Design and Implementation of a Supervisory Safety Controller for a 3DOF Helicopter

by

Mariya A. Ishutkina

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author .....
Department of Aeronautics and Astronautics
May 15, 2004

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
Chairman, Department Committee on Graduate Students

Design and Implementation of a Supervisory Safety Controller for a 3DOF Helicopter

by

Mariya A. Ishutkina

Submitted to the Department of Aeronautics and Astronautics
on May 15, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This research effort presents the design and implementation of a supervisory controller for a 3DOF helicopter. This safety critical system is used in undergraduate laboratories in the Department of Aeronautics and Astronautics at MIT. There already exists a framework for designing a supervisory safety controller for motions about one axis. It is based on an analytical description of the safety region in state space. However, this framework cannot be easily extended to more complicated systems such as a 3DOF helicopter. In this thesis we present a different approach which uses a real-time simulation of linearized plant dynamics with a feedback law to ensure the system's safety. We describe the development of the system model, the design and implementation of the supervisory safety controller, integration of the safety controller as part of a remote laboratory and its evaluation based on its performance during laboratory exercises.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

This work would not be completed without the help and support of many individuals who I would like to acknowledge here.

First and foremost, I would like to thank my thesis supervisor Prof. Eric Feron for his patience, mentorship and support.

I would like to thank the department of Aeronautics and Astronautics for lending me the 3DOF helicopter.

I would also like to thank Prof. Antonio Vicino and Dr. Marco Casini of the University of Siena for the cooperation in making the helicopter operable from the Internet.

I would also like to thank the students in the lab; in particular, Vlad Gavrilets, Jan De Mot, Chris Dever, Tom Schouwenaars and Louis Breger, for the help and knowledge they have provided. I would like to thank the students in 35-303 for the chat sessions and calling me up in the middle of the night to tell me that the helicopter took off by itself.

I would like to thank Igor Tarashansky for helping me in generating ideas and finding bugs in the code.

Finally, I would like to thank my family for providing the moral support and kindness.

Funding for this research has been provided by the Air Force Research Laboratory grant F33615-01-C-1850.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Design of supervisory control for safety critical systems is a challenging problem because of its conflicting objectives. On one hand, the designer wants to guarantee the safety of the system, and on the other hand, he wants to give maximum freedom to the user in his choice of command inputs. For example, aircraft use automatic flight envelope protection to improve overall confidence and safety of the system while enabling aggressive maneuvering close to the envelope limits [19]. Some techniques have been developed to analyze the safety of a decoupled motion of a body about a single axis. These techniques use an analytical state space description of the safety region and do its verification using Hybrid Input/Output Automata [10]. In fact, it is possible to prove that the system's safety can be guaranteed using a feedback law consisting of a bang-bang controller followed by a linear quadratic regulator (LQR) [1]. However, this framework cannot be easily extended to more complicated nonlinear systems because it is analytically difficult to characterize a region in the state space where the safe operation of a plant can be guaranteed by a feedback control law. Instead, we show that a supervisory controller consisting of a real-time lookahead simulation of the closed-loop dynamics, which runs in parallel with the user's controller, is sufficient to ensure the system's safety. The supervisor takes as inputs the current system states and user's controller outputs and decides whether a feedback law in the form of a LQR would be able to keep the machine in a safe region by a timely override of user's control inputs. We do not have a theoretical proof that this scheme would keep the system always safe, but it has been implemented and works well in practice.

The safety critical system used in this research effort is a 3 Degree-of-Freedom (DOF) helicopter (Figure 1-1). This system has been a part of a laboratory facility provided by the MIT Department of Aeronautics and Astronautics since 1998. The laboratory facility has a total of six 3DOF helicopters. Three types of experiments are usually performed in the laboratory: a plant identification experiment, a 1DOF control experiment (e.g. control of the elevation angle θ), and a 3DOF control experiment (e.g. control of the elevation angle θ and travel velocity $\dot{\psi}$). Each year a hundred students on average use the machines for a period of six weeks. Over the lifetime of the lab, two machines have had to be replaced because of the unintentional misuse by the students. This 3DOF helicopter is a complex system which can be easily damaged if it hits the table support or strains its own joints. Direct interaction with the equipment can also present physical danger to an inexperienced user due to fast moving parts. The goal of this research is to reduce the hardware damage

and maintenance costs by redesigning the laboratory while preserving its functionality and providing a safe, educational and satisfying laboratory experience for the students.

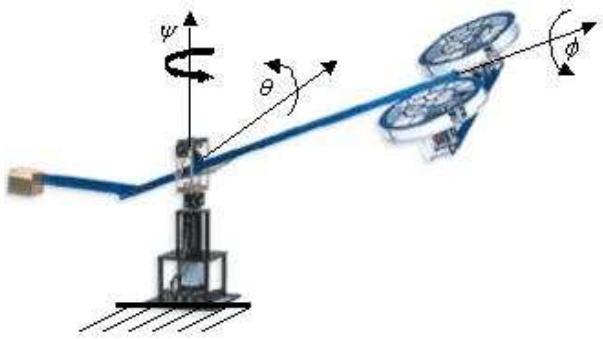


Figure 1-1: An illustration of the 3DOF helicopter [12].

As part of this research, we have made the 3DOF helicopter accessible from the Internet as a remote laboratory. The remote laboratories allow for utilization of the shared resources, make the laboratory experience more efficient, provide off-hours accessibility to the equipment, and provide educational opportunities for distance-learning students [6]. Remote laboratories also open up collaborations between different universities and allow students to use facilities not available in their own school. Of course, operation of such laboratories raises concerns about the robustness and safety of the equipment, as well as security. Therefore, most of the systems used as experimental platforms for remote laboratories are inexpensive and often non-safety critical systems. Some of the remote laboratories used in other universities include DC motors, water tanks, inverted pendulums, 2DOF helicopters [7], [4], [8]. In fact, the manufacturer of the 3DOF helicopter used in this research developed a remote laboratory, WebLab [2], but it was never made available to the public. The supervisory controller developed in this research effort allows us to use this safety critical system as an experimental platform for a remote laboratory.

To summarize, this research effort consists of three objectives. Firstly, a supervisory controller which ensures the safety of the system has to be developed; secondly, the controller has to be implemented and integrated as part of a remote laboratory to be available for distance learning; and, lastly, the functionality of the modified laboratory and student experiences using traditional and remote laboratories need to be evaluated and compared.

1.2 Thesis Overview

As the first step in the supervisory controller design, we develop a nonlinear dynamic model using basic physical principles. Then, we obtain the linearized model to be used for controller design. The system description and model development are presented in Chapter 2.

The linearized system model is used to develop a LQR controller which is a part of the supervisory controller. The supervisory controller prevents the 3DOF helicopter from hitting the table's surface or stressing the joints. It consists of three parts: initialization, real-time monitoring, ensuring the system's safety by a timely override of the user-designed controller, and landing. The design of the supervisory controller is described in Chapter 3.

Then we integrate the supervisory controller into the remote laboratory so that the 3DOF helicopter can be operated remotely through a standard web browser with the visual feedback to the user provided via live streaming video. The remote operation is hosted by the Automatic Control Telelab at the University of Siena, Italy [4]. The Telelab's architecture and implementation are reviewed in Chapter 4.

To compare student experiences using the remote and traditional laboratories, a set of laboratory experiments is performed . The experimental subjects are advanced undergraduate students taking a feedback control systems course. They are given two surveys to fill out: the first one as they are doing the laboratory and the second one to be filled out once the laboratory is completed. The results of the comparison are presented in Chapter 5.

In Chapter 6, the summary and contributions of the research are provided and future work is discussed.

Chapter 2

System Description and Model Development

2.1 System Description

The experimental platform used in this research is a three degree-of-freedom (3DOF) helicopter. The assembly is depicted in Figure 1-1, and in Figures 2-1 and 2-2 we show the schematics of the helicopter.

The 3DOF helicopter is mounted on a table top and its primary components are the main beam, the twin rotor assembly and the counterweight. The main beam is mounted on a bearing and slip-ring assembly which allows the rotor assembly to rotate in continuous circles without entangling the wires. This rotational motion is called *travel*. It occurs about a vertical axis which goes through the slip-ring and is perpendicular to the table. At the bearing and slip-ring assembly there is a pivot point which allows the main beam to raise and lower. This motion is described as *pitch* or *elevation*, and it occurs about an axis which goes through the slip-ring assembly and is parallel to the table. At the longer end of the main beam, there is another bearing whose axis is parallel to the beam. It allows a set of twin rotors driven by DC motors to pivot around that bearing. The rotational motion of the rotors is referred to as *roll*, and it occurs about an axis going through the main beam. The DC motors can provide either collective or differential (cyclic) voltage. The collective voltage results in the pitch motion of the main beam, and the differential voltage results in

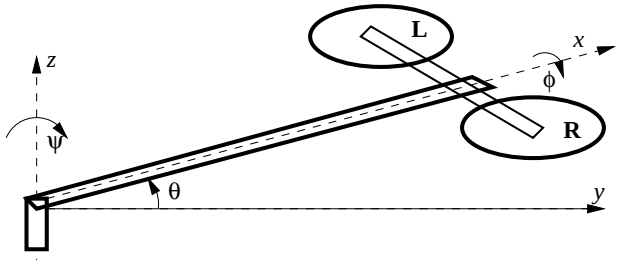


Figure 2-1: Schematic of the 3DOF helicopter

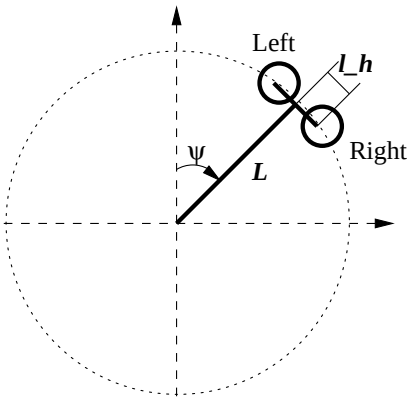


Figure 2-2: Top view

the roll motion. The roll motion of the rotors in turn gives rise to the travel motion of the assembly. At the other end of the main beam, there is a counterweight which reduces the power requirements on the motors by reducing the effective weight of the rotor assembly to only 31 grams in the horizontal position, but keeping the moment of inertia at 0.93 Nm. In order to derive the dynamic equations of the system, a coordinate system is used with its origin at the bearing and slip-ring assembly, with travel (ψ) being the circular motion of the main beam, pitch (θ) being the up and down motion of the beam, and roll (ϕ) being the motion of the rotor assembly. The corresponding angles are shown in Figure 1-1.

2.1.1 Nonlinear Model

The 3DOF helicopter model is derived applying Newton's second law to the rate of change of angular momentum. J_{xx}, J_{yy} and J_{zz} denote the corresponding moments of inertia. The moments of inertia terms can be found using the definition assuming that the point mass weight is concentrated at the motors and the counterweight. The moment of inertia of the main beam is approximated by a hollow cylinder formula whose dimensions can be measured and whose mass can be estimated based on the density of aluminum. The cross-product terms of the moments of inertia are difficult to measure; and, since they are usually small, they can be neglected. As shown in Figure 2-2, L is the length of the main beam from the slip-ring pivot to the rotor assembly and l_h is the distance from the rotor pivot to each of the propellers. The pitch motion of the helicopter is controlled by collective thrust applied to both propellers, defined as T_L and T_R, and is defined to be positive up from the horizontal position. If the force generated by the left motor is greater than the force generated by the right, then the resulting thrust differential produces a positive change in roll angle. If the roll angle is non-zero, then the components of thrust will produce a torque around the travel axis. Positive roll results in positive change of travel angle. There are also two moments due to the pendulum like motions of the appropriate centers of mass around the x and y axes. Lastly, there is aerodynamic drag affecting the travel motion of the helicopter. All the constants have been determined experimentally and are given in Table 2.1.

The equations of motion are as follows:

$$
\begin{aligned}
J_{yy}\ddot{\theta} &= -Mgl_\theta \sin(\theta+\theta_0) + (T_L+T_R)L\cos(\phi) && (2.1)\\
J_{xx}\ddot{\phi} &= (T_L-T_R)l_h - mgl_\phi \sin(\phi) && (2.2)\\
J_{zz}\ddot{\psi} &= (T_L+T_R)L\cos(\theta)\sin(\phi) + (-T_L+T_R)l_h\sin(\theta)\sin(\phi) - \text{Drag} && (2.3)\\
& && (2.4)
\end{aligned}
$$

where

$$
\begin{aligned}
T_L &= C_T^L \rho(\Omega_L R)^2 \pi R^2 && (2.5)\\
T_R &= C_T^R \rho(\Omega_R R)^2 \pi R^2 && (2.6)\\
\text{Drag} &= \frac{1}{2}\rho(\dot{\psi}L)^2(S_0 + S_0'\sin(\phi)) && (2.7)
\end{aligned}
$$

- R is the radius of the helicopter blades
- Ω is the frequency of the rotating blades (L = left motor and R = right motor)
- C_T is the thrust coefficient given by Equation 3.157 of Padfield [11]
- S_0 and S_0' are the effective drag coefficients times the reference area

- M is the total mass of the helicopter assembly

- m is the mass of the rotor assembly

Thrust Calculations

In order to obtain the nonlinear equations of motion, we develop an accurate model for the thrust based on the model presented in Padfield [11, pp. 115–124]. The equations for thrust are given in the previous section, where the thrust coefficient is

$$C_T = \frac{a_0\sigma}{2}(\theta_0(\frac{1}{3}+\frac{\mu^2}{2})+(\frac{\mu_z-\lambda_0}{2})+\frac{1}{4}(1+\mu^2)\theta_{tw}). \tag{2.8}$$

Here, a_0 is the blade lift curve, σ is the solidity ratio ($\sigma = \frac{N_{blades}Chord}{\pi R}$), θ_0 is the pitch at the root of the blade, θ_{tw} is the blade twist, λ_0 is the inflow, μ_z is the normalized inflow velocity ($\mu_z = \frac{V_z}{\Omega R}$), and μ is the advance ratio ($\mu = \frac{V}{\Omega R}$) . The varying parameters are μ_z and μ. They can be determined geometrically as

$$\begin{aligned}
\mu_z^L &= \frac{1}{\Omega_L R}[-\dot{\psi}\cos(\theta)\sin(\phi)L - \dot{\theta}\cos(\phi)L - \dot{\phi}l_h] \\
\mu^L &= \frac{1}{\Omega_L R}[\dot{\psi}\cos(\theta)\cos(\phi)L - \dot{\theta}\sin(\phi)L] \\
\mu_z^R &= \frac{1}{\Omega_R R}[-\dot{\psi}\cos(\theta)\sin(\phi)L - \dot{\theta}\cos(\phi)L + \dot{\phi}l_h] \\
\mu^R &= \frac{1}{\Omega_R R}[\dot{\psi}\cos(\theta)\cos(\phi)L - \dot{\theta}\sin(\phi)L]
\end{aligned}$$

Since C_T and λ_0 depend on each other, an iterative routine is implemented to compute $C_T(\mu, \mu_z)$ [11]. To determine the constant parameters in Eq. (2.8), the helicopter's thrust is computed in the hover position. Hover is defined as the position in which the main beam of the helicopter is horizontal and all the angles are zero. The rotors are weighted in this position and the thrust, denoted by $T_L^e + T_R^e$, needed to keep them in that position is taken to be equal to the weight. A strobe light is then used to determine the frequency of rotation for the blades at hover, which is denoted by Ω^e. A strobe light is also used to build a lookup table which describes the relationship between the frequencies of revolution (Ω_L and Ω_R) of the motors and the voltages applied to them.

Pendulum Effect

As the rotors rotate about the x-axis, the center of mass of the rotor assembly is displaced in a pendulum-like motion which results in a restoring torque. In order to find the displacement of the center of mass, which can also be referred to as the length of the pendulum, l_ϕ, the following procedure is used. The equation for a simple pendulum in Eq. (2.9) is reduced to the linearized equation for small angles in Eq. (2.10) giving an expression for the natural

frequency in Eq. (2.12):

$$J_{xx}\ddot{\phi} - mgl_\phi \sin(\phi) = 0 \quad (2.9)$$

$$J_{xx}\ddot{\phi} + mgl_\phi \phi = 0 \quad (2.10)$$

$$\ddot{\phi} + \frac{gl_\phi}{{l_h}^2}\phi = 0 \quad (2.11)$$

$$\omega = \frac{\sqrt{gl_\phi}}{l_h} \quad (2.12)$$

Then the period of oscillation is measured and $T = \frac{2\pi}{\omega}$ is used to find the length of the pendulum l_ϕ.

For the pitch motion of the helicopter, an analogous procedure can be used to find the displacement of the center of mass of the main beam (l_θ). The restoring torque term in Eq. (2.1) should also be modified in order to account for the non-zero angle θ_0 at which the whole assembly is in equilibrium. Equilibrium in this case is defined by the state in which both motors are turned off and the assembly is resting at its natural equilibrium. The elevation angle is defined to be zero at the hover position.

Drag Calculations

The drag force is computed using the standard equation Drag $= \frac{1}{2}C_D\rho v^2 A$, where C_D is the drag coefficient, ρ is the density of air, v is the flow speed, and A is the cross-sectional area of the body. In our case Drag $= \frac{1}{2}\rho(\dot{\psi}L)^2(S_0 + S_0' \sin(\phi))$, where S_0 and S_0' are the effective drag coefficients times the reference area.

2.1.2 Linearized Model

In this section we describe the linearization of the nonlinear model around the hover position. First, the thrust $T = C_T(\mu, \mu_z)\rho(\Omega R)^2\pi R^2$ is linearized using the Taylor series expansion

$$T = T^e + \left.\frac{\partial T}{\partial \Omega}\right|_{hover} \Delta\Omega + \left.\frac{\partial T}{\partial C_T}\frac{\partial C_T}{\partial \mu}\right|_{hover} \Delta\mu + \left.\frac{\partial T}{\partial C_T}\frac{\partial C_T}{\partial \mu_z}\right|_{hover} \Delta\mu_z$$

where $\mu^e = \mu_z^e = 0$, $\theta^e = \phi^e = \psi^e = 0$, $T^e = T_L^e = T_R^e$, $\Omega^e = \Omega_R^e = \Omega_L^e$ are the hover equilibrium conditions.

Padfield is used to find $\frac{\partial C_T}{\partial \mu_z}$, and the $\frac{\partial C_T}{\partial \mu}$ term is approximated numerically by solving for different values of C_T around the hover position while varying μ and keeping μ_z constant [11, p. 219]. As a result, the following equations are obtained:

$$\left.\frac{\partial T}{\partial \Omega}\right|_{hover} = 2\rho(\Omega^e R^2)\pi R^2\left(\frac{a_0\sigma}{2}\left(\frac{1}{3}\theta_0 - \frac{\lambda_0}{2} + \frac{1}{4}\theta_{tw}\right)\right.$$

$$\left.\frac{\partial T}{\partial \mu}\right|_{hover} = \rho(\Omega^e R)^2\pi R^2 \left.\frac{\partial C_T}{\partial \mu}\right|_{hover}$$

$$\left.\frac{\partial T}{\partial \mu_z}\right|_{hover} = \rho(\Omega^e R)^2\pi R^2 \frac{2a_0\sigma\lambda_0^e}{16\lambda_0^e + a_0\sigma}$$

and

$$\begin{aligned}
\Delta\mu_z^L &= \frac{1}{\Omega^e R}[-L\dot{\theta} - l_h\dot{\phi}] \\
\Delta\mu_z^R &= \frac{1}{\Omega^e R}[-L\dot{\theta} + l_h\dot{\phi}] \\
\Delta\mu^L &= \frac{1}{\Omega^e R}[L\dot{\psi}] \\
\Delta\mu^R &= \frac{1}{\Omega^e R}[L\dot{\psi}]
\end{aligned}$$

Finally, using the evaluated Taylor series for T_L and T_R and defining $\Delta T_L = T_L - T_L^e$ and $\Delta T_R = T_R - T_R^e$, the linearized system equations are written as:

$$\begin{aligned}
\ddot{\theta} &= \frac{1}{J_{yy}}[L(\Delta T_L + \Delta T_R) - Mgl_\theta \cos(\theta_0)\Delta\theta] \\
\ddot{\phi} &= \frac{1}{J_{xx}}[l_h(\Delta T_L - \Delta T_R) - mgl_\phi\Delta\phi] \\
\ddot{\psi} &= \frac{1}{J_{zz}}[L(T_R^e + T_L^e)\Delta\phi]
\end{aligned}$$

where

$$\begin{aligned}
\Delta T_L + \Delta T_R &= \frac{\partial T}{\partial\Omega}\Delta\Omega_L + \frac{\partial T}{\partial\Omega}\Delta\Omega_R - 2\frac{\partial T}{\partial\mu_z}\bigg|_{hover}\frac{1}{\Omega^e R}L\dot{\theta} + 2\frac{\partial T}{\partial\mu}\bigg|_{hover}\frac{1}{\Omega^e R}L\dot{\psi} \\
\Delta T_L - \Delta T_R &= \frac{\partial T}{\partial\Omega}\Delta\Omega_L + \frac{\partial T}{\partial\Omega}\Delta\Omega_R - 2\frac{\partial T}{\partial\mu_z}\bigg|_{hover}\frac{1}{\Omega^e R}l_h\dot{\phi}
\end{aligned}$$

The linear simulation step response and the system step response are compared in Figure 2-3 and Figure 2-4. The final system state matrices are presented in Appendix B.

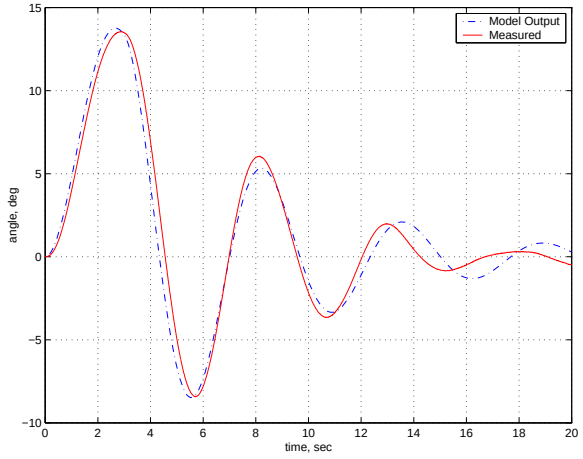


Figure 2-3: Roll response

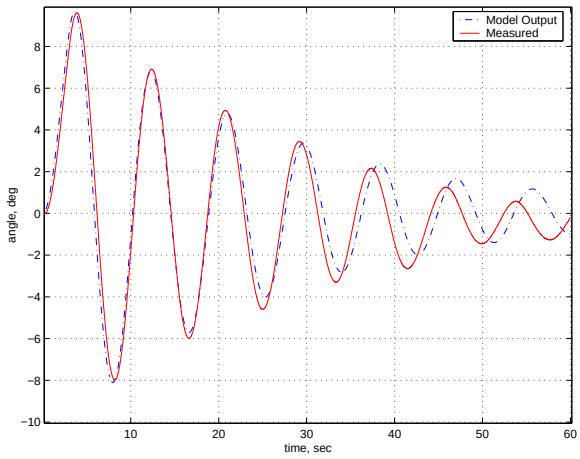


Figure 2-4: Pitch response

2.2 Frequency Response Identification

As part of a different research project, both linear and nonlinear models of the 3DOF helicopter have been developed using frequency response identification methods [5]. In that research, the models are derived using the Comprehensive Identification from FrEquency Responses (CIFER) tool developed by NASA [16]. These models assume that the DC motors behave as first-order systems in response to voltage inputs and hence use the collective and cyclic voltages as the model inputs. Since this description eliminates the need to model the thrust coefficient and makes the nonlinear model quite simple and intuitive, the models described below are provided to remote laboratory users for simulation.

2.2.1 Nonlinear model derived using CIFER

Figure 2-1 shows a simplified representation of a 3DOF helicopter, where the three degrees of freedom are: the roll motion (ϕ), the pitch or elevation motion (θ), and the travel motion (ψ). We also denote the rolling moment generated by the rotors as τ_{cyc}, and the elevation moment as τ_{coll}.

Using a combination of first-principle modeling and identification, the coupled nonlinear dynamic equations are:

$$\ddot{\psi} = -a_1\dot{\psi} + a_2(\alpha\tau_{coll} + 1)\sin\phi \tag{2.13}$$

$$\ddot{\phi} = -b_1\dot{\phi} - b_2\sin\phi + b_3\tau_{cyc} \tag{2.14}$$

$$\ddot{\theta} = -d_1\dot{\theta} - d_2\sin\theta + d_3\tau_{coll}\cos\phi \tag{2.15}$$

where

- Eq. (2.13) describes travel dynamics. Here, $-a_1\dot{\psi}$ is the aerodynamic drag (as well as some amount of mechanical friction at the hinge) and $-a_2(\alpha\tau_{coll} + 1)\sin\phi$ is the propulsive thrust component.
- Eq. (2.14) describes roll dynamics. Here, the second order equation of motion accounts for the lightly-damped oscillatory characteristics: $-b_1\dot{\phi}$ is the rotor damping, $-b_2\sin\phi$ is the restorative spring torque (pendulum) and $b_3\tau_{cyc}$ is the rotor torque.
- Eq. (2.15) describes elevation dynamics. Here, the second order equation of motion accounts for the pendulum-like oscillatory characteristic due to the presence of the counterweight. $-d_1\dot{\theta}$ is the aerodynamic and mechanical damping torque; $-d_2\sin\theta$ is the restorative spring torque and $d_3\tau_{coll}\cos\phi$ is the vertical lift thrust component.

The helicopter system is actuated by two rotors each driven by an electric DC motor. The rotors can operate collectively, to produce lift (elevation torque), or differentially to produce a rolling torque. The two modes can be superimposed. The actuation control inputs are:

- V_{cyc} – electric voltage that results in differential change in the two rotor speeds. A positive voltage produces a positive rolling moment, and
- V_{col} – electric voltage that controls the speed of the two propellers collectively. A positive voltage produces a positive lifting moment.

The first-order actuation dynamics account for the inertial and electric motor lag involved in the development of the rotor thrust and can be described by the following differential equations:

$$\begin{aligned} \dot{\tau}_{cyc} &= -c_1\tau_{cyc} + c_2 V_{cyc} \\ \dot{\tau}_{coll} &= -e_1\tau_{coll} + e_2 V_{coll} \end{aligned}$$

The model parameters as well as their identified values are given in Table 2.2.

2.2.2 Linear model derived using CIFER

A nonlinear model described in the previous section is linearized using the Taylor series expansion technique. The linearized dynamics can be presented as:

$$\delta\dot{\mathbf{x}} = \left[\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right]_* \delta\mathbf{x} + \left[\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right]_* \delta\mathbf{u},$$

where * denotes $\mathbf{x} = \mathbf{x}_{trim}$, $\mathbf{u} = \mathbf{u}_{trim}$, and $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_{trim}$ and $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{trim}$. The state vector is $\mathbf{x} = [\psi\ \dot{\psi}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \tau_{cyc}\ \tau_{coll}]^T$, and the input vector is $\mathbf{u} = [V_{cyc}\ V_{coll}]^T$. Then, the linearized system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ is given by

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\tau_{cyc}} \\ \dot{\tau_{coll}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -a_1 & a_2(\alpha\tau^*_{coll}+1)\cos\phi^* & 0 & 0 & 0 & 0 & a_2\alpha\sin\phi^* \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -b_2\cos\phi^* & -b_1 & 0 & 0 & b_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -d_3\tau^*_{coll}\sin\phi^* & 0 & -d_2\cos\theta^* & -d_1 & 0 & d_3\cos\phi^* \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -e_1 \end{bmatrix} \begin{bmatrix} \delta\psi \\ \delta\dot{\psi} \\ \delta\phi \\ \delta\dot{\phi} \\ \delta\theta \\ \delta\dot{\theta} \\ \delta\tau_{cyc} \\ \delta\tau_{coll} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ c_2 & 0 \\ 0 & e_2 \end{bmatrix} \begin{bmatrix} V_{cyc} \\ V_{coll} \end{bmatrix}$$

For the trim condition at hover, we have $\dot{\psi} = \dot{\phi} = \dot{\theta} = 0$ and $\psi = \phi = \theta = 0$, and the matrices reduce to the ones given in Appendix B. The coefficients have been adjusted to fit the model better by observing the system step responses and comparing them to the simulation.

The nonlinear and linearized simulations are implemented in Simulink and the final diagram of the helicopter model is shown in Figure 2-5. For reference, the nonlinear simulation implemented as a MATLAB S-function is provided in Appendix A.

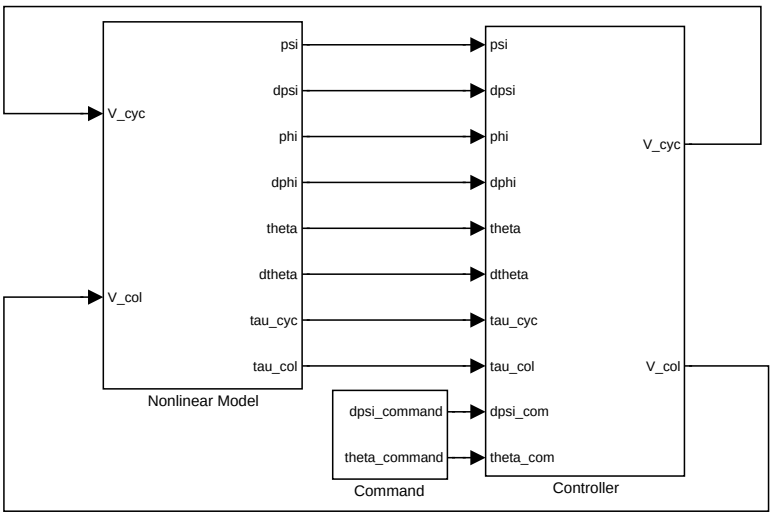


Figure 2-5: Simulink simulation model

Table 2.1: Parameter Values

Parameter	Value	Unit	Description
m	1.15	kg	mass of the rotor assembly
M	3.57	kg	mass of the whole setup
m_0	0.031	kg	effective mass at hover
L	0.66	m	length from pivot point to the heli body
l_h	0.177	m	length from pivot point to the rotor
J_{xx}	0.036	Nm	moment of inertia about x-axis
J_{yy}	0.93	Nm	moment of inertia about y-axis
J_{zz}	0.93	Nm	moment of inertia about z-axis
R	0.1	m	radius of the rotor
ρ	1.23	kg/m^3	density of air
g	9.8	m/s^2	gravitational constant
l_ϕ	0.004	m	length of pendulum for roll axis
l_θ	0.014	m	length of pendulum for pitch axis
θ_0	0.45	rad	neutral theta == setup in equilibrium
θ_{tw}	-0.93		blade twist
σ	0.14		rotor solidity
a_0	4.5		rotor blade C_{L_α}
S_0	0.012	m^2	C_D times the reference area when $\phi = 0$
S_0'	0.168	m^2	C_D times the reference area when $\phi = \pi$
Ω_L^e	183	rad/s	left rotor frequency at hover
Ω_R^e	183	rad/s	right rotor frequency at hover
T_L^e	0.15	N	left rotor thrust at hover
T_R^e	0.15	N	right rotor thrust at hover

Table 2.2: CIFER Nonlinear Parameter Values

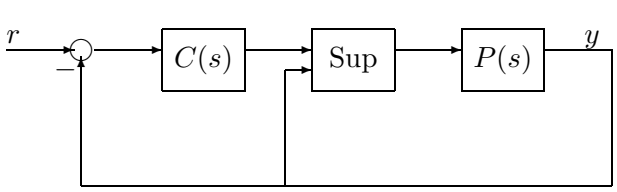
Parameter	a_1	a_2	b_1	b_2	b_3	c_1	c_2
Value	0.2517	0.2105	0.3290	1.5664	16.2	7.32	1
Parameter	d_1	d_2	d_3	e_1	e_2	α	
Value	0.1011	0.504	1.34	6.16	1	4	

Chapter 3

Controller Development

3.1 Supervisory Controller

The main goal of a supervisory control for a safety critical system is to ensure the system's safety while interfering with the user's controller as little as possible. A common architecture of the supervisory control is shown in Figure 3-1. In the figure, the supervisor takes as inputs the current system states and user's controller outputs and decides whether to feedthrough or override the user's control inputs to keep the system safe.



r = user reference
y = state measurement
$C(s)$ = user controller
$P(s)$ = Plant
Sup = supervisory controller

Figure 3-1: Block diagram of the supervisory control

3.1.1 Elevation Safety

One way to design a supervisory controller is to analytically define the safe region of the plant in the state space and then verify if a certain feedback law can guarantee the safe operation of the system. This approach is used to analyze the safety of a decoupled elevation motion of the 3DOF helicopter [10]. In [10], the simplified equation of elevation dynamics is considered and an analytical state space description of the safe region is determined by solving the equation of motion assuming the worst possible user input. Following the description of the safe region, a feedback law in the form of a bang-bang controller followed by a LQR controller is shown to guarantee the safety of a 1DOF system. The verification of system safety is then done using the modeling framework of Hybrid Input/Output Automata. This verification framework is able to account for both continuous (plant dynamics) and discrete (sampling rate) aspects of the system. The supervisory controller is designed such that it is running in parallel with the user's controller observing the elevation angle and velocity and overriding the user's controller if the estimated next worst possible state of the system is not in the safe region.

3.1.2 Full 3DOF Safety

It is difficult to extend this technique to the three degree of freedom motion of the helicopter. In particular, in order to obtain an analytical description of a safe region in three dimensions, one would have to find a closed form solution to a set of nonlinear differential equations describing the system dynamics. The nonlinear differential equations would have to be solved for the case of the worst possible user input, which is not easily determined for the full 3DOF motion of the helicopter. Hence, we follow a different approach to ensure the safety of the system. Namely, we use a lookahead simulation as a part of a structured supervisory controller which operates in three phases:

1. Initialization of the Plant.

2. Real-time monitoring of the Plant with lookahead simulation.

3. Returning the Plant to its initial position after finishing the experiment.

Initialization

The 3DOF helicopter used in this research can hit the table either if the elevation angle is too low following the negative V_{col} command or if the roll angle is too high following a large V_{cyc} command. The mechanical system is designed such that when it is in hover position and the roll angle is 90°, the rotor assembly barely clears the table's surface. The roll dynamics of the plant are highly undamped and are the fastest in the system, followed by the elevation dynamics which are in turn followed by the travel dynamics. The mechanical setup is mounted on the edge of the table such that when it is not used, it is resting on the table's surface. The initial position coordinates are $[\theta, \psi] = [-20^\circ, -180^\circ]$. If one moves the 3DOF helicopter by 180° in travel while applying hover voltage, it is then located over a free space not constrained by the table's surface. The coordinates of this position are $[\theta, \psi] = [0^\circ, 0^\circ]$. While in this position, the 3DOF helicopter can still strain the pitch joint if excessive collective voltage is applied. We also note that student's initial trial of the controller usually destabilizes the system and is most dangerous for system's integrity. Following these observations, we design a supervisory controller such that it always initializes the plant by bringing it to the $[0, 0]$ position before the user's controller is turned on. The initialization takes place during a preset time after the user turns on the 3DOF helicopter. This is done so that the user knows exactly when his controller becomes active. Our Initialization controller completes its operation in thirty seconds.

Real-time monitoring

Once the initialization is complete, the supervisory control is achieved through real-time monitoring of the system. We design the supervisory controller assuming the user's controller is arbitrary. The supervisory controller consists of a real-time lookahead simulation of the dynamic response of the closed-loop system which runs in parallel with the user's controller. The supervisor takes as inputs the current system states and user's controller outputs and decides whether a feedback law would be able to keep the machine in a safe region by a timely override of user's control inputs. The simulated dynamic response of the closed-loop system consists of a system model linearized over an appropriate trim point and a LQR controller whose command is to bring the 3DOF helicopter to the nearest trim state.

The following trim points are chosen for the linearization of the model presented in Chapter 2: $[\theta, \dot{\psi}] = [0, 0], [0.175, 0], [-0.175, 0], [0, 0.5], [0, 1]$. The units are $[\theta, \dot{\psi}] = [rad,\ rad/sec]$. Then, a LQR controller for each set of trim points is designed based on an appropriate linearized model. The controller used in the lookahead simulation is the actual controller which would be used should the supervisor decide to override user's commands. It would have been ideal if we could use a combination of nonlinear dynamic simulation with an appropriate LQR controller, however, the nonlinear simulation cannot be implemented because it takes more time than the real-time system clock can afford. This difficulty can potentially be alleviated in the future. We simulate the system response for a lookahead time of six seconds, which is determined experimentally and is constrained by the real-time nature of the process.

The safe region is determined by considering the physical limitations of the plant and describes the relationship between the plant's elevation, roll and travel angles. The safe region is not uniform since the movement of the 3DOF helicopter is less constrained around the $[0, 0]$ position and more constrained when the rotors have the possibility of hitting the table. We determine the maximum and minimum elevation angles which can be reached without straining the upper and lower joints as a result of excessive collective commands. We also derive a relationship between the elevation and roll angles allowing us to check if the rotors are able to clear the table support. This relationship depends on the current travel position of the plant. The safe region used for the lookahead simulation is a subset of the actual safe region. The safe region in the simulation has been determined experimentally.

In addition to the real-time simulation, we are using the hardbound saturations on the user reference inputs and voltage commands enforced by the physical limitations of the system.

We do not have a theoretical proof that this scheme would keep the system always safe, but it has been implemented and works well in practice. Experimental testing of this supervisory controller keeps the system out of the unsafe region in 95% of the runs and can be made even better if the safe region in the simulation is defined more conservatively. The rare cases of over-stressed joints are still safer compared to the flipped machines of the unsupervised laboratory.

Landing

When the user sends the stop command to the 3DOF helicopter, the Landing controller brings the machine to hover over the initial position and safely lands it on the table.

3.1.3 Supervisor in action

Figure 3-2 shows the "θ with super_lqr" data from an actual run of the user's controller of the elevation angle. It takes t_0 seconds for the machine to initialize at the beginning of the experiment. At t_0, the user's controller is turned on and when it is deemed unsafe at time t_1, the supervisor subsequently overrides the user's commands and brings the machine to the initial position. The elevation angle reaches the maximum of 0.614 radians which is marked by the lower horizontal line in the figure. Note that the supervisor is implemented such that the user can only view the data between the times t_0 and t_1. The other plot on the figure is "θ no super_lqr" . This plot shows the data from the run of the same user's elevation angle controller with the supervisory controller consisting of the initialization and landing phases but no real-time monitoring. As can be seen in the figure, the user's controller goes

unstable and eventually hits the support joint at elevation angle 0.643 radians marked by the upper horizontal line in the figure.

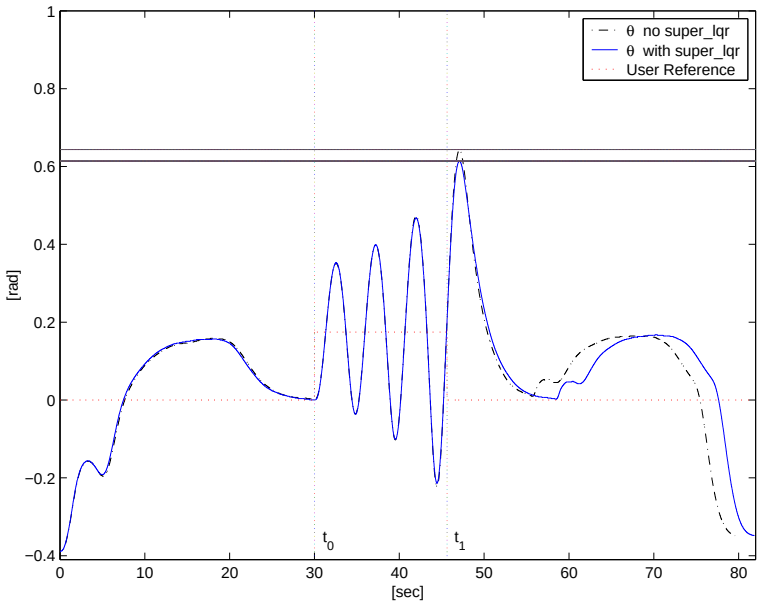


Figure 3-2: An illustration of an Elevation Controller run

3.2 Control System Design

Since full state feedback is available, a linear quadratic regulator was chosen as the system controller due to its inherent robustness properties [1], [14]. Several LQR controllers are used for each task of the supervisory controller outlined in the previous section. The pseudocode of the supervisory controller is presented in Algorithm 3.2.1.

Algorithm 3.2.1: SUPERVISORY CONTROLLER($states, user\_col, user\_cyc$)

$$
\begin{array}{l}
\textbf{if } (StopExperiment) \\
\textbf{then } \begin{cases} land\_lqr(-20°, 180°) \\ V\_col = land\_col \\ V\_cyc = land\_cyc \end{cases} \\
\textbf{else if } (time < 30\ sec) \\
\textbf{then } \begin{cases} init\_lqr(0,0) \\ V\_col = init\_col \\ V\_cyc = init\_cyc \end{cases} \\
\textbf{else } \begin{cases} safe\_flag \leftarrow do\_lookahead\_sim(states, u\_col, u\_cyc) \\ \quad \textbf{if } (safe\_flag) \begin{cases} V\_col = u\_col \\ V\_cyc = u\_cyc \end{cases} \\ \quad \textbf{else } \begin{cases} super\_lqr(\theta\_trim, \dot{\psi}\_trim) \\ V\_col = super\_col \\ V\_cyc = super\_cyc \end{cases} \end{cases} \\
\textbf{return } (V\_col, V\_cyc)
\end{array}
$$

3.2.1 Real-time monitoring controller (*super_lqr*)

The real-time monitoring uses a LQR with the Elevation and Travel Velocity ($[\theta, \dot{\psi}]$) as the tracking variables. Here we present the design procedure for this controller. Our task is to compute the feedback gains $\mathbf{K}$ of $\mathbf{u} = \mathbf{K}\mathbf{x}$ for the plant input which is defined as

$$\mathbf{u} = [\Omega_{sum} \;\; \Omega_{diff}]^T$$

where $\Omega_{sum} = \frac{\Omega_L + \Omega_R}{2}$ and $\Omega_{diff} = \frac{\Omega_L - \Omega_R}{2}$.

We start out by augmenting the original state vector to include the integrators in order to guarantee zero steady state error:

$$\mathbf{x} = [\theta \; \psi \; \phi \; \dot{\theta} \; \dot{\psi} \; \dot{\phi}]^T \Longrightarrow \bar{\mathbf{x}} = [\theta \; \phi \; \dot{\theta} \; \dot{\psi} \; \dot{\phi} \; \theta_i \; V_i]^T$$

where $\dot{\theta}_i = \theta^{cmd} - \theta$ and $\dot{V}_i = \dot{\psi}^{cmd} - \dot{\psi}$.

The feedback gains for this system are calculated using a linear quadratic regulator approach with diagonal Q and R matrices used in the quadratic cost function.

$$\int_0^\infty (x^T Q x + u^T R u)\, dt$$

The initial entries of Q and R are picked using Bryson's rule [3] and the details of obtaining final Q and R matrices and feedback gains $\mathbf{K}$ using the **lqr** function in MATLAB are presented in Appendix C. The feedback gains are designed such that the closed loop frequencies of the system are lower than its natural frequencies and the damping ratio is near 0.7. Finally, we partition the 2x7 matrix $\mathbf{K}$ such that

$$\mathbf{u} = \mathbf{K}\mathbf{x} \Longrightarrow \mathbf{u} = \mathbf{K}_1(\mathbf{x}_1^{cmd} - \mathbf{x}_1) - \mathbf{K}_2\mathbf{x}_2 - \mathbf{K}_3\mathbf{x}_3$$

where $\mathbf{x}_1 = [\theta \; \dot{\psi}]^T$, $\mathbf{x}_2 = [\phi \; \dot{\theta} \; \dot{\phi}]^T$, and $\mathbf{x}_3 = [\theta_i \; V_i]^T$.

Using the partitioned feedback gains, we present the final controller structure in Figure 3-3:

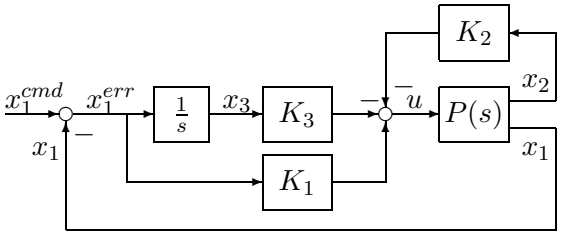


Figure 3-3: Traditional LQR setup

The final controller is presented in Algorithm 3.2.2. This implementation ensures the proper initialization of the control $\mathbf{u}$ and limits the control to the physical voltage saturation limits. It also ensures that the integrators, which are incremented using Euler's method, do not wind up in the case of external disturbances.

Algorithm 3.2.2: ELEVATION/TRAVELVELOCITY CONTROLLER($\mathbf{x}$)

comment: Compute $\mathbf{u} = \mathbf{Kx}$

$\mathbf{x_1^{err}} \leftarrow \mathbf{x_1^{cmd}} - \mathbf{x_1}$
if $INITIALIZED$
then $\begin{cases} \mathbf{x_3} \leftarrow \mathbf{x_3} + \mathbf{x_1^{err}} dt \\ \mathbf{u} = \mathbf{K_1 x_1^{err}} - \mathbf{K_2 x_2} - \mathbf{K_3 x_3} \\ \mathbf{u} = LIMIT(\mathbf{u}, \mathbf{u_{min}}, \mathbf{u_{max}}) \end{cases}$
else $\begin{cases} \mathbf{u} \leftarrow 0 \\ INITIALIZED \leftarrow 1 \end{cases}$
$\mathbf{x_3} \leftarrow \mathbf{K_3^{-1}}(-\mathbf{u} + \mathbf{K_1 x_1^{err}} - \mathbf{K_2 x_2})$
return ($\mathbf{u}$)

3.2.2 Initialization controller (*init_lqr*)

We use a LQR with the Elevation and Travel Angles ($[\theta, \psi]$) as the tracking variables to initialize the 3DOF helicopter. This controller is designed similar to the *super_lqr* except the state is augmented to be

$$\mathbf{x} = [\theta\ \psi\ \phi\ \dot{\theta}\ \dot{\psi}\ \dot{\phi}]^T \Longrightarrow \bar{\mathbf{x}} = [\theta\ \psi\ \phi\ \dot{\theta}\ \dot{\psi}\ \dot{\phi}\ \theta_i]^T$$

where $\dot{\theta}_i = \theta^{cmd} - \theta$ is an added integrator state. Since we use large travel commands, we need to limit the travel rate and hence have a different structure for the travel loop of the controller. This modified structure is presented in Figure 3-4, where "Sat" block represents the enforced saturation of the travel rate.

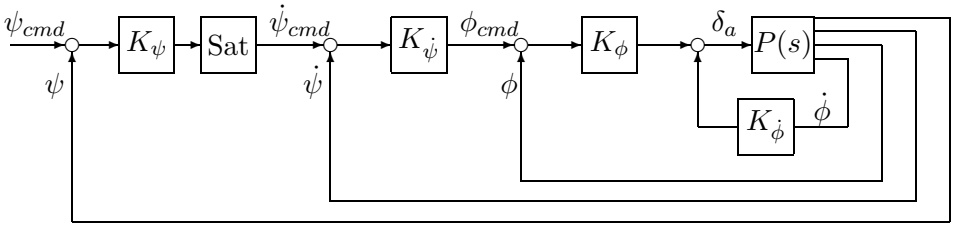


Figure 3-4: ψ loop setup with rate limit

3.2.3 Landing controller (*land_lqr*)

The *land_lqr* controller consists of two parts, the first one brings the rotor assembly to hover above the initial starting point and is identical to the *init_lqr*. The second one ensures a smooth landing on the table's surface from the hover position using the decaying exponential command for the elevation angle in the form:

$$\theta^{cmd} = \theta_0 \exp^{-\frac{t-t_0}{\tau}}$$

where time constant τ was determined experimentally and θ_0 is the elevation angle at time t_0.

Chapter 4

Implementation

4.1 Hardware Structure

The supervisory controller is implemented on a 3DOF helicopter manufactured by Quanser [12]. The hardware setup consists of a mechanical system shown in Figure 1-1, sensors, amplifiers, data acquisition board and a computer. The mechanical system is powered by a pair of amplifiers driving the motors while the angle measurements are done using mounted optical encoders. The slip-ring transmits the electric signals to and from the 3DOF helicopter. An ISA MultiQ3 data acquisition board enables the computer to access measurements from the sensors and to command voltages to the amplifiers. We use a PC with a Pentium III processor. A second Pentium III PC equipped with a webcam provides visual feedback via live streaming video.

4.2 Software Structure

4.2.1 Traditional Laboratory

The traditional laboratory setup uses the WinCon software developed by Quanser for the Windows OS [12]. It controls the operation of the 3DOF helicopter in real-time. The WinCon application allows students to design their controllers in Simulink (similar to the diagram in Figure 2-5), then to replace the nonlinear simulation block with the ISA MultiQ3 Input/Output block in order to interface with the actual machine. Once the Simulink diagram is completed, it is built using MATLAB Real-Time Workshop (RTW) [9] to generate C code. Finally, the Visual C++ compiler (VC++) produces a WinCon client library file. The WinCon client runs in real-time while the WinCon Server is a separate graphical interface which allows users to change the input parameters or plot real-time data. The WinCon Sever and Client communicate over a TCP/IP connection. The remote laboratory's architecture is similar to the traditional one.

4.2.2 Remote Laboratory

The objective of making the system remotely available has been achieved in collaboration with the University of Siena Automatic Control Telelab (ACT) [4]. The overall architecture of the remote laboratory is shown in Figure 4-1. Firstly, a client web browser connects to the University of Siena web server optionally submitting a user-defined controller as a Simulink model. Secondly, the web server provides the client with the information necessary

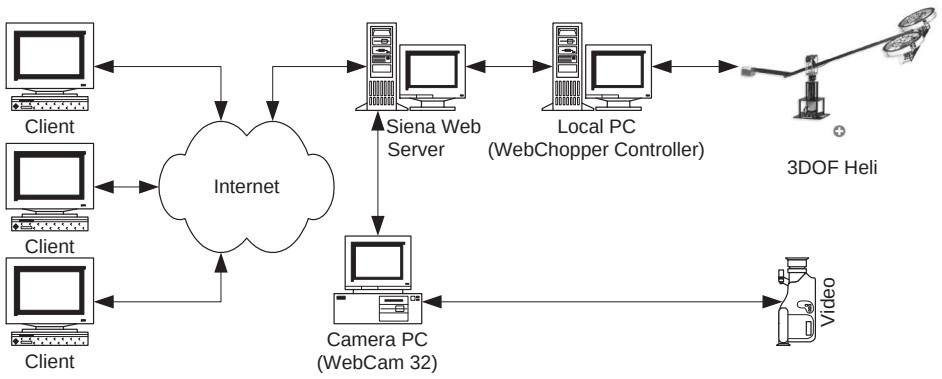


Figure 4-1: The remote laboratory architecture

to connect to the local server. The local server directly controls the 3DOF helicopter. Then, the client establishes a connection with the local server and optionally provides an integrated model of the user-defined controller. If the integrated model is provided, the local server first compiles it and builds an executable. Finally, the local server starts a new process controlling the 3DOF helicopter and provides a Java applet as a user interface to this process. The Java applet communicates with the controlling process over a TCP connection. The client also connects to the webcam, which provides live streaming video of the process, using another Java applet.

This remote architecture gives users an opportunity to use either a predefined controller, such as an Elevation PID controller, or to design their own controllers in Simulink. Once uploaded to the local server, a user-defined controller is built with RTW. The resulting code is compiled and linked with the ACT library to produce an executable. For maximum portability, the client is a standard web browser that can handle HTML forms and run Java applets. In fact, the user does not need to install any software to run the remote laboratory. The applet allows the client to change parameters, such as controller gains, and references, such as step or sinusoidal inputs, and to observe the outputs in real-time on graphical displays. Once the experiment session is completed, the output data are stored in the MATLAB format and can be downloaded for off-line analysis. The session is limited to five minutes to prevent a single user from hogging the system.

The local server programs run on Windows 98 due to incompatibility of the existing Quanser software and an ISA card with more recent version of Windows. The Real-Time Extension software [13] (provided with the current version of WinCon) and a Q8 PCI card from Quanser should enable the use of more recent version of Windows. We are using MATLAB 5.3/Simulink 3.0 environment in addition to the Visual C++ 6.0 compiler. The supervisor is written in C S-function format and a sample template file of the Simulink model used to generate an executable for the Elevation Controller is shown in Appendix D. The HTTP server used on the local machine is Apache 1.3.20 with PHP 4.0.6.

4.2.3 Connection Management Module

The experiments used in the University of Siena Telelab are non-safety critical systems; therefore, managing client connections is not an important issue for the ACT architecture.

In our case, we cannot simply stop the executable controlling the 3DOF helicopter when the client clicks the "Stop Experiment" button or when the connection between the client and the server is lost. To ensure that the supervisory controller lands the 3DOF helicopter, we have developed an additional software module to supplement the ACT architecture. The software module consists of a proxy server which runs on the local machine and checks the signals coming from the client and modifies them accordingly to ensure that the supervisor completes the landing before the machine is turned off.

4.2.4 Telepresence

The users are able to observe the experiment using graphical interface displaying a live streaming video. Webcam32 [17] software captures images from a video camera and makes them available on the Internet. A JavaCamPush applet displays the real-time streaming video from Webcam32 in the user's browser. Since Webcam32's built-in HTTP is not compatible with the most common version of Java (Java 1.4.2.), the applet is served from a HTTP 1.1 compatible web server, Apache 1.3.29, instead. It is important to note that a separate PC is required to run the camera. Webcam32 software requires a lot of RAM in order to work efficiently and might interfere with the control experiments if the camera is placed on the same computer as the actual setup.

4.3 Laboratory Description

In this section we give an overview of the available remote experiments. The 3DOF helicopter is currently setup to have three experiments: identification of the plant, elevation angle controller, and elevation angle and travel velocity controller.

4.3.1 Identification Experiment

This experiment has been set up so that users can collect open loop data by applying collective and cyclic voltages to the plant at hover. The collective voltage (V_{col}) for the identification is limited to $[-0.2, 0.3]$ volts, and the cyclic voltage (V_{cyc}) is limited to $[-0.2, 0.2]$ volts. The users can apply either a step or a sine input as the reference. This choice of reference inputs allows users to analyze step response data and fit them to a second degree model or to do a frequency sweep of the system and obtain a more accurate system description using Bode analysis techniques. To facilitate the identification of the elevation dynamics, we are also providing an option of using a "Roll Controller", which keeps the 3DOF helicopter from swinging in the roll. Without the roll controller, any plant disturbance, such as air currents in the room, induce the motion of the 3DOF helicopter in the travel direction.

4.3.2 Elevation Control Experiment

In this experiment, users can test the elevation angle controllers they have designed for the 3DOF helicopter. Users can test either a predefined Elevation PID controller whose parameters (Proportional, Integral, and Derivative controller gains) can be changed, or design and upload their own controllers.

In order to implement a user-defined controller, the template Simulink model shown in Figure 4-2 has to be used. This template consists of two subsystems: one for the controller ("ACT_CONTROLLER") and one for the references ("ACT_REFERENCE"). Both

subsystems can be modified using the Simulink library blocks. For instance, one can increase the number of available reference inputs by adding other Simulink *Sources* to the "ACT_REFERENCE" subsystem [9]. The predefined PID controller serves as an example for a user-defined controller and is shown in Figure 4-3. Just like for a predefined controller, a user-defined controller can have parameters which can be changed in real-time. To do that, users must prefix the names of the parameters in the "ACT_CONTROLLER" subsystem with "ACT_TP_" (ACT Tuning Parameter). Ultimately, the user-defined controller is integrated with the main Simulink template file shown in Appendix D. The main Simulink diagram consists of the following subsystems: "ACT_REFERENCE", "ACT_CONTROLLER", "ACT_GRAPHICS", "Supervisor C S-function", "Heli 3DOF", and "Roll_Ctrl". The "ACT_REFERENCE" and "ACT_CONTROLLER" subsystems consist of default references and a predefined PID controller unless a user uploads his own references and a controller as described above. The "Supervisor C S-function" contains the supervisor code which initializes, monitors safety and lands the machine. The "Heli 3DOF" subsystem contains an ISA MultiQ3 Input/Output block which provides an interface with the actual machine. The "ACT_GRAPHICS" block sets up real-time plotting of the outputs and saves them at the end of the experiment in MATLAB (.mat) file format. The "Roll_Ctrl" subsystem gives students an option to use a roll controller to stabilize the roll motion of the plant.

A screen shot of the user interface for testing the elevation controller is shown in Figure 4-4.

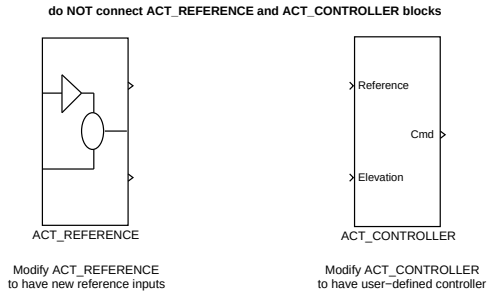


Figure 4-2: A template Simulink diagram for implementing user-defined controller

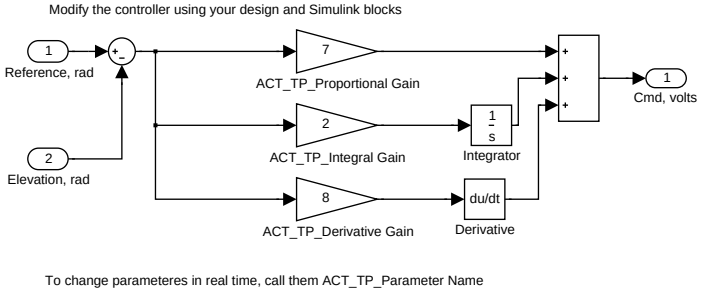


Figure 4-3: A Simulink diagram of the "ACT_CONTROLLER" subsystem

4.3.3 LQR Experiment

The LQR Experiment setup can be used in graduate level classes for designing controllers using full state feedback. This setup allows students to control the elevation and travel axes of the 3DOF helicopter. There is no predefined controller for this experiment, so users are expected to use the template file similar to the one in Figure 4-2 to implement their controllers. The user template and main template file structures are practically identical to the ones used in the Elevation Control Experiment.

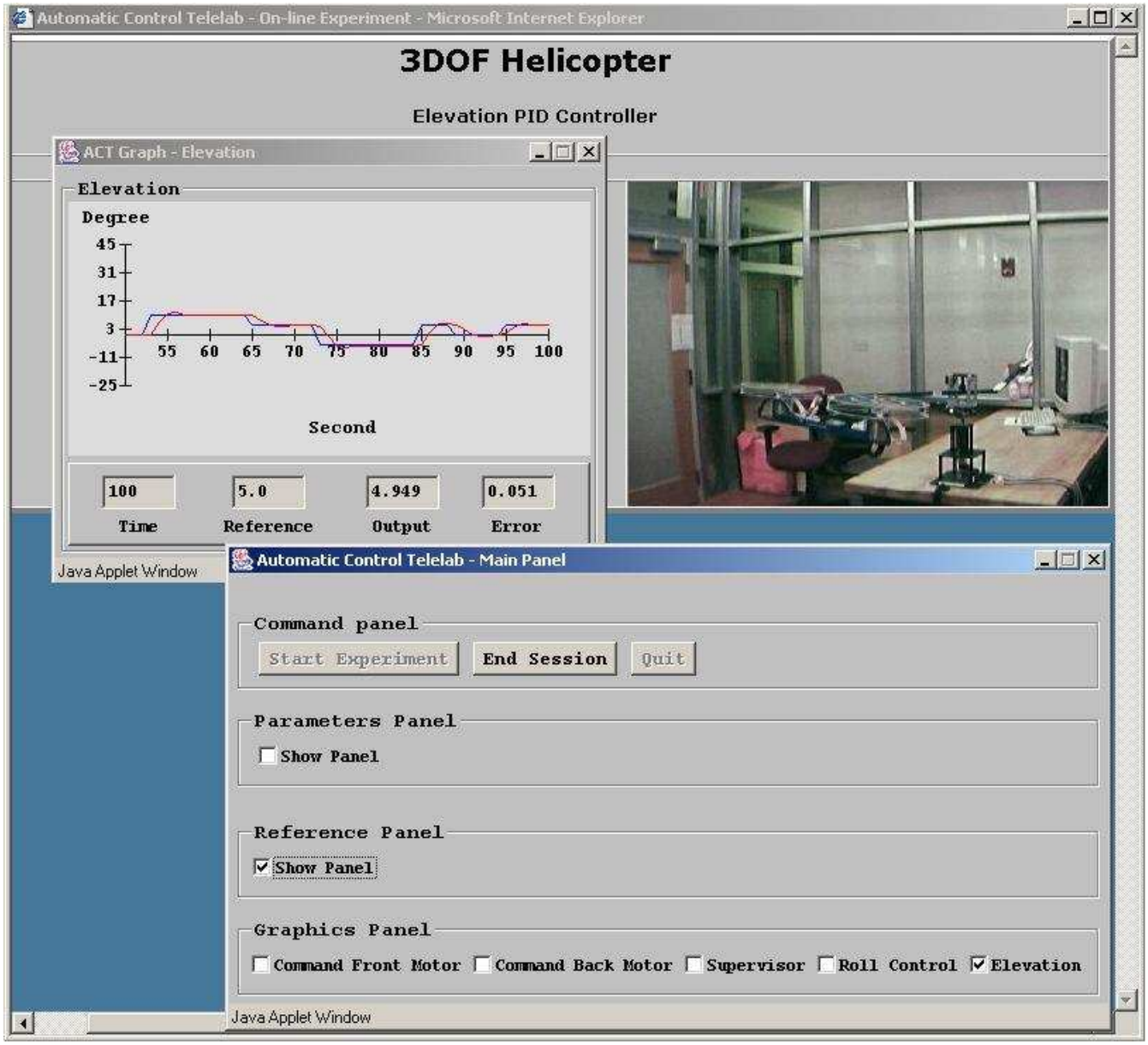


Figure 4-4: A screen shot of the Web interface

Chapter 5

Laboratory Evaluation

5.1 Description of Experiments

In this chapter we evaluate the 3DOF helicopter remote laboratory. This evaluation is based on the comparison of the student experiences using the traditional and remote laboratories. Since MIT's Department of Aeronautics and Astronautics routinely runs the control laboratories using the 3DOF helicopters, we have had a unique opportunity to analyze the laboratories using a two-group design. In a two-group design, the *control* group of subjects gets no treatment, and the *experimental* group gets some treatment [18]. The treatment in this case is the remote control laboratory with the supervisory controller, and the measure of comparison is the satisfaction of laboratory requirements and student satisfaction.

In our experiment, the *control* group consisted of ten students performing the traditional laboratory, and the *experimental* group consisted of ten students performing the remote laboratory. The subjects for the experiments were advanced undergraduate students taking a feedback control systems course. All students had prior experience using the 3DOF helicopter system. The 3DOF helicopter laboratory experiment consisted of two parts: the identification of the elevation plant dynamics and the design of an elevation angle controller. Both groups of students were required to satisfy identical laboratory requirements. To collect students' feedback, they were given two surveys to fill out: the first one as they were doing the laboratory and the second one to be filled out once the laboratory was completed. The in-lab survey asked students to record the type of reference inputs they used each time they turned on the machine. It also asked students if they recorded the data for each run, if the machine became unstable or if it was hard landed on the table. The post-lab survey asked students to rate their laboratory experience on a numerical scale upon the completion of the assignment. The survey is included in Appendix E along with the compiled responses in Tables E.2 and E.3. The averaged results of student responses to the post-lab survey are presented in Table E.1. Only six remote laboratory users and ten traditional laboratory users submitted their surveys for analysis. All students submitted their lab reports. We include sample graded laboratory reports of remote and traditional users on a CD. The remote and traditional laboratory instructions are included in Appendices F and G. The analysis of the survey responses and the laboratory reports is presented below.

5.1.1 Identification Experiment

The first part of the laboratory involves the identification of the elevation dynamics of the plant around the hover position. In particular, students apply step and sinusoidal inputs

to the system, collect the system response data and analyze it using the techniques from the course lectures. To get an average system response, students are expected to collect several sets of data for each combination of amplitude and frequency. Since the traditional laboratory is run open-loop from the moment the machine is turned on, it is difficult to bring the machine to the same initial conditions near the hover position. In the traditional laboratory, students usually do data collection with a partner because one person has to stabilize the machine in the horizontal position while the other person records the system response. In the remote laboratory, the supervisory controller initializes the system and then waits for user's reference input while running open-loop with the predefined hover voltages. The supervisory controller ensures that the machine always starts from the same position and hence makes the data sets more repeatable and reduces the number of times the data have to be collected for each reference input.

The student responses showed that the average number of hours spent collecting data in the traditional laboratory was 5.4 hours, while for the remote laboratory users it was 4.8 hours. The remote laboratory users indicated that it was easy to initialize the machine for the data collection while the traditional laboratory users considered it difficult. We also found that the remote laboratory users did not record data during 11% of the runs, while the traditional users did not take data during only 2% of the runs. We attribute this to the fact that the remote laboratory users had to get acquainted with the laboratory procedures, while the traditional laboratory users had used the setup before. The students also reported that the remote laboratory crashed their Internet browser in 5% of the cases, and two times the 3DOF helicopter was hard landed on the table when the Internet connection was dropped. Currently, we deal correctly with the dropped connections in the software module described in Chapter 4.

To compare the technical aspects of the laboratory, we asked students to record the reference inputs they used. The control group of students used the gains within $[0.1, 0.35]$ range and the frequencies within $[0.1, 50]$ range. The remote group used the gains within $[0.1, 0.3]$ range and the frequencies within $[0.05, 100]$ range. These ranges show that the remote laboratory users were able to adequately identify the plant by observing its response to certain inputs. Remote users also indicated that the supervisor took over the control in about 7% of the runs.

5.1.2 Elevation Control Experiment

For the controller part of the laboratory, the students were required to design an elevation angle controller with specified bandwidth and phase margin requirements. They were expected to use either a phase-lead/phase-lag compensation or a PID design technique. Two remote laboratory users used a PID controller while the rest of the users used a phase-lead/phase-lag compensator or a phase-lead compensator. The difference in controller choice was probably due to the fact that a PID controller was a predefined controller in the remote laboratory. For a predefined controller, students needed only to change the gain parameters in order to achieve the desired response. However, any other type of compensator had to be first implemented in Simulink before being tested on the system. Overall, the performance of a closed-loop system was comparable for both groups of students. It was difficult to judge how many times the students tested their controllers on the actual system and how many times the system became unstable because students were not diligent in completing the in-lab surveys. Laboratory reports indicated that both groups of students tested their controllers within a $[-5^\circ, 15^\circ]$ range. This range allowed the remote laboratory users to test

stable controllers without the interference of the supervisory controller. Survey responses also showed that the remote laboratory users had adhered more closely to a "healthy" controller design procedure, which recommends testing the controller in Simulink before using it on the actual system. The amount of time spent by both groups of students testing their controller on the system was practically the same.

5.2 Overall Comparison

Here, we summarize the survey responses presented in Appendix E. The survey was mainly structured around the three important aspects of the laboratory: its availability, the clarity of the instructions, and the safety and integrity of the equipment.

Even though the remote laboratory was available twenty four hours a day, the demand was so high that we needed to schedule the access to the machine. To eliminate time conflicts, we asked the students to sign up for using the remote laboratory. The schedule was posted on the class website and updated as students changed their preferences. However, several students did not adhere to the schedule making it difficult to access the machine the night before the deadline. Nevertheless, most students praised the ease of access to the remote laboratory. In fact, later in the semester, as students were doing a different laboratory assignment, some students expressed interest in performing the remote laboratory due to the scheduling conflicts with the traditional one.

According to the survey, the opinions concerning the clarity of the instructions for running the machine and changing the input parameters were strictly divided. Half of the remote laboratory respondents indicated that the instructions were very difficult to understand, while the other half found them very easy. Apparently, some students were confused about the units used in the laboratory because the real-time displays showed data in degrees but the MATLAB data were saved in radians. Some students were confused on the conceptual level. Specifically, they could not understand how the references used for the identification part had units of volts, and the references used in the control experiment had units of radians. Once these points were clarified, the rest of the students found the instructions easier to understand. In addition, some students had problems implementing their controller in the Simulink diagram. Currently, we are providing several template files for user-defined controllers to make the implementation easier.

As for the safety and integrity of the equipment, the traditional laboratory did not have any built-in safety features. The students were requested to use a *kill-switch* to turn off the amplifiers should the machine become unstable. The traditional laboratory also did not have built-in initialization and landing. Its users indicated that on average each student hard landed the machine on the table about 8 times while running the experiments. The remote laboratory was hard landed on the table only twice when the connection was accidentally dropped. The hard landing of the helicopter led to the fracture of the rotor support structure in at least two machines used in the traditional laboratory. The traditional laboratory users indicated that in addition to hitting the table's surface, the machine hit or strained the bearing about 3.5 times on average while running the experiments. The remote laboratory users indicated that the machine strained the bearing about 10 times. However, we have found that the question was posed ambiguously in the survey and the remote laboratory users thought we had asked how many times a supervisory controller took over during the experiments. The oral feedback provided by the remote lab users indicated that the machine strained the bearing about 1.5 times on average while running

the experiments. Observing the recorded output data, we determined that the supervisory control took over successfully in about 95% of the cases before straining the top of the structure and the remote system hit the table's surface twice.

The grading of the laboratory was done objectively by an undergraduate grader. The average grade of the remote laboratory users was $\sim 6\%$ lower than the average grade of the traditional laboratory users. The average was lower due to the fact that several remote laboratory students did not have the time to test disturbance rejection because of scheduling conflicts close to the deadline. The remote laboratory grade average was adjusted later by the course instructor to be equal to the one of the traditional laboratory reports. As for the overall laboratory experience, both groups found it to be both stimulating and frustrating.

5.3 Discussion of the Results

We would like to point out certain positive aspects of the remote laboratory based on the students' feedback.

- The remote laboratory allows students to reduce the data collection time necessary for system identification. This is usually a very tedious procedure, so students appreciate easier initialization of the machine in the remote laboratory.

- We believe that the remote laboratory, which has a supervisory controller, improves students' controller design technique. It encourages students to test the simulated response of a closed loop system before trying it on a real system. The structure of the remote laboratory encourages the students to follow this procedure because the initialization and landing phases add more time overhead. This controller design technique is usually employed by successful hardware projects in the real world. For example, the miniature acrobatic helicopter designers first test their controllers in software simulation, then hardware in the loop simulation and only then test them on the actual hardware [15]. This design procedure allows for considerable savings in the setup costs.

- The comparison of the student reference inputs and controller commands indicates that, both for the identification and the elevation control experiments, the stable controllers can be tested within the desired range without the supervisory controller interference. This indicates that the basic functionality is preserved in the remote laboratory setting.

- The supervisory control reduces hardware maintenance because it overrides excessive user inputs that lead to misalignment and structural damage.

- The remote laboratory provides a safe environment for the students because they are not in physical contact with the hardware.

- The remote laboratory is convenient because of its high availability.

- The remote laboratory saves resources. It allows for use of a single machine by a large number of students. Moreover, the remote laboratory setup can be placed anywhere clearing the prime laboratory space for other projects in the department.

We also observe some negative aspects of the remote laboratory. Since the laboratory is available at all times, it prompts students to do their assignment at the last minute. This is not a good working habit. However, stricter enforcement of the schedule should be able to fix this problem. Also, the students found the remote laboratory slightly less motivating than the traditional one. This can be potentially improved through the modifications in the telepresence. For instance, we plan to provide an audio feedback to the students by using a microphone in addition to the video camera. We can also equip the system with a physical device which would produce a loud buzzing sound whenever the helicopter is being taken over by a supervisory controller. This could make the remote laboratory a more educationally entertaining and stimulating experience.

Chapter 6

Summary and Future Work

6.1 Results and Conclusions

In this thesis we have designed and successfully implemented a remote laboratory for a safety-critical system. The main contributions of this work are:

- The development of a system model for a 3 DOF helicopter.
- The development of a structured supervisory controller which ensures the system's safety.
- The implementation of the supervisory controller on a real physical system as part of a remote laboratory.
- The evaluation of the new laboratory based on students' responses and graded laboratory reports.

The remote laboratory developed in this research is available at

http://webchopper.lids.mit.edu

The overall reaction of the students to the highly available software-supervised machine has been extremely positive and encouraging. The assessment of the two student groups shows that while the remote and traditional laboratories both meet the educational objectives, the remote laboratory also ensures the safety of the equipment and the students. The remote laboratory is an attractive practical alternative to the traditional one. It simplifies maintenance of the equipment, allows for reuse of the resources and saves prime laboratory space. Furthermore, the ease of access to the experimental platform encourages interdepartmental collaboration and opens new possibilities for distance-learning students.

6.2 Future Work

Telepresence is an important part of the laboratory that needs to be improved. In particular, we need to address the hands-on aspect of the laboratory so that the students realize that they are working with a safety critical system. As a first step, we plan to add a microphone to the setup and implement an audio alarm for the students to hear when the supervisor turns on.

We are working on a stronger theoretical framework for the real-time supervisory control. In particular, instead of simulating the closed loop system response consisting of a linearized plant and a given LQR controller, we can ensure the system's safety by finding over approximation of the reachable sets of the system. Specifically, one can pose the finite-time reachability set problem as a Linear Matrix Inequalities (LMI) problem. The LMI, in turn, can be solved using MATLAB LMI toolbox to find the ellipsoid containing the reachable set. If that ellipsoid intersects the unsafe region, the supervisor will then override user's control inputs. This method presents an attractive alternative because one can derive LMI for the nonlinear system dynamics and use the LMI toolbox to solve the inequalities without drastically increasing the computational cost. The reachable sets would still have to be evaluated at each time step as the real plant dynamics evolve, but their computation should take less time than the current real-time simulation. Other controller-design techniques, such as H2/H-Infinity optimal control, are going to be investigated.

The concept of the structured supervisory controller designed for the 3DOF helicopter can potentially be extended to other safety-critical systems. In particular, we are considering extending the setup to other safety critical experimental platforms in the department, such as the Aero/Astro wind tunnel.

Since the student feedback and the results have been positive, it has been proposed that we equip other machines in the department with the supervisory safety controller in the near future. The remote laboratory should be made autonomous from the University of Siena server and become part of the MIT OpenCourseWare to be available for distance learning.

Appendix A

```
/*=================================================================
* nonlinear_smodel.c
*
* The calling syntax is:
*
*     [THETA, PSI, PHI] = nonlinear_smodel(OMEGA_LEFT. OMEGA_RIGHT)
*
*  INPUTS:
*      u = [Omega_leftMotor, Omega_rightMotor] in SI units [rad/sec]
*
*  OUTPUTS:
*      y = [THETA, PSI, PHI] in SI units [rad]
*
*  STATES:
*      x = [THETA, PHI, PSI, DTHETA, DPHI, DPSI]
*
*=================================================================*/

#define S_FUNCTION_NAME nonlinear_smodel_mod
#define S_FUNCTION_LEVEL 2

#include ‘‘simstruc.h’’
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define PI 3.14159265358979

#define THETA 0
#define PHI   1
#define PSI   2
#define DTHETA 3
#define DPHI   4
#define DPSI   5

#define OMEGAL  0
#define OMEGAR  1

#define TINYLAMBDA 1e-6
```

```
#define ABS(A)     ((A)<0?-(A):(A))
#define MIN(A,B)   (((A)<(B))?(A):(B))
#define MAX(A,B)   (((A)>(B))?(A):(B))
#define LIMIT(A,B,C) (((A)<(B))?(B):(MIN(A,C)))

static int_T doThrustInflowIteration (real_T *CT, /* Thrust coefficient */
real_T *lambda0,               /* Inflow coefficient */
real_T mi,                     /* Advance ratio */
real_T mi_z,                   /* normalized w */
real_T theta0,                 /* collective pitch */
real_T theta_tw,               /* blade twist */
real_T s,                      /* rotor solidity */
real_T a0);                    /* rotor blade C_{L_\alpha} */

#define U(element) (*uPtrs[element])  /* Pointer to Input Port0 */

/*====================*
* S-function methods *
*====================*/

/* Function: mdlInitializeSizes ===============================================
*    The sizes information is used by Simulink to determine the S-function
*    block's characteristics (number of inputs, outputs, states, etc.).
*/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, 6);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 2);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 3);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
```

```
    /* Take care when specifying exception free code - see sfuntmpl.doc */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes =========================================
*    Specifiy that we have a continuous sample time.
*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
/* Function: mdlInitializeConditions ========================================
*    Initialize both continuous states to zero.
*/
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetContStates(S);
    x0[THETA] = 0.0;
    x0[PHI] = 0.0;
    x0[PSI] = 0.0;
    x0[DTHETA] = 0.0;
    x0[DPHI]   = 0.0;
    x0[DPSI]   = 0.0;
}

/* Function: mdlOutputs =======================================================
*      y = Cx + Du
*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    real_T            *x    = ssGetContStates(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    y[0] = x[THETA];
    y[1] = x[PSI];
    y[2] = x[PHI];
}
#define MDL_DERIVATIVES
/* Function: mdlDerivatives =================================================
*      xdot = Ax + Bu
*/
static void mdlDerivatives(SimStruct *S)
{
    real_T            *dx   = ssGetdX(S);
```

```
real_T            *x    = ssGetContStates(S);
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

/* define constant parameters */
const real_T rho = 1.23;    /* [kg/m^3] density of air */
const real_T R_rotor = 0.1; /*[m] Radius of propeller */
const real_T l_h = 0.177;   /* (m)length from pivot point to one motor */
const real_T L = 0.66;      /* (m)length from pivot point to heli body */
const real_T Jyy = 0.93;    /* [kg*m2] */
const real_T Jzz = 0.93;    /* [kg*m2] */
const real_T Jxx = 0.0360;  /* [kg*m2] */
const real_T mh = 0.031;      /*[kg] effective mass at hover */
const real_T m_rotors = 1.15; /*[kg] mass of the rotors (from specs) */
const real_T g = 9.8;         /*[m/s^2] */
const real_T pend_rotors = 0.004;/*[m] length of pendulum for rotors */
const real_T pend_beam = 0.014;  /*[m] length of pend for long beam */
const real_T mass_all = 3.57;    /*[kg] mass of the whole setup */
const real_T thetaNeutral = 0.45;  /*[rad] neutral theta  */
const real_T theta0 = 0.91;      /* collective pitch */
const real_T theta_tw = -0.93;   /* blade twist */
const real_T sigma = 0.14;       /* rotor solidity */
const real_T a0 = 4.5;           /* rotor blade C_{L_\alpha} */
const real_T Cd = 2;             /* aerodynamic drag coefficient */
const real_T SrfA0 = 0.006;      /* [m^2] horz surface area of heli */
const real_T SrfA  = 0.084;      /* [m^2] surface area when roll != 0 */

/* declare Thrust */
real_T ThrustL = 0.0;
real_T ThrustR = 0.0;
real_T CtL     = 0.0;
real_T CtR     = 0.0;
real_T lambdaL = 0.0;
real_T lambdaR = 0.0;

/* declare normalized velocities */
real_T mu_zL = 0.0;
real_T mu_zR = 0.0;
real_T mu_L   = 0.0;
real_T mu_R   = 0.0;
real_T temp_OR_L = 0.0;
real_T temp_OR_R = 0.0;

/* inputs */
double u_left  = U(OMEGAL);
double u_right = U(OMEGAR);

/* other variables */
double cPhi = cos(x[PHI]);
```

```
    double sPhi = sin(x[PHI]);
    double cTheta = cos(x[THETA]);
    double sTheta = sin(x[THETA]);

    /* define normalized velocities */
    temp_OR_L = 1/((u_left  + 0.1)*R_rotor);
    temp_OR_R = 1/((u_right + 0.1)*R_rotor);
    mu_zL = temp_OR_L*(-x[DPSI]*cTheta*sPhi*L-x[DTHETA]*cPhi*L - x[DPHI]*l_h);
    mu_zR = temp_OR_R*(-x[DPSI]*cTheta*sPhi*L-x[DTHETA]*cPhi*L + x[DPHI]*l_h);
    mu_L  = temp_OR_L*( x[DPSI]*cTheta*cPhi*L-x[DTHETA]*sPhi);
    mu_R  = temp_OR_R*( x[DPSI]*cTheta*cPhi*L-x[DTHETA]*sPhi);

    /* find Thrust */
    doThrustInflowIteration (&CtL,&lambdaL,mu_L,mu_zL,theta0,theta_tw,sigma,a0)
    doThrustInflowIteration (&CtR,&lambdaR,mu_R,mu_zR,theta0,theta_tw,sigma,a0)
    ThrustL = CtL*rho*u_left*u_left*R_rotor*R_rotor*R_rotor*R_rotor*PI;
    ThrustR = CtR*rho*u_right*u_right*R_rotor*R_rotor*R_rotor*R_rotor*PI;

    /* find derivatives */
    dx[THETA] = x[DTHETA];
    dx[PHI]   = x[DPHI];
    dx[PSI]   = x[DPSI];
    dx[DTHETA] = 1/Jyy*((ThrustL + ThrustR)*L*cPhi -
                 mass_all*g*pend_beam*sin(x[THETA] + thetaNeutral));
    dx[DPHI]   = 1/Jxx*((ThrustL - ThrustR)*l_h - m_rotors*g*pend_rotors*sPhi);
    dx[DPSI]   = 1/Jzz*((ThrustL + ThrustR)*L*sPhi*cTheta +
                 (ThrustR - ThrustL)*l_h*sPhi*sTheta -
                 0.5*rho*(x[DPSI]*L)*(x[DPSI]*L)*Cd*(SrfA0 + SrfA*sPhi));
}

static int_T doThrustInflowIteration (real_T *CT,/* Thrust coefficient */
real_T *lambda0,  /* Inflow coefficient */
real_T mi,        /* Advance ratio */
real_T mi_z,      /* normalized w */
real_T theta0,    /* collective pitch */
real_T theta_tw,  /* blade twist */
real_T s,         /* rotor solidity */
real_T a0)        /* rotor blade C_{L_\alpha} */
{
/* the following is based mainly on:
G.D. Padfield: ‘‘Helicopter Flight Dynamics: The Theory and Application
of Flying Qualities and Simulation Modeling’’, page 123 */

static const real_T SOLVEGAIN = 0.6; /* Stabil. gain for NR iteration */
static const int_T MAXIT = 1000;     /* max number of iterations */
static const real_T DLAMBDA0 = 1.e-8;/* maxError (~ 1% lambda0 hover) */

int_T it;                     /* iteration counter */
```

```
real_T oldlambda = *lambda0;  /* used for convergence test */
real_T h, Lambda, sqrtLambda; /* funcsdefined in Padfield, pg. 123 */
for (it = 0; it < MAXIT; it++)
    {
      Lambda = mi * mi + (*lambda0 - mi_z) * (*lambda0 - mi_z);
      /* avoid singularity close to vortex ring state (lambda==mi_z) */
      if (Lambda < TINYLAMBDA) Lambda = TINYLAMBDA;
      sqrtLambda = sqrt (Lambda);

      *CT = a0 * s / 2 * (theta0 * (1. / 3. + mi * mi / 2) +
      (mi_z - *lambda0) / 2. +
      (1 + mi * mi) * theta_tw / 4.);
      /* NOTE: in the above formula (Padfield eq. 3.157, pg. 123),
      the term in theta1_sw and p_w has been neglected
      (second order terms, and a contribution that is not
      directed along the normal of the rotor disk anyway...) */

      h = -(2 * (*lambda0) * sqrtLambda - (*CT)) /
      (2 * sqrtLambda + a0 * s / 4 - (*CT) / Lambda
      * (mi_z - (*lambda0)));
      *lambda0 += SOLVEGAIN * h;

      if (fabs (*lambda0 - oldlambda) < DLAMBDA0)  /* if converged... */
      break; /* exit the loop */
      oldlambda = *lambda0;
    }

    /* update CT to the best estimate */
    *CT =
    a0 * s / 2 * (theta0 * (1. / 3. + mi * mi / 2) + (mi_z - *lambda0) / 2.
    + (1 + mi * mi) * theta_tw / 4.);

    return it;/* if equal to MAXINT, then didn't have convergence at all!*/
}

/* Function: mdlTerminate =============================================
*/
static void mdlTerminate(SimStruct *S)
{
}
#ifdef  MATLAB_MEX_FILE   /* Is this file being compiled as a MEX-file? */
#include ``simulink.c''   /* MEX-file interface mechanism */
#else
#include ``cg_sfun.h''    /* Code generation registration function */
#endif
```

Appendix B

Linearized Model:

The state is given by

$$\mathbf{x} = [\theta \ \psi \ \phi \ \dot{\theta} \ \dot{\psi} \ \dot{\phi}]^T \tag{B.1}$$

and the input is given by

$$\mathbf{u} = [\Omega_{sum} \ \ \Omega_{diff}]^T \tag{B.2}$$

where $\Omega_{sum} = \frac{\Omega_L + \Omega_R}{2}$ and $\Omega_{diff} = \frac{\Omega_L - \Omega_R}{2}$.
Then $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ is given by

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ -0.5264 & 0 & 0 & -0.0810 & 0.0027 & 0 \\ 0 & 0 & 0.2129 & 0 & 0 & 0 \\ 0 & 0 & -1.4150 & 0 & 0 & -0.3486 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.0027 & 0 \\ 0 & 0 \\ 0 & 0.0209 \end{pmatrix}$$

$$eig(\mathbf{A}) = \begin{pmatrix} 0 \\ -0.0405 + 0.7244j \\ -0.0405 - 0.7244j \\ -0.1743 + 1.1767j \\ -0.1743 - 1.1767j \\ -0.0000 \end{pmatrix}$$

This linear model was obtained by manually linearizing the nonlinear model. It is also possible to obtain the linearized model using the **linmod** function in MATLAB.
As can be seen in the eigenvalues plots, the linear CIFER model directly captures the motor poles. The linearized model derived using basic physical principles captures motor poles indirectly through the lookup table which describes the relationship between the frequencies of revolution of the motors and the voltages applied to them.

Linear Model using CIFER:

The state is given by

$$\mathbf{x} = [\psi \ \dot{\psi} \ \phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \tau_{cyc} \ \tau_{coll}]^T \tag{B.3}$$

and the input is given by

$$\mathbf{u} = [V_{cyc} \ \ V_{coll}]^T \tag{B.4}$$

Then $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ is given by

$$\mathbf{A} = \begin{pmatrix} 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2517 & 0.1671 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.4220 & -0.3260 & 0 & 0 & 16.2000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5544 & -0.1008 & 0 & 1.34 \\ 0 & 0 & 0 & 0 & 0 & 0 & -7.3200 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6.16 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.9500 & 0 \\ 0 & 1.4000 \end{pmatrix}$$

$$eig(\mathbf{A}) = \begin{pmatrix} 0 \\ -0.2517 \\ -0.1630 + 1.1813j \\ -0.1630 - 1.1813j \\ -0.0504 + 0.7429j \\ -0.0504 - 0.7429j \\ -7.3200 \\ -6.1600 \end{pmatrix}$$

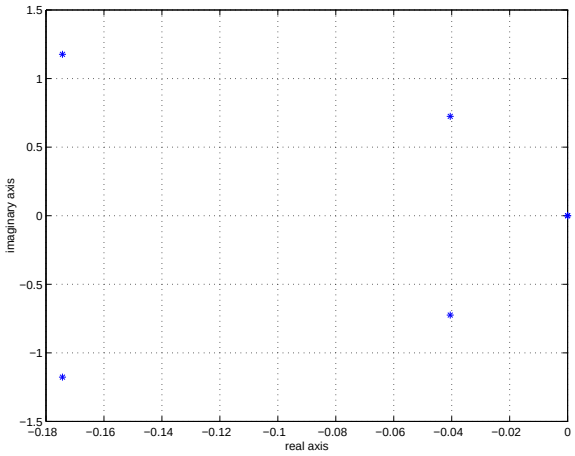


Figure B-1: $eig(A_{Linearized})$

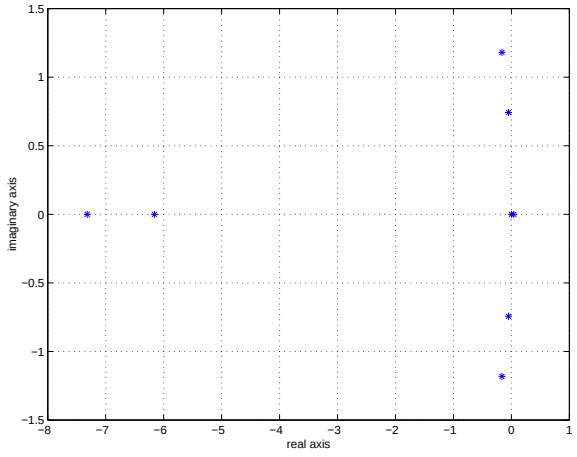


Figure B-2: $eig(A_{CIFER})$

Appendix C

LQR controller for θ and $\dot{\psi}$:

The augmented state is given by

$$\bar{\mathbf{x}} = [\theta \ \phi \ \dot{\theta} \ \dot{\psi} \ \dot{\phi} \ \theta_i \ V_i]^T \tag{C.1}$$

where where $\dot{\theta}_i = \theta^{cmd} - \theta$ and $\dot{V}_i = \dot{\psi}^{cmd} - \dot{\psi}$. The input is given by

$$\mathbf{u} = [\Omega_{sum} \ \ \Omega_{diff}]^T \tag{C.2}$$

where $\Omega_{sum} = \frac{\Omega_L + \Omega_R}{2}$ and $\Omega_{diff} = \frac{\Omega_L - \Omega_R}{2}$.

Then $\dot{\bar{\mathbf{x}}} = \bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{u}}$ is given by

$$\bar{\mathbf{A}} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ -0.5264 & 0 & -0.0810 & 0.0027 & 0 & 0 & 0 \\ 0 & 0.2129 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.4150 & 0 & 0 & -0.3486 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.0000 & 0 & 0 & 0 \end{pmatrix}$$

$$\bar{\mathbf{B}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0.0027 & 0 \\ 0 & 0 \\ 0 & 0.0209 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Set initial $\mathbf{R}$:

$$\mathbf{R} = \begin{pmatrix} \bar{u}_1^{-2} & 0 \\ 0 & \bar{u}_2^{-2} \end{pmatrix}$$

where $\bar{u}_1 = 10$ and $\bar{u}_2 = 10$.

Set initial $\mathbf{Q}$ using Bryson's rule [3]:

$$\mathbf{Q} = \begin{pmatrix} \bar{\theta}^{-2} & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & \bar{\phi}^{-2} & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \dot{\bar{\theta}}^{-2} & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & \dot{\bar{\psi}}^{-2} & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \dot{\bar{\phi}}^{-2} & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \bar{\theta}_i^{-2} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & \bar{V}_i^{-2} & 0 \end{pmatrix}$$

where initial $\mathbf{Q}$ is given by:

$$\begin{aligned} \bar{\theta} &= 0.14\ [rad] \equiv spring \\ \bar{\phi} &= 0.07\ [rad] \equiv spring \\ \dot{\bar{\theta}} &= \pi \quad [rad/sec] \equiv dashpot \\ \dot{\bar{\psi}} &= \pi/2\ [rad/sec] \equiv dashpot \\ \dot{\bar{\phi}} &= \pi/4\ [rad/sec] \equiv dashpot \\ \bar{\theta}_i &= 0.7\ \ [radsec] \equiv \theta_{steady\ state} \\ \bar{V}_i &= 10\ \ [rad] \qquad \equiv \dot{\psi}_{steady\ state} \end{aligned}$$

after playing with the controller in MATLAB to get desirable closed loop properties, the final parameters in $\mathbf{Q}$ are as follows:

$$\begin{aligned} \bar{\theta} &= 0.14\ [rad] \\ \bar{\phi} &= 0.49\ [rad] \\ \dot{\bar{\theta}} &= 0.3142\ [rad/sec] \\ \dot{\bar{\psi}} &= 3.14\ [rad/sec] \\ \dot{\bar{\phi}} &= 1.3352\ [rad/sec] \\ \bar{\theta}_i &= 0.3\ [radsec] \\ \bar{V}_i &= 0.5 \end{aligned}$$

Using MATLAB's [K] = **lqr**(A_bar, B_bar, 100*diag(Q), R), one obtains the following K:

$$\mathbf{K} = \begin{pmatrix} 865.1398 & 0.0531 & 828.3765 & 0.9939 & 0.0057 & -333.3333 & -0.0117 \\ -0.1474 & 245.7066 & 0.0437 & 768.0719 & 154.8216 & 0.0195 & -200.0000 \end{pmatrix}$$

and MATLAB's **damp**(A_bar - B_bar*K) produces the following:

$$\begin{pmatrix} Eigenvalue & Damping & Freq.(rad/s) \\ -3.15e-01+3.14e-01i & 7.08e-01 & 4.45e-01 \\ -3.15e-01-3.14e-01i & 7.08e-01 & 4.45e-01 \\ -4.45e-01+0.00e+00i & 1.00e+00 & 4.45e-01 \\ -9.48e-01+1.07e+00i & 6.63e-01 & 1.43e+00 \\ -9.48e-01-1.07e+00i & 6.63e-01 & 1.43e+00 \\ -1.48e+00+1.52e+00i & 6.97e-01 & 2.12e+00 \\ -1.48e+00-1.52e+00i & 6.97e-01 & 2.12e+00 \end{pmatrix}$$

The closed loop eigenvalues are shown in Figure C-1.

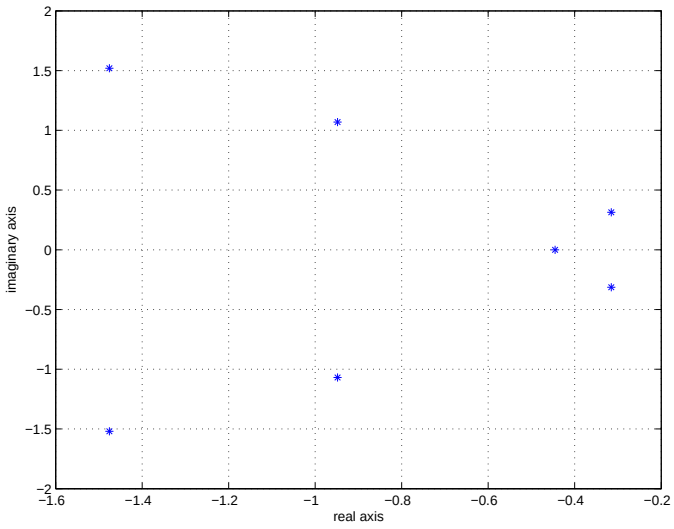


Figure C-1: $eig(\bar{A} - \bar{B}K)$

Appendix D

Figure D-1 shows the template Simulink diagram used for the Elevation controller experiment.

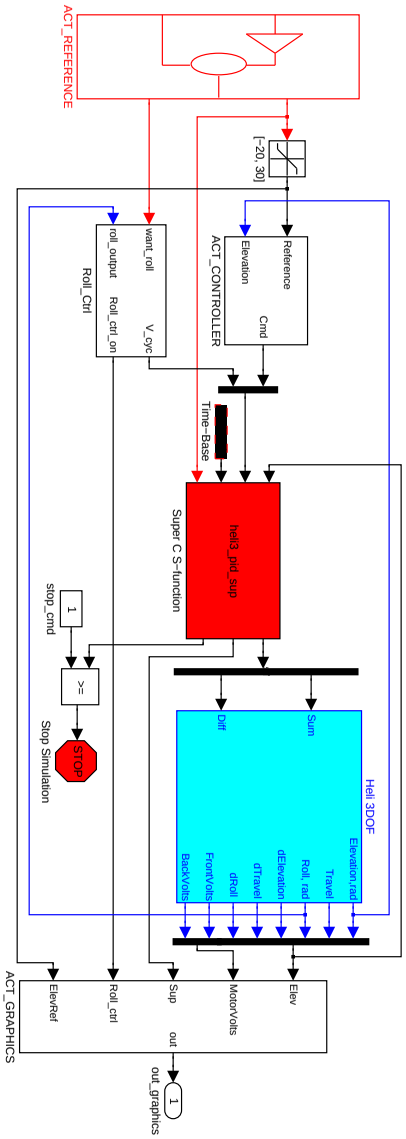


Figure D-1: Template Simulink model for elevation controller

Appendix E

The compiled response to the questionnaire is presented below. The numbers have been averaged for presentation. Six remote laboratory students submitted the survey, and ten traditional laboratory users submitted the survey. In Table E.1 we present the mean and standard deviation of the data for the remote and traditional laboratories. The complete survey responses are presented in the next two sections.

Table E.1: Student Responses to Post-Lab Survey

Question #	RemMean	TradMean	RemStDev	TradStDev
1	Yes	Yes	N/A	N/A
2	N/A	N/A	N/A	N/A
3	4.8	5.4	1.1	3.2
4	1.3	4.2	0.7	2.0
5	4.2	4.1	3.2	2.2
6	2.5	3.3	1.1	1.7
7	4.3	3.4	1.7	1.7
8	4.3	4.5	1.6	1.6
9	5.2	4.2	0.7	1.9
10	4.0	2.1	1.8	1.6
11	3.7	1.5	1.5	0.8
12	1.2	2.3	0.4	1.5
13	1.7	3.1	0.7	1.6
14	4.3	3.2	1.5	1.5
15	5.2	2.3	1.5	1.0
16	9.7*	3.5	7.5	5.9
17	0.2	7.9	0.4	7.4
18	5.0	4.5	0.7	1.7
19	4.8	4.6	1.3	1.7
20	4.3	4.6	0.7	1.6
21	2.3	2.5	1.8	1.6

$*$: As was mentioned in Chapter 5, this question was misinterpreted by the students. They thought we had asked how many times the supervisor took over the control before the stops were hit. Oral feedback provided by students indicates that, on average, the 3DOF helicopter hit the stops about 1.5 times while doing the remote laboratory assignment.

The actual questionnaire is presented on the next page.

Massachusetts Institute of Technology
Department of Aeronautics and Astronautics
16.30 Estimation and Control of Aerospace Systems
Spring 2004: Post Quanser Lab Questionnaire

Please, fill out the following questionnaire and submit it with your lab report. DO NOT put your name on this questionnaire (unless you really want to):

1. Have you used Quanser in your other classes prior to 16.30?

   `Yes  No`

2. For 16.30, which lab did you do:

   `Web lab  Regular Lab`

3. How much time did you spend to collect data for the identification part of the lab?

4. For the identification experiment, was it easy to initialize the Quanser (bring it to hover)?

   `Extremely Easy  1  2  3  4  5  6  7  Extremely Difficult`

5. How much time did you spend in the actual lab working on your controller?

6. How accurately did you follow the healthy controller design procedure of:
   (1) obtaining a linear model
   (2) designing the controller
   (3) testing the controller using the model
   (4) testing the controller on the actual equipment

   `Extremely Closely  1  2  3  4  5  6  7  Quite Loosely`

7. How easy was it to access the actual equipment?

   `Extremely Easy  1  2  3  4  5  6  7  Extremely Difficult`

8. Were the lab requirements easy to understand?

   `Extremely Easy  1  2  3  4  5  6  7  Extremely Difficult`

9. Is it easy to do the lab without staff support and presence during the sessions?

   `Extremely Easy  1  2  3  4  5  6  7  Extremely Difficult`

10. How easy was it to figure out which files should be used to run the lab?

    `Extremely Easy  1  2  3  4  5  6  7  Extremely Difficult`

11. How easy was it to understand how to change input parameters?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

12. How easy was it to turn on the data displays (plots of angles and rates)?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

13. How easy was it to manipulate the data displays?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

14. How easy was it to understand the instructions on running the machine?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

15. How easy was it to implement your controller in the final Simulink diagram?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

16. How many times did your Quanser hit the stops/table because of your controller design?

17. How many times did you hard land the machine on the table after finishing the experiment?

18. If you worked with a partner, how easy would it have been to perform the identification lab on your own?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

19. What about the controller design lab?

```
Extremely Easy   1   2   3   4   5   6   7   Extremely Difficult
```

20. How did you find the overall lab experience?

```
Dull             1   2   3   4   5   6   7   Stimulating
```

21. How did you find the overall lab experience?

```
Frustrating      1   2   3   4   5   6   7   Satisfying
```

22. Please, write below any additional comments about the lab and your experiences:

E.1 Survey results for the remote laboratory users

Table E.2: Survey results for the remote lab

Question #	#1	#2	#3	#4	#5	#6
1	Yes	Yes	Yes	Yes	Yes	Yes
2	N/A	N/A	N/A	N/A	N/A	N/A
3	4	4	4	5.5	4	7
4	1	1	1	3	1	1
5	6	10	2	1	1	5
6	3	4	3	1	3	1
7	2	6	3	6	6	3
8	4	5	5	2	7	3
9	6	5	5	6	5	4
10	5	2	3	2	7	5
11	2	3	4	2	5	6
12	1	1	1	2	1	1
13	1	1	2	2	3	1
14	3	3	5	7	3	5
15	5	5	3	7	7	4
16	2	7	6	20	20	3
17	0	N/A	1	N/A	0	0
18	5	6	5	N/A	N/A	4
19	4	6	6	N/A	N/A	3
20	4	4	4	6	4	4
21	3	2	1	1	1	6

Responses to Question 22:

- Weblab failed a few times → very frustrating, otherwise very useful and convenient. Many thanks to Masha. The Quanser hit the stops 2 times, could not test disturbance rejection (the system took over before collision).
- Purpose of lab should be more straight forward. Shorter, quicker examples. Shorter, but more labs would be better for me.
- Be very clear about what units all the numbers are in, please. Deg, rad, volt, times 10? This cost us hours.
- WebLab units not consistent. Roll/travel controller not good enough. Simulink diagram made no sense. Weblab access (windows only, java version, scheduling) frustrating.
- It would be nice if we could script the data collection, so students wouldn't sit for 7 hours collecting data. Notes on how many data sets should be sufficient to get a reasonable model would be nice. The ability to resize the graphs so we can better see the transient response of the system. Wider angle camera. Better description of the input fields and how to make a working Simulink controller(i.e. we input degrees not volts when designing the controller).

- The lab experience overall was somewhat frustrating, not because of any problems with the equipment (other than units), but only because it took such a long time (we spent about 20 hours in the lab — this included performing our analyses, but did not include writing our reports). Our time spent may have been above average because it took me longer than expected to develop the Mathematica scripts we used for processing our data. We also spent a long time trying to figure out why the observed compensated system behavior was different from our predictions, without success. In general, the web set-up was very well done, and the lab directions were clear. The general idea of the lab was also fun and interesting.

E.2 Survey results for the traditional laboratory users

Table E.3: Survey results for the traditional lab

Question #	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
3	10	10	3.5	3	5	2.5	10	3.5	5	1.5
4	1	3	6	2	4	7	5	6	6	2
5	5	1	8.5	2	3	5.5	5	6	3	1.5
6	2	2	3	4	2	6	1	5	2	6
7	5	3	4	2	6	6	3	3	1	1
8	6	4	4	2	7	5	4	5	6	2
9	6	5	3	3	1	6	5	6	6	1
10	1	2	5	1	5	1	3	1	1	1
11	1	1	1	1	1	3	3	2	1	1
12	1	2	5	2	1	5	3	2	1	1
13	1	2	6	2	3	6	3	3	3	2
14	4	2	5	1	2	5	3	4	5	1
15	1	2	2	2	2	3	4	4	2	1
16	5.5	0	2	0	1	5	20	1	0	0
17	20	3	20	1	5	3	1	15	10	1
18	3	N/A	4	2	6	7	5	N/A	N/A	N/A
19	2	N/A	4	3	6	7	5.5	N/A	N/A	N/A
20	5	6	5	6	1	6	3	4	4	6
21	1	2.5	2	5	N/A	1	3	2	2	6

Responses to Question 22:

- TAs should be present in the lab more often. Lab should be open after 5pm. Instruction sheet should be much clearer, clearly bulleting the needed data, what's going to be graded. Instruction sheet should be more step by step, not in machine operation, but in outlining what exactly needs to be done to properly complete the lab.

- It was good time for a first iteration of a controller design – not enough time or opportunities to ask for help to IMPROVE first design.

- Hard landed machine on the table almost every time – damn thing needs landing gear, without a partner it's hard to avoid this. Cool project, pain in the ass to carry out.

- The total time I've spent on this lab has been something close to over twenty hours. As this is a twelve unit class, that may want to be considered. Another option would be to not have the lab be due on a week when most people have midterms. In general, there need to be better directions, and more office hours for help. If something is done incorrectly in this lab, the error propagates through the results, causing the need to go back and redo previous portions of the lab. Thus it is important for students to have the ability to ask questions and show data in person.

- Clearer instructions would be helpful, such as plot Input with output!!! Don't have TA's office hours at beginning of week.
- Would help to have a staff member downstairs giving a tutorial or something before the actual lab.
- Didn't work with partner. I really enjoyed playing with the Quanser and felt that the lab was interesting and educational. However, this lab was in fact almost identical to what we did in 16.060 (except that we characterized pitch instead of elevation in .060). I would have liked to develop a more sophisticated controller for the lab, such as one that would simultaneously control all 3 degrees of freedom.

Appendix F

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Aeronautics and Astronautics
16.30: Estimation and Control of Aerospace Systems
Spring 2004

Remote Lab 1: Elevation Dynamics and Control
Due: 03/15/2004

Objective

The objectives of this lab are:

1. To use a collective motor voltage to experimentally determine the plant transfer function for the elevation axis of the Quanser from both step response and frequency response characteristics.

2. To design a controller for the elevation axis which achieves bandwidth of about 3 rad/sec and phase margin of at least 45° with no steady state error.

3. To implement the controller and compare its performance to that of he open loop system in terms of step response characteristics and disturbance rejection.

Administrative

You can do this lab individually or with one partner. Should you have any questions, e-mail *ishut@mit.edu*.

Introduction

Quanser helicopter is a mechanical device that essentially emulates the flight of a reduced degree of freedom (DOF) helicopter. Instead of the usual six DOF of a free-flying helicopter, the Quanser only exhibits three: the roll motion ϕ, the elevation motion θ, and the travel motion ψ. The Quanser system is actuated by two rotors and the inputs to the system are V_{cyc}, which is an electric voltage that results in differential change in the two rotor speeds, and V_{col}, which is an electric voltage that controls the speed of the two propellers collectively. The voltage is limited to 3 volts. The outputs of the system are the three angles: the roll ϕ, the elevation θ, and the travel ψ. Please keep the limits of the system in mind, in particular, the voltage is limited to $[-3, 3]$ Volts, pitch motion should be limited to $[-40°, 40°]$, the elevation motion should be limited to $[-15°, 25°]$.

Getting Started to do Identification

1. You should install Java (*http://java.sun.com/products/archive/j2se/1.4.0_03/*). The camera does not work with Java 1.4.2 so if you have that version, uninstall it and install Java 1.4.0. It is best to use Internet Explorer. Now, make sure you can see Java applets. You might want to check if you can see the camera by going to http://webchopper.mit.edu/javacampush. The experiment is set-up to run only from MIT IP addresses, let *ishut@mit.edu* know if that is problem.

2. Connect to http://webchopper.mit.edu, click run experiment, then here. You'll be taken to the University of Siena interface, scroll down to the 3DOF Helicopter experiment.

3. Click "Control Experiment". Enter appropriate personal data, and click "Identification" → "Run Experiment".

4. Unless there is a "Process Busy" message, which means someone else is using the machine, a java window should pop up. On the right you should have a camera window, if it is showing "Loading...Please Wait...", right click your mouse and choose "Restart".

5. This experiment has been set up for you to collect open loop data by applying V_{col} and V_{cyc}. The collective voltage for the identification is limited to [-0.2, 0.3] volts, and the cyclic voltage is limited to +/- 0.2 volts. For this lab, you will only be changing the V_{col}. You can apply either a Step input, or a Sine input, do not forget to click "Update Reference" if you want changes to take effect. For you information, at hover, $V_{col} = 1.55$ volts. In the Sine input options, *Frequency* has units of Hertz (1 Hz = 6.28 rad/sec), and *Center* is the offset of the sine on the y-axis. In order to keep the Quanser from swinging in the roll, you have an option to turn on a "Roll Controller" by changing "Reference Panel" → "Roll Cont" → "Step Value" to some nonzero number.

6. Click "Start Experiment". It takes 30 seconds for the experiment to initialize. Click on "Reference Panel" → "Show", here you can change your inputs. To see the output click on the appropriate box in the Graphics Panel. Note that when the supervisor control is on (plot shows 1), such as during initialization, stopping or when it overrides the user's actions, the outputs are shown to be zero. For visualization purposes, the reference input's amplitude has been multiplied by 10 and all the angles have units of degrees.

7. When you are done, click "End Session". You can save the outputs for later plotting in Matlab in either .mat or .zip format. You can follow these commands to plot the results:

   - load *filename.mat*
   - who
   - plot(rt_tout, rt_ElevationOutput);

   All the angles here have units of radians. Note that the time to do an individual experiment is limited to 5 minutes. Also, if the helicopter approaches its safety boundaries,

a supervisor controller turns on and shuts off your experiment, so you will have to start over.

Getting Started to test Controller

There are two ways to test your controller. The first one is using the predefined PID controller, where you just input Proportional, Integral, and Derivative gains for the elevation. The second one is by downloading your own controller designed in Simulink, you can also set that one up to change gains in real time.

Predefined PID

1. Before you plug in your controller, you should test it with the developed transfer function in Simulink.

2. If you choose to use a PID controller. Follow the links to the "Control Experiment", but now click "Run Experiment" for the "Elevation PID Controller".

3. Unless there is a "Process Busy" message, which means someone else is using the machine, a java window should pop up. On the right you should have a camera window, if it is showing "Loading...Please Wait...", right click your mouse and choose "Restart".

4. Click "Start Experiment". After 30 second initialization, a pre-set PID controller is turned on, you can change the appropriate gains by opening "Parameters Panel". Do not forget to click "Update Reference" if you want changes to take effect. In the "Reference Panel" you can choose an appropriate input, the units are in degrees (they are internally converted to radians). Here, you also have an option of turning on the Roll Controller by changing "Reference Panel" $\rightarrow$ "Roll Cont" $\rightarrow$ "Step Value" to some nonzero number. You can see if the "Roll Cont" in on (on = 1) by clicking on "Roll Control" button in the Graphics Panel. To see the output click on the appropriate box in the Graphics Panel. Note that when the supervisor control is on, such as during initialization, stopping or when it overrides the user's actions, the outputs are shown to be zero.

5. When you are done, click "End Session". You can save the outputs for later plotting in Matlab in either .mat or .zip format. Note that the time to do an individual experiment is limited to 5 minutes.

User-defined Elevation Controller

1. Before you plug in your controller, you should test it with the developed transfer function in Simulink.

2. If you choose not to use a PID controller. Follow the links to the "Control Experiment". To the left of the "Elevation PID Controller" you will see "Download Model", download the model onto your hard drive and modify it with your controller in Simulink. Remember that angles are measured in radians. If you want to modify some parameters real time, give them names in the form of *ACT_TP_Var Name*,

the way it is done in the template file. It IS case-sensitive.**Save it as a Simulink 3.0 version.** Now go back to that page and upload your controller into "Controller Model" field. "Controller Data" should be left blank, and the "Sample time" should be set to 10 msec. Press "Send Controller". It takes some time for you controller to compile.

3. Click "Start Experiment". After 30 second initialization, you controller is turned on.

Lab Procedure

1. Obtain and record the open-loop step response of the transfer function from the motor collective voltage to the elevation. From this data, identify the transfer function as well as you can.

2. Obtain experimental steady-state frequency response data to produce a Bode magnitude and phase plots for the transfer function from the motor collective voltage to the elevation angle. Since you don't get perfect sinusoids and you can't make perfect measurements, at each frequency, you should have a range of possible magnitudes and a range of possible phases. In getting the frequency response, let the system come to steady state at each frequency you try. You may want to run your signals through gains and/or summers with offset constants so that you can get the input and the output at the same magnitude and offset on the same plot in order to make a good phase measurement. Pick your frequencies wisely; start at dc, then 0.03 Hz and up. Please remember that the amplification of the plant is different at different frequencies. Pick the size of your inputs large enough to see the output easily (10-15 degrees is good).

3. Design a controller for the elevation axis that achieves a bandwidth of about 3 rad/sec and phase margin of at least 45° with no steady state error. Test your controller by making a Simulink simulation of your plant and the controller. Finally, implement your controller on the plant. Obtain and record the response of the elevation of your controlled system to a step command input of a change in elevation. Obtain and record the step response of the elevation of your controlled system to a step disturbance at the controller output or plant input.

Post Processing, Requirements and Write-up

- The write-up should be clear and concise. Include graphs or tables of your data as appropriate.

- Include the final model obtained using the open loop step response and the model obtained using frequency identification. Compare the performance of controlled system to the response of the open loop system. Compare the response of the controlled system to the one simulated.

Appendix G

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Aeronautics and Astronautics
16.30: Estimation and Control of Aerospace Systems
Spring 2004

Traditional Lab 1: Elevation Dynamics and Control
Due: 03/15/2004

Objective

The objectives of this lab are:

1. To use a collective motor voltage to experimentally determine the plant transfer function for the elevation axis of the Quanser from both step response and frequency response characteristics.
2. To design a controller for the elevation axis which achieves bandwidth of about 3 rad/sec and phase margin of at least 45° with no steady state error.
3. To implement the controller and compare its performance to that of the open loop system in terms of step response characteristics and disturbance rejection.

Administrative

The machines are located in the lab in Room 33-017. The lab is open 9am-5pm weekdays. There will be a TA available for you on Monday 3-5pm downstairs in the lab to answer any questions. You can do this lab individually or wi th one partner.

Introduction

Quanser helicopter is a mechanical device that essentially emulates the flight of a reduced degree of freedom (DOF) helicopter. Instead of the usual six DOF of a free-flying helicopter, the Quanser only exhibits three: the roll/pitch motion ϕ, the elevation motion θ, and the travel motion ψ. The Quanser system is actuated by two rotors and the inputs to the system are V_{cyc}, which is an electric voltage that results in differential change in the two rotor speeds, and V_{col}, which is an electric voltage that controls the speed of the two propellers collectively. The outputs of the system are the three angles: the pitch ϕ, the elevation θ, and the travel ψ. For this lab, you are going to be working with the elevation mode. Please keep the limits of the system in mind, in particular, the voltage is limited to

$[-5, 5]$ Volts, pitch motion should be limited to $[-40°, 40°]$, the physical limits on elevation are $[-25°, 30°]$.

Getting Started on Computer

1. On the computer, make yourself a directory in C:/MATLAB6p1/work/16.30/. Copy the files lab1_ident.mdl and lab1_ctrl.mdl to your directory. IMPORTANT NOTE: at the end of the lab session, you should save all of your work to a floppy disk (or scp it to your Athena account) and clean out the directory on the hard drive. You only need to save your .mdl files and your data. Simulink creates a lot of runtime files that you don't need to save. IF YOU LEAVE YOUR FILES ON THE HARD DRIVE THEY MAY BE DELETED.

2. Make sure you are working in your own directory. Double click on lab1_ident.mdl to start Matlab. You should see the beginnings of a Simulink model appear. The "Quanser Subsystem" block is the computer's interface to the Quanser. It applies the given voltages to the two engines and reads the elevation angle in radians. Be aware that the Quanser takes all measurements to be zero when you click Run. The elevation angle has been adjusted so that it reads zero at hover if you start on the table.

3. You will need to use step and sine inputs to obtain the model. Work with the file lab1_ident.mdl. Obtain the corresponding input blocks in the Simulink Library Browser → Sources → Step and adjust parameters accordingly. Click OK. Connect the blocks to the Collective voltage inputs. The hover voltage has been set for you to keep the Quanser horizontal. You are also given an option to turn on a *Pitch Controller* in order to keep the Quanser from swinging. To do that double-click on the "Quanser Subsytem" block and check off the box for the "Turn on the Pitch Controller".

4. Build the Simulink model by choosing Tools/Real Time Workshop/Build (or Ctrl-B).

5. Before you start your system, set up your scope to capture the data. In the WinCon Server window choose Plot/New/Scope. Open the 'Elevation' folder. The measurements are done in radians. In the Axis menu of the scope plot choose Auto Scaled. In the Update menu, choose Real time and choose Buffer, set it to the time you want to run the experiment.

6. Press Run and record the response. After the Quanser reaches a steady state, go to Update- Freeze Plot. You can now save the data to the Matlab workspace by choosing 'File/Save/Save to Workspace'. Or you can press Stop and save the data as before.

7. The plot data is now stored in Matlab. To see its name, type *who* at the Matlab prompt. Two arrays have been created: PLOT_TIME stores the x-axis (time) and lab1_elev_rad_ stores the y-axis (elev angle). We need to save the data to a different Matlab variable. We also need to truncate the vector (so that all of our responses are the same length). We will keep the first 25,000 elements of the response. To rename lab1_elev_rad_ to STEP1 and truncate, type 'STEP1 = lab1_elev_rad_(1:25000,1);' at the Matlab prompt. Note that your plot data may have a slightly different name. Plot the response in Matlab: 'plot(PLOT_TIME, STEP1)'.

8. To implement your controller, you might want to use lab1_ctrl.mdl file and modify it accordingly. Again, you have an option to turn on the Pitch Controller by double clicking on the "Quanser Subsystem".

Lab Procedure

1. Record the number of the machine and the position of the counterweight. Should anything go wrong, apply the *kill switch*, i.e. flip the lighted button on the table.

2. Turn on the machine so that it is at hover, always start on the table to ensure that the angle is zero when the arm is horizontal.

3. Obtain and record the open-loop step response of the transfer function from the motor collective voltage to the elevation. From this data, identify the transfer function as well as you can.

4. Obtain experimental steady-state frequency response data to produce a Bode magnitude and phase plots for the transfer function from the motor collective voltage to the elevation angle. Since you don't get perfect sinusoids and you can't make perfect measurements, at each frequency, you should have a range of possible magnitudes and a range of possible phases. In getting the frequency response, let the system come to steady state at each frequency you try. You may want to run your signals through gains and/or summers with offset constants so that you can get the input and the output at the same magnitude and offset on the same plot in order to make a good phase measurement. Pick your frequencies wisely; start at dc, then 0.3 rad/sec and up. Please remember that the amplification of the plant is different at different frequencies. Pick the size of your inputs large enough to see the output easily (10-15 degrees is good).

5. Design a controller for the elevation axis that achieves a bandwidth of about 3 rad/sec and phase margin of at least 45° with no steady state error. Test your controller by making a Simulink simulation of your plant and the controller. Finally, implement your controller on the plant. Obtain and record the response of the elevation of your controlled system to a step command input of a change in elevation. Obtain and record the step response of the elevation of your controlled system to a step disturbance at the controller output or plant input.

Post Processing, Requirements and Write-up

- The write-up should be clear and concise. Include graphs or tables of your data as appropriate.

- Include the final model obtained using the open loop step response and the model obtained using frequency identification. Compare the performance of controlled system to the response of the open loop system. Compare the response of the controlled system to the one simulated.

Bibliography

[1] B. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice Hall, New York, NY, 1990.

[2] J. Apkarian and A. Dawes. Interactive control education with virtual presence on the web. In *Proceedings of the American Control Conference*, pages 3985–3990, Chicago, Illnois, June 2000.

[3] A. E. Bryson. *Control of spacecraft and aircraft*. Princeton University Press, Princeton, NJ, 1994.

[4] M. Casini, D. Prattichizzo, and A. Vicino. The automatic control telelab: a user-friendly interface for distance learning. *IEEE Transactions on Education*, 46(2), May 2003.

[5] C. W. Dever. *Flexible Maneuvering for Autonomous Vehicles*. PhD dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2004.

[6] W.E. Dixon, D.M. Dawson, B.T. Costic, and M.S. de Queiroz. Towards the standardization of a matlab-based control systems laboratory experience for undergraduate students. In *Proceedings of the American Control Conference*, pages 1161–1166, June 2001.

[7] H.H. Hahn and M.W. Spong. Remote laboratories for control education. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pages 895–900, December 2000.

[8] Z. Jing, C. Jianping, C.C Ko, B.A. Chen, and S.S. Ge. A web-based laboratory on control of a two-degree-of-freedom helicopter. In *Proceedings of the 40th IEEE Conference on Decision and Control*, volume 3, pages 2821–2826, December 2001.

[9] *http://www.mathworks.com*.

[10] S. Mitra, Y. Wang, N. Lynch, and E. Feron. Safety verification of model helicopter controller using hybrid Input/Output automata. In *HSCC'03, Hybrid System: Computation Control*, Prague, the Czech Republic, April 3-5 2003.

[11] G. D. Padfield. *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*. AIAA Education Series, Reston, VA, 1995.

[12] *http://www.quanser.com*.

[13] *http://www.vci.com/products/windows_embedded/index.asp*.

[14] M. G. Safonov and M. Athans. Gain and phase margin for multiloop lqg regulators. *IEEE Transactions on Automatic Control*, 22(2):173–179, 1977.

[15] K. Sprague, V.Gavrilets, D. Dugail, B. Mettler, I. Martinos, and E. Feron. Design and applications of an avionics system for a miniature acrobatic helicopter. In *Digital Avionics Systems Conference*, October 2001.

[16] M. B. Tischler, M. G. Cauffman, G. Villere, L. Pierce, and D. Hermstad. *http://caffeine.arc.nasa.gov/cifer/*.

[17] *http://www.webcam32.com/support/webcam32help/index.html*.

[18] C.D. Wickens, S.E. Gordon, and Y. Liu. *An introduction to human factors engineering*. Longman, New York, 1998.

[19] I. Yavrucuk, S. Unnikrishnan, and J. V. R. Prasad. Envelope protection in autonomous unmanned aerial vehicles. In *Proceedings of the 59th American Helicopter Society Annual Forum*, May 2002.