EE-479 Digital Control System
Project 1
Flexible Link

Purpose

This control project involves positioning the flexible link to a set point using a state feedback controller to damp out the vibration at the tip of the link as quickly as possible with minimal vibrations. The objectives of this project are:

- To obtain a linear state-space model for the Flexible Link module
- To design a state feedback controller that damps out the vibrations at the tip of the beam.
- Build the compensated servo plant in SIMULINK and simulate offline to obtain the response to a given command input.
- Build the WinCon application, implement and test the system on the real-time hardware

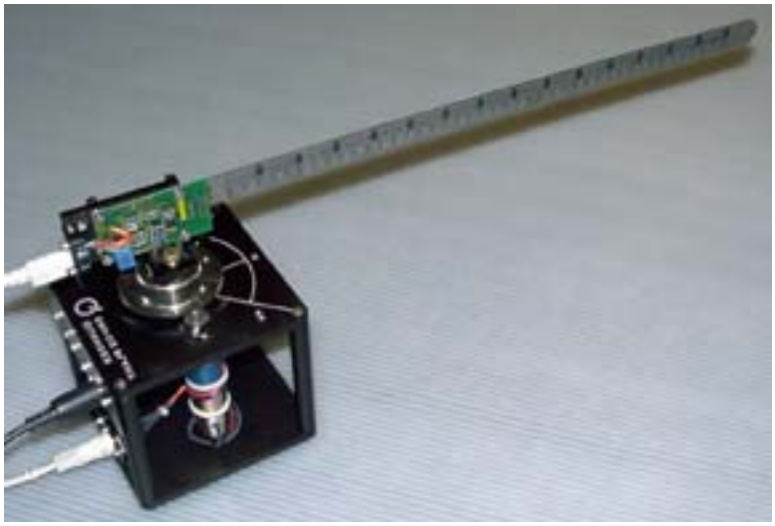

Figure 1.1 Flexible Link module coupled to SRV02 Servomotor in high gear ratio

Introduction

This project addresses the problem of controlling a flexible link with a state feedback controller. Numerous applications are mechanical actuators, control of robot arms, space shuttle arms, and flexible space structures.

The flexible link module includes a SRV02 servomotor in the high gear ratio configurations and a flexible link. The flexible link consists of a single lightweight arm that flexes and bends during rotation. Tip deflection $\alpha(t)$ is measured with a strain gauge at the motor end of the link. The strain gage output is calibrated to output 1 Volt per inch of deflection at the tip. A DC motor located at the base actuates the arm. An optical encoder attached to the shaft of the DC motor is used to measure the angular position of the shaft $\theta(t)$.

Because of the flexibility of the link, rotating the base of the link causes the entire link to oscillate. The objective of this challenge is to design a feedback controller to position the tip of the link to a desired set point as quickly as possible with minimal vibration by a control input at the base away from the tip of the link.

.

1. Mathematical Modeling

1.1 Servomotor Model

In the position control experiment (EE-371 Lab 5) the servomotor model was developed with the following block diagram as shown in Figure 1.2.

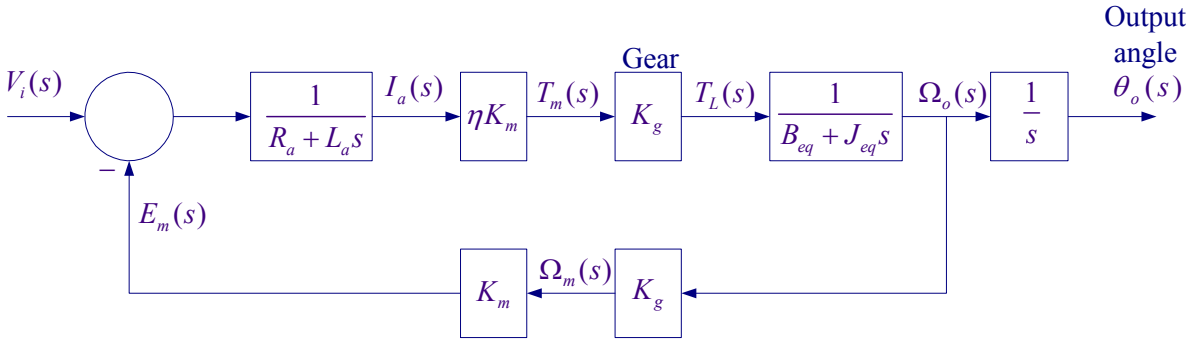

Figure 1.2 Servo plant block diagram

$$\frac{\theta_o(s)}{V_i(s)} = \frac{\frac{\eta K_m K_g}{R_a J_{eq}}}{s\left(s + \frac{B_{eq}}{J_{eq}} + \frac{\eta K_m^2 K_g^2}{R_a J_{eq}}\right)} \tag{1.1}$$

or

$$\frac{\theta_o(s)}{V_i(s)} = \frac{a_m}{s(s + b_m)} \tag{1.2}$$

Where

$$a_m = \frac{\eta K_m K_g}{R_a J_{eq}}, \qquad b_m = \frac{B_{eq}}{J_{eq}} + \frac{\eta K_m^2 K_g^2}{R_a J_{eq}} \tag{1.3}$$

Also from the block diagram the s-domain output torque $T_L(s)$ is

$$T_L(s) = \frac{\eta K_m K_g}{R_a}\left[V_i(s) - K_m K_g \Omega_0(s)\right]$$

Therefore the expression for the output torque in time-domain is

$$T_L(t) = \frac{\eta K_m K_g}{R_a} v_i(t) - \frac{K_m^2 K_g^2}{R_a}\dot{\theta}(t) \tag{1.4}$$

In this project the servomotor is arranged for the high gear ratio as shown in Figure 1.3. For this configuration the gear ratio is $K_g = (14)(5)$.

Figure 1.3 High-gear ratio configurations

The system parameters are as follows:
Armature resistance, $R_a = 2.6\ \Omega$

Motor voltage constant, $K_m = 0.00767$ V-s/rad

Motor torque constant, $K_\tau = 0.00767$ N-m/A

Armature inertia, $J_m = 3.87\times10^{-7}$ Kg m^2

Tachometer inertia, $J_{tach} = 0.7\times10^{-7}$ Kg m^2

High gear ratio, $K_g = (14)(5)$

Equivalent viscous friction referred to the secondary gear $B_{eq} = K_g^2 B_m + B_L \simeq 4\times10^{-3}$ Nm/(rad/s)

Motor efficiency due to rotational loss $\eta_{mr} \simeq 0.87$

Gearbox efficiency, $\eta_{gb} \simeq 0.85$

$\eta = \eta_{mr}\eta_{gb} = (0.87)(0.85) = 0.7395$

Gear inertia:

$J_{120} = 4.1835\times10^{-5}$ Kg m^2

$J_{72} = 5.4435\times10^{-6}$ Kg m^2

$J_{24} = 1.0081\times10^{-7}$ Kg m^2

Load inertia, $J_L = J_{120} + 2(J_{72}) + J_{24} = 5.2823\times10^{-5}$ Kg m^2

$$J_{eq} = K_g^2(J_m + J_{tach}) + J_L = (14\times 5)^2(3.87+0.70)\times 10^{-7} + 5.2823\times 10^{-5} = 0.0023 \text{ Kg m}^2$$

1.2 Flexible Link Model

The parameters of the flexible module are defined as follows:

θ	Servo gear angular displacement
ω	Servo gear angular velocity
α	Link angular deflection
υ	Link angular velocity
γ	Total deflection $\gamma = \theta + \alpha$
L	Flexible link length ($L = 15$ inches $= 0.381$ m)
D	End point arc length deflection ($D = \alpha L$)
m	Mass of flexible link ($m = 65$ gm)
J_{arm}	Link's moment of inertia ($J_{arm} = \frac{1}{3}mL^2 = \frac{1}{3}(0.065)(0.381)^2 = 0.0031452$)
ω_{FL}	Link's damped natural frequency (f_L measured to be 3Hz, $\omega_{FL} = 6\pi$)
K_{stiff}	Link's stiffness ($K_{stiff} = \omega_{FL}^2 J_{FL} = (6\pi)^2(0.0031452) = 1.1175$)
K_{Gage}	Strain gage calibration factor (1 Volt/inch)

Consider the Flexible link schematic shown in Figure 1.4.

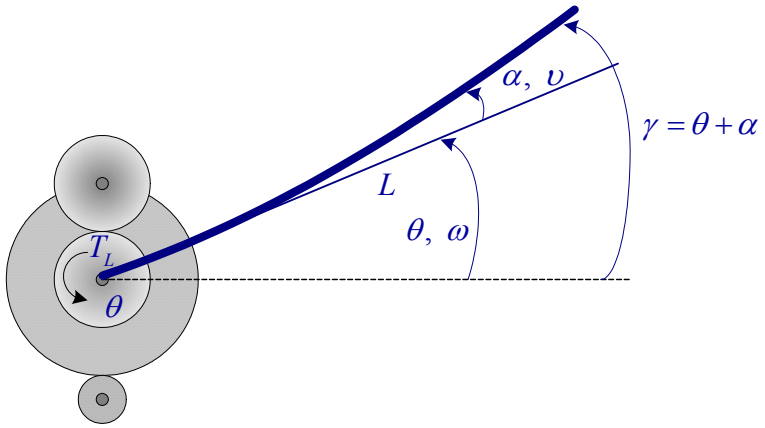

Figure 1.4 A schematic picture of the Flexible Link.

From the above definition

$$\dot{\theta} = \omega$$
$$\dot{\alpha} = \upsilon \qquad (1.5)$$

If J_{arm} is the link's moment of inertia, the torque due to the link acceleration is

$$T_{J_{arm}} = J_{arm}\frac{d^2\gamma}{dt^2} = J_{arm}(\ddot{\theta} + \ddot{\alpha}) = J_{arm}(\dot{\omega} + \dot{\upsilon}) \qquad (1.6)$$

The link torque due to torsional spring stiffness K_{stiff} is assumed to be proportional to the link's deflection α, i.e.,

$$T_{K_{stiff}} = K_{stiff}\alpha \tag{1.7}$$

$$T_{J_{arm}} + T_{K_{stiff}} = 0$$

or

$$J_{arm}\left(\dot{\omega} + \dot{\upsilon}\right) + K_{stiff}\alpha = 0 \tag{1.8}$$

The servomotor output torque in addition to overcoming the inertia torque due to J_{eq} and frictional torque, it is assumed to overcome the torque due to link's acceleration, i.e.,

$$J_{eq}\dot{\omega} + B_{eq}\omega + J_{arm}\left(\dot{\omega} + \dot{\upsilon}\right) = T_L \tag{1.9}$$

Substituting for $J_{arm}\left(\dot{\omega} + \dot{\upsilon}\right)$ from (1.8) into (1.9), we have

$$\dot{\omega} = \frac{K_{stiff}}{J_{eq}}\alpha + \frac{1}{J_{eq}}T_L - \frac{B_{eq}}{J_{eq}}\omega \tag{1.10}$$

Substituting for T_L from (1.4), yields

$$\dot{\omega} = \frac{K_{stiff}}{J_{eq}}\alpha - \frac{\eta K_m^2 K_g^2 + B_{eq}R_a}{J_{eq}R_a}\omega + \frac{\eta K_m K_g}{J_{eq}R_a}v_i \tag{1.11}$$

Substituting for $\dot{\omega}$ from (1.11) into (1.9), we obtain

$$\dot{\upsilon} = \frac{K_{stiff}(J_{eq} + J_{arm})}{J_{eq}J_{arm}}\alpha + \frac{\eta K_m^2 K_g^2 + B_{eq}R_a}{J_{eq}R_a}\omega - \frac{\eta K_m K_g}{J_{eq}R_a}v_i \tag{1.12}$$

We now obtain an state-space model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\,\boldsymbol{x}(t) + \boldsymbol{B}\,u(t) \tag{1.13}$$

$$y(t) = \boldsymbol{C}\,\boldsymbol{x}(t) \tag{1.14}$$

For the combined servomotor and the flexible link module, we choose the state variables $\boldsymbol{x}(t)$ as

$$\boldsymbol{x}(t) = \begin{bmatrix}\theta & \alpha & \omega & \upsilon\end{bmatrix}^T$$

Writing (1.5), (1.11) and (1.12) in matrix form, we obtain the following state-space model

$$\begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \\ \dot{\omega} \\ \dot{\upsilon} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{K_{stiff}}{J_{eq}} & -\dfrac{\eta K_m^2 K_g^2 + B_{eq} R_a}{J_{eq} R_a} & 0 \\ 0 & -\dfrac{K_{stiff}(J_{eq} + J_{arm})}{J_{eq} J_{arm}} & \dfrac{\eta K_m^2 K_g^2 + B_{eq} R_a}{J_{eq} R_a} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \omega \\ \upsilon \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{\eta K_m K_g}{J_{eq} R_a} \\ -\dfrac{\eta K_m K_g}{J_{eq} R_a} \end{bmatrix} v_i \quad (1.15)$$

2. Regulator Design

(a) Pole-placement

Stat-space feedback is the most important aspect of modern control system. By proper state feedback, unstable systems can be stabilized or damping of oscillatory systems can be improved. One basic approach is known as the *pole-placement design*. Pole placement design allows the placement of poles at specified locations, provided the system is controllable**.** This results in a regulator with constant gain vector $\boldsymbol{K}$. However, the basic pole placement method is strongly dependent on the availability of an accurate model of the system. Other approach to the design of regulator systems is the optimal control problem where a specified mathematical performance criterion is minimized. In optimal regulator design random disturbances can be rejected and the closed-loop system can be made insensitive to changes of plant dynamics.

In pole-placement design the control is achieved by feeding back the state variables through a regulator with constant gains. Consider a linear continuous-time controllable system, modeled in state-space form as given by (1.13)-(1-14). The block diagram of the system with the following state feedback control is shown in Figure 1.5.

$$u(t) = -\boldsymbol{K}\ x(t) \quad (1.16)$$

where $\boldsymbol{K}$ is a $1 \times n$ matrix of constant feedback gain. The control system input is assumed to be zero. The purpose of this system is to return all state variables to values of zero when the states have been perturbed.

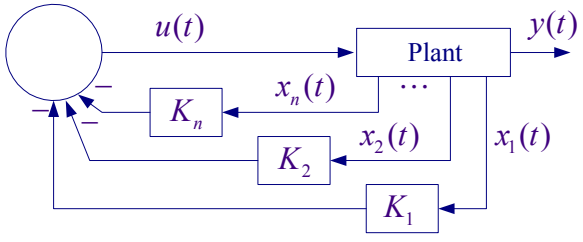

Figure 1.5 Control system design via pole placement.

Substituting (1.16) into (1.13), the closed-loop system state-variable representation become

$$\dot{\boldsymbol{x}}(t) = [\ \boldsymbol{A} - \boldsymbol{BK}\ ]\ \boldsymbol{x}(t) = \boldsymbol{A}_f \boldsymbol{x}(t) \qquad (1.17)$$

The design objective is to find the gain matrix $\boldsymbol{K}$ such that the characteristic equation for the controlled system is identical to the desired characteristic equation. The derivation when matrix $\boldsymbol{A}$ is in phase variable control canonical form is straightforward. When the system is not in phase variable form, first the system is transformed into the phase variable control canonical form; refer your textbook and lecture notes or to "Computational Aids in Control Systems using MATLAB, Hadi Saadat, McGraw-Hill 1993, Chapter 8, page 170." A custom-made function named **[K, Af] = placepol(A, B, C, P)** is developed for the pole placement design. A, B, C are system matrices and P is a row vector containing the desired closed-loop poles. This function returns the gain vector K and the new system matrix Af. Also, the MATLAB Control System Toolbox contains two functions for pole-placement design. Function **K = acker(A, B, P)** is for single input systems, and function **K = place(A, B, p)** suitable for multi-input systems. The plant described by (1.13) with the system matrix having dimension $n \times n$ is completely state controllable if and only if the controllability matrix S has a rank of n.
[Refer to Chapter 8 p. 175 in the above reference]. A function **S = cntrable(A, B)** is developed which returns the controllability matrix S and determines whether or not the system is state controllable. Also, the function **ctrb(A, B) in** MATLAB Control system toolbox can be used to determine the controllability matrix. In pole-placement design, it was assumed that all state variables are available for feedback. However, in practice it is impractical to install all the transducers, which would be necessary to measure all of the states. If the state variables are not available an observer or estimator is designed.

(b) Optimal Regulator

Instead of the pole-placement design described above, the controller can be designed using the Linear Quadratic Regulator (LQR). The object is to determine the optimal controller $u(t) = -\ \boldsymbol{K}\ \boldsymbol{x}(t)$ such that a given performance index $\boldsymbol{J} = \int (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}) dt$ is minimized. The performance index is selected to give the best performance. The choice of the elements of $\boldsymbol{Q}$ and $\boldsymbol{R}$ allows the relative weighting of individual state variables and individual control inputs. For example, using an identity matrix $\boldsymbol{Q}$ for weights all the states equally. As a starting point you may use a diagnol matrix with values $Q = diag([260 \quad 3600 \quad 2 \quad 1])$ and $R = 1$.The MATLAB Control System Toolbox function **[k, S] = lqr2(A, B, Q, R)** calculates the optimal feedback matrix K such that it minimizes the cost function J subject to the constraint defined by the state equation. Also returned is S, the steady-state solution to the associated algebraic Riccati equation. Refer to your textbook and lecture notes or to "Computational Aids in Control Systems using MATLAB, Hadi Saadat, McGraw-Hill 1993, Chapter 8, page 180

2.2 Closed-loop poles specifications − Pole-placement

The state equation obtained in (1.15) results in a fourth order characteristic equation. For the compensated closed-loop system we choose the poles as a desired pair of dominant second-order poles, and select the rest of the poles to have real parts corresponding to sufficiently damped modes so that the system will mimic a second-order response with reasonable control effort. Pick the low-frequency modes and the high-frequency modes as follows:

- A pair of dominant second-order poles with a time constant of $\tau = 0.1$ second and a damping ratio of $\zeta = 0.8$.
- Select two real poles with very small time constants $\tau_3 = 0.05$ second and $\tau_4 = 0.008$ second.

3. Pre Laboratory Assignment

3.1 Open-loop Analysis

Define the servomotor and flexible link parameters in a script m-file and determine the system $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices. Use **damp(A)** to obtain the roots, damping ratios and natural frequencies of the uncompensated characteristic equation. Comment on the open-loop system stability and obtain the open-loop step response. Consider a step input of 30°. To find the step response for θ as output, define C = [1 0 0 0], D = 0, and use

```
[numo, deno] = ss2tf(A, B, C, D, 1)
Gp = tf(numo, deno)
ltiview('step', 30*Gp)
```

Use function **S = cntrable(A, B)** or **S = ctrb(A, B)** to determine the system controllability.

3.2 Regulator Design

Based on the design specifications given in section 2.2 locate the desired closed-loop poles and form the vector **P** containing the four poles. Use function **[K, Af] = placepol(A, B, C, P)** to obtain the gain vector **K** and the new system matrix **Af**. Add the following statements to obtain the compensated transfer function and the step response to a 30° step input

```
[K, Af] = placepol(A, B, C, P)
[numc, denc] = ss2tf(Af, B, C, D, 1)
Tc = tf(numc, denc)
ltiview('step', 30*Tc)
```

Use **damp** function to obtain the compensated system roots and confirm the pole-placement design. Comment on the system response.

Alternatively apply LQR design with $Q = diag([260 \quad 3600 \quad 2 \quad 1])$ and $R = 1$ and find the state feedback gain matrix K.

3.3 Digital Simulation

The Simulink simulation diagram for the compensated system with state feedback named "FlexLink_sim.mdl" is constructed as shown in Figure 1.6. In the Simulink block diagram you can replace A, B, C, D, Ks(1), Ks(2), Ks(3), and Ks(4) with the computed values. Alternatively, you can place all equations in a script m-file to compute the parameters, which are sent to the MATLAB Workspace, then simulate the Simulink diagram. From the Simulink/Simulation Parameters select the Solver page and for Solver Option Type select Fixed-step and ode4 (Runge-Kutta) and set the fixed step size to 0.001.

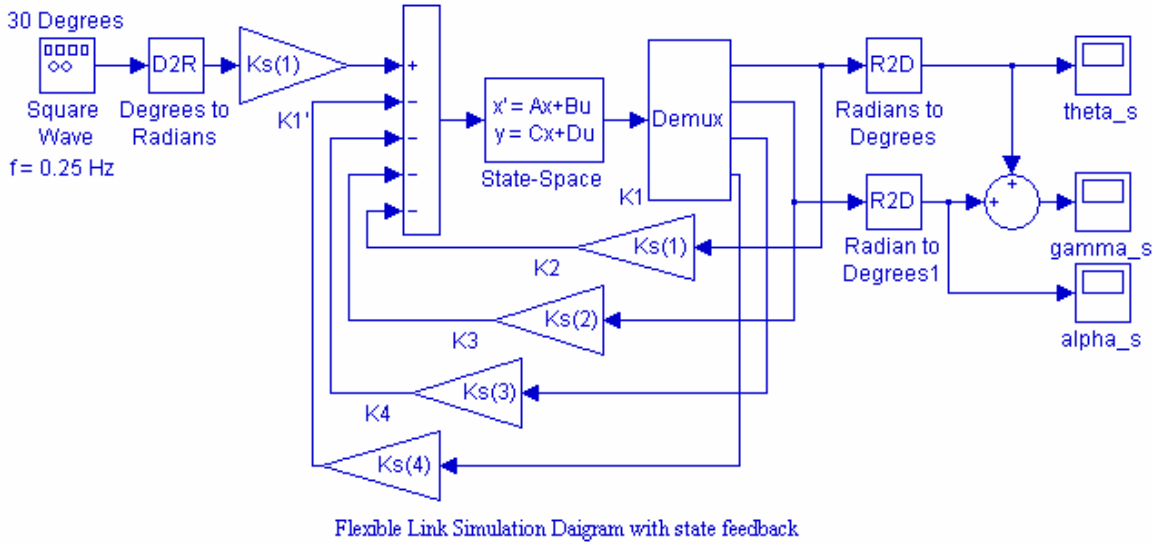

Figure 1.6 Simulink simulation diagram for the Flexible Link

Obtain the response to a square input of amplitude 30 degrees and frequency 0.25 Hz.

4. Laboratory Procedure

When you have finished testing your model in Simulink, it has to be prepared for implementation on the real-time hardware. This means the plant model has to be replaced by the I/O components that form the interfaces to the real plant.

4.1 Creating the Implementation model

The SRV02-E FlexGage Interface

One encoder is used to measure the angular position of the servomotor, θ and a strain-gage sensor is used to measure the link deflection α. We must use two observers to reconstruct the state variables ω, and υ which are not directly measured. These are the derivatives of θ, and α. Remembering that we cannot implement a pure derivative, we use the low-pass band-limited observer $\frac{50s}{s+50}$. This filter is tuned to reduce the noise introduced in the numerical derivative. Open a Simulink page and get the part Encoder Input. Double-click on the Encoder input block to open its dialog box and set the Channel Use to 0. Since the encoder has 4096 signal periods per revolution, get a gain block and set its value to $-2*pi/4096$. A negative sign is used for the encoder gain, because the negative feedback gain is already implemented in the encoder wiring (i.e. positive voltage $\Rightarrow$ negative counts). Get a Transfer function block $\frac{250}{s+250}$ for the low-pass band-limited filter. Use another block for the low-pass band-limited differentiator $\frac{250s}{s+250}$ to obtain ω. Get an analog input for getting the link deflection from the strain-gage sensor. Double-click to open its dialog box and set the Channel to Use to 1. Get a gain block and set its value to $-1/19$ for the Strain Gage calibration factor and pass the signal through a band-limited filter $\frac{25}{s+25}$ to reduce the noise. Add another low-pass band-limited filter $\frac{25s}{s+25}$ to obtain the derivative of α as shown in Figure 1.7. This filter is tuned to reduce the noise introduced in the numerical derivative.

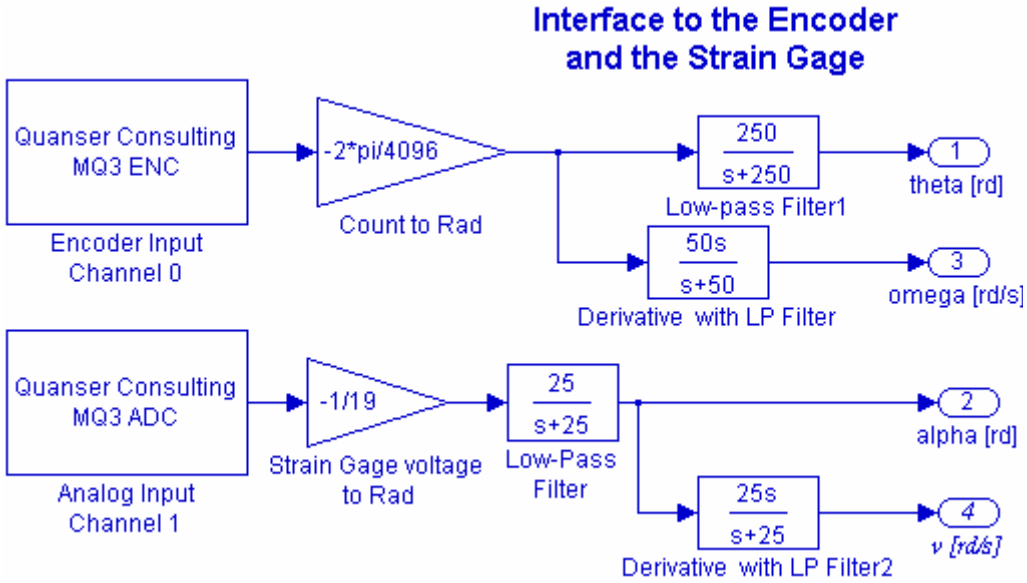

Figure 1.7 Content of the subsystem Encoder and Strain Gage Input.

To create a subsystem, enclose the blocks within a bounding box (Do not place any outport blocks). Choose **Create Subsystem** from the **Edit menu**. Simulink replaces the selected blocks with a subsystem block. Double-click to open the subsystem block. Notice that the Simulink automatically adds Outport blocks. Label the Outports appropriately as shown in Figure 1.8. Rename and title subsystem to Encoder and Strain Gage Input, and save it as FlexLink_Interface.mdl. You know have the following subsystem as shown in Figure 1.8.

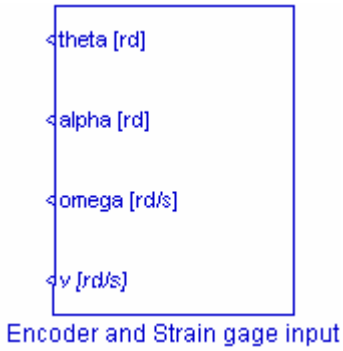

Figure 1.8 Subsystem block for Encoder and Strain Gage Interface

If you double-click on the above subsystem it would display the underlying system as shown in Figure 1.7.

The implementation model can be added on the previously constructed Simulink model. This would enable you to obtain the simulation and actual results simultaneously. Open FlexLink_Sim.mdl (your simulation model), save it under a new name (say FlexLink_Imp.mdl). Start constructing the implementation diagram. Copy the FlexLink_Interface.mdl (constructed in part 4.1, Figure 1.8) to the clipboard and paste it on a new Simulink page as shown in Figure 1.9. Get the Analog Output block from the Quanser MultiQ3 library and set the Channel Use to 0. Use a Signal Generator with amplitude 30 degrees, and frequency 0.25 Hz, and use a D2R block to convert to radians. Add four state feedback gain blocks and connect to the Interface outputs via some Sliders Gain blocks. Complete the feedback loops and connect the resulting signal from the summing block to the Quanser Analog output. Place as many Scopes as you like to monitor the phase angles, etc.

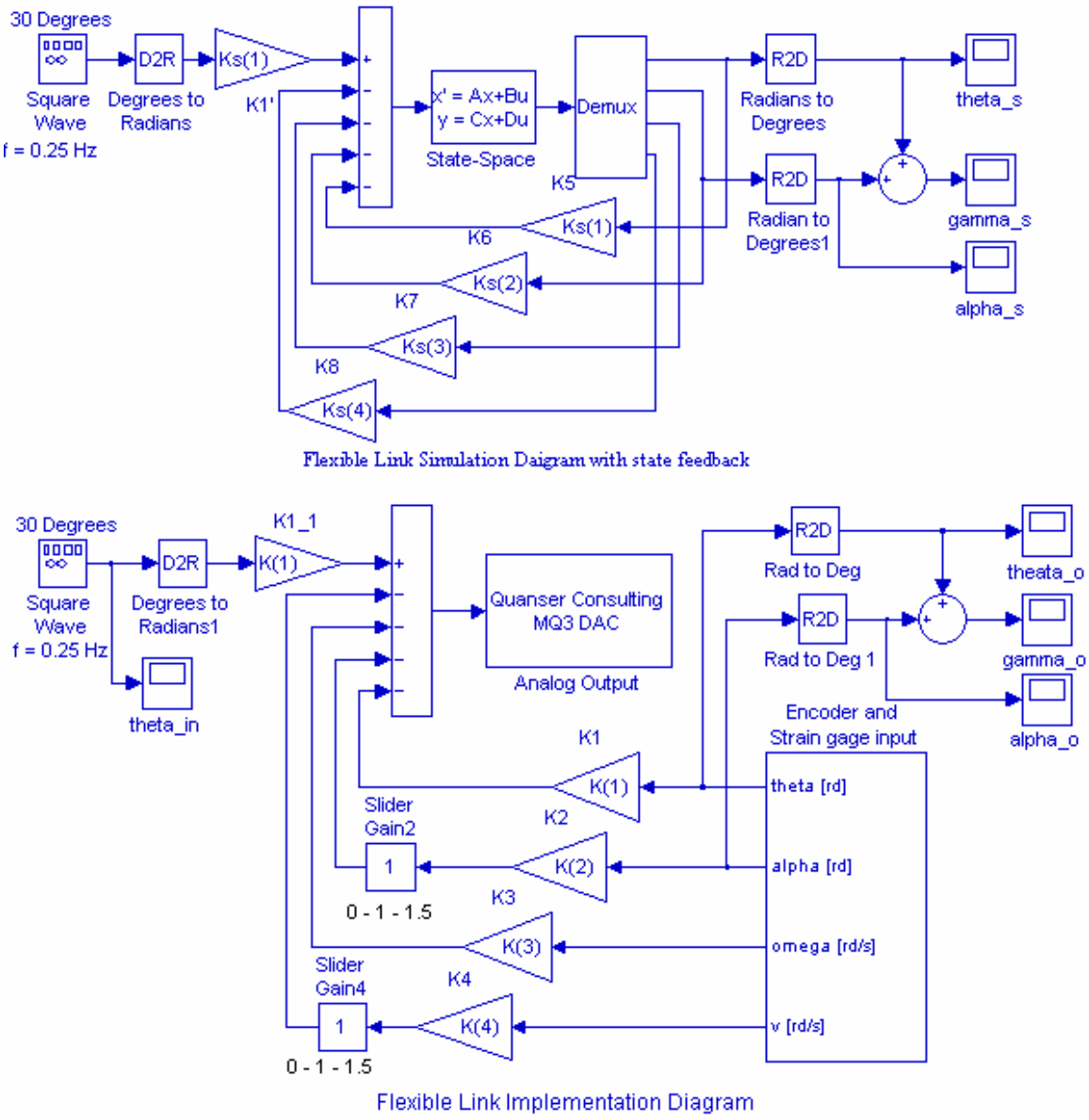

Figure 1.9 Simulation and Implementation diagram for Flexible Link project.

Your completed model should be the same as shown in Figure 1.9. Set the state feedback gains to the values found in part (3.2), or run the m-file that returns the K array. The gains in Simulink Simulation diagram are renamed to Ks(1) – Ks(4), so that if the value of the variables K are changed at the MATLAB prompt for fine tuning, the values in the Simulation diagram are not changed.

4.2 Wiring diagram

Attach the Flexible Link module to the SRV02 as shown in Figure 1.1. Tighten the two thumb screws very well. Clamp the SRV02 to the table so that it does not tip when the link is oscillating.
Using the set of leads, universal power module (UPM), SRV-02 DC-motor, and the connecting board of the MultiQ3 data acquisition board, complete the wiring diagram shown in Figure 1.10 as follows:

From	To	Cable
Flexible Link sensor	S1, S2 on UPM	6 pin mini Din to 6 pin mini Din
Encoder on SRV02	MultiQ/Encoder 0	5 pin Din to 5 pin Din
Motor on SRV02	UPM/To Load	6Pin to 4 Pin Din, **Gain 1 Cable**
D/A #0 on MultiQ	UPM – From D/A	RCA to 5 Pin Din
A/D # 0, 1, 2, 3, on MultiQ	UPM- TO A/D	5 Din to 4xRCA

Figure 1.10 Wiring diagram for Flexible Link Project.

4.3 Compiling the model

In order to run the implementation model in real-time, you must first build the code for it. Turn on the UPM. Start WinCon, Click on the MATLAB icon in WinCon server. This launches MATLAB. In the Command menu set the Current Directory to the path where your model FlexLink_Imp.mdl is. Before building the model, you must set the simulation parameters. Pull down the Simulation dialog box and select Parameters. Set the Start time to 0, the Stop time to 10, for Solver Option use Fixed-step and ode4 (Runge-Kutta) method set the Fixed-step size, i.e., the sampling rate to 0.001. In the Simulation drop down menu set the model to **External**. Set the Matrix gains K to the values found in part (3.2), or run the m-file that returns the values of the matrix K .

Start the WinCon **Server** on your laptop and then use **Client Connect**, in the dialog box type the proper Client workstation IP address. Generate the real-time code corresponding to your diagram by selecting the "**Build**" option of the WinCon menu from the Simulink window. The MATLAB window displays the progress of the code generation task. Wait until the compilation is complete. The following message then appears: "### *Successful completion of Real-Time Workshop build procedure for model: FlexLink_Imp*".

4.4 Running the code

Following the code generation, WinCon Server and WinCon Client are automatically started. The generated code is automatically downloaded to the Client and the system is ready to run. To start the controller to run in real-time, click on the **Start** icon from the WinCon Server window shown in Figure 1.11. It will turn red and display STOP. The Flexible Link should have negligible oscillations at the tip of the beam during cyclic rotation.

Clicking on the **Stop** icon will stop the real-time code and return to the green button.

Figure 1.11 WinCon Server

If at any point the system is not behaving as expected, immediately press STOP on the WinCon server If you hear a whining or buzzing in the motor you are feeding high frequency noise to the motor or motor is subjected to excessive voltage, immediately stop the motor. Ask the instructor to recheck the implementation diagram and the state feedback gains before proceeding again.

4.5 Plotting Data

You can now plot in real-time any variables of your diagram by clicking on the "**Plot/New/Scope**" button in the WinCon Server window and selecting the variable you wish to visualize. Select "gamma_o" and click OK. This opens one real-time plot. From Scope pull-down menu, select Buffer and set the Buffer Size to 10. To plot more variables in that same window, click on "**File/Variables…**" from the Scope window menu. The names of all blocks in the Simulink model diagram appear in a Multiple Select Variable Tree. You can then select the variable(s) you want to plot. In this case, select "theta_in" and "gamma_os. From the File menu you can Save and Print the graph. If you choose Save As M-File … you save the plot as M-file. Now at the MATLAB prompt if you type the file name you can obtain the MATLAB Figure plot. You can type grid to place a grid on the graph or edit the Figure as you wish.

If you are sufficiently happy with your results and the actual response is close to the simulated response, you can move on and begin the report for this project. A control design usually involve some form of fine-tuning, and will more than likely be an iterative process. If the actual response deviates from the desired response by large values you can fine tune by adjusting the Slider Gains to get a response to meet the design specifications more closely.

In WinCon Server, use File Save, this saves the compiled controller including all plots as a **.wpc** (WinCon project) file. In case you want to run the experiment again, from WinCon Server use File/Open to reload this **.wcp** file, and run the project in real time independent of MATLAB/Simulink.

To prevent excessive wear to the motor and gearbox run the experiment for a short

To see the actual vibration of the Flexible link without the Strain Gage feedback, set the Slider Gains corresponding to the feedback gains for α , and υ to 0. In this mode while there is feedback for the servomotor position control, there is no strain Gage feedback control. Click on the **Start** icon, you should now see clearly the oscillation at the tip of the beam. Obtain a plot of gamma_o, theta_in and gamma_os in one graph, and a plot of alpha_o and alpha_os in another graph.

5. Project Report

Discuss the assumptions and approximations made in modeling the servomotor and the Flexible Link. Your report must include the detailed servo plant and the Flexible Link block diagrams. Summarize the state feedback gains and comment on the simulation results due to a step input in part 3.3. Comment on your results; how does experimental response compare to simulated response? Discuss the reason for any deviation in the actual transient response and the simulated response. Estimate the actual steady-state error if any, and discuss the reasons for the steady-state error.