

# INTRODUCTION LITERATURE

## INTRODUCTION

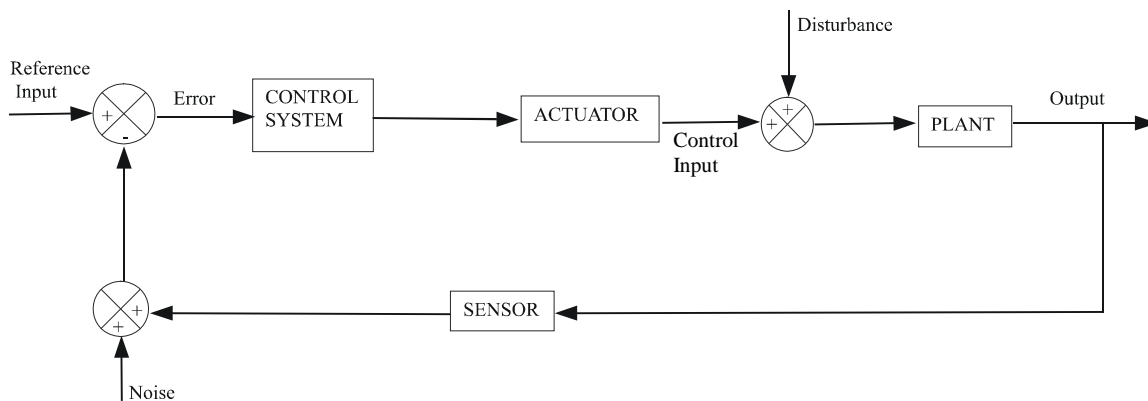
The increasing availability of low-cost sensors, actuators, and microprocessors is revolutionizing the design of mechanical engineering systems including manufacturing systems, consumer products, and air and ground vehicles. Thus, it is essential for the modern mechanical engineer to understand the design and implementation of control systems.

The objective of the ME 86 Controls Laboratory is to enable students to experimentally investigate the calibration, response characteristics, modeling, and control of thermal, fluid, and mechanical systems. This lab manual provides background information and the laboratory exercises. For detailed background information, however, students are referred to their ME 440 textbook.

In this chapter, we introduce the elements common to all control systems and the seven, state-of-the-art experimental systems used in the laboratory.

### Control System Elements

Figure 1 shows a block diagram with the elements (represented by blocks) common to every control system: sensor, actuator, plant, and controller. The signals (represented by arrows) include a reference input, error, control input, system output, disturbance input, and noise input. The reference input is the desired value of the system output. The difference between the output (as measured by a sensor) and the reference input is the error signal. The controller acts on the error signal to produce a control input through an actuator to the plant. The control objective is to reduce the error signal to zero despite the disturbances and noise in the sensor.



**Figure 1:** Control System Elements

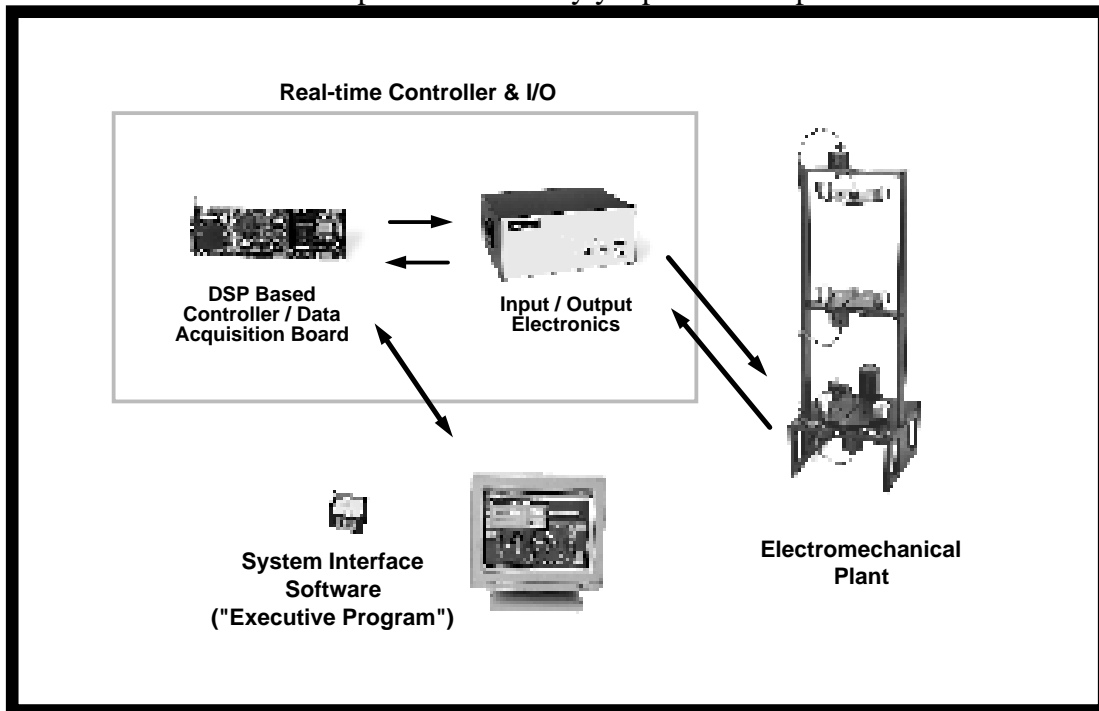
### ME 86 Laboratory Experiments

Seven experiments are used in the ME 86 lab: Torsion System, Rectilinear System, Moment Gyro, Industrial Servo, Inverted Pendulum, Magnetic Levitation, and Fluid

Process Control. All the experiments except the Fluid System were designed and built by Educational Control Products, Inc. (ECP). The Fluid System is from Feedback Instrumentation, Ltd.

All of the ECP plants are comprised of the three subsystems shown in Figure 2. The first of these is the electromechanical plant which consists of the mechanism, its actuator and sensors. The next subsystem is the real-time controller unit which contains the digital signal processor (DSP) based real-time controller, servo/actuator interfaces, servo amplifiers, and auxiliary power supplies. The DSP is capable of executing control laws at high sampling rates allowing the implementation to be modeled as continuous or discrete time. The controller also interprets trajectory commands and supports such functions as data acquisition, trajectory generation, and system health and safety checks. A logic gate array performs motor commutation and encoder pulse decoding. Two optional auxiliary digital-to-analog converters (DAC's) provide for real-time analog signal measurement. This controller is representative of modern industrial control implementation.

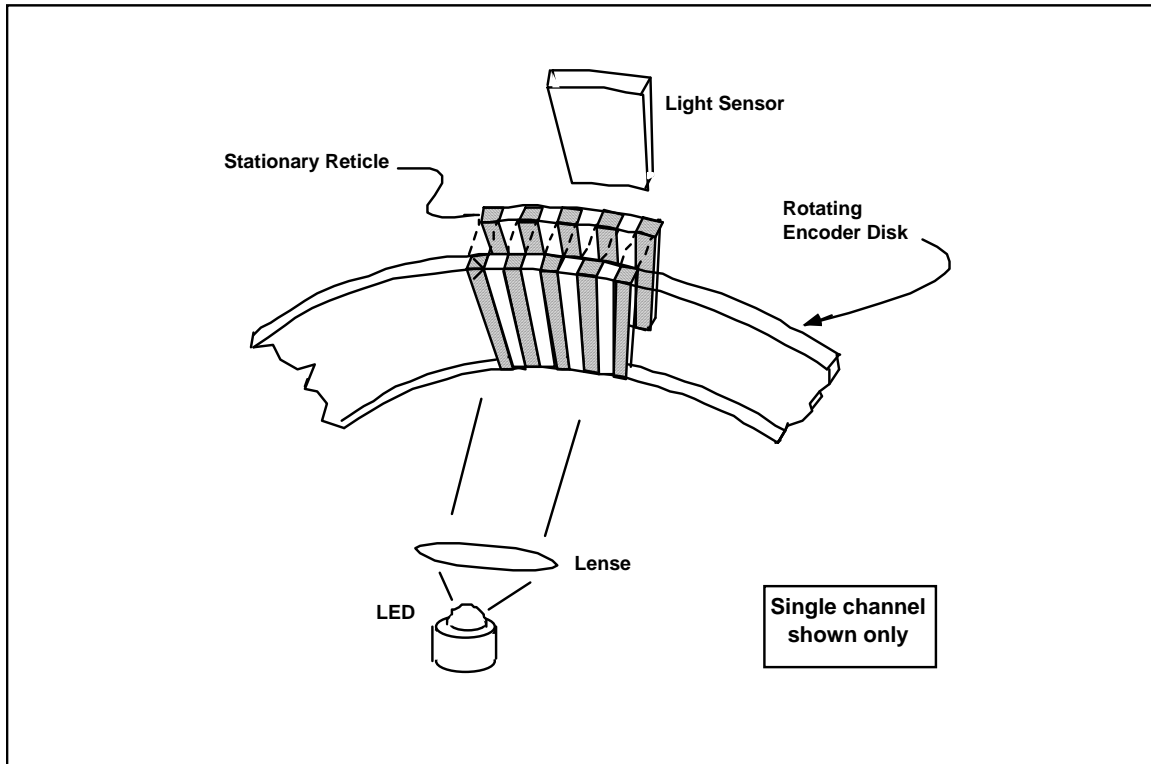
The third subsystem is the executive program which runs on a PC under the DOS or Windows™ operating system. This menu-driven program is the user's interface to the system and supports controller specification, trajectory definition, data acquisition, plotting, system execution commands, and more. Controllers may assume a broad range of selectable block diagram topologies and dynamic order. The interface supports an assortment of features which provide a friendly yet powerful experimental environment.



**Figure 2:** ECP Experiment Components

## CALIBRATION

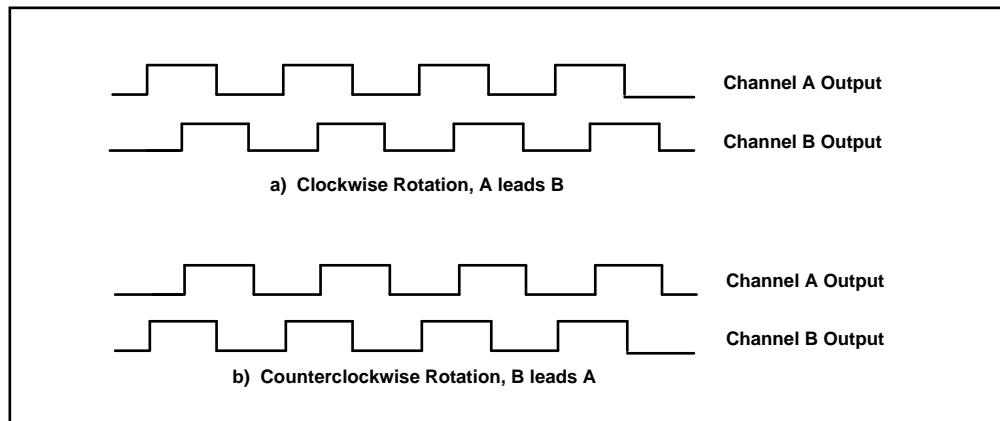
Prior to implementing a feedback control system, the sensors must be calibrated. The calibration process checks the linearity of the sensor and reduces spurious gain factors in the feedback loop that may cause the theoretically predicted and actual performance to differ. In this chapter, we discuss four of the sensors used in the laboratory experiments: Optical encoder, bead thermistor, laser displacement, and flow rate sensors.



**Figure 3:** The Operation Principle of Optical Incremental Encoders

### Optical Encoder Rotary Displacement Sensor

Incremental rotary shaft encoders are used in the Torsion, Control Moment Gyro, and Industrial Servo Trainer Experiments. These encoders measure the incremental displacement of a rotating shaft. The encoders are an optical type whose principle of operation is depicted in Figure 3. A low power light source is used to generate two 90 degrees out of phase square wave signals on the detectors as the moving plate rotates with respect to the stationary plate. The gate array uses the A and B channel phasing to decode direction and detects the rising and falling edge of each to generate 4x resolution – see Figure 4. The pulses are accumulated continuously within 24-bit counters (hardware registers). The contents of the counters are read by the controller once every servo cycle time and extended to 48-bit word length for high precision numerical processing. Thus the accumulation of encoder pulses provides an angular position measurement (signal) for the servo routines.



**Figure 4:** Optical Encoder Output

### **Bead Thermistor Temperature Sensor**

The Thermal Control Experiment includes a bead thermistor sensor. A thermistor changes resistance with temperature. The bead thermistor is exposed to the process air and electrically connected in bridge circuit that produces an output voltage at the “Y” terminal on the user interface.

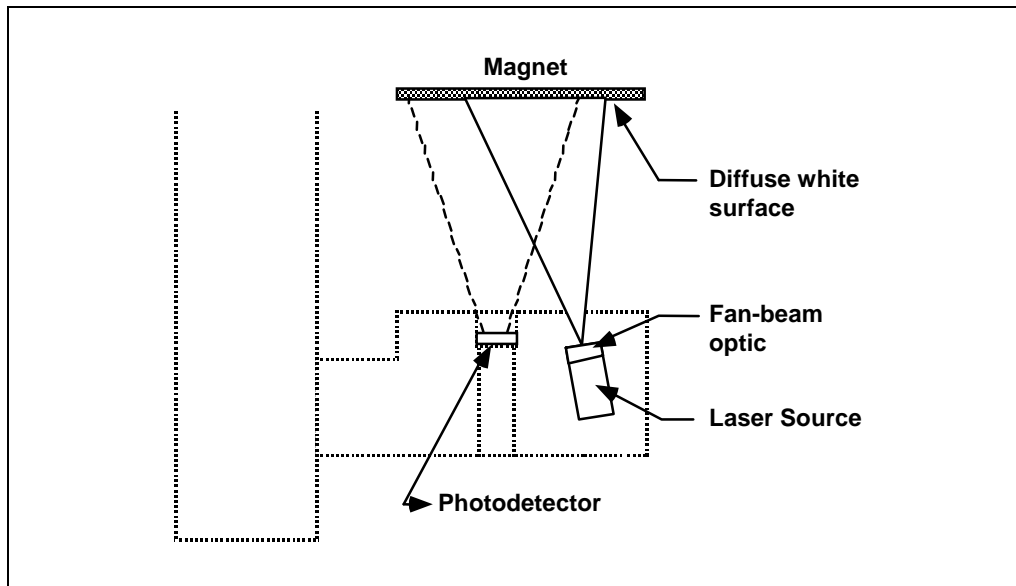
### **Laser Displacement Sensor**

The Magnetic Levitation apparatus uses laser light amplitude sensors for feedback. The laser sensor is depicted functionally in Figure 5 and is a proprietary ECP design. The monochromatic, coherent laser beam is spread via a single hemicylindrical optical element into a fan shape. This beam is projected on the diffuse white surface of the magnet. A photo detector views the beam and generates a voltage proportional to the amount of beam power incident on it. The white surface properties and laser/detector view factors and geometries are designed to maximize the sensor gain slope (change in output v. change in position) through the sensor operational range of 5.cm.

Because of the fan beam shape and the fact that the reflected light is a combination of specular and diffuse, the power on the detector diminishes with the inverse and inverse square of the magnet distance. Additionally, the geometry of the laser, detector, and magnet and the resulting cutoff of the detector’s view of the beam as a function of magnet height give rise to a linear dependency at small magnet distances.

Proprietary electronic circuitry makes the design immune to stray light noise, such as turning room lights on and off, and rejects most induced electronic disturbances. Thus a relatively low noise signal is output to the analog-to-digital converter (ADC) located in the Control Box. The resulting digital word, scaled at  $2^{15}$  (32,768) counts = 10 V is made available to the DSP board for subsequent real-time processing. A laser temperature signal is also output to the ADC and subsequently to the DSP board. This is used to compensate for the laser’s inherent reduction in emitted power as a function of temperature (approx. 15% maximum for the operational temperatures of the Magnetic

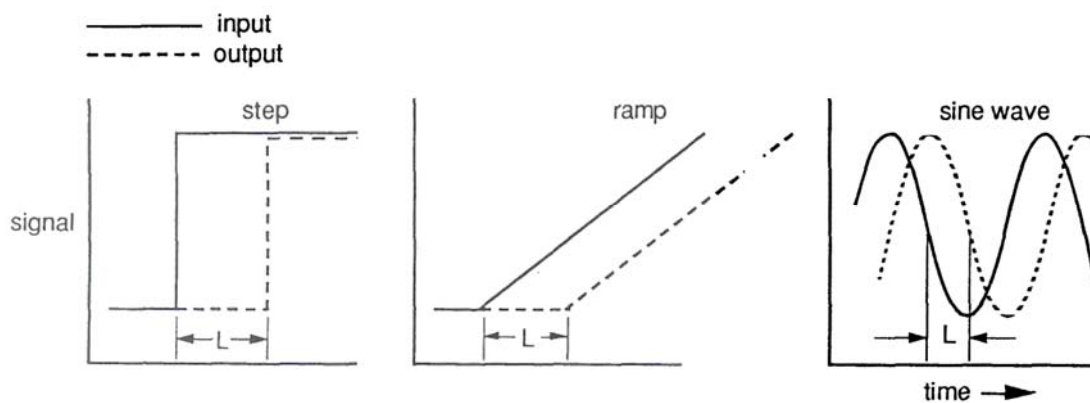
Levitation apparatus). This compensation is accomplished in background processing and is transparent to the user.



**Figure 5:** Laser Sensor Functional Diagram

## STEP RESPONSE

The open loop step response of a dynamic system tells a lot about its characteristics. In this chapter, we study the step responses of the ME 86 laboratory experiments. The Torsion, CMG, and Industrial Servo Trainer experiments have very similar dynamics. The input is essentially a torque applied to a freely spinning inertia. The Magnetic Levitation experiment has inherent nonlinearities and inherent stiffness in the magnetic force. The Thermal Control experiment has typical first order response for thermal systems with a delay associated with transport of the heated air down the duct. The Fluid Process Control experiment has first order response associated with filling a tank. The digital pendulum experiment has coupled pendulation and translation.

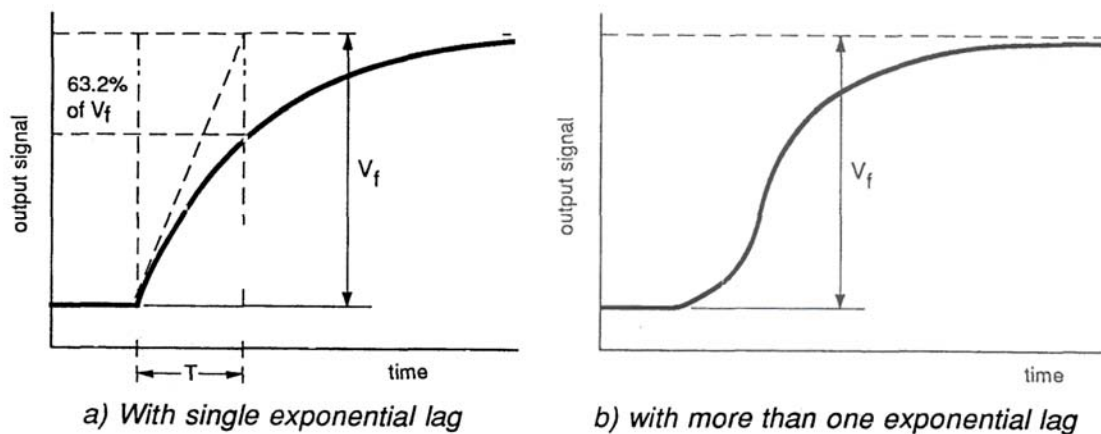


**Figure 6:** The effect of a pure delay on step, ramp, and sinusoidal signals

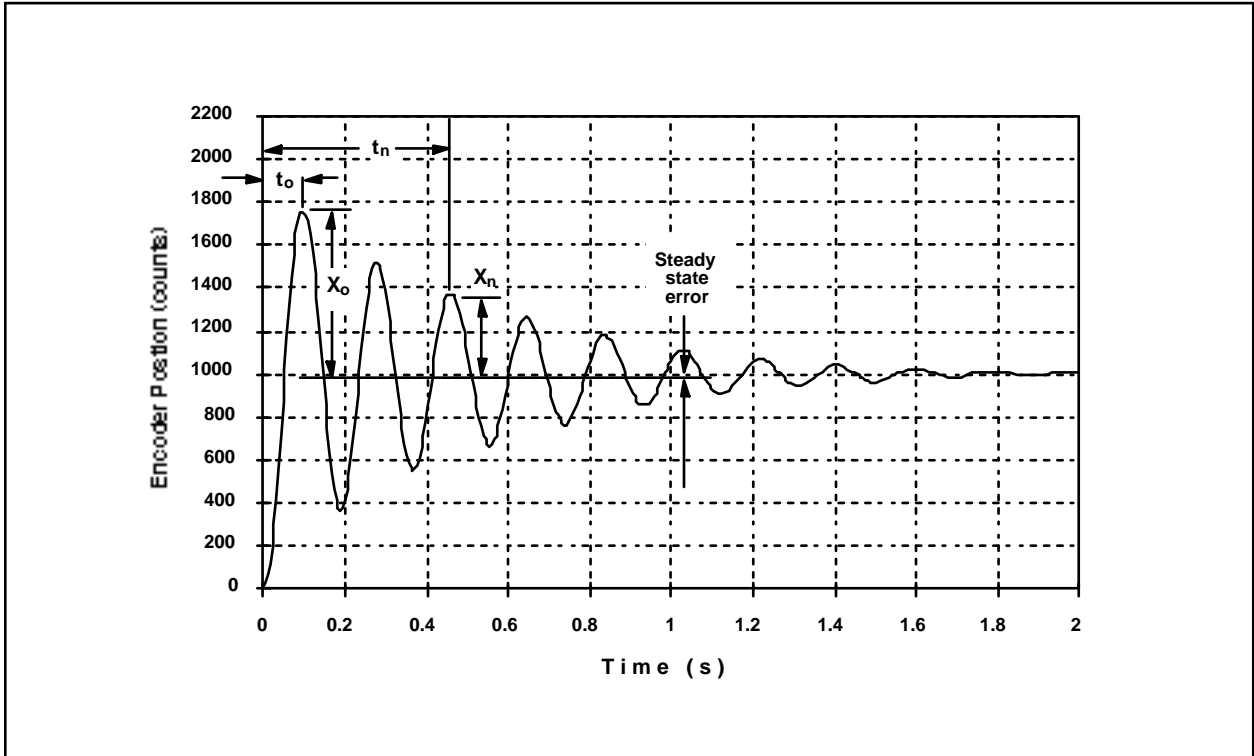
The step response can be used to measure several key parameters for a physical system. First, any pure delays in the system time shift the output relative to the input (see Fig. 6). For first order systems, the time constant can be calculated as shown in Fig. 7. For second order systems, the typical step response is shown in Fig. 8. We can measure the damped natural frequency  $\omega_d$  and damping ratio  $\zeta$  as follows. First, measure the amplitude and time to peak of the first one or two consecutive cycles as shown in Figure 8 (The smaller amplitude of the later cycles becomes dominated by nonlinear friction effects and does not reflect the salient system dynamics. From a stability point of view it is conservative to use the lower damping calculated from the early cycles.) Measure the reduction from the initial cycle amplitude  $X_0$  to the last cycle amplitude  $X_n$  for the  $n$  cycles.

The following relationship is associated with the *logarithmic decrement* for under-damped second order systems:

$$\frac{\zeta}{\sqrt{1-\zeta^2}} = \frac{1}{2\pi n} \ln\left(\frac{X_0}{X_n}\right) \quad (1)$$



**Figure 7:** Calculating the time constant ( $T$ ) from the time response of a first order system (a). The shape of the step response changes for higher order systems (b).



**Figure 8:** Typical step response for a second order system

For small  $\zeta$  Eq. (1) becomes

$$\zeta = \frac{1}{2\pi n} \ln\left(\frac{X_o}{X_n}\right) \quad (2)$$

Solve for  $\zeta$  by first estimating it via Eq. (2) then solving Eq (1) by trial and error<sup>1</sup>.

Divide the number of cycles,  $n$ , by the time taken to complete them ( $t_n - t_o$ ).

Convert the resulting frequency in Hz to radians/sec. This *damped frequency*,  $\omega_d$ , is related to the *natural frequency*,  $\omega_n$ , according to:

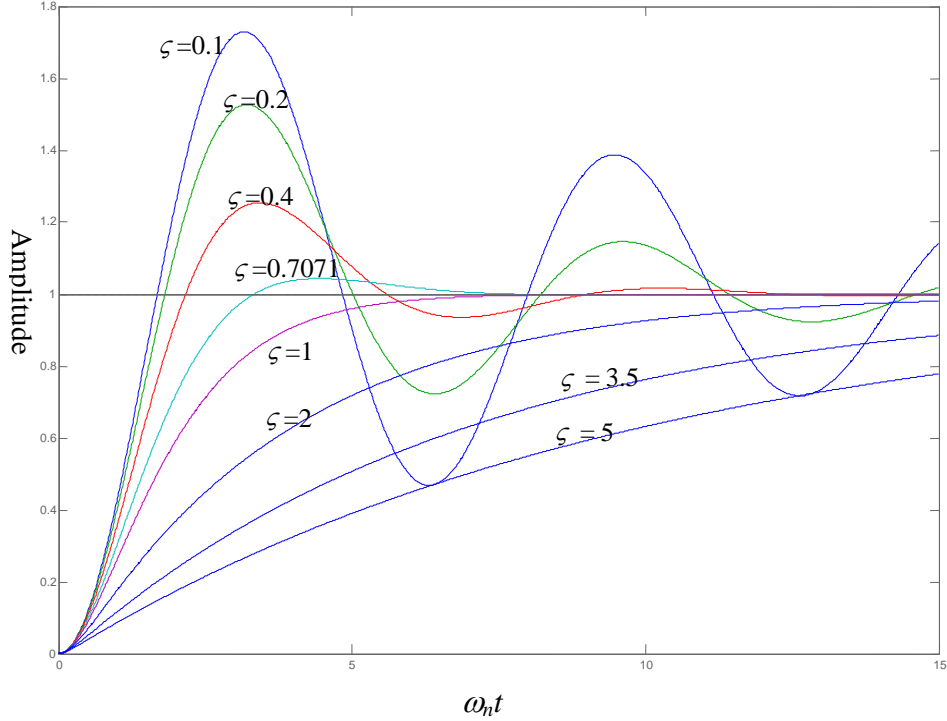
$$\omega_n = \frac{\omega_d}{\sqrt{1-\zeta^2}} \quad (3)$$

which becomes

$$\omega_n = \omega_d \quad (4)$$

for lightly damped systems. Figure 9 shows representative step response curves for second order systems with different damping ratios. The time axis is normalized with respect to the undamped natural frequency. One may compare the step response obtained experimentally with those in Fig. 9 to get approximate values of  $\omega_d$  and  $\zeta$ .

<sup>1</sup>Two significant digit precision is sufficient here and may often be obtained via Eq (2) for the lightly damped cases but is imprecise for other more highly damped system tests.

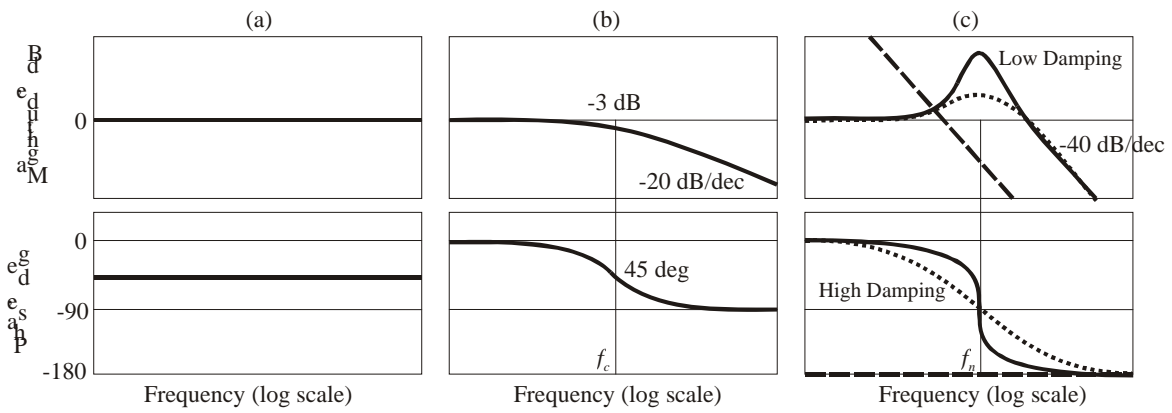


**Figure 9:** Step response curves for second order systems

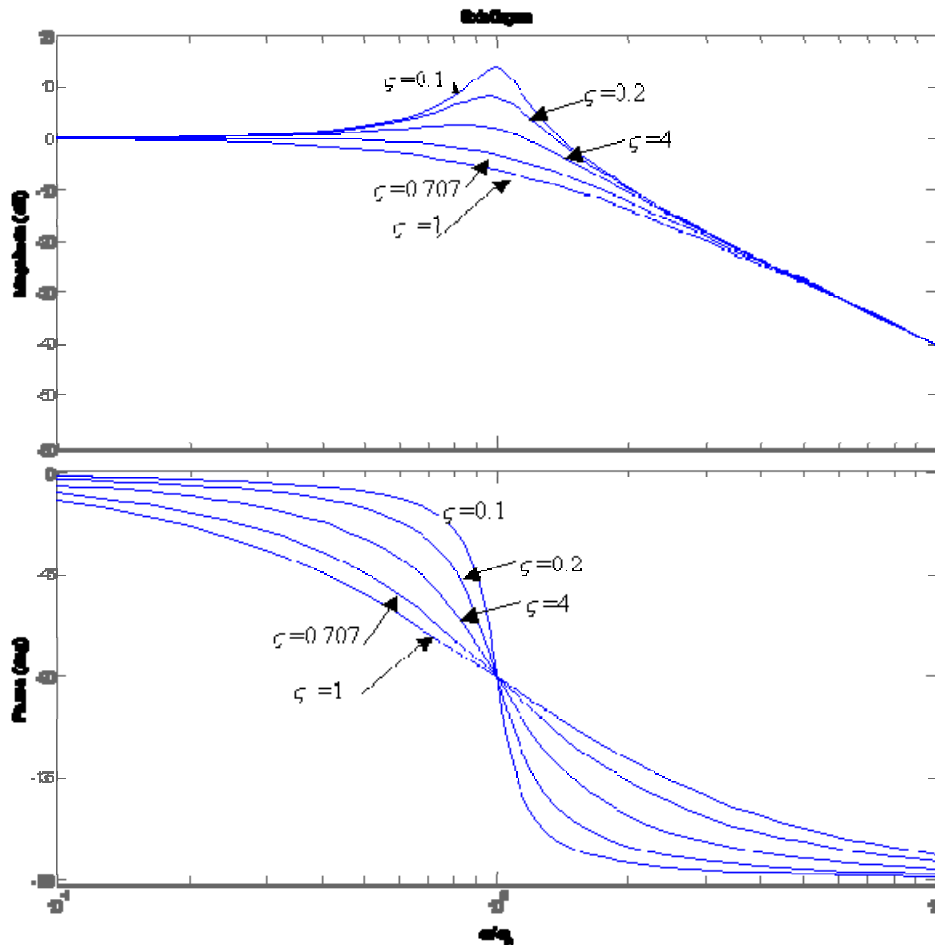
## FREQUENCY RESPONSE

The frequency response of a linear system is the output response to sinusoidal inputs. The output response is at the same frequency as the input with differing magnitude and phase. In this chapter, we study the frequency response of the ME 86 laboratory experiments. These experiments exhibit the frequency response characteristics of delay, first order (and higher order) lag, and under-damped second order response.

The frequency response characteristics of linear systems are best understood using a bode plot of magnitude ( $M_{dB} = 20 \log_{10} M$ ) and phase (deg) versus log frequency. The frequency response of a pure delay is shown in Fig. 10(a). The gain is constant at unity through the entire frequency range. The frequency response of a first order lag is shown in Fig. 10(b). The gain is fairly constant up to the corner frequency  $f_c$ . Above the corner frequency, the amplitude rolls off with a slope of  $-20$  dB/decade. The frequency responses of under-damped second order systems with different damping ratios are shown in Fig. 10(c). The resonant peak at  $f = f_n$  gets larger with smaller damping ratio. The gain is constant below the resonant frequency and rolls off with a slope of  $-40$  dB/decade above the resonant frequency. The frequency response of a second order system without stiffness and damping (pure inertia) is also shown in Fig. 10(c). The gain rolls off with a constant  $-40$  dB/dec and the phase is constant at  $180$  deg. Figure 11 shows the frequency response curves for second order systems with different damping ratios. The horizontal frequency axis is normalized by the undamped natural frequency.



**Figure 10:** Frequency response (a) Pure delay (delay time = Phase (rad)/frequency (rad/s)); (b) first order lag; (c) Second order systems with pure inertia (dashed), highly damping (dotted), and light damping (solid).



**Figure 11:** Frequency response curves for a second order system

## MODELING

Accurate modeling is an important part of control system design. Model-based simulation and analysis establish the stability and performance of a control system without expensive and time-consuming experimental trials. In this chapter, we introduce SIMULINK and develop SIMULINK models of the ME 86 experiments.

SIMULINK enables the input of the differential equations that model a given physical system in state space or transfer function form. Governing equations are typically in the form of first or second order differential equations. The Laplace transform is used to generate the system transfer function. For second order equations, additional states are defined (e.g. velocities) to put into state space form.

Parameter identification is the determination of the correct parameters to put into your mathematical model. Based on your calibrated sensors, the experimental step, and frequency response, the best set of parameters can be determined. The DC (or steady state) gain between a constant voltage input and constant output is used to determine a scale factor  $K$  that relates the steady state output to the steady state input. The step response can be used to identify time constants, damping ratios, and natural frequencies. If we compare our governing differential equation with the following standard forms:

First order system:  $\tau \dot{y} + y = Ku$

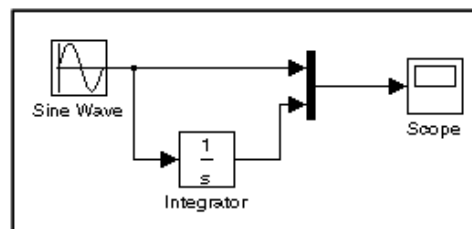
Second order system:  $\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2y = Ku$

we can relate the measured time constants  $\tau$ , damping ratios  $\zeta$ , and undamped natural frequencies  $\omega_n$  to the system parameters. Thus, many system parameters can be determined from the experimental measurements.

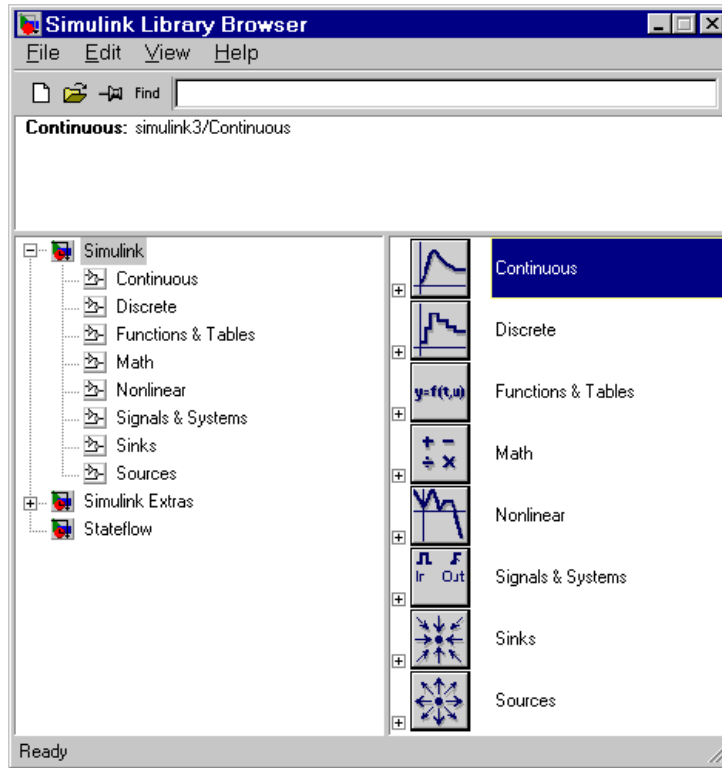
### Introduction to Simulink

This exercise shows you how to build a model using many of the model building commands and actions you will use to build your own models. The instructions for building this model in this section are brief. Please refer to Matlab on-line help for more information.

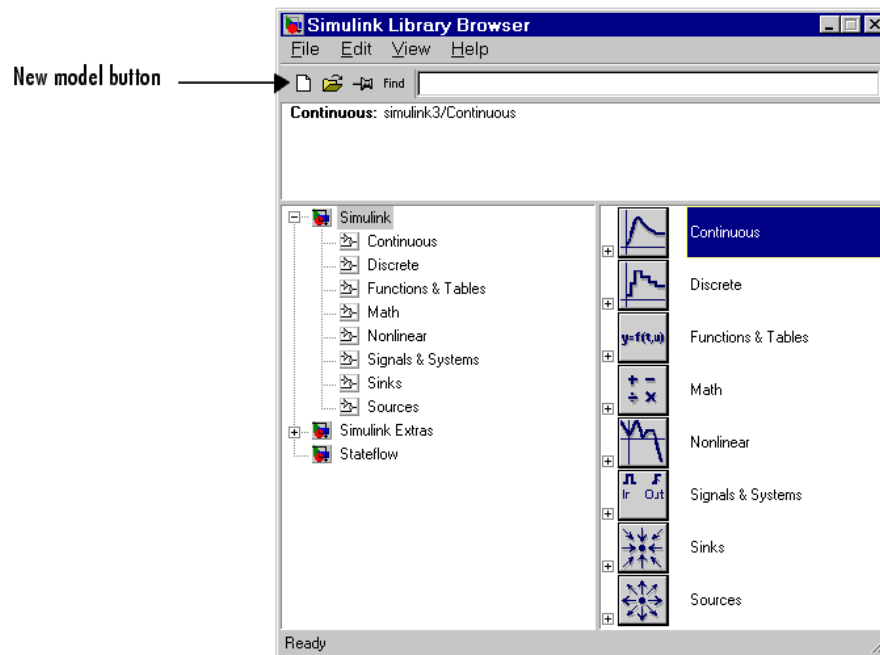
The model integrates a sine wave and displays the result, along with the sine wave. The block diagram of the model looks like this.



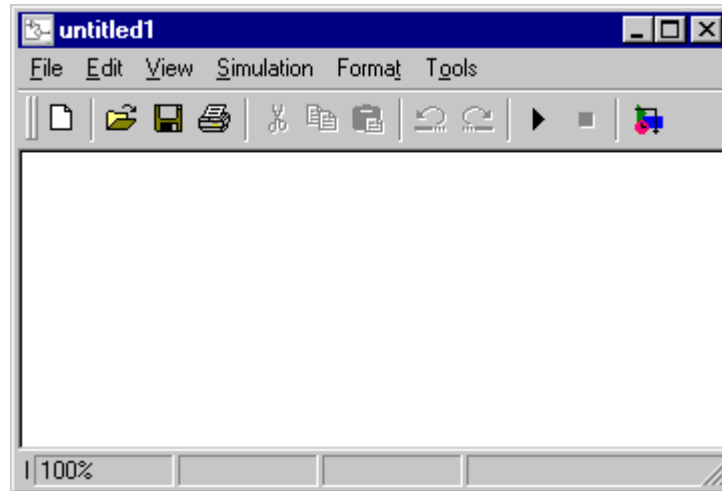
To create the model, first type `simulink` in the MATLAB command window and the Simulink Library Browser appears.



To create a new model on Windows, select the **New Model** button on the Library Browser's toolbar.



Simulink opens a new model window.



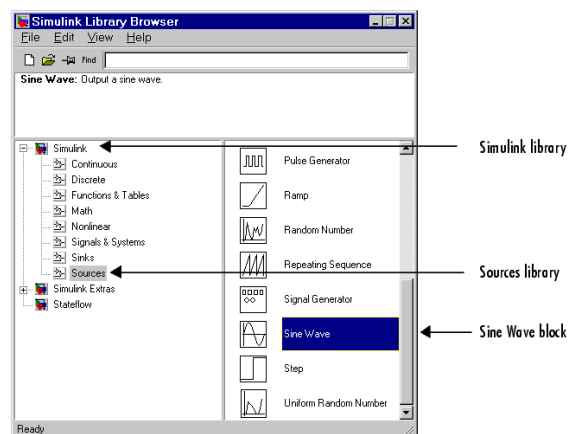
To create this model, you will need to copy blocks into the model from the following Simulink block libraries:

- Sources library (the Sine Wave block)
- Sinks library (the Scope block)
- Continuous library (the Integrator block)
- Signals & Systems library (the Mux block)

You can copy a Sine Wave block from the Sources library, using the Library Browser or the Sources library window.

To copy the Sine Wave block from the Library Browser, first expand the Library Browser tree to display the blocks in the Sources library. Do this by clicking on the Sources node to display the Sources library blocks. Finally click on the Sine Wave node to select the Sine Wave block.

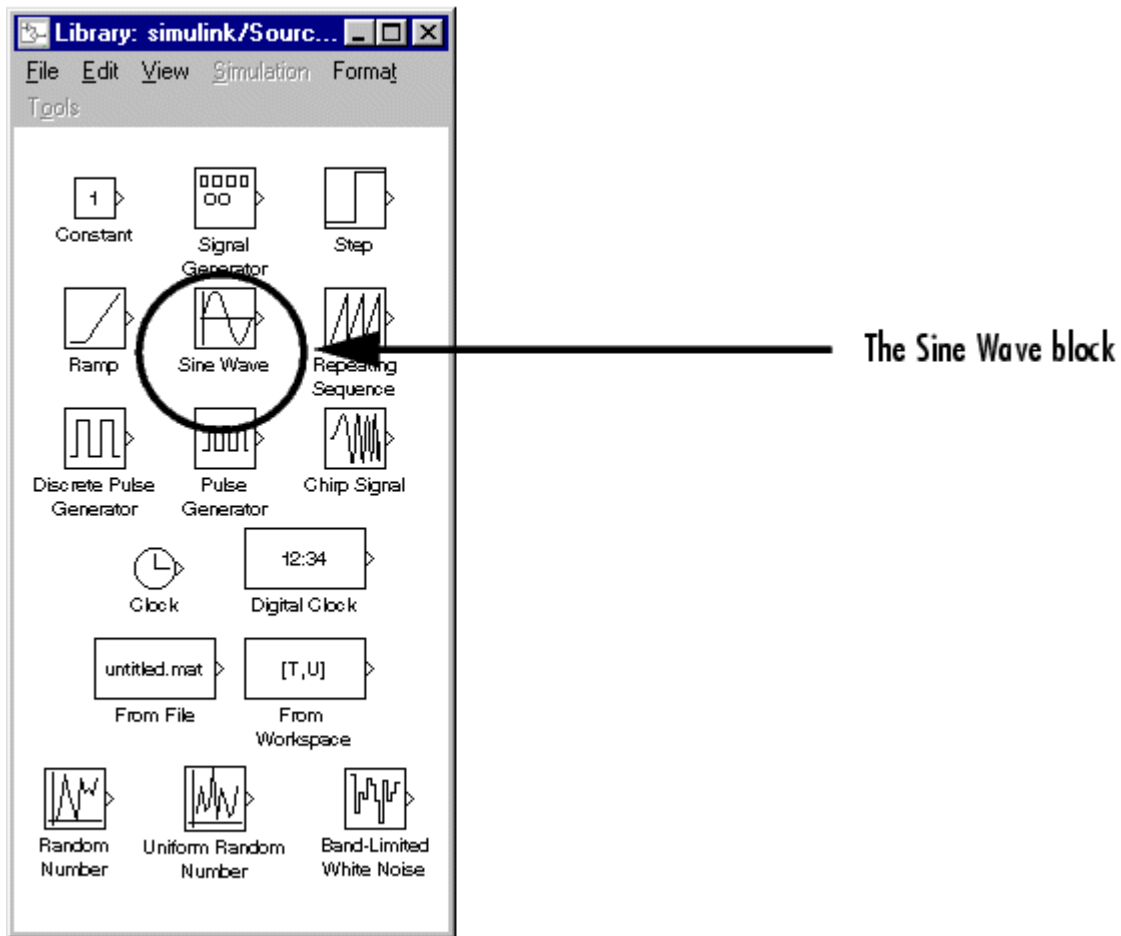
Here is how the Library Browser should look after you have done this.



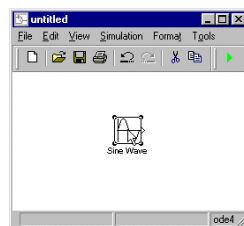
Now drag the Sine Wave block from the browser and drop it in the model window. Simulink creates a copy of the Sine Wave block at the point where you dropped the node icon.

To copy the Sine Wave block from the Sources library window, open the Sources window by double-clicking on the Sources icon in the Simulink library window. (You can open the Simulink library window by right-clicking the Simulink node in the Library Browser and then clicking the resulting **Open Library** button.)

Simulink displays the Sources library window.

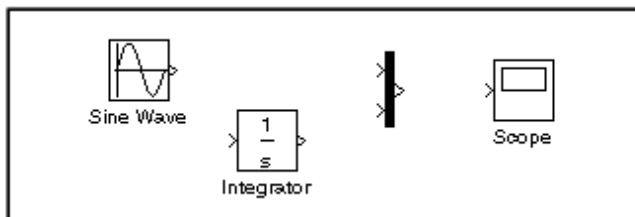


Now drag the Sine Wave block from the Sources window to your model window.

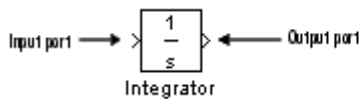


Copy the rest of the blocks in a similar manner from their respective libraries into the model window. You can move a block from one place in the model window to another by dragging the block. You can move a block a short distance by selecting the block, then pressing the arrow keys.

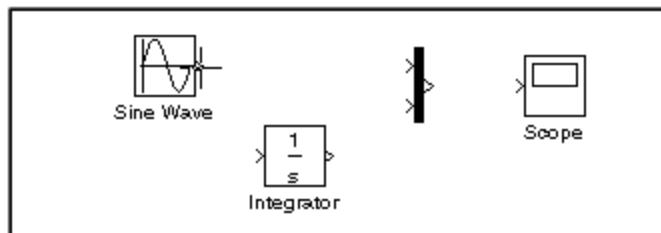
With all the blocks copied into the model window, the model should look something like this.



If you examine the block icons, you see an angle bracket on the right of the Sine Wave block and two on the left of the Mux block. The > symbol pointing out of a block is an *output port*; if the symbol points to a block, it is an *input port*. A signal travels out of an output port and into an input port of another block through a connecting line. When the blocks are connected, the port symbols disappear.

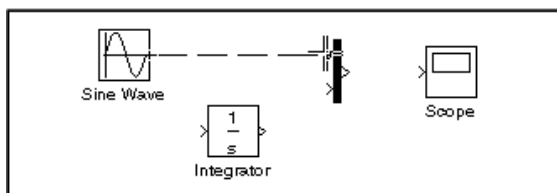


Now it's time to connect the blocks. Connect the Sine Wave block to the top input port of the Mux block. Position the pointer over the output port on the right side of the Sine Wave block. Notice that the cursor shape changes to cross hairs.

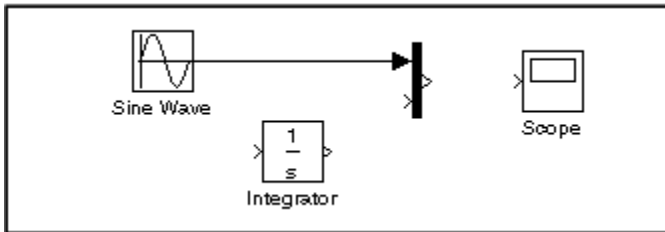


Hold down the mouse button and move the cursor to the top input port of the Mux block.

Notice that the line is dashed while the mouse button is down and that the cursor shape changes to double-lined cross hairs as it approaches the Mux block.



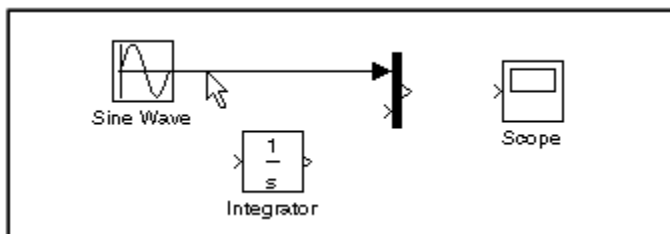
Now release the mouse button. The blocks are connected. You can also connect the line to the block by releasing the mouse button while the pointer is inside the icon. If you do, the line is connected to the input port closest to the cursor's position.



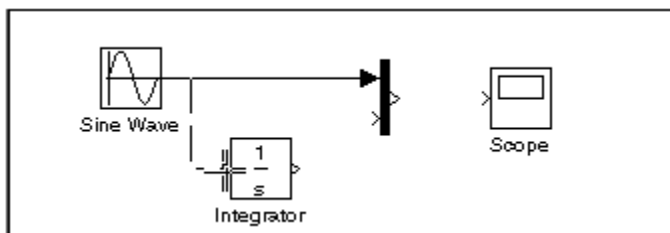
If you look again at the model at the beginning of this appendix, you'll notice that most of the lines connect output ports of blocks to input ports of other blocks. However, one line connects a *line* to the input port of another block. This line, called a *branch line*, connects the Sine Wave output to the Integrator block, and carries the same signal that passes from the Sine Wave block to the Mux block.

Drawing a branch line is slightly different from drawing the line you just drew. To weld a connection to an existing line, follow these steps:

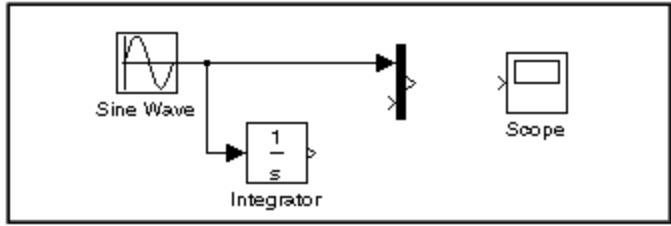
1. First, position the pointer *on the line* between the Sine Wave and the Mux block.



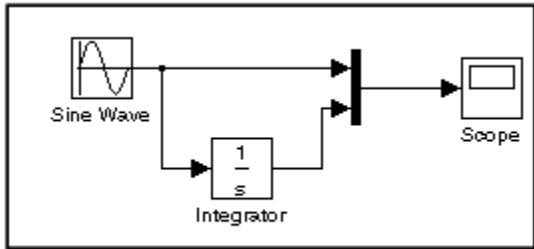
2. Press and hold down the **Ctrl** key (or click the right mouse button). Press the mouse button, then drag the pointer to the Integrator block's input port or over the Integrator block itself.



3. Release the mouse button. Simulink draws a line between the starting point and the Integrator block's input port.

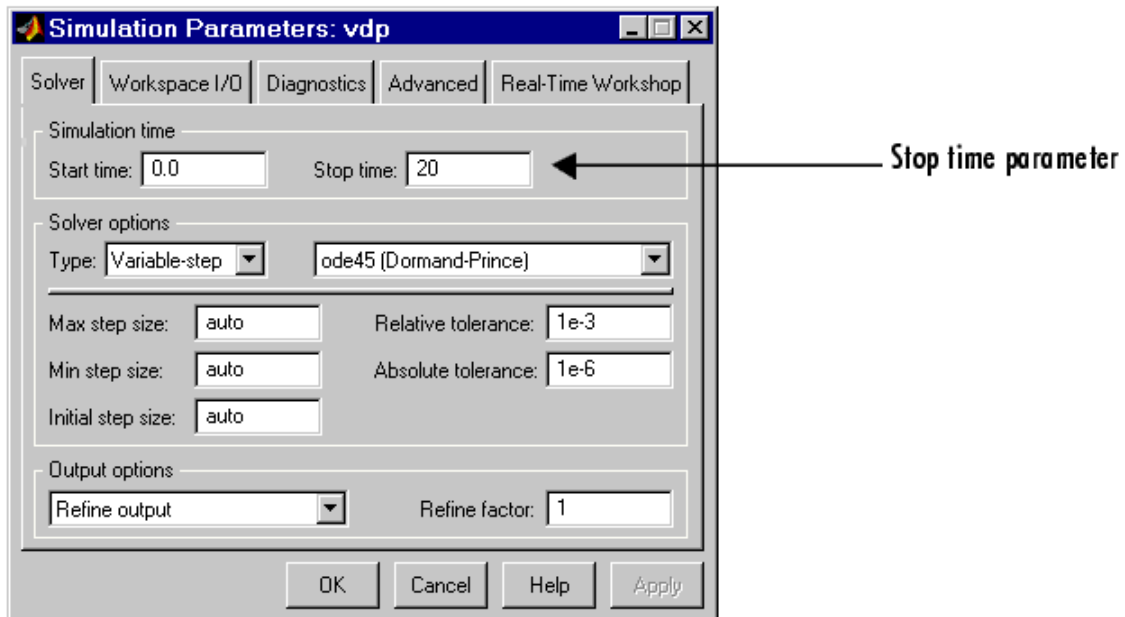


Finish making block connections. When you're done, your model should look something like this.



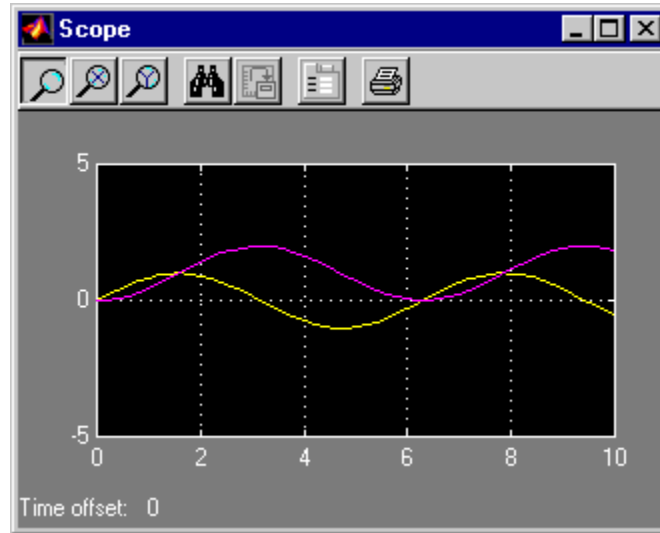
Now, open the Scope block to view the simulation output. Keeping the Scope window open, set up Simulink to run the simulation for 10 seconds. First, set the simulation parameters by choosing **Simulation Parameters** from the **Simulation** menu.

On the dialog box that appears, notice that the **Stop time** is set to 10.0 (its default value).



Close the **Simulation Parameters** dialog box by clicking on the **OK** button. Simulink applies the parameters and closes the dialog box.

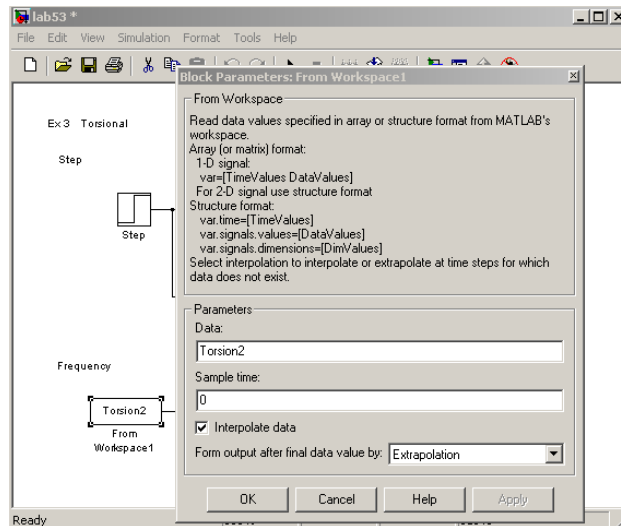
Choose **Start** from the **Simulation** menu and watch the traces of the Scope block's input.



The simulation stops when it reaches the stop time specified in the **Simulation Parameters** dialog box or when you choose **Stop** from the **Simulation** menu.

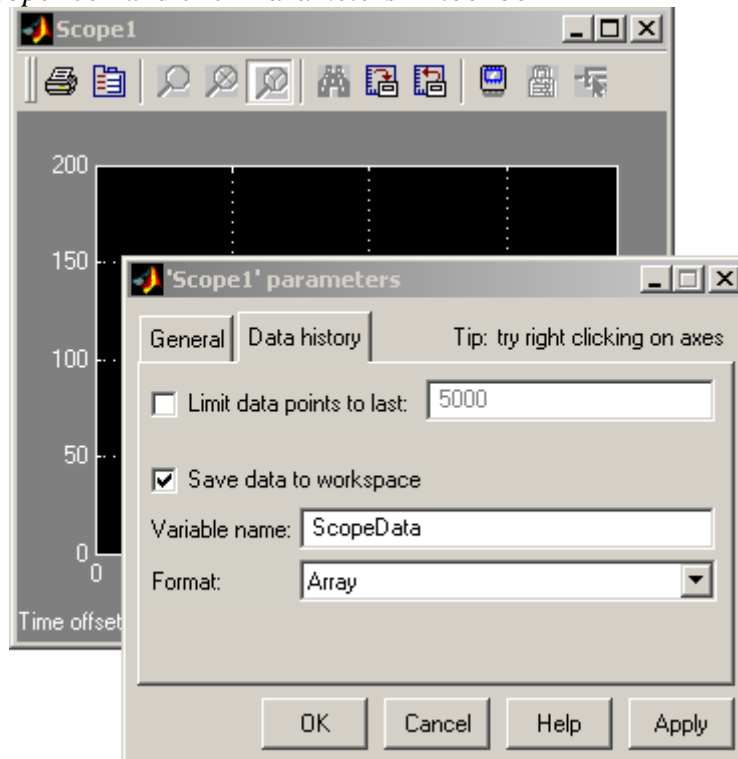
To save this model, choose **Save** from the **File** menu and enter a filename and location. That file contains the description of the model.

It may be easier to use the experimental input rather than the step input using the “Input from Workspace” block under “Sources”.



Create a matrix which includes experimental input: first column is for time, and second column is for input data. The matrix's name should be same as the name in *Parameters/Data*

Double click *Scope* icon and click *Parameters* in tool box



Click *Data history*, select *Save data to workspace*, where *variable name* is the output variable in workspace, which can be used to plot the simulation result with experimental result in one figure. Change Format to *array*.

In the simulation window, click *simulation/simulation parameters...*, change the *simulation time* to the suitable one. It is different for each experiment.

To terminate Simulink and MATLAB, choose **Exit MATLAB**. You can also type `quit` in the MATLAB command window. If you want to leave Simulink but not terminate MATLAB, just close all Simulink windows.

## PID CONTROL

In this chapter, we introduce the concept of feedback control. Using the previously developed Simulink models we design Proportional-Integral-Derivative (PID) controllers that improve the response over that of the open loop. These controllers are then implemented on the experimental test stands and tuned to achieve the best performance.

### PID Control

PID control is the most widely used control technique in practice with many applications in HVAC, vehicles, and manufacturing. PID controllers are error driven with gains that weight the current (proportional), future (derivative), and past (integral) contribution to the overall control efforts. The proportional term improves the response speed. The derivative term increases the system stability or damping. The integral term improves the

steady state performance, removing offsets in the step response. Mathematically, the PID controller is represented by the transfer function

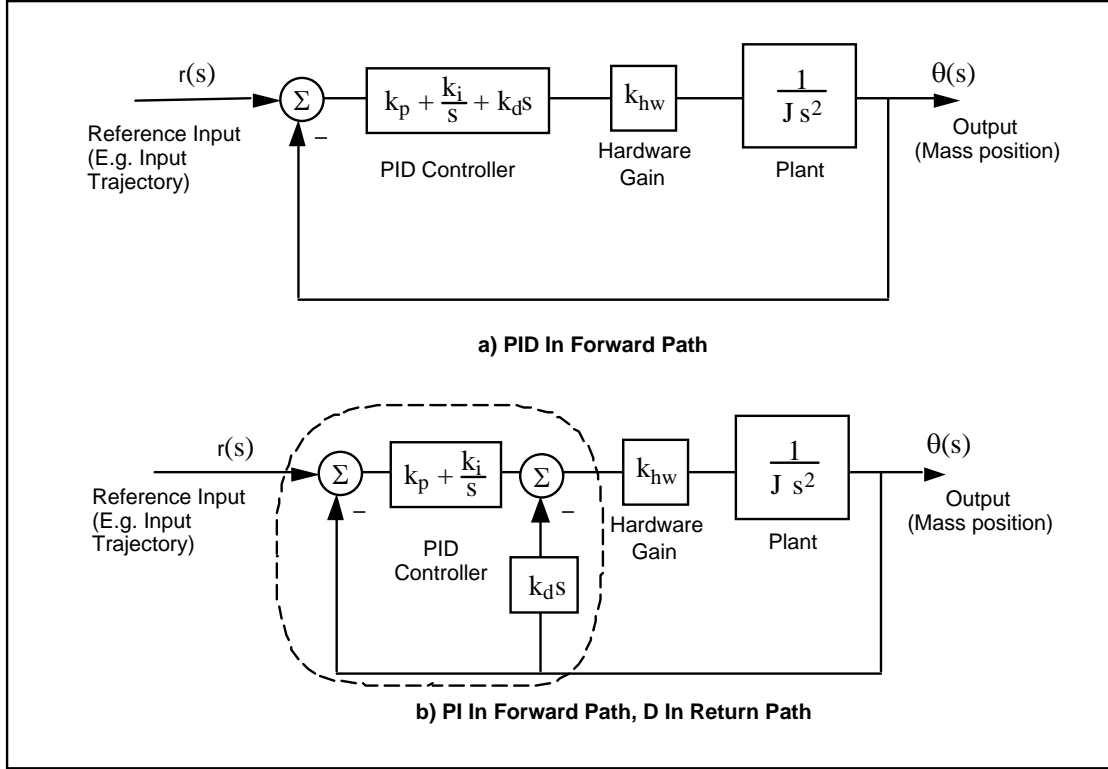
$$C(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k_d s^2 + k_p s + k_i}{s} \quad (5)$$

where the  $k_p$ ,  $k_i$ , and  $k_d$  are the proportional, integral, and derivative gains, respectively. The ECP software for the Torsion and Industrial Servo Trainer Experiments has options for PID control implementation in the two forms shown in Fig. 12. Thus, implementation simply involves inputting the three gain values. Note that the block diagram includes a hardware gain  $k_{hw}$  for the amplifier that must be independently measured. The second form in 12 (b) often provides better performance because the reference input is not differentiated which can cause problems for step inputs (infinite derivative!). For the CMG and Magnetic Levitation experiments, there is no preprogrammed PID controller and the following control code must be executed every control cycle (sample time =  $T_s$ ):

Discrete Time PID Control Algorithm	$e = r - \theta$ $u_i = u_{i_{last}} + T_s k_i e$ $u_d = \frac{k_d}{T_s} (e - e_{last})$ $u = k_p e + u_i + u_d$ $u_{i_{last}} = u_i$ $e_{last} = e$	Execute code every sample time
--	--	---

(6)

where  $\theta$  is the measured angle,  $r$  is the reference input,  $u_i$  is the integral control term using Euler integration, and  $u_d$  is the derivative control term using backward differencing. At the end of the control loop, the current values of the integral and error are stored in  $u_{i_{last}}$  and  $e_{last}$  so that they can be used in the next cycle through the control loop.



**Figure 12:** PID Control Block Diagram for a Rigid Body Plant.

### PID Control Design

A number of methods exist for PID control tuning. First, trial and error can be used with either a simulation model or the actual hardware. Although there are only three numbers to choose, this technique can be time-consuming, may result in a non-optimal solution, and can be dangerous when using actual hardware. Second, the Zeigler-Nichols method can be used with systems that go unstable under proportional control at sufficiently high gain. Third, for second order systems, the pole placement method can be used to choose gains that place the closed loop poles at desired locations. Finally, the root locus method can be used to tune one gain at a time.

The poles of the closed loop transfer function govern the stability and performance of a feedback system. The closed loop transfer function is determined by solving for the input/output relationship of the feedback system. The closed loop transfer function for the block diagram in Fig. 12(a) is

$$\frac{\theta(s)}{r(s)} = \frac{\left( \frac{k_p s + k_i + k_d s^2}{s} \right) \left( \frac{k_{hw}}{J s^2} \right)}{1 + \left( \frac{k_p s + k_i + k_d s^2}{s} \right) \left( \frac{k_{hw}}{J s^2} \right)} = \frac{k_{hw} (k_p s + k_i + k_d s^2)}{J s^3 + k_{hw} (k_d s^2 + k_p s + k_i)} \quad (7)$$

where, for a single feedback loop, the closed loop transfer function equals the forward gain over one plus the loop gain. Note that the closed loop system has three poles. There are also three gains that can be chosen to adjust the characteristic polynomial coefficients

to achieve any desired closed loop poles. If we wish to place all of the closed loop poles at  $s = -1$ , for example, then the desired closed loop characteristic equation is

$$(s+1)(s+1)(s+1) = s^3 + 3s^2 + 3s + 1 \quad (8)$$

If we compare Eq. (8) with Eq. (7), we observe that

$$\frac{k_{hw}k_d}{J} = 3, \quad \frac{k_{hw}k_p}{J} = 3, \quad \frac{k_{hw}k_i}{J} = 1, \text{ or} \quad (9)$$

$$k_d = \frac{3J}{k_{hw}}, \quad k_p = \frac{3J}{k_{hw}}, \quad k_i = \frac{J}{k_{hw}}$$

so the control gains can be solved if the inertia and hardware gain are known.

The closed loop performance can be estimated by the following simple rules. First, all closed loop poles must be in the left half of the complex plane to ensure stability. Second, the 1% settling time  $t_s$  is the time required for the response to settle to within 1% of its final value. The settling time is related to the pole(s) with the largest real part as follows  $t_s = \frac{4.6}{\sigma}$ , where  $\sigma$  is the magnitude of the real part. Third, for second order systems with a closed loop characteristic equation given by

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \quad (10)$$

$\sigma = \zeta\omega_n$  and the % overshoot ( $M_p = (\text{Maximum Value} - \text{Initial Value})/(\text{Final Value} - \text{Initial Value}) * 100\%$ ) can be estimated by  $M_p = 0.6(1 - \zeta)$ . Finally, the steady state error can be calculated using the final value theorem

$$e(\infty) = \lim_{s \rightarrow 0} [sE(s)] = \lim_{s \rightarrow 0} [s(r(s) - \theta(s))] = \lim_{s \rightarrow 0} [s(1 - G_{cl}(s))r(s)] \quad (11)$$