = (logged_data[i][0] - init_time)*0.001

    with open('Experiment2.csv', 'w') as f:
        writer = csv.writer(f, delimiter =',')
        writer.writerows(logged_data)
```

Review carefully the code listed above, for discussion in the final report. To land the CrazyFlie, slowly put your hand on top of the vehicle.

Procedure

1. Place the CrazyFlie in the center of the floor mat and away from any other objects.

2. Open **lab8_part2_push.py**

3. Make sure the MIN_DISTANCE is set to 0.4 meters.

4. Make sure the VELOCITY is set to 0.5 m/s.

5. Run **lab8_part2_push.py:**
   python3 lab8_part2_push.py

6. The CrazyFlie should reach a steady hover before interfering. Place your hand or a flat object in front of the push sensor and observe as the CrazyFlie maintains a minimum distance of 0.2 m from the incoming object. Make observations regarding the return to steady state hover.

7. Try to imitate a static obstacle with your hand or object, commanding the CrazyFlie to maintain the minimum distance.

8. Save the position data.

9. Try to imitate an incoming obstacle with an object moving slower than 0.5m/s.

10. Save the position data.

11. Try to imitate an incoming obstacle with an object moving faster than 0.5m/s. Do not hit the vehicle!

12. Save the position data.

13. Close everything. **DO NOT SAVE THE CHANGES.**

Analysis & Lab Report

Part A

- Plot all data sets and discuss the differences. You can directly import csv file into Matlab for plotting. Your plots should look like example plots shown in Figure 8.4.

- Discuss and explain your observations of the hover performance of the Crazyflie flight very close to the ground.

- Discuss the impact of increasing the proportional and derivative gains on the waypoint controller performance.

Part B

- Plot all datasets and include a brief discussion on your experimental observations and the plotted results.

- Discuss the observed vehicle response to stationary versus moving obstacles.

- Discuss any important changes to the obstacle avoidance algorithm in **lab8_part2_push.py** you would suggest for it to be useful for flying through a field of stationary and moving obstacles.