Louisiana Tech University
ELEN 479 Automatic Control Systems Lab
Lab 3

INTRODUCTION TO STATE SPACE

Objective:

- Model dynamics of electro-mechanical system using state space representation
- Understand the difference between Transfer Functions and State Space Representations
- Linearize a dynamic system about an equilibrium point and analyze system stability

Theory:

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:

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

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, $\dot{\mathbf{x}}$, 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, almost 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, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$.

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 is 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:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{3}$$

where $\mathbf{x}$ is the vector of state variables (n by 1), $\dot{\mathbf{x}}$ is the time derivative of the state vector (n by 1), $\mathbf{u}$ is the input or control vector (p by 1), $\mathbf{y}$ is the output vector (q by 1), $\mathbf{A}$ is the system matrix (n by n), $\mathbf{B}$ is the input matrix (n by p), $\mathbf{C}$ is the output matrix (q by n), and $\mathbf{D}$ is the feedforward matrix (q by p).

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, $\mathbf{C}$, 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 $\mathbf{D}$ 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 given below:

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

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 Laplace Transform Table.

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

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

For zero initial condition:

$$L\left\{\frac{d^n f}{dt^n}\right\} = s^n F(s) \tag{6}$$

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:

$$a_n \frac{d^n y}{dt^n} + \cdots + a_1 \frac{dy}{dt} + a_0 y(t) = b_m \frac{d^m u}{dt^m} + \cdots + b_1 \frac{du}{dt} + b_0 u(t) \tag{7}$$

The Laplace transform of this equation is given below

$$a_n s^n Y(s) + \cdots + a_1 s Y(S) + a_0 Y(s) = b_m s^m U(s) + \cdots + b_1 s U(s) + b_0 U(s) \tag{8}$$

Where $Y(s)$ and $U(s)$ are the Laplace Transform of $y(t)$ and $u(t)$, respectively. Note that when finding transfer functions, we assume that 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:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \tag{9}$$

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

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

Linearization of Non-Linear Systems

Most components found in physical systems have nonlinear characteristics. In practice, we may find that some devices have moderate nonlinear characteristics, or nonlinear properties that would occur if they were driven into certain operating regions. For these devices, the modeling by linear-system models may give quite accurate analytical results over a relatively wide range of operating conditions.

However, there are numerous physical devices that possess strong nonlinear characteristics. For these devices, a linearized model is valid only for limited range of operation and often only at the operating point at which the linearization is carried out. More importantly, when a nonlinear system is linearized at an operating point, the linear model may contain time-varying elements.

Although there are many ways to linearize a system but for this lab we will be concentrating on linearizing a non-linear system using the state space approach.

A nonlinear system can be represented by the following vector-matrix state equations:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{11}$$

Where $\mathbf{x}(t)$ represents the (n by 1) state vector; $\mathbf{u}(t)$, the (p by 1) input vector; and $\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t)]$, an ($n$ by 1) function vector. In general, $\mathbf{f}$ is a function of the state vector and the input vector.

Being able to represent a nonlinear and/or time-varying system by state equations is a distinct advantage of the state-variable approach over the transfer-function method, since the latter is strictly defined only for linear time-invariant systems. As a simple example, the following nonlinear state equations are given:

$$\frac{dx_1(t)}{dt} = x_1(t) + x_2^2(t) \tag{12}$$

$$\frac{dx_2(t)}{dt} = x_1(t) + u(t) \tag{13}$$

Because nonlinear systems are usually difficult to analyze and design, it is desirable to perform a linearization whenever the situation justifies it. A linearization process that depends on expanding the nonlinear state equations into a Taylor series about a nominal operating point or trajectory is now described. All the terms of the Taylor series of order higher than the first are discarded. and the linear approximation of the nonlinear state equations at the nominal point results. Let the nominal operating trajectory be denoted by $x_0(t)$, which corresponds to the nominal input $u_0(t)$ and some fixed initial states. Expanding the nonlinear state equation of Equation 11 into a Taylor series about $x(t) = x_0(t)$ and neglecting all the higher-order terms yield:

$$\dot{x}_i(t) = f_i(x_0, u_0) + \sum_{j=1}^{n} (x_j - x_{0,j}) \frac{\partial f_i(x,u)}{\partial x_j}\bigg|_{x_0,u_0} + \sum_{j=1}^{n} (u_j - u_{0,j}) \frac{\partial f_i(x,u)}{\partial u_j}\bigg|_{x_0,u_0} \tag{14}$$

Where $i = 1, 2, \ldots, n$. Let:

$$\begin{aligned} \delta x_i &= x_i - x_{0i} \\ \delta u_j &= u_j - u_{0j} \\ \delta \dot{x}_i &= x_i - x_{0i} \\ \delta \dot{x}_{0i} &= f(x_0, u_0) \end{aligned}$$

Equation 14 can then be written as:

$$\dot{x}_i(t) = \sum_{j=1}^{n} \Delta x_j \frac{\partial f_i(x,u)}{\partial x_j}\bigg|_{x_0,u_0} + \sum_{j=1}^{n} \Delta u_j \frac{\partial f_i(x,u)}{\partial u_j}\bigg|_{x_0,u_0} \tag{15}$$

Equation 15 may be written as:

$$\Delta\dot{x} = A^*\Delta x + B^*\Delta u \tag{16}$$

Where

$$A^* = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$
$$B^* = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_n} \end{bmatrix}$$

An example will be demonstrated in class to serve to illustrate the linearization the procedure just described.

Stability

For our purposes, we will use the Bounded Input Bounded Output (BIBO) definition of stability which states that a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.

The transfer function representation is especially useful when analyzing system stability. If all poles of the transfer function (values of s for which the denominator equals zero) have negative real parts, then the system is stable. If any pole has a positive real part, then the system is unstable. If we view the poles on the complex s-plane, then all poles must be in the left-half plane (LHP) to ensure stability. If any pair of poles is on the imaginary axis, then the system is marginally stable, and the system will tend to oscillate. A system with purely imaginary poles is not considered BIBO stable. For such a system, there will exist finite inputs that lead to an unbounded response. The poles of an LTI system model can easily be found in MATLAB using the pole command, an example of which is shown below:

```
> s = tf('s');
G = 1/(s^2+2*s+5)
pole(G)

G =
         1
   -------------
   s^2 + 2 s + 5
Continuous-time transfer function
ans =
     -1.0000 + 2.0000i
     -1.0000 - 2.0000i
```

Thus, this system is stable since the real parts of the poles are both negative. The stability of a system may also be found from the state-space representation. In fact, the poles of the transfer function are the eigenvalues of the system matrix A. We can use the eig command to calculate the eigenvalues using either the LTI system model directly, `eig(G)`, or the system matrix as shown below.

```
> [A,B,C,D] = ssdata(G);
eig(A)
ans =
      -1.0000 + 2.0000i
      -1.0000 - 2.0000i
```

Controllability and Observability

A system is controllable if there always exists a control input, $u(t)$, that transfers any state of the system to any other state in finite time. It can be shown that an LTI system is controllable if and only if its controllability matrix, $\mathcal{C}$, has full rank (i.e. if $\text{rank}(\mathcal{C}) = n$ where n is the number of states variables). The rank of the controllability matrix of an LTI model can be determined in MATLAB using the commands `rank(ctrb(A,B))` or `rank(ctrb(sys))`.

All of the state variables of a system may not be directly measurable, for instance, if the component is in an inaccessible location. In these cases, it is necessary to estimate the values of the unknown internal state variables using only the available system outputs. A system is observable if the initial state, $x(t_0)$, can be determined based on knowledge of the system input, $u(t)$, and the system output, $y(t)$, over some finite time interval $t_0 < t < t_f$. For LTI systems, the system is observable if and only if the observability matrix, $\mathcal{O}$, has full rank (i.e. if $\text{rank}(\mathcal{O}) = n$ where n is the number of state variables). The observability of an LTI model can be determined in MATLAB using the command `rank(obsv(A,C))` or `rank(obsv(sys))`.

Controllability and observability are dual concepts. A system A, is controllable if and only if a system A', B' is observable. This fact will be useful when designing an observer, as we shall see below.

Lab Instructions

Modeling of a Magnetic Ball Suspension System in State Space

Figure 1 shows the diagram of a magnetic-ball-suspension system. The objective of the system is to control the position of the steel ball by adjusting the current in the electromagnet through the input voltage $v(t)$.

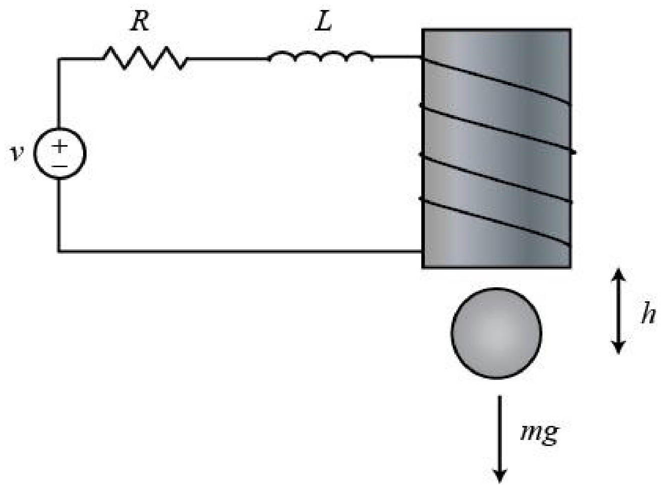

Figure 1: Magnetic ball suspension system

The differential equations of the system are:

$$M\frac{d^2h(t)}{dt^2} = Mg - K\frac{i^2(t)}{h(t)} \tag{17}$$

$$v(t) = Ri(t) + L\frac{di(t)}{dt} \tag{18}$$

Where,

- $v(t)$ = input voltage,
- $h(t)$ = ball position from initial position,
- $i(t)$ = current through the electromagnet,
- R = winding resistance,
- L = winding inductance,
- M = mass of the ball,
- g = gravitational acceleration,
- K = coefficient that determines the magnetic force on the ball,

Let us define the state variables as $x_1(t) = h(t)$, $x_2(t) = \frac{dh(t)}{dt}$, and $x_3(t) = i(t)$. The state equations of the system are:

$$f_1(x,u) = \frac{dx_1(t)}{dt} = x_2(t) \tag{19}$$

$$f_2(x,u) = \frac{dx_2(t)}{dt} = g - \frac{K}{M}\frac{x_3^2(t)}{x_1(t)} \tag{20}$$

$$f_3(x,u) = \frac{dx_3(t)}{dt} = -\frac{R}{L}x_3(t) + \frac{1}{L}v(t) \tag{21}$$

Let us linearize the system about the equilibrium point $h_0(t) = x_{0,1} =$ constant. Then,

$$x_{0,2}(t) = \frac{x_{0,1}(t)}{dt} = 0 \tag{22}$$

$$\frac{d^2h_0(t)}{dt^2} = 0 \tag{23}$$

The nominal value of $i(t)$ is determined by substituting Equation 23 into Equation 17. Thus

$$i_0(t) = x_{0,3}(t) = \sqrt{Mgx_{0,1}} \tag{24}$$

The linearized state equation is expressed in the form of Equation 16, with the coefficient matrices $\mathbf{A}^*$ and $\mathbf{B}^*$ evaluated as

$$\mathbf{A}^* = \begin{bmatrix} 0 & 1 & 0 \\ \frac{Kx_{0,3}^2}{Mx_{0,1}^2} & 0 & \frac{-2Kx_{0,3}}{Mx_{0,1}} \\ 0 & 0 & -\frac{R}{L} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{Kg}{x_{0,1}} & 0 & -2K\sqrt{\frac{g}{Mx_{0,1}}} \\ 0 & 0 & -\frac{R}{L} \end{bmatrix} \tag{25}$$

$$\mathbf{B}^* = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} \tag{26}$$

Function Reference

Relevant function descriptions pulled from MATLAB documentation. These can all be used in a solution to the lab tasks.

`sys=ss(A,B,C,D)` creates a continuous-time state-space model object of the following form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

For instance, consider a plant with Nx states, Ny outputs, and Nu inputs. The state-space matrices are:

- A is an Nx-by-Nx real- or complex-valued matrix.
- B is an Nx-by-Nu real- or complex-valued matrix.
- C is an Ny-by-Nx real- or complex-valued matrix.
- D is an Ny-by-Nu real- or complex-valued matrix.

`P = pole(sys)` returns the poles of the SISO or MIMO dynamic system model sys. The output is expressed as the reciprocal of the time units specified in `sys.TimeUnit`. The poles of a dynamic system determine the stability and response of the system.

An open-loop linear time-invariant system is stable if:

- In continuous-time, all the poles of the transfer function have negative real parts. When the poles are visualized on the complex s-plane, then they must all lie in the left-half plane (LHP) to ensure stability.
- In discrete-time, all the poles must have a magnitude strictly smaller than one, that is they must all lie inside the unit circle.

`X = zeros(sz)` returns an array of zeros where size vector `sz` defines `size(X)`. For example, `zeros([2 3])` returns a 2-by-3 matrix.

`sz = size(A)` returns a row vector whose elements are the lengths of the corresponding dimensions of `A`. For example, if `A` is a 3-by-4 matrix, then `size(A)` returns the vector `[3 4]`. If `A` is a table or timetable, then `size(A)` returns a two-element row vector consisting of the number of rows and the number of table variables.

`y = lsim(sys, u, t, x0)` returns the system response `y`, sampled at the same times `t` as the input. For single-output systems, `y` is a vector of the same length as `t`. For multi-output systems, `y` is an array having as many rows as there are time samples (`length(t)`) and as many columns as there are outputs in `sys`. This syntax does not generate a plot. The optional vector `x0` specifies initial state values when `sys` is a state-space model.

`[num,den] = ss2tf(A,B,C,D)` converts a state-space representation of a system into an equivalent transfer function. `ss2tf` returns the Laplace-transform transfer function for continuous-time systems and the Z-transform transfer function for discrete-time systems.

`I = eye(n)` returns an n-by-n identity matrix with ones on the main diagonal and zeros elsewhere.

`s = tf('s')` creates special variable `s` that you can use in a rational expression to create a continuous-time transfer function model. Using a rational expression can sometimes be easier and more intuitive than specifying polynomial coefficients.

`Y = inv(X)` computes the inverse of square matrix `X`.

- `X^(-1)` is equivalent to `inv(X)`.

- `x = A\b` is computed differently than `x = inv(A)*b` and is recommended for solving systems of linear equations.

Task 1 You already have the state space matrices in equations 25 and 26. Show the steps needed to derive these matrices from equations 17 and 18. Linearize the magnetic ball suspension system using the state space approach about the equilibrium point $h_e = 0.01$ m (where nominal current is about 7 A) and derive $\mathbf{A}^*$ and $\mathbf{B}^*$. Use $M = 0.05$ kg, $K = 0.0001$, $L = 0.01$ H, $R = 1\ \Omega$, $g = 9.81\ \mathrm{m\,s^{-2}}$. In the output we will only be measuring the height of the ball $h(t)$.

Task 2 Use MATLAB to write a script to model the system using your linearized matrices. Implement in your code to find the poles of the system.

Task 3 Add a constant input $v(t) = 0$ with initial condition $x_0 = [0.01\ 0\ 0]^T$ into your code and run a linear simulation in MATLAB to view the open loop response to the system.

Task 4 Repeat Task 3 in Simulink.

Task 5

- Use MATLAB's own built-in function to convert the state space representation to transfer function representation
- Use Equation 10 to compute transfer function from state space model symbolically in MATLAB. You will need to initialize the variable `s` with the function `tf()`.

Post Lab 3 Since this is a post-lab, finding the appropriate functions is up to you. They can all be found by searching for keywords in MATLAB's help website.

1. Find the poles and zeroes of the system and generate a pole-zero map. Draw or printout that map in your notebook.
2. Comment on the output of Task 2 and the positions of the poles of the system. What do they tell you about system stability?
3. Do you think the system is stable? If yes, why?
4. Is the system observable with the output provided?
5. Is the system controllable with the inputs provided?