Chapter 3

Modeling and Digital Simulation Case Studies

One of the objectives of this session is to get you acquainted with the basics of Simulink, a graphical modeling, simulation, and prototyping environment used extensively in industry. We will not be able to cover the vast capability of Simulink with a few examples; you are encouraged to explore various features and graphical programming techniques on your own.

The other objectives of this lab are to find the mathematical model for some basic physical systems, to obtain a digital simulation diagram for the resulting differential equations, and to obtain the system's step response and investigate the effect of damping on the system response.

3.1 Pre-Lab Reading

Read through this chapter and do the pre-lab exercises. More help can be found in Appendix C.

The dynamic performance of physical systems is obtained by utilizing the related physical laws governing them. Many dynamic systems contain energy storage elements such as masses and springs (in mechanical systems) or inductors and capacitors (in an electric circuit). The principle of conservation of energy prohibits instantaneous changes in the state variables. Therefore, the system will go through some transients before settling to their steady-state values.

All physical systems are nonlinear to some extent. In order to model the system with linear time-invariant differential equations (for transfer function or state space representations), the system must first be linearized. Alternatively, its range of operation may be confined to a linear range.

The next step in designing a practical control system is to simulate the model on a computer to obtain the system response to various signals and disturbances. Then we can introduce appropriate controllers to achieve the desired system response. This process of design and analysis is repeated until a satisfactory control system is obtained and then the design may be implemented on the hardware.

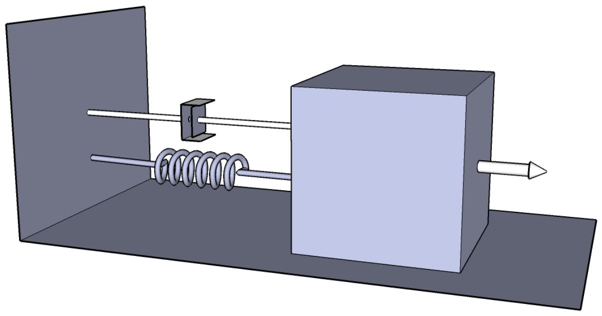
One of the most powerful tools for modeling and simulation of dynamic systems is Simulink, a toolbox extension of MATLAB. Simulink's strongest selling point is that it is very visual and intuitive. A system in block diagram representation is built easily and the simulation results are displayed quickly. Simulation algorithms and parameters can be changed in the middle of a simulation with natural results, thus providing the student with a readily accessible learning tool for simulating many of the operational problems found in the real world.

Simulink is particularly useful for studying the effects of nonlinearities on the behavior of the a system. As such, it is also an ideal research tool. Simulink has many advanced features for simulating a complex control system, such as the creation of new subsystem blocks and masking blocks through m-files, C programs, or other Simulink models for easy integration in your system's model. This allows an extension of the Simulink graphical functions to suit your own needs of analysis and design. The Simulink demos and User's Guide are very helpful in explaining the advanced usage and extension of the Simulink block library. For more information on Simulink, see Chapter 1 in [2] or browse [3].

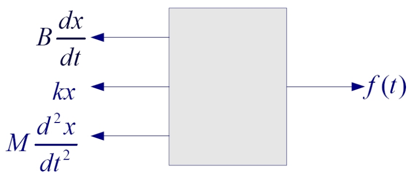
3.2 Laboratory Procedure

3.2.1 Case Study 1: Mechanical Translational System

Consider a simple mechanical system consisting of a mass, spring, and shock absorber—known as a *dashpot* or *piston*—shown in Figure 3.1a. In the figure, M is the mass, B is the damping coefficient, k is the spring constant, $f(t)$ is the external force, and $x(t)$ is the displacement of the mass. Three forces (and inertia) influence the motion of the mass, namely the applied force, the damping force, and the spring force as shown in the free-body diagram in Figure 3.1b.



(a) Mechanical translational system



(b) Free-body diagram

Figure 3.1

Applying Newton's Second Law, we have

$$M\frac{d^2x(t)}{dt^2} + B\frac{dx(t)}{dt} + kx = f(t) \tag{3.1}$$

The transfer function model is obtained by taking the Laplace transform (with zero initial conditions), which results in:

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{Ms^2 + Bs + k} \tag{3.2}$$

You may note that for complicated mechanical systems, it is easier to draw the electric circuit force-voltage analogy in place of the free-body diagram. In the force-voltage analogy, mass (M) is analogous to inductance, spring compliance $(1/K)$ is analogous to capacitance, damping coefficient (B) is analogous to resistance, and velocity $(\dot{x})$ is analogous to current. The key point in drawing the electric circuit analogy is to identify the displacement or velocity of each element and draw the circuit accordingly. The circuit can be drawn in the s-domain to find the transfer function or in the time domain which is suitable for obtaining the state space model. The electric circuit analogy for this mechanical system is shown in Figure 3.2.

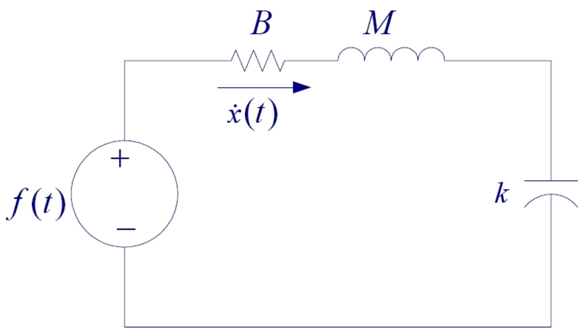


Figure 3.2: Electric circuit analogy for the mechanical system in Fig. 3.1a

With Kirchhoff's voltage law applied to the electric circuit, we get

$$M\frac{d\dot{x}(t)}{dt} + B\dot{x}(t) + k\int_0^t \dot{x}(\sigma) = f(t) \tag{3.3}$$

which is of course mathematically equivalent to (3.1). Equation (3.1) can also be written in state space form by selecting the two state variables as displacement and velocity: $x_1(t) = x(t)$ and $x_2(t) = \dot{x}(t)$. The first-order differential equations then become

$$\begin{aligned} \frac{dx_1(t)}{dt} &= x_2(t) \\ \frac{dx_2(t)}{dt} &= \frac{1}{M}f(t) - \frac{k}{M}x_1(t) - \frac{B}{M}x_2(t) \end{aligned} \tag{3.4}$$

For convenience, we will define multiple outputs: $y_1 = x_1(t)$ and $y_2 = x_2(t)$. Thus the state space expression of this system is

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\frac{k}{M} & -\frac{B}{M} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} f(t) \\ \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \end{aligned} \tag{3.5}$$

The simulation diagram for Equations (3.4) is shown in Figure 3.3

With the system initially at rest, a constant force of $f(t) = 32$ Newtons is applied at time $t = 0$. The system in question has a mass of 2 kg and a spring constant of 32 kg/s^2. We are told that the damping coefficient, B, may be adjusted to obtain a desirable response.

The system's characteristic equation—from Eq. (3.2)—is given by

$$s^2 + \frac{B}{M}s + \frac{k}{M} = 0 \tag{3.6}$$

which may be compared to the standard second-order transfer function

$$s^2 + 2\zeta\omega_n + \omega_n^2 = 0 \tag{3.7}$$

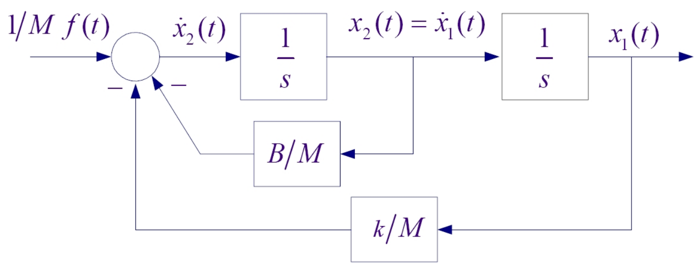


Figure 3.3: Simulation diagram for the mechanical system in Case Study 3.2.1

Pre-lab Exercises

1. The dashpot damping is adjusted to $B = 2$ Ns/m. Determine the natural frequency of oscillation (ω_n), damping ratio (ζ), percent overshoot ($PO = 100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$), peak time ($t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$), and settling time ($t_s \approx 4\tau$).

2. The dashpot damping is adjusted to $B = 56$ Ns/m. Determine the damping ratio, ζ, response time constants τ_1 and τ_2, and settling time $t_s \approx 4\max(\tau_1, \tau_2)$.

3. Determine the friction coefficient B for the response to be critically damped. What is the response time constant and approximate settling time?

Lab Exercises (Digital Simulation using Simulink)

To create a Simulink block diagram representation of the system shown in Figure 3.3 double click on the Simulink icon on the MATLAB toolbar.An untitled window for designing and simulating a new model will open. Double click on the Simulink library icon, this will open nine subsystems libraries. Open the Source Library and drag the Step Input block to the open new model window. Double click on the Step Input to open its dialog box, set the parameters Step time to 0, and Final value to 32. Get two integrators from the Continuous library, three Gain blocks on one Sum block from the Math Library, one Scope and one XY Graph from the Sink library, one Mux block from Signals and Systems library. Open the Sum block dialog box and enter the required summing point signs + - -. Once you have dragged all the required blocks and placed them on the new model window, join the in-ports and out-ports to create the simulation model. The purpose of Mux block (Multiplex) is to combine the velocity and displacement signals into a composite signal so as to display both signals on one Scope. XYGraph is used to display the state trajectory, i.e., velocity versus displacement plot. Connect x_1 to the first input and x_2 to the second input of the XY Graph. Open the XY Graph dialog box, set the x-axis limits to 0, 2, the y-axis limits to ± 4 and Sample time to 0.01. An mfile named `MSFanimation.m` has been developed for animating the motion of the mass-spring-friction system during simulation. To add this animation, get an S-Function block form the functions & Tables library, place it on your model window and connect its input terminal to the signal coming from the output of the Mux block. Make sure that you have obtained `MSFanimation.m` and `InAmin.m` files from your instructor and placed it on a folder in the current directory. Open the S-Function block for its name enter `MSFanimation`, and for the S-Function parameter enter 0.01. Set the gain blocks to the given values and the damping coefficient specified part 1 of the Pre-lab exercises.

Before starting simulation, you must set the simulation parameters. Pull down the Simulation dialog box and select Simulation Parameters. Set the start time to 0 and the stop time to a suitable value. For the Solver option, select Variable-step and any of the Continuous integration routines such as `ode45` or `ode23`. For more accurate step response you may change the Relative tolerance from 10^{-3} to 10^{-5}. If you select Fixed-step, again make sure you select a Continuous integration routine such as `ode4` (Runge-Kutta). You can also change the step size from auto to a small value such as 0.0001. Follow the same procedure in the remaining case studies in this lab and make sure the Solver option is *not* set to Discrete.

Simulate and obtain a print of the Scope. The scope yellow trace will not print well. Also, the Simulink XY Graph cannot be printed. A script m-file named `plotscope.m` has been developed which captures the scope plot and produces a Figure plot. At the MATLAB prompt, type `plotscope` and then click on the Scope Figure (outside the plot area) and hit return. You can add labels and a legend or edit the graph. You can use this procedure for the XY Graph or the animation plot.

An alternative way to obtain a Figure plot is to place two To Workspace blocks from the Sink library and connect their inputs to the x_1 and x_2 signals and defining `x1` and `x2` for the variables. The time array can be obtained by feeding a Clock block into another To Workspace block and defining a variable `t` for time. After performing the simulation you can use the `plot` function to obtain the desired Figure plot.

Repeat the simulation for the value of B in questions 2 and 3 in the Pre-lab. Document the plots obtained for the above three cases, determine and summarize the time-domain specifications for each case. Comment on the nature of each response and discuss the effect of damping coefficient on the resulting response.

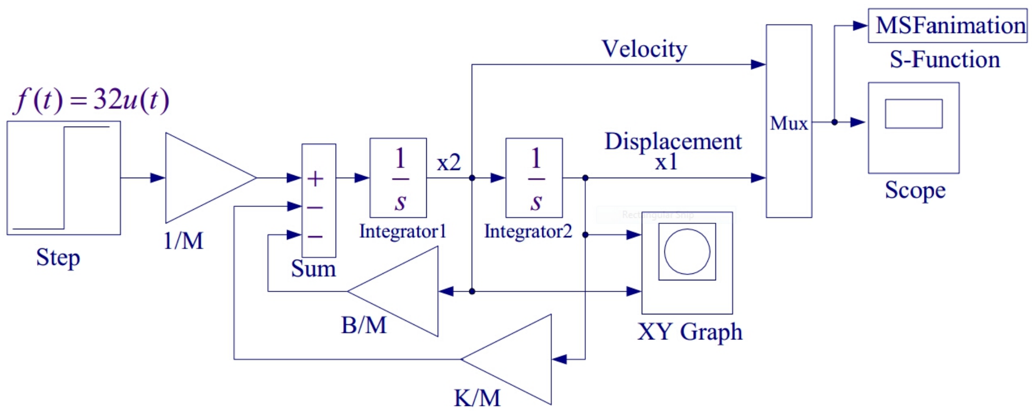


Figure 3.4: Simulink diagram for the mechanical system in Case Study 1.

3.2.2 Case Study 2: Simple Pendulum

Consider the simple pendulum illustrated in Figure 3.5 where a mass of m kg hangs from a hinge by a rod of length l meters. The rod is light enough that its mass can be neglected. The rod is displaced by angle θ radians from the equilibrium position. Assume a viscous friction for the motion with a damping coefficient of B kgs/m. The tangential velocity of the mass is $l\dot{\theta}$. The tangential forces acting to restore the pendulum to equilibrium are

$$F_T = -mg\sin(\theta) - Bl\dot{\theta} \tag{3.8}$$

where g is the gravitational acceleration. Also from Newton's law, we have

$$F_T = ml\ddot{\theta}. \tag{3.9}$$

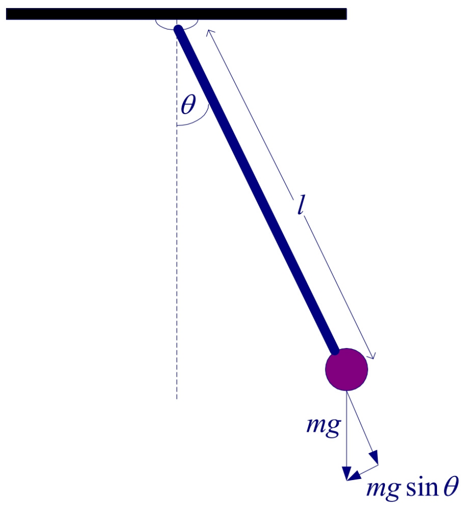


Figure 3.5: Simple pendulum.

Combining the above equations, we get

$$\ddot{\theta} + \frac{B}{m}\dot{\theta} + \frac{g}{l}\sin(\theta) = 0. \tag{3.10}$$

Equation (3.10) is nonlinear because of the $\sin(\theta)$ term. We can now write the above equation in state variable form. Let $x_1 = \theta$ and $x_2 = \dot{\theta}$ (angular velocity). Then,

$$\dot{x}_1 = x_2 \tag{3.11a}$$

$$\dot{x}_2 = -\frac{B}{m}x_2 - \frac{g}{l}\sin(x_1). \tag{3.11b}$$

The simulation diagram for equations (3.11) is shown in Figure 3.6. The mass is displaced from the equilibrium by 0.5 radians (28.65°) at time $t = 0$. The mass is 0.5 kg, the rod length is 0.613 m, and the gravitational acceleration is 9.81 m/s^2. Two cases of frictional coefficient will be considered:

1. $B = 0.05$ kgs/m
2. $B = 4.0$ kgs/m

Lab Exercises (Digital Simulation using Simulink)

Create a Simulink block diagram presentation of the system shown in Figure 3.5. Drag all the required blocks and place them on a new model window. For the nonlinear term, get the `Fcn` block from the Function & Tables library. Use `u` for the input variable name, e.g. $16 * \sin(u)$. Specify the gain B/m and connect all blocks to create the simulation model shown in Figure 3.7.

Open the last integrator dialog box and set the initial condition to 0.5 for the angular displacement. The initial velocity is zero. Therefore, set the initial condition parameter for the first integrator to 0. Place one scope to display the angle θ (signal x_1) and another scope to display the velocity signal x_2. Use an XY Graph to display the state trajectory, i.e., velocity versus displacement plot. Connect x_1 to the first input and x_2 to the second input. Open the XY Graph dialog box, and set the x-axis limits to $-5, 0.5$, the y-axis limits to ± 2, and the sample time to 0.01.

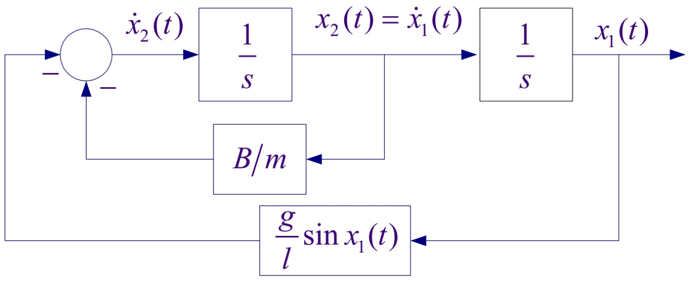


Figure 3.6: Simple pendulum.

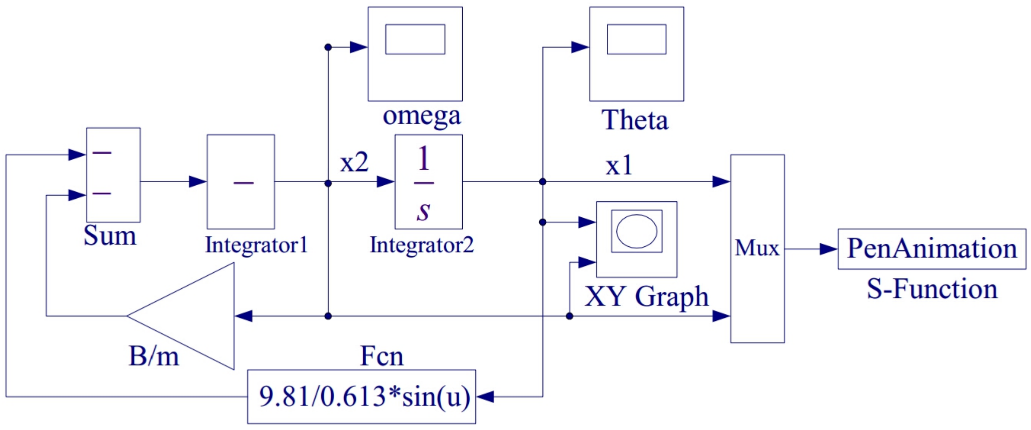


Figure 3.7: Simulink diagram for the pendulum in Case Study 2.

An m-file named `PenAnimation.m` has been developed for animating the pendulum swing during simulation. To add this animation, get a Mux block with two inputs. Connect x_1 to the top in-port and x_2 to the lower in-port. Next get an S-Function block from the Functions & Tables library, place it on your model window and connect its in-port terminal to the signal coming from the Mux block. Make sure that you have obtained this m-file from your instructor and placed it on a folder in the path of MATLAB. Open the S-Function block, for its name enter `PenAnimation` and for the S-Function parameter, enter 0.01.

1. Set the friction coefficient to 0.05 kgs/m.
   Simulate and print the zero-input response (natural response) for the angular displacement and state trajectory. You may want to reduce the simulation final time to a suitable value. Comment on the nature of the response. When a system returns to its equilibrium point after a disturbance, the system is said to be *stable*. Is the system stable about its equilibrium point $\theta = 0$? How would you describe the system stability if the damping coefficient B were neglected? Use `plotscope` to capture the Scope trace and XY Graph.

2. Repeat the above for $B = 4$ kgs/m.

Linearization

Many control systems are designed to return to their equilibrium position when subjected to a small disturbance. Nonlinear systems are often *linearized* assuming small deviation from the equilibrium point. The nonlinear differential equation describing the motion of the pendulum can be linearized if the initial angle of deflection is small.

Let $\theta = \theta_0 + \Delta\theta$ and substitute this θ into (3.10) and expand the sine term. For small $\Delta\theta$, we assume that $\sin(\Delta\theta) \approx \Delta\theta$, $\cos(\Delta\theta) \approx 1$, and $\cos(\theta_0) \approx 1$. Show that (3.10) results in the linearized differential equation

$$\frac{d^2\Delta\theta}{dt^2} + \frac{B}{m}\frac{d\Delta\theta}{dt} + \frac{g}{l}\frac{d\Delta\theta}{dt}. \tag{3.12}$$

This approximation is reasonably accurate for $-\pi/4 \leq \theta \leq \pi/4$.

3. The state variable in terms of the small changes $\Delta\theta$ and $\frac{d\Delta\theta}{dt}$ is the same as in (3.11), except $\sin x_1$ is replaced by x_1. Copy the Simulink nonlinear model and paste it on the same model window, replace the `Fnc` block with a gain bloc and set the parameter to the value given by g/l. Eliminate the S-function in the duplicate model. In order to validate the linearized model use a Mux block with two inputs and connect the in-ports to the x_1 signal of each model and a Scope to its out-port terminal. Simulate for $B = 0.05$ in both models and obtain the response. State if the linearized model response is in close agreement with the nonlinear model. The characteristic equation of the linearized model is

$$s^2 + \frac{B}{m}s + \frac{g}{l} = 0. \tag{3.13}$$

4. For the given values of l and m, find the value of B for critically damped response. Set the B to this value in both models and repeat the simulation. Comment on the response.

3.2.3 Case Study 3: Nonlinear Differential Equation with Saturation

One of the useful features of Simulink is the availability of nonlinear blocks such as `switch`, `relay`, `deadzone`, `backlash`, `rate-limiter`, `saturation`, `Coulomb friction`, and many other nonlinear functions. These are very useful for studying effects of nonlinearities on the behavior of the system. This study deals with the simulation of a nonlinear differential equation. The angular displacement of a dynamic system is given by

$$\ddot{\delta} + 4\dot{\delta} + a_0 \sin(\delta) = 30u(t) \tag{3.14}$$

where $u(t)$ is a unit step, δ is constrained to be between $\pm 2\pi$, and

$$a_0 = \begin{cases} 35.6, & 0 \leq t \leq t_c \\ 15, & t_c < t < \infty \end{cases} \tag{3.15}$$

where t_c is a threshold switching time. Large values of t_c may result in an unbounded response. Transforming to state variable form, let $x_1 = \delta$ and $x_2 = \dot{\delta}$. Then

$$\dot{x_1} = x_2 \tag{3.16a}$$
$$\dot{x_2} = -a_0 \sin x_1 - 4x_2 + 30u(t). \tag{3.16b}$$

The Simulink diagram for the above system is shown in Figure 3.8. Equation (3.15) is represented by the `switch` block and (3.14) is represented by the `saturation` block.

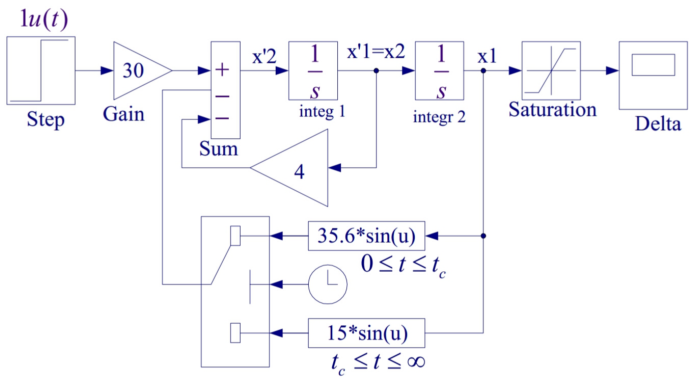


Figure 3.8: Simulink diagram for Case Study 3.

Lab Exercises (Digital Simulation using Simulink)

Construct the Simulink diagram as in Figure 3.8. In the `step` block, set the step time to 0 and the final time to 1.

Open the Simulation Parameters dialog box. Set the stop time to 5 seconds and select `ode45`. Obtain the response for $t_c = 0.4$ and $t_c = 3$. Comment on the behavior of the response for each case.

3.2.4 Case Study 4: Inverted Pendulum

This fourth study deals with the classic problem of balancing an inverted pendulum. This study demonstrates the control of an inherently unstable system of balancing systems that occurs in the areas of missile stabilization and robotics. This study also demonstrates the linearization of a nonlinear system.

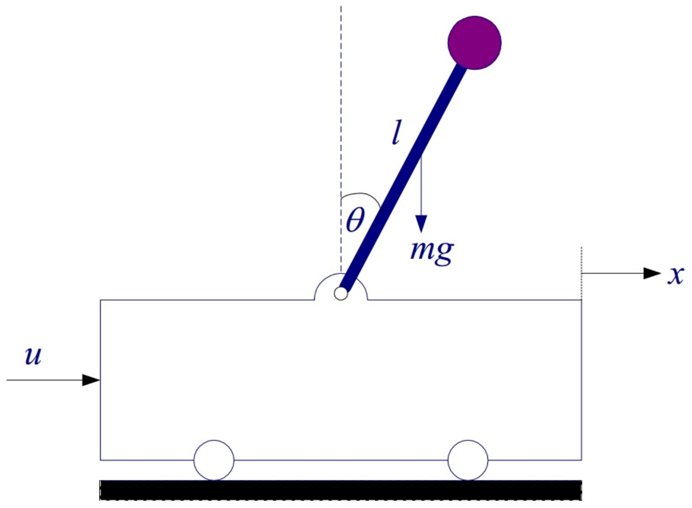


Figure 3.9: Simulink diagram for Case Study 3.

Figure 3.9 shows an inverted pendulum of length l and mass m supported by a frictionless pivot on a cart of mass M. It is to be balanced be means of a force u applied to the cart. That is, the cart must be moved in such a way that the pendulum is in the upright position. In a physical system there would be sensors to measure the position and velocity of the cart and angle θ measured from the vertical position. This is also a model of the attitude control of a space booster on takeoff. This is similar to balancing a broomstick on the palm of your hand. The equilibrium condition is when $\theta(t)$ and $\dot{\theta}(t)$ return to zero. The visual location of your hand and the position of the broomstick and the proper movement of your hand is the required feedback without which it is not possible to balance the broomstick. The differential equations describing the motion of the system by summing the forces on the pendulum result in the following nonlinear equations:

$$(M+m)\ddot{x} + (mL\cos\theta)\ddot{\theta} = (mL\sin\theta)(\dot{\theta})^2 + u \tag{3.17a}$$

$$(mL\cos\theta)\ddot{x} + mL^2\ddot{\theta} = mgL\sin\theta. \tag{3.17b}$$

Pre-Lab Exercise

Linearize equations (3.17) in the neighborhood of the zero initial states. *Hint: Substitute θ for $\sin\theta$, 1 for $\cos\theta$ and $0 for (\dot{\theta})^2$.* With the state variables defined as $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = x$, and $x_4 = \dot{x}$, show that the linearized state equation is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{ML}g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-m}{M}g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-1}{ML} \\ 0 \\ \frac{1}{M} \end{bmatrix} u. \tag{3.18}$$

If we want to have all the state variables available as output, we define the C matrix as the identity and D as a 4×1 zero matrix.

Lab Exercises (Digital Simulation using Simulink)

The parameters of the inverted pendulum are $M = 4$ kg, $m = 0.2$ kg, $L = 0.5$ m, and $g = 9.81$ m/s^2. Let the initial state vector be

$$x_0 = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \\ 0 \end{bmatrix}. \tag{3.19}$$

In a MATLAB script file, define the system parameters and the A, B, C, D matrices.

Save the file as `Lab4CS4aData.m`.

Launch Simulink, open a new model, get the `State-Space` block from the Continuous library, and construct the above state model. Double click on the `State-Space` block to open its dialog box and for the parameters type `A`, `B`, `C`, and `D`. Note that MATLAB is case-sensitive. For the initial condition, type `x0` (or however you defined the initial conditions in your script). In the Simulation Parameters dialog box set the Start time to 0 and the Stop time to 3. For the Solver option, use a variable step size and `ode45` algorithm.

Connect a `Demux` block to separate the signals, and use two scopes to display θ and x. Get an `Inport` block from the Source library and use it as an input terminal. Also get an `Outport` block from the Sink library and connect it to the signal for θ as shown in Figure 3.10.

Save the Simulink model as `Lab4CS4a.mdl`.

Run the script m-file `Lab4CS4aData` at the MATLAB prompt to calculate the A, B, C, and D matrices. These values are now defined and are available in Simulink. Start the simulation in Simulink and obtain a plot of θ and x. Comment on the stability of the system. MATLAB provides

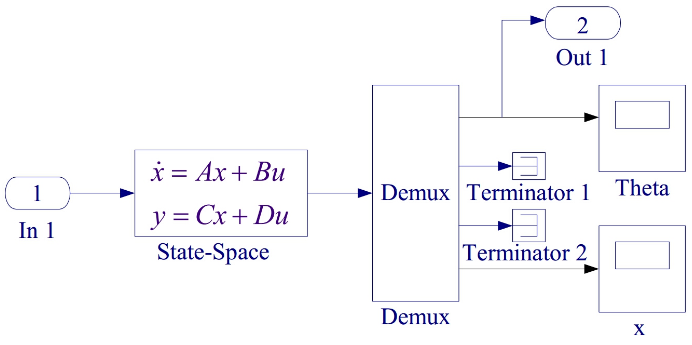


Figure 3.10: Simulink diagram for the inverted pendulum in Case Study 4.

the function `linmod` to extract a linear model in state variables or as a transfer function model using the Simulink file name as an argument. At the MATLAB prompt type the following commands to obtain the linearized transfer function model and roots of the characteristic equation:

Example Code:

```
[num, den] = linmod('Lab4CS4a');
r = roots(den);
```

You may have found that the angle θ increases without limit, i.e. the response is unbounded. Also you may find that a root of the characteristic equation is positive. This again confirms an unbounded response and we say that the system is unstable, that is, the inverted pendulum will fall over unless a suitable control force via state feedback is used.

The purpose is to design a control system such that for a small initial disturbance the pendulum can be brought back to the vertical position ($\theta = 0$), and the cart can be brought back to the reference position ($x = 0$).

One approach in modern control systems, accomplished by the use of state feedback, is known as *pole-placement design*. The pole-placement method allows all roots of the system characteristic equation to be placed in the desired locations. This results in a regulator with a constant gain vector K. In pole-placement design the control is achieved by feeding back the state variables through a regulator with constant gains. Consider the control system presented in state-variable form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.20a}$$

$$y(t) = Cx(t). \tag{3.20b}$$

Consider the block diagram of the system shown in Figure 3.11 with state feedback control

$$u(t) = -Kx(t), \tag{3.21}$$

where K is a $k \times 1$ 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.

Substituting (3.21) into (3.20), the closed-loop system variable representation is

$$\dot{x}(t) = [A - BK]x(t) = A_f x(t). \tag{3.22}$$

The design objective is to find the gain matrix K such that the characteristic equation for the controlled system is identical to the desired characteristic equation. A custom-made function named `placepole` is developed for the pole placement design. The syntax is

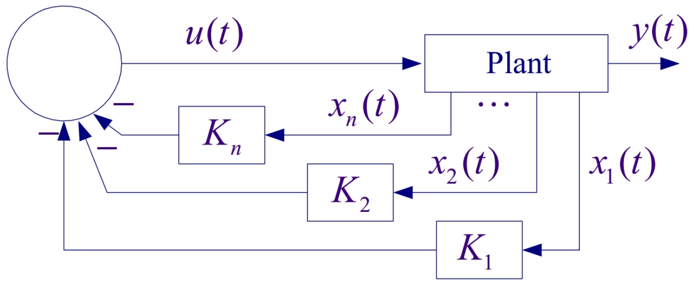


Figure 3.11: Control system design via pole-placement.

Example Code:

```
[K, Af] = placepole(A, B, C, P)
```

where A, B, and 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 closed-loop system matrix A_f. Also, the MATLAB Control System Toolbox contains two function for pole-placement design (`acker` and `place`).

An aspect of state variable design is state feedback design. In this study we use the custom function `placepole` and design a state feedback controller to place the closed-loop poles at

$$P = \{-1 \pm 0.5j, -4, -5\}.$$

In your `Lab4CS4aData4.m` script, append a definition of P and use it to call `placepole`.

In Simulink, open `Lab4CS4a.mdl`. Add the state feedbacks and set the gains to $K(1)$, $K(2)$, $K(3)$, and $K(4)$ as shown in Figure 3.12 and save the model as `Lab4CS4b.mdl`. Run the script file to evaluate K for use in Simulink. Then state the simulation in Simulink and obtain a plot of θ and x. Comment on the stability of the system. To see an inverted pendulum animation, make sure you have obtained the m-file named `InvPenAnimation.m`. After simulation, type `InvPenAnimation` at the MATLAB prompt.

At the MATLAB prompt type the following commands to obtain the linearized transfer function model and roots of the characteristic equation:

Example Code:

```
[num, den] = linmod('Lab4CS4b');
r = roots(den);
```

Check for the roots of the compensated system. Are they the same as the specified values? Is the system stable—that is—will the pendulum return to the vertical equilibrium position?

3.3 Take-Home Assignment

Experiment 4.1 and Experiment 4.2 in Cyber Exploration Laboratory at the end of Chapter 4 in [1].

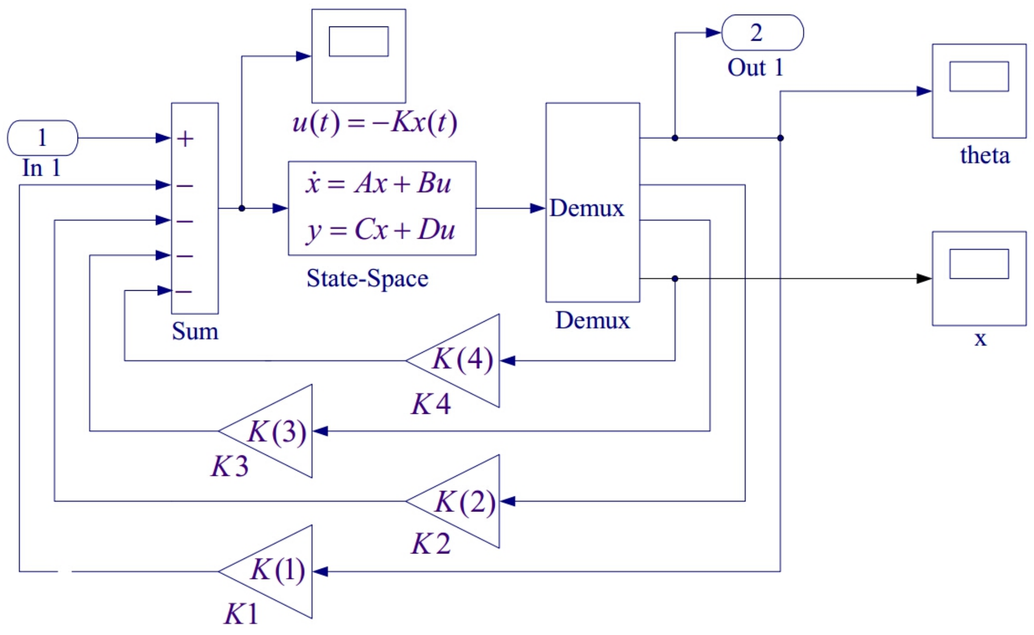


Figure 3.12: Control of an inverted pendulum via pole placement.

Bibliography

[1] Norman Nise. Control Systems Engineering, 6th ed. Wiley. 2010.

[2] Hadi Saadat. Computational Aids in Control Systems Using MATLAB. McGraw-Hill. 1993.

[3] Hadi Saadat website. http://people.msoe.edu/ saadat/matlab.htm