Control Systems Implementation Laboratory: Control and Automation System Design Challenges

Kevin M. Passino and Nicanor Quijano
The Ohio State University
Department of Electrical Engineering
2015 Neil Avenue, Columbus Ohio, 43210

March 29, 2002

Abstract

This document lists a variety of control and automation system design challenges for the experiments that we have in our laboratory. Challenges are defined via a brief description of each plant and in some cases by providing a model of the plant. This document is not intended to be a *complete* compilation of plant models and design challenges. Plant model development is discussed in more detail in literature available in the lab for some of the experiments. The student should try to identify their own challenges that they would like to address in the laboratory. Moreover, the students will naturally encounter many specific challenges as they attempt to implement control strategies.

Contents

1 Conventional Control Systems

1.1 Control System Design Challenges

Here, we list some *generic* challenges encountered in control systems development, list some methods to overcome them, and then in the subsequent subsections outline the *specific* challenges for some of the plants in our laboratory. Moreover, we briefly outline a list of different methods you may consider to apply to the plants in our laboratory.

Assuming you use feedback control, the design objectives (also called "closed-loop specifications" or "performance objectives"), i.e., what you want the control system to achieve, can involve many factors, including:

- Tracking.
- Reducing the effects of adverse conditions and uncertainty.
- Behavior in terms of time responses (e.g., stability, a certain rise-time, overshoot, and steady state tracking error).
- Engineering goals such as cost and reliability.

You can consider applying a wide array of control techniques to the plants in our laboratory. The popular methods for control are the following:

- *Proportional-integral-derivative (PID) control:* Over 90% of the controllers in operation today are PID controllers (or at least some form of PID controller like a P or PI controller). This approach is often viewed as simple, reliable, and easy to understand. Often, heuristics are used quite effectively to tune PID controllers (e.g., the Zeigler-Nichols tuning rules).
- *Classical control:* Lead-lag compensation, Bode and Nyquist methods, root-locus design, and so on.
- *State-space methods:* State feedback, observers, and so on.
- *Optimal control:* Linear quadratic regulator, use of Pontryagin's minimum principle or dynamic programming, and so on.
- *Robust control:* H_2 or H_∞ methods, μ-synthesis, quantitative feedback theory, loop shaping, and so on.
- *Nonlinear methods:* Feedback linearization, Lyapunov redesign, sliding mode control, backstepping, and so on.
- *Adaptive control:* Model reference adaptive control, self-tuning regulators, nonlinear adaptive control, and so on.
- *State and parameter estimation:* Observers, Kalman filters, etc.
- *Stochastic control:* Minimum variance control, linear quadratic gaussian (LQG) control, stochastic adaptive control, and so on.
- *Discrete event systems:* Petri nets, automata, supervisory control, infinitesimal perturbation analysis, and so on.
- *Hybrid systems:* Control of systems that can most conveniently be represented by a combination of continuous time differential equations and discrete event system models.
- *Intelligent control:* Neural networks, rule-based control (fuzzy, expert), planning systems, attentional systems, learning (adaptive) control, genetic algorithms for off-line design and on-line tuning, foraging methods, game-theoretic approaches, general hierarhical and distributed intelligent control.

Most of these methods rely on the existence of "design models" and "truth models." Hence, a challenge that is present for each of the plants in the laboratory is how to develop models, or improve on the ones that we already have. There are two basic approaches for model development: physics-based model development and system identification. A typical challenge is to try to develop a more accurate design model, but one that is also useful for the systematic development of a controller. Another challenge is to develop a truth model and test its validity in simulation against the actual experiment.

1.2 DC Servo

The Quanser DC servo is shown in Figure 1. The input is the voltage V_{in}. There are three sensors. There are two different ones for shaft angle θ, a potentiometer and an encoder. There is also a tachometer for measuring $\dot{\theta}$. This DC servo serves as the actuation for the flexible joint, flexible link, ball and beam, and rotational inverted pendulum that are discussed below.

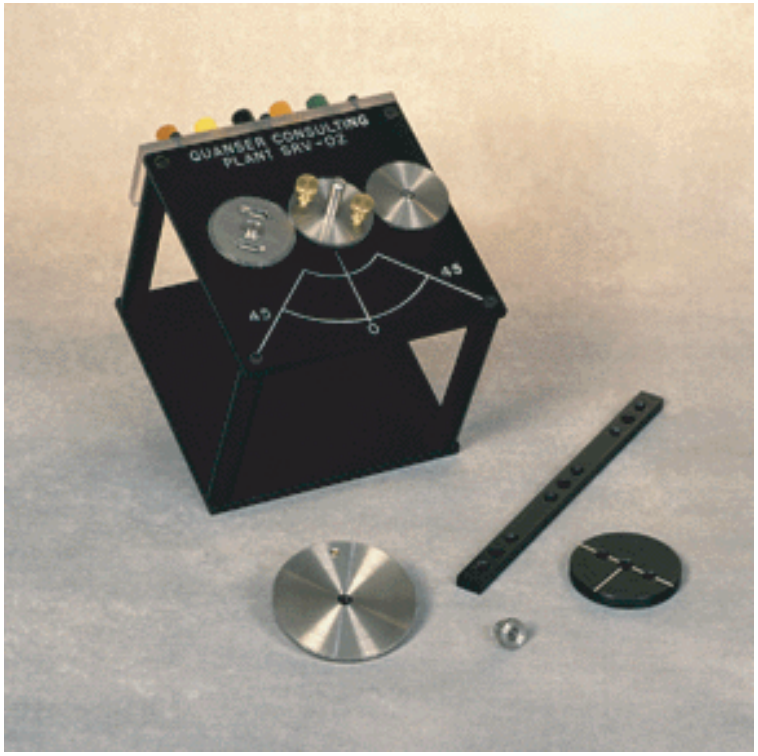

Figure 1: DC servo.

Suppose that you are concerned with position control. The transfer function for this case is

$$\frac{\theta(s)}{V_{in}(s)} = \frac{1}{s(s\frac{R_m J_{eq}}{K_m K_g} + K_m K_g)} \tag{1}$$

Here, $J_{eq} = K_g^2 J_m + J_I$. The parameters values are $R_m = 2.6\ \Omega$,$K_m = 0.00767\ V/rad/sec$, $K_g = 14:1$, $J_m = 3.87 \times 10^{-7}\ Kg.m^2$, and $J_I = 7.22 \times 10^{-6}\ Kg.m^2$ and this gives

$$\frac{\theta(s)}{V_{in}(s)} = \frac{1}{s(0.0020s + 0.1074)} \tag{2}$$

The root locus design for a unity feedback control system is given in Figure 2. From this figure, you can see that a proportional controller can speed up the response, but if the gain is too high, then you can expect overshoot in response to a step reference input. How would a PD controller work? A PID? Note that

this plant can be damaged if you place high frequency oscillations into the DC servo in V_{in} (see Quanser documentation) and hence this is a concern for experiments (flexible link, flexible joint, ball and beam, pendulum). This must be carefully taken into consider in design and implementation. .

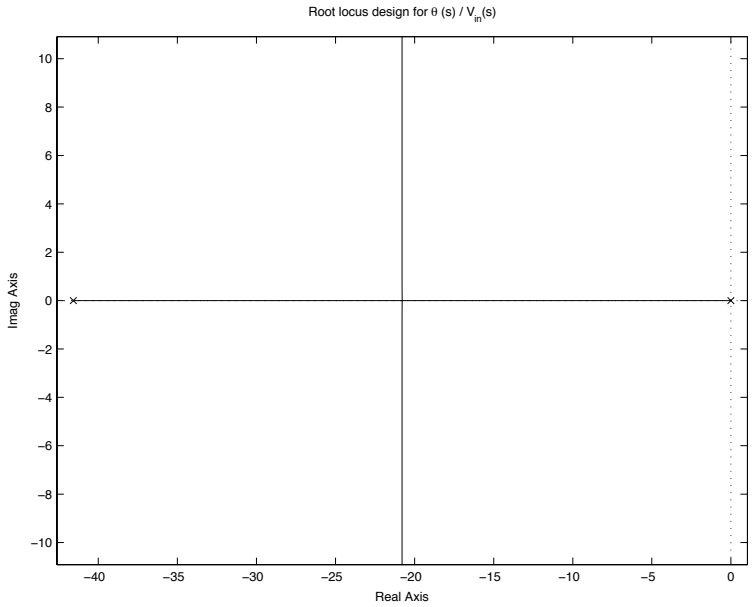

Figure 2: Root locus for control system for DC servo

1.3 Flexible Joint

The flexible joint is shown in Figure 3. A mass can be added to the end point of the arm and there are 9 different ways that springs can be attached, and there are three different types of springs each with a different spring constant. This allows for studying a variety of flexibility effects. There is an encoder for measuring the angle of the link relative to the motor shaft angle, and of course the sensors from the DC servo can be used.

The purpose of the experiment is to design a controller which allows you to command a desired tip angle position. For this experiment, we have the following variables:

- θ is the DC servo output shaft angle.
- K_{stiff} is a linear estimate of the joint stiffness, and is given by: $K_{stiff} = \frac{\delta M}{\delta \theta}|_{\theta=0}$, where M is the restoring moment. There are three different springs that are included with the experiment, springs #1, #2, and #3. There are three possible springs each with different stiffness characteristics and several anchor points on the flexible link (A, B, C for one end of the spring and 1, 2, and 3 for the other end). For the case where we use spring #2 at anchors $[A, 3]$, we have that $K_{stiff} = 1.6108\ Nm/rad$.
- α is the relative angle of the arm to the DC servo output shaft angle, i.e. it is the measurement of the angular deflection of the arm.
- The total inertia of the motor output is given by J_{hub}, and in this case is equal to 0.0021 Kgm^2.
- The total inertia of the arm is given as J_{load}, and in this case is equal to 0.0059 Kgm^2.
- R is the distance from the reference point of the joint to where you fix the spring in the anchor points. For the case described above, the distance is equal to 0.0762 m.
- K_g is the gear ratio, and for this case is equal to 14:1.

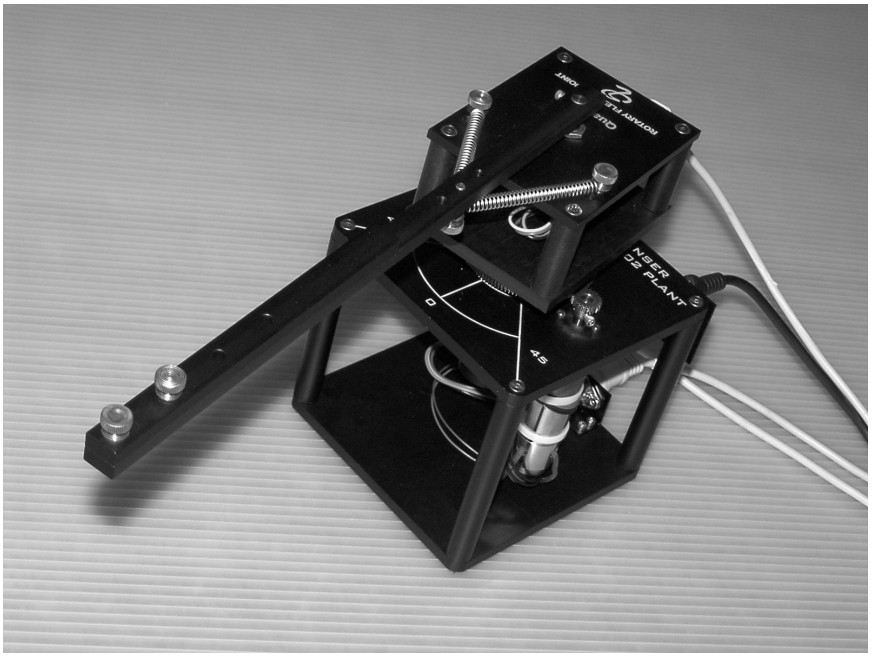

Figure 3: Flexible joint.

- K_m is one of the motor torque constant, and for this case it is equal to 0.00767 $V/rad/sec$.

We have:

```
L = 0.0318;          % m #1: 0.0254, #2: 0.0318 #3: default
d = 0.0318;          % m A: default; B: 0.0254; C: 0.0191
r = 0.0318;          % m
J_hub = 0.0021;      % Kgm^2
J_load = 0.0059;     % #0: 0.0021; #1: 0.0042; #2: 0.0048; #3: default, Kgm^2
R = 0.0762;          % #1: 0.101; #2: 0.089; #3: default, m
Km = 0.00767;        % V/rad/sec
Kgi = 14;            % 14:1
Kge = 5;             % it could be also 5:1
Fr = 1.33;           % #1: 0.83; #2: 1.33; #3: 1.78, N (refers to a spring property)
K = 368;             % #1: 220; #2: 368; #3: 665, N/m (refers to a spring property)
Rm = 2.6;            % Ohms

D       =   r^2 + (R-d)^2;
Kstiff  =   (2*R/D^(3/2))*((D*d-R*r^2)*Fr + (D^(3/2)*d - D*L*d + R*r^2*L)*K)
Kg      =   Kgi*Kge;
```

A state space model for this plant

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{3}$$

has $u = V_{in}$,

$$x = \left[\theta, \alpha, \dot{\theta}, \dot{\alpha}\right]^\top$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_{stiff}}{J_{hub}} & \frac{-K_m^2 K_g^2}{R J_{hub}} & 0 \\ 0 & \frac{-K_{stiff}(J_{load}+J_{hub})}{J_{hub} J_{load}} & \frac{K_m^2 K_g^2}{R J_{hub}} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{K_m K_g}{R J_{hub}} \\ \frac{-K_m K_g}{R J_{hub}} \end{bmatrix}$$

With the above parameter values, Matlab gives:

```
A =

            0            0   1.0000e+00            0
            0            0            0   1.0000e+00
            0   7.6705e+02  -5.2795e+01            0
            0  -1.0401e+03   5.2795e+01            0


B =

            0
            0
   9.8333e+01
  -9.8333e+01


C =

     1     0     0     0
     0     1     0     0
     1     1     0     0


eigenvalues =

           0
 -8.9913e+00 + 1.8254e+01i
 -8.9913e+00 - 1.8254e+01i
 -3.4813e+01


Transfer function:
-4.974e-14 s^3 + 98.33 s^2 - 7.458e-11 s + 2.685e04
---------------------------------------------------
      s^4 + 52.8 s^3 + 1040 s^2 + 1.441e04 s


zeros =

  1.9770e+15
  3.1017e-13 + 1.6523e+01i
  3.1017e-13 - 1.6523e+01i
```

```
poles =

           0
 -3.4813e+01
 -8.9913e+00 + 1.8254e+01i
 -8.9913e+00 - 1.8254e+01i


Transfer function:
-6.395e-14 s^3 - 98.33 s^2 - 4.184e-11 s
----------------------------------------
 s^4 + 52.8 s^3 + 1040 s^2 + 1.441e04 s


zeros =

  1.9770e+15
  3.1017e-13 + 1.6523e+01i
  3.1017e-13 - 1.6523e+01i


poles =

           0
 -3.4813e+01
 -8.9913e+00 + 1.8254e+01i
 -8.9913e+00 - 1.8254e+01i


Transfer function:
-5.684e-14 s^3 - 2.046e-12 s^2 - 5.821e-11 s + 2.685e04
------------------------------------------------------
        s^4 + 52.8 s^3 + 1040 s^2 + 1.441e04 s


zeros =

  1.9770e+15
  3.1017e-13 + 1.6523e+01i
  3.1017e-13 - 1.6523e+01i


poles =

           0
 -3.4813e+01
 -8.9913e+00 + 1.8254e+01i
 -8.9913e+00 - 1.8254e+01i
```

The three transfer functions are listed in the following order: V_{in} to θ, V_{in} to α, and then V_{in} to $\theta + \alpha$.

Notice that several terms in the numerators are essentially zero compared to the size of the other terms so that some of the zero locations are essentially at infinity as you would expect.

The objective is to regulate α to zero while moving the endpoint position to a specific angle. What type of compensation would be effective for this plant? What should the closed-loop specifications be? Can you measure the entire state and hence use state feedback control? Can you, via the use of state estimation, use only a subset of the available measurements and still come up with an effective control strategy? Can you derive a nonlinear model? Could you derive one of a form that is both accurate and of a form that is useful for some nonlinear controller design method (e.g., backstepping or feedback linearization)? What types of disturbances would you typically encounter for this plant? What if there is a weight change at the link endpoint? What if you change the configuration of the springs or the spring type to change the flexibility effects? How could you model this? Would an adaptive or robust control approach be useful to be able to construct a controller that could cope with a wide range of different endpoint masses and flexibility effects?

1.4 Flexible Link

The flexible link is show in Figure 4. As with the flexible joint the objective is fast endpoint position control. There is a strain gage sensor that is calibrated to provide one volt output for every inch of deflection at the tip, and of course the sensors from the DC servo can be used.

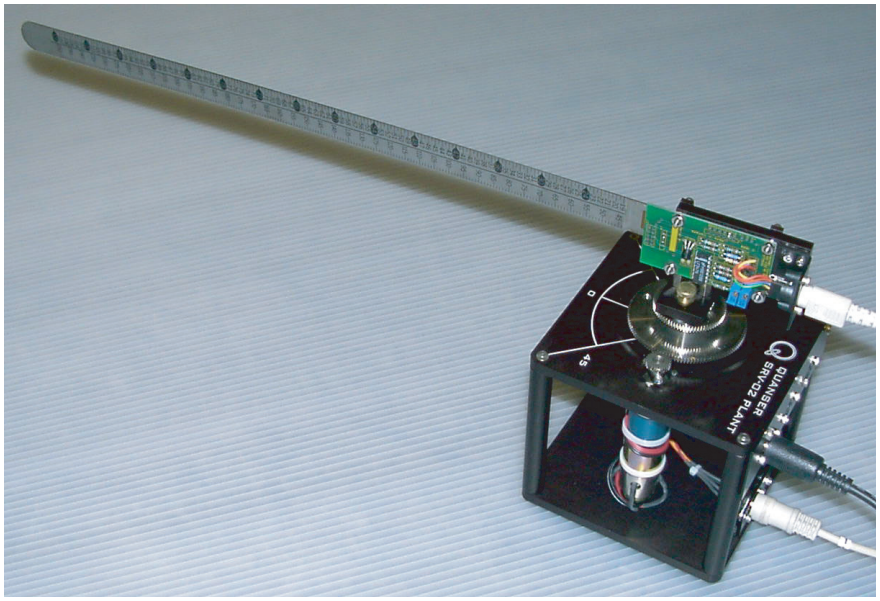

Figure 4: Flexible link.

This experiment is similar to the flexible joint experiment except that the link is flexible rather than the joint. The state space model has $u = V_{in}$, $x = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^\top$, and $y = \theta + \alpha$ with

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 2165 & -53.6 & 0 \\ 0 & -2797 & 53.6 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 99 \\ -99 \end{bmatrix}$$

and

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

The entire state, except for $\dot{\alpha}$ can be measured and $\dot{\alpha}$ can typically be estimated in software via a filtered derivative calculation.

In this case, since the matrix C is defined as above, the output will be the total deflection of the tip. Using these matrices, and the command ss2tf in Matlab, we obtain the transfer function

$$G_4(s) = \frac{-7.816 \times 10^{-14} s^3 - 6.366 \times 10^{-12} s^2 - 8.731 \times 10^{-11} s + 6.257 \times 10^4}{s^4 + 53.6 s^3 + 2797 s^2 + 33875 s} \tag{4}$$

Notice that the first three terms in the numerator are essentially zero compared to the constant term in the numerator. The poles and zeros are:

- Poles:
  1. 0
  2. -15.3251
  3. $-19.1374 + 42.9440i$
  4. $-19.1374 - 42.9440i$
- Zeros (as expected the zeros are essentially at ∞):
  1. $-4.6429 \times 10^5 + 8.0412 \times 10^5 i$
  2. $-4.6429 \times 10^5 - 8.0412 \times 10^5 i$
  3. 9.2849×10^5

The root locus plot for a unity feedback control system is shown in Figure 5. What type of compensation would be effective for this plant? What should the closed-loop specifications be? Can you measure the entire state and hence use state feedback control? Can you, via the use of state estimation, use only a subset of the available measurements and still come up with an effective control strategy? Can you derive a nonlinear model? Could you derive one of a form that is both accurate and of a form that is useful for some nonlinear controller design method (e.g., backstepping or feedback linearization)? What types of disturbances would you typically encounter for this plant? What if there is a weight change at the link endpoint? How could you model this? Would an adaptive or robust control approach be useful to be able to construct a controller that could cope with a wide range of different endpoint masses?

1.5 Ball and Beam

The ball and beam experiment is shown in Figure 6. There is a sensor for ball position, and of course the sensors from the DC servo can be used. The objective is to balance the ball at a desired location.

For the ball and beam experiment we have the distance x, the radius of the output gear r, gravity g, the angle of the beam α, the length of the beam L, and the angle of the DC servo as θ. With these variables,

$$\frac{X(s)}{\theta(s)} = \frac{-5rg}{7Ls^2} \tag{5}$$

so that the dynamics can be approximated as a double integrator. The parameter values are $g = 9.8\ m/s^2$, $L = 43.18\ cm$, and $r = 2.54\ cm$.

The root locus for a unity feedback cases covers the $j\omega$ axis. Based on this root locus what type of compensation would you expect to be effective? A PD controller? What should the closed-loop specifications be? What is the state? What is the state space model? Can you measure the entire state and hence use state feedback control? Can you derive a nonlinear model? Could you derive one of a form that is both accurate and of a form that is useful for some nonlinear controller design method (e.g., backstepping or feedback linearization)? What types of disturbances would you typically encounter for this plant? What if there is a weight change at one end of the beam? How could you model this? Would an adaptive or robust control approach be useful to be able to construct a controller that could cope with a wide range of different such masses?

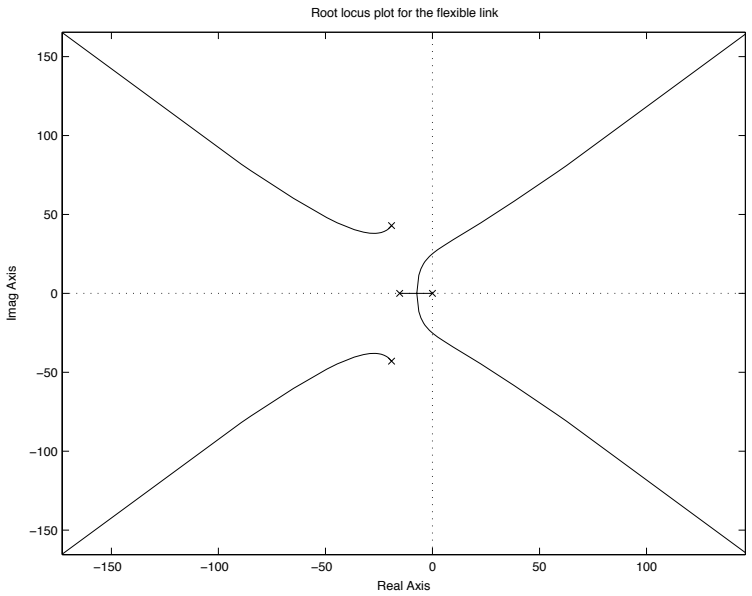

Figure 5: Root locus for Equation (4).

Figure 6: Ball and beam.

1.6 Inverted Pendulum

The rotational inverted pendulum is shown in Figure 7. There is a sensor for measuring the pendulum angle, and of course the sensors from the DC servo can be used. The inverted pendulum is a classical very well-studied and simple nonlinear control experiment.

The model for our inverted pendulum has

$$x = [\theta, \alpha_u, \dot{\theta}, \dot{\alpha}_u]^\top$$

Where θ is the angle made by the base and α_u is the angle made by the pendulum relative to the vertical. Sensors are provided for all the state elements except for $\dot{\alpha}_u$ (but perhaps you can use software to estimate it and $\dot{\theta}$ so that you do not need the tachometer for it). The state space model can be derived using physics and data sheets provide the parameters.

There are several types of control problems that can be considered:

1. *Swing-up:* Design a controller that will "destabilize" the equilibrium corresponding to the pendulum hanging down.

Figure 7: Pendulum.

2. *Balancing:* Design a controller that will balance the pendulum in its inverted position.

3. *Mode-control:* Design one controller that will switch modes from swing up to balancing so that the pendulum starts in the hanging position, is swung up, and then the balancing controller "catches" it and balances it in the inverted position (what Quanser calls "self-erecting").

The system basically behaves in a linear fashion about the inverted position so a linear controller works quite well for balancing. A simple linear controller can be used for swing-up. Mode switching involves some simple heuristics to switch the controllers.

The main challenges for this experiment come in only when you add some types of disturbances. For instance, challenges include adding weights to the penulum end point, perhaps even ones that will move (e.g., a bottle full of bolts). Then, the challenge is to design a single controller that can handle a range of such disturbances.

1.7 Water Tanks

The tank experiment is shown in Figure 8. This experiment has two coupled water tanks, tank #1 and #2, a single pump input, and level sensors for the two tanks.

You can study a variety of level control problems:

1. Level control in tank #1 (not using tank #2).

2. Pump feeds tank #1 which in turn feeds tank #2, and you try to design a controller to regulate the level in tank #2. Different orifice sizes can be used resulting in different plant characteristics.

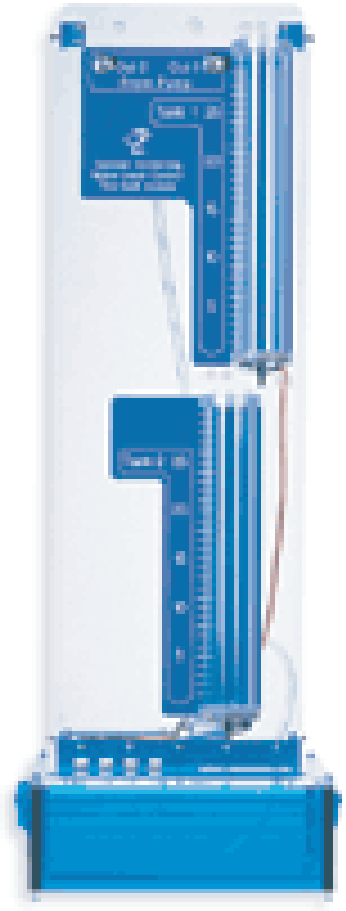

Figure 8: Tanks.

3. The pump feeds both the tanks and tank #1 also feeds tank #2. Try to regulate the level of tank #2.

Simple linear controllers (e.g., PI) will work reasonably well; however, there are delays in the process that complicate achieving high performance control. Robust control design in the face of the possibility of encountering different orifice sizes for the *same* controller implemented on different plants is a challenge. Moreover, nonlinear and stochastic effects also complicate the problem of achieving highly accurate level control.

1.8 Cube

The cube is shown in Figure 9. The objective is to design a controller that will balance the cube on its edge. One problem is that the angle that the vertical of the wedge makes with the horizontal cannot be measured, but its rate can. Simple integration of the rate does not result in an accurate estimate since the rate signal is biased. So, in a state space model of the cube there are two states that can be measured, two that can be easily estimated via software differentiation or integration, and one that is best estimated via the development of an observer. Weights can be added for disturbances.

A variety of challenges can be envisioned, but especially interesting is the need for an observer for the controlled variable.

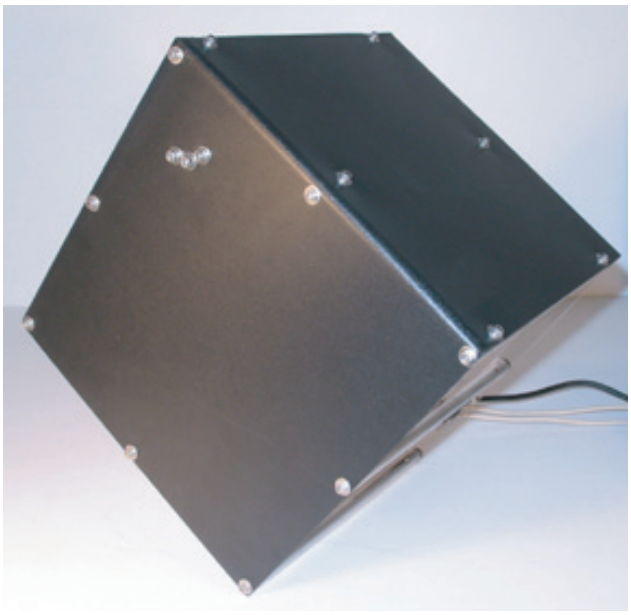

Figure 9: Cube.

1.9 Helicopter

The helicopter "model" is mounted on a fixed base and the two propellers are driven by two DC motors (two plant inputs) that control pitch and yaw as shown in Figure 10. There are measurements of the pitch and yaw via encoders. The objective is to design a controller that can allows you to command a desired pitch and yaw. Also, you can study the performance of the system with a human in the loop via the use of a joy stick.

This is a multi-input multi-output (MIMO) control system design problem that is coupled due to pitch-yaw coupling. Disturbances can be added via weights.

2 Complex Hierarchical and Distributed Control Systems

In this section we overview the plants that we have in the laboratory for the study of hierarchical and distributed control. Our motivation for including such problems is the presence of a wide variety of complex hierarchical and distributed control systems operating in industry. For example, there are commercial distributed control systems (DCS) that are typically composed of a network of sequencing logic and PID algorithms and are used for process control and manufacturing systems.

While a student may gain some experience with such control systems while in industry, it is typically not possible to have full-scale versions of such control systems available at a university due to cost constraints. Yet, it seems likely that the demands of industrial automation will press the universities to provide educational experiences in this area (e.g., research on principles and theory of complex control systems and laboratory experience with such systems). This provides the prime motivation for including a set of plants and control system design challenges for complex hierarchical and distributed control systems.

2.1 Automation Design Challenges

There are many application-specific challenges you will encounter in trying to implement a control system (e.g., interfacing issues with the plant and operator), and these can be particularly numerous and difficult for hierarchical distributed control systems. Here we try to provide a brief overview of some typical design challenges. First, it is important to keep in mind that all the control system design problems that we outlined in the last section are also encountered here (e.g., modeling, robustness, disturbance rejection, performance

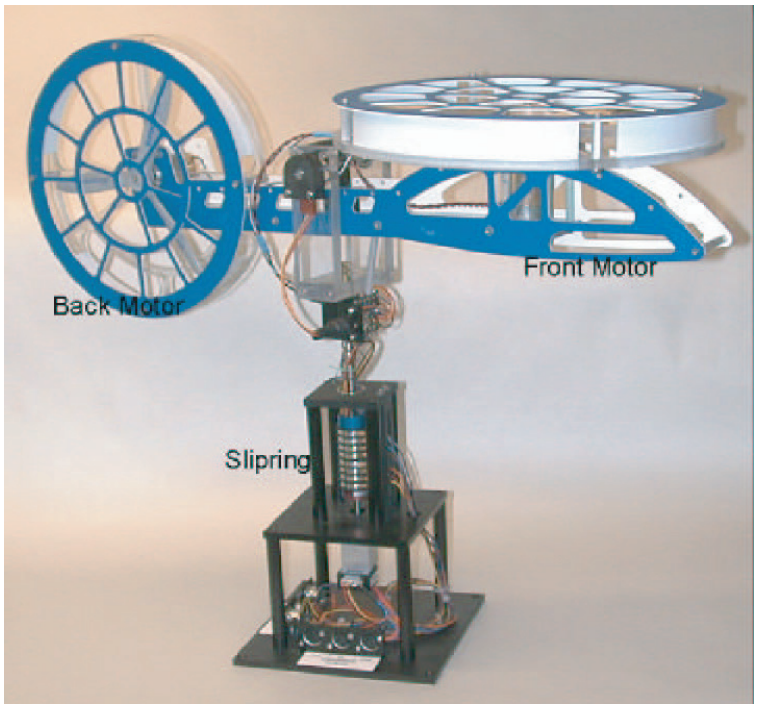

Figure 10: Helicopter.

specifications in terms of time responses). Here, we seek to identify some of the *additional* challenges that you could encounter in constructing a more sophisticated *automation* system that may be composed of many closed-loop control systems each with its own goals, and many other subsystems.

Some of the challenges that are encountered include:

- The various components of the control system may be implemented on different computer platforms (e.g., a workstation or PC) that use different operating systems.
- There is often the need to establish communications between the different platforms that are efficient and flexible.
- There are system timing issues where different portions of the overall system may be operating at different sampling times and in synchronous or asynchronous modes.
- There may be *many* control inputs and sensed outputs and interactions between spatially distributed parts of the plant.
- Parts of the plant may be best modeled with ordinary or partial differential equations while others may be most accurately described with a discrete event system (DES) model. This results in the need for hybrid system models and methods.
- It must be easy to establish hierarchies in the controller.
- It must be easy to develop the complex control systems so that coding and debugging can be implemented by *teams* of programmers (e.g., you may need special monitoring, design, and code generation tools).
- Sometimes, for large projects, it is useful to use software engineering methodologies (e.g., the "spiral method").

There are also classes of generic problems that are encountered in complex industrial automation systems, including the following:

1. *Sequencing and scheduling:* There are typically problems that demand the sequencing of operations (e.g., in a manufacturing system) and often the plant is best modeled by a finite state machine, Petri net, some other DES model, or even a hybrid system model. There is a need for closed-loop scheduling for dynamical systems that is robust and can be implemented in real-time.

2. *Resource allocation:* In many applications you must control what gets certain resources and at which times they get them. Sometimes such "resources" can be split, and other times they may not be. For instance, in flexible manufacting systems there is the problem of controlling (scheduling) which parts a machine (a resource) should process (the control problem is to measure queues of incoming parts and decide which queue to service). In the "gas lift optimization" problem in an oil field you have to allocate natural gas to different oil wells to maximize the overall production rate.

3. *Process-wide optimization:* An important problem is how to optimize decisions taken across the *entire* process from inventory control to the end product to ensure minimization of cost of production, yet the highest quality product.

4. *Hierarchical and distributed control:* As discussed above hierarchical and distributed control is often used in industry. It arises due to the need to "divide and conquer" large complex automation problems.

5. *Hybrid control:* There is often a need for hybrid combinations of linear, nonlinear, and discrete event type controllers that may operate synchronously or in an asychronous fashion.

It should be clear that there is no way that you can be expected to develop a full complex hierarchical and distributed control system. You can, however, address some of these fundamental challenges via the experiments that are described next since they have been constructed specifically to address typically-encountered "generic" challenges.

2.2 Building Temperature Control

The building temperature control experiment is shown in Figure 11. There are 8 rooms, each of which has a temperature sensor and heater (control input). Also, there are 4 fans that can be proportionally controlled (but which blow air only in one direction). The building has two floors which can be easily separated for study in isolation. Each floor has one wall that is easily removable for the study of different floor plans. There are many windows and doors, each of which can be manually opened or shut (e.g., to study the interactions between the rooms when a door is left open, or when a window is open and there is then an ambient temperature effect). Moreover, the fans can be placed in any window or door to move air in or out of a room. Also, it is easy to add more fans so that more plant inputs can be added (e.g., one coupling the two floors).

The plant currently has $8+4 = 12$ control inputs and 8 temperature sensor outputs and hence is a MIMO control problem. There are a wide variety of ways to induce disturbances. It is easy to study multi-zone and hierarchical distributed temperature control.

2.3 Distributed Dynamic Resource Allocation for Balls in Tubes

In this experiment there are two sets of four tubes each of which holds a ball that is levitated via sensing its height with an ultrasonic sensor and actuation via a proportionally controlled fan. One control objective could be to blow air so as to maximize the height of the balls in the tubes (i.e., to keep them levitated at a maximum height). The problem is that there is a manifold at the bottom of each set of tubes that couples them so that air must be allocated by the fans to the different tubes to maximize ball height (maximizing the height of only one ball will result in lowering the height of several other balls). Also, there is an electrically controllable disturbance on the manifold, and the two manifolds can be coupled to study cross-coupling effects between all eight tubes.

The experiment is only partially implemented as of the writing of this document (Spring'02). This quarter the challenge for this experiment is to finish the construction of the experiment, get the individual controllers operating for ball height regulation for the 8 tubes, and then design and test dynamic resource allocation strategies for the entire set of tubes.

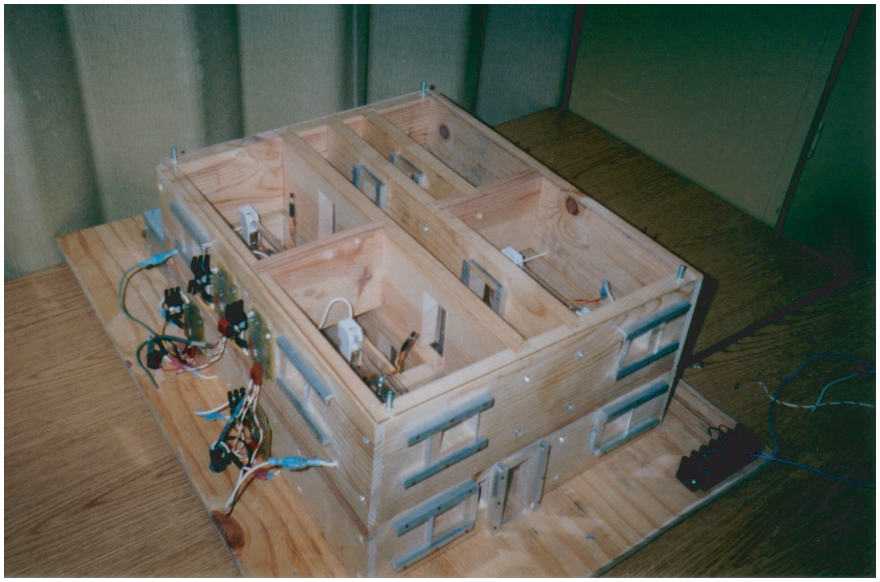

Figure 11: Building temperature control experiment.

2.4 Cooperative Scheduling for Multi-Agent Fire Control

In this experiment the problem is dynamically direct a motor shaft-mounted laser and fire it at a set of targets as they appear (like in an arcade). Moreover, there are two such "firing systems" that can be cooperatively controlled so as to maximize the point value of targets that are eliminated. There are PID loops for position control for the motors. The challenge is to eliminate as many high value targets in the shortest amount of time. The problem involves real-time scheduling of what to fire at by recognizing target appearance patterns.

The experiment is only partially implemented as of the writing of this document (Spring'02). A team of two persons will be assigned to complete the construction of the experiment and fully test it.

2.5 Uniform Planar Hybrid Temperature Control

We are also constructing an experiment to study how to control the temperature to be uniform on a flat surface. This involves implmenting a "grid" of zones each of which has a temperature sensor and heater. Then, it is possible to reconfigure the communications that are allowed between zone controllers and study hierarchical and distributed temperature control. Moreover, we plan to achieve a high number of zones via switching the application of a single analog input to many different heaters in different zones. This switching and the communications allow for a variety of DES-type system inputs that can be used in conjunction with the traditional continous voltage inputs to the heaters. Moreover, it is likely to provide an interesting "current resource allocation" problem for control of the heaters.

2.6 Software Development Tools for Complex Control Systems

The National Institute of Standards and Technology in the U.S. has developed over many years a software package, called the real-time control system (RCS) library, that can be quite useful in the development and implementation of hierarchical and distributed control systems.

The RCS software tool allows for controller implementation on a wide variety of platforms, and it uses a "neutral message language" to ensure consistent and flexible cross-platform communcations between "modules" that implement the various components of the system. Timing issues are easy to deal with via the given timer functions. Hierarchies are simple to establish, especially with the "design tool" which is a graphical environment for creating hierarchical controllers. The controller operation can be monitored and directed by a human operator at all its levels (and modules) via a graphical "diagnostics tool." Automatic code

generation is provided and tools are available to facilitate team-based development of a complex controller.

It is interesting to note that RCS is a free software package that can be used to solve many of the problems listed above, and is well-developed enough to use in practical industrial control and automation problems. A book covering all the details is:

> V. Gazi, M. Moore, K. Passino, W. Shackleford, F. Proctor, and J. Albus, *The RCS Handbook: Tools for Real Time Control Systems Software Development*, John Wiley and Sons, Pub., NY, 2000.