Control Tutorials for MATLAB and Simulink

The first step in the control design process is to develop appropriate mathematical models of the system to be controlled. These models may be derived either from physical laws or experimental data. In this section, we introduce the state-space and transfer function representations of dynamic systems. We then review some basic approaches to modeling mechanical and electrical systems and show how to generate these models in MATLAB for further analysis.

Key MATLAB commands used in this tutorial are: `ss` , `tf`

Contents

Dynamic Systems

Dynamic systems are systems that change or evolve in time according to a fixed rule. For many physical systems, this rule can be stated as a set of first-order differential equations:

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{1}$$

In the above equation, $\mathbf{x}(t)$ is the **state vector**, a set of variables representing the configuration of the system at time t. For instance, in a simple mechanical mass-spring-damper system, the two state variables could be the position and velocity of the mass. $\mathbf{u}(t)$ is the vector of external inputs to the system at time t, and $\mathbf{f}$ is a (possibly nonlinear) function producing the time derivative (rate of change) of the state vector, $d\mathbf{x}/dt$, for a particular instant of time.

The state at any future time, $\mathbf{x}(t_1)$, may be determined exactly given knowledge of the initial state, $\mathbf{x}(t_0)$, and the time history of the inputs, $\mathbf{u}(t)$, between t_0 and t_1 by integrating Equation (1). Though the state variables themselves are not unique, there is a minimum number of state variables, n, required in order to capture the "state" of a given system and to be able to predict the system's future behavior (solve the state equations). n is referred to as the **system order** and determines the dimensionality of the **state-space**. The system order usually corresponds to the number of independent energy storage elements in the system.

The relationship given in Equation (1) is very general and can be used to describe a wide variety of different systems; unfortunately, it may be very difficult to analyze. There are two common simplifications which make the problem more tractable. First, if the function $\mathbf{f}$ does not depend explicitly on time, i.e. $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, then the system is said to be **time invariant**. This is often a very reasonable assumption because the underlying physical laws themselves do not typically depend on time. For time-invariant systems, the parameters or coefficients of the function $\mathbf{f}$ are constant. The state variables, $\mathbf{x}(t)$, and control

inputs, $\mathbf{u}(t)$, however, may still be time dependent.

The second common assumption concerns the linearity of the system. In reality, nearly every physical system is nonlinear. In other words, $\mathbf{f}$ is typically some complicated function of the state and inputs. These nonlinearities arise in many different ways, one of the most common in control systems being "saturation" in which an element of the system reaches a hard physical limit to its operation. Fortunately, over a sufficiently small operating range (think tangent line near a curve), the dynamics of most systems are approximately **linear**. In this case, the system of first-order differential equations can be represented as a matrix equation, that is, $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$.

Until the advent of digital computers (and to a large extent thereafter), it was only practical to analyze **linear time-invariant (LTI)** systems. Consequently, most of the results of control theory are based on these assumptions. Fortunately, as we shall see, these results have proven to be remarkably effective and many significant engineering challenges have been solved using LTI techniques. In fact, the true power of feedback control systems are that they work (are **robust**) in the presence of the unavoidable modeling uncertainty.

State-Space Representation

For continuous linear time-invariant (LTI) systems, the standard state-space representation is given below:

(2) $$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

(3) $$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$

where $\mathbf{x}$ is the vector of state variables (nx1), $\dot{\mathbf{x}}$ is the time derivative of the state vector (nx1), $\mathbf{u}$ is the input or control vector (px1), $\mathbf{y}$ is the output vector (qx1), A is the system matrix (nxn), B is the input matrix (nxp), C is the output matrix (qxn), and D is the feedforward matrix (qxp).

The output equation, Equation (3), is necessary because often there are state variables which are not directly observed or are otherwise not of interest. The output matrix, C, is used to specify which state variables (or combinations thereof) are available for use by the controller. Also, it is often the case that the outputs do not directly depend on the inputs (only through the state variables), in which case D is the zero matrix.

The state-space representation, also referred to as the time-domain representation, can easily handle **multi-input/multi-output (MIMO)** systems, systems with non-zero initial conditions, and nonlinear systems via Equation (1). Consequently, the state-space representation is used extensively in "modern" control theory.

Transfer Function Representation

LTI systems have the extremely important property that if the input to the system

is sinusoidal, then the output will also be sinusoidal with the same frequency as the input, but with possibly different magnitude and phase. These magnitude and phase differences are a function of frequency and capture what is known as the **frequency response** of the system.

Using the **Laplace transform**, it is possible to convert a system's time-domain representation into a frequency-domain input/output representation, known as the **transfer function**. In so doing, it also transforms the governing differential equation into an algebraic equation which is often easier to analyze.

The Laplace transform of a time domain function, $f(t)$, is defined below:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt \tag{4}$$

where the parameter $s = \sigma + j\omega$ is a complex frequency variable. It is very rare in practice that you will have to directly evaluate a Laplace transform (though you should certainly know how to). It is much more common to look up the transform of a time function in a table such as the one found here: Laplace Transform Table

The Laplace transform of the nth derivative of a function is particularly important:

$$\mathcal{L}\left\{\frac{d^n f}{dt^n}\right\} = s^n F(s) - s^{n-1} f(0) - s^{n-2} \dot{f}(0) - ... - f^{(n-1)}(0) \tag{5}$$

Frequency-domain methods are most often used for analyzing LTI **single-input/single-output (SISO)** systems, e.g. those governed by a constant coefficient differential equation, as shown below:

$$a_n \frac{d^n y}{dt^n} + ... + a_1 \frac{dy}{dt} + a_0 y(t) = b_m \frac{d^m u}{dt^m} + ... + b_1 \frac{du}{dt} + b_0 u(t) \tag{6}$$

The Laplace transform of this equation is given below:

$$a_n s^n Y(s) + ... + a_1 s Y(s) + a_0 Y(s) = b_m s^m U(s) + ... + b_1 s U(s) + b_0 U(s) \tag{7}$$

where $Y(s)$ and $U(s)$ are the Laplace Transforms of $y(t)$ and $u(t)$, respectively. Note that when finding transfer functions, we always assume that the each of the initial conditions, $y(0)$, $\dot{y}(0)$, $u(0)$, etc. is zero. The transfer function from input $U(s)$ to output $Y(s)$ is, therefore:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0} \tag{8}$$

It is useful to factor the numerator and denominator of the transfer function into what is termed **zero-pole-gain** form:

$$G(s) = \frac{N(s)}{D(s)} = K \frac{(s - z_1)(s - z_2)...(s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2)...(s - p_{n-1})(s - p_n)} \tag{9}$$

The **zeros** of the transfer function, $z_1, \ldots, z_m$, are the roots of the numerator polynomial, i.e. the values of s such that $N(s) = 0$. The **poles** of the transfer function, $p_1, \ldots, p_n$, are the roots of the denominator polynomial, i.e. the values of s such that $D(s) = 0$. Both the zeros and poles may be complex valued (have both real and imaginary parts). The system **Gain** is $K = b_m/a_n$.

Note that we can also determine the transfer function directly from the state-space representation as follows:

$$G(s) = \frac{Y(s)}{U(s)} = C(s\mathbf{I} - A)^{-1}B + D \tag{10}$$

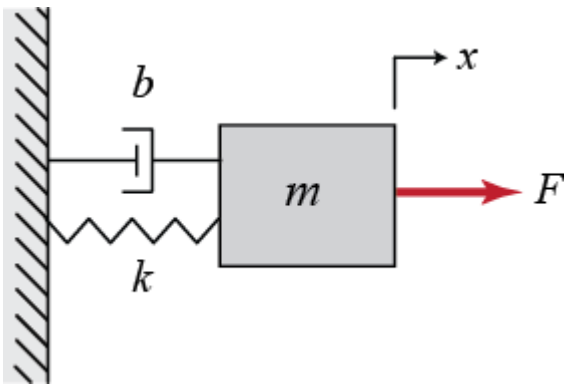
Mechanical Systems

Newton's laws of motion form the basis for analyzing mechanical systems. **Newton's second law**, Equation (11), states that the sum of the forces acting on a body equals the product of its mass and acceleration. **Newton's third law**, for our purposes, states that if two bodies are in contact, then they experience the same magnitude contact force, just acting in opposite directions.

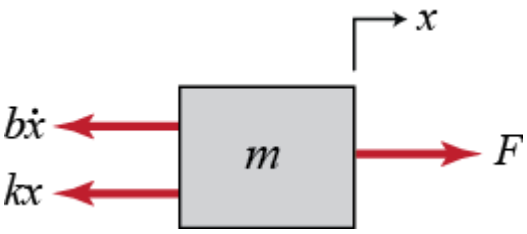
$$\Sigma\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{x}}{dt^2} \tag{11}$$

When applying this equation, it is best to construct a **free-body diagram (FBD)** of the sysetm showing all of the applied forces.

Example: Mass-Spring-Damper System



The free-body diagram for this system is shown below. The spring force is proportional to the displacement of the mass, x, and the viscous damping force is proportional to the velocity of the mass, $v = \dot{x}$. Both forces oppose the motion of the mass and are, therefore, shown in the negative x-direction. Note also that $x = 0$ corresponds to the position of the mass when the spring is unstretched.



Now we proceed by summing the forces and applying Newton's second law, Equation (11), in each direction. In this case, there are no forces acting in the y -direction; however, in the x-direction we have:

(12) $$\Sigma F_x = F(t) - b\dot{x} - kx = m\ddot{x}$$

This equation, known as the **governing equation**, completely characterizes the dynamic state of the system. Later, we will see how to use this to calculate the response of the system to any external input, $F(t)$, as well as to analyze system properties such as stability and performance.

To determine the state-space representation of the mass-spring-damper system, we must reduce the second-order governing equation to a set of two first-order differential equations. To this end, we choose the position and velocity as our state variables.

(13) $$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

The position variable captures the potential energy stored in the spring, while the velocity variable captures the kinetic energy stored by the mass. The damper only dissipates energy, it doesn't store energy. Often when choosing state variables it is helpful to consider what variables capture the energy stored in the system.

The state equation in this case is:

(14) $$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

If, for instance, we are interested in controlling the position of the mass, then the output equation is:

(15) $$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

Entering State-Space Models into MATLAB

Now we will demonstrate how to enter the equations derived above into an `m-file` for MATLAB. Let's assign the following numerical values to each of the variables.

```
m                   mass                                1.0 kg
k                   spring constant                     1.0 N/m
```

```
b                   damping constant                        0.2 Ns/m
F                   input force                             1.0 N
```

Create a new `m-file` and enter the following commands.

```
m = 1;
k = 1;
b = 0.2;
F = 1;

A = [0 1; -k/m -b/m];
B = [0 1/m]';
C = [1 0];
D = [0];

sys = ss(A,B,C,D)
```

```
sys =

  A =
         x1    x2
   x1     0     1
   x2    -1  -0.2

  B =
       u1
   x1   0
   x2   1

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

The Laplace transform for this system assuming zero initial conditions is

(16) $$ms^2X(s) + bsX(s) + kX(s) = F(s)$$

and, therefore, the transfer function from force input to displacement output is

(17) $$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$

Entering Transfer Function Models into MATLAB

Now we will demonstrate how to create the transfer function model derived above within MATLAB. Enter the following commands into the `m-file` in which you defined the system parameters.

```
s = tf('s');
```

```
sys = 1/(m*s^2+b*s+k)

sys =

         1
  ---------------
  s^2 + 0.2 s + 1

Continuous-time transfer function.
```

Note that we have used the symbolic `s` variable here to define our transfer function model. We recommend using this method most of the time; however, in some circumstances, for instance in older versions of MATLAB or when interfacing with SIMULINK, you may need to define the transfer function model using the numerator and denominator polynomial coefficients directly. In these cases, use the following commands:

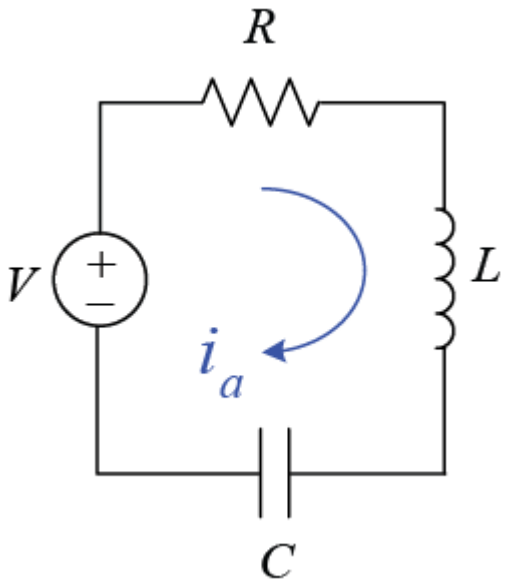
```
num = [1];
den = [m b k];
sys = tf(num,den)

sys =

         1
  ---------------
  s^2 + 0.2 s + 1

Continuous-time transfer function.
```

Electrical Systems

Like Newton’s laws for mechanical systems, Kirchoff’s circuit laws are fundamental analytical tools for modeling electrical systems. **Kirchoff’s current law (KCL)** states that the sum of the electrical currents entering a node in a circuit must equal the sum of electrical currents exiting the node. **Kirchoff’s voltage law (KVL)** states that the sum of voltage differences around any closed loop in a circuit is zero. When applying KVL, the source voltages are typically taken as positive and the load voltages are taken as negative.

Example: RLC Circuit

We will now consider a simple series combination of three passive electrical elements: a resistor, an inductor, and a capacitor, known as an **RLC Circuit**.



Since this circuit is a single loop, each node only has one input and one output; therefore, application of KCL simply shows that the current is the same throughout the circuit at any given time, $i(t)$. Now applying KVL around the loop and using the sign conventions indicated in the diagram, we arrive at the following **governing equation**.

(18) $$V(t) - Ri - L\frac{di}{dt} - \frac{1}{C}\int idt = 0$$

We note that that the governing equation for the RLC circuit has an analogous form to the mass-spring-damper mechanical system. In particular, they are both second-order systems where the charge (integral of current) corresponds to displacement, the inductance corresponds to mass, the resistance corresponds to viscous damping, and the inverse capacitance corresponds to the spring stiffness. These analogies and others like them turn out to be quite useful conceptually in understanding the behavior of dynamical systems.

The state-space representation is found by choosing the charge on the capacitor and current through the circuit (inductor) as the state variables.

(19) $$\mathbf{x} = \begin{bmatrix} q \\ i \end{bmatrix}$$

where,

(20) $$q = \int idt$$

The state equation is, therefore:

(21) $$\dot{\mathbf{x}} = \begin{bmatrix} i \\ \frac{di}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} q \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V(t)$$

We choose the current as ouput as follows:

(22) $$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} q \\ i \end{bmatrix}$$

The transfer function representation may be found by taking the Laplace transform as we did for the mass-spring-damper or from the state-space equation as follows:

(23) $$\frac{I(s)}{V(s)} = C(s\mathbf{I} - A)^{-1}B + D = \begin{bmatrix} 0 & 1 \end{bmatrix} \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}$$

(24) $$\Rightarrow \quad \frac{I(s)}{V(s)} = \frac{s}{Ls^2 + Rs + \frac{1}{C}}$$

The RLC state-space and transfer function models can be entered into MATLAB using the same procedure as discussed for the mass-spring-damper system above.

System Identification

In this section, we have seen how to model systems using basic physical principles; however, often this is not possible either because the parameters of the system are uncertain, or the underlying processes are simply not understood. In these cases, we must rely on experimental measurements and statistical techniques to develop a system model, a process known as **system identification**.

System identification may be performed using either time-domain or frequency-domain data, see the Introduction: System Identification page for further details. A couple of system identification activities can also be found from the **Hardware** tab located at the top of this window.

Also refer to MATLAB's System Identification Toolbox for more information on this subject.

System Conversions

Most operations in MATLAB can be performed on either the transfer function, the state-space model, or the zero-pole-gain form. Furthermore, it is simple to transfer between these forms if the other representation is required. If you need to learn how to convert from one representation to another, see the Introduction: System Conversions page.