© Copyright F.L. Lewis 1999
All rights reserved

**Updated:Tuesday, August 05, 2008**

---

## STATE VARIABLE (SV) SYSTEMS

A natural description for dynamical systems is the nonlinear state-space or state variable (SV) equation

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \qquad (1)$$

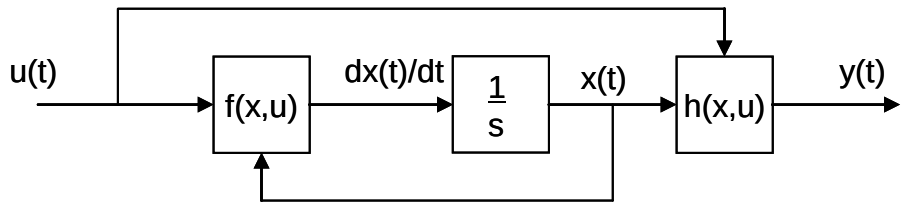
with $x(t) \in R^n$ the internal state, $u(t) \in R^m$ the control input, and $y(t) \in R^p$ the measured output. These equations are nonlinear and can capture very general system behaviors. The nonlinear state equation follows in a natural manner from a physical analysis of naturally occurring systems, using, for instance, Hamilton's equations of motion or Lagrange's equation of motion

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = F ,$$

with $q(t)$ the generalized position vector, $\dot{q}(t)$ the generalized velocity vector, and F(t) the generalized force vector. The Lagrangian is L= K-U, the kinetic energy minus the potential energy. Using Lagrange's equation, one can derive the nonlinear function f(x,u). The output function h(x,u) depends on the measurements selected by the design engineer.

Note that the SV model has *m* inputs, *n* states, and *p* outputs, so it can represent complicated *multivariable* (multi-input/multi-output) systems such as modern aerospace systems, automobiles, submarine vehicles, etc.

The nonlinear state-space system has the structure shown in the figure. Note that it contains an integrator, which acts as the memory of this dynamical system.



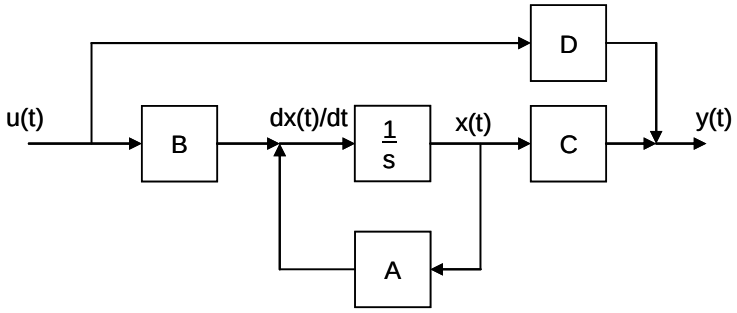

Nonlinear State-Space System

The linear state-space equations are given by

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

where A is the system or plant matrix, B is the control input matrix, C is the output or measurement matrix, and D is the direct feed matrix. This description is said to be time-invariant if A, B, C, D are constant matrices. These equations are obtained directly from a physical analysis if the system is inherently linear.

The linear state-space system has the form shown in the figure. It has more structure than the nonlinear SV system. For that reason, control system design is easier for linear SV systems. Note that the feedback is determined only by the system A matrix. The direct feedback matrix D is often equal to zero for many systems.




Linear State-Space System

If the system is nonlinear, then the state equations are nonlinear. In this case, an approximate linearized system description may be obtained by computing the Jacobian matrices

$$A(x,u) = \frac{\partial f}{\partial x}, \quad B(x,u) = \frac{\partial f}{\partial u}, \quad C(x,u) = \frac{\partial h}{\partial x}, \quad D(x,u) = \frac{\partial h}{\partial u}.$$

These are evaluated at a nominal set point (x,u) to obtain constant system matrices A,B,C,D, yielding a linear time-invariant state description which is approximately valid for small excursions about the nominal point.

The Jacobian resulting from differentiation of a *p*-vector function *h* with respect to an *m*-vector variable $u = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}^T$ is a $p \times m$ matrix found as

$$\frac{\partial h}{\partial u} = \begin{bmatrix} \frac{\partial h}{\partial u_1} & \frac{\partial h}{\partial u_2} & \cdots \end{bmatrix}.$$

### *Example 1*

A second-order differential equation of the sort occurring in robotic systems is

$$m\ddot{q} + mL\dot{q}^2 + mgL\sin q = \tau$$

where q(t) is an angle and τ(t) is an input torque. By defining the state $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ as

$$x_1 = q(t), \quad x_2 = \dot{q}(t)$$

one may write the state equation

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -Lx_2^2 - gL\sin x_1 + \frac{1}{m}u$$

where the control input is u(t)= τ(t). To solve the second-order differential equation one requires two initial conditions, e.g. $q(0), \dot{q}(0)$. Thus, there are two state components. The state components correspond to energy storage variables. For instance, in this case one could think of potential energy $mgh$ (the third term in the differential equation, which involves $q(t)$), and rotational kinetic energy $m\omega^2$ (the second term, which involves $\omega = \dot{q}(t)$).

This is a nonlinear state equation. One can place it into the form (1) simply by noting that $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ and writing

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -Lx_2^2 - gL\sin x_1 + \frac{1}{m}u \end{bmatrix} \equiv f(x,u)$$

This defines *f(x,u)* as the given nonlinear function 2-vector.

By computing the Jacobians, the linear SV representation is found to be

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -gL\cos x_1 & -2Lx_2 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u = Ax + Bu\,.$$

Evaluating this at a nominal equilibrium point of x=0, u=0 yields the linear time-invariant state description

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -gL & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u = Ax + Bu$$

which describes small excursions about the origin.

The function *f(x,u)* is given by the dynamics of the system. By contrast, the output function *h(x,u)* is given by which measurements the engineer decides to take.

It is easy to measure angles using an optical encoder. However, measuring angular velocities requires a tachometer, which is more expensive. Therefore, suppose we decide to measure the robot joint angle $x_1 = q$. Then the output function is given as

$$y = h(x,u) = \begin{bmatrix} 1 & 0 \end{bmatrix} x .$$

This equation is linear and directly defines the matrices $C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = 0$.

---

## COMPUTER SIMULATION

Given the state-space description it is very easy to simulate a system on a computer. All one requires is a numerical integration routine such as Runge-Kutta that computes the state derivative using $\dot{x} = f(x,u)$ to determine x(t) over a time interval. MATLAB has an integration routine 'ode23' that can be used to simulate any nonlinear system in state-space form. For linear systems, MATLAB has a variety of simulation routines including 'step' for the step response, etc., however, it is recommended that 'ode23' be used even for linear systems, since it facilitates controller design and simulation.

### *Example 2*

To use 'ode23' one must write a function M-file that contains the state equations. To simulate the nonlinear system in Example 1, one may use the M-file:

```
function xdot= robot(t,x) ;
g= 9.8 ; L= 1 ; m= 10 ; u= sin(2*pi*t)
xdot(1)= x(2) ;
xdot(2)= -g*L*sin(x(1)) - L*x(2)^2 + u/m ;
```

where we have assumed that L= 1m, m= 10 Kg. The control input is set to be a sinusoid. Using vector operations one could write the simpler M-file:

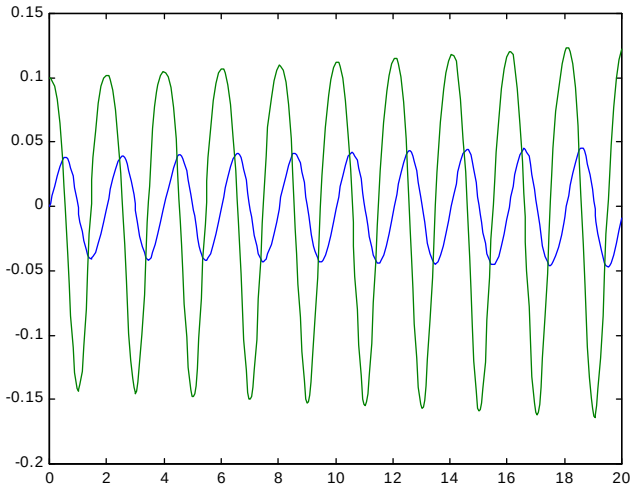
```
function xdot= robot(t,x) ;
g= 9.8 ; L= 1 ; m= 10 ; u= sin(2*pi*t)
xdot= [x(2) ;    -g*L*sin(x(1)) - L*x(2)^2 + u/m] ;
```

Note that the semicolon is used to terminate each line to avoid printing to the screen during computations.

Given this M-file, stored in a file named 'robot.m', let's say, the following command lines compute and plot the time response over the time interval 0 to 20 sec:

```
tint= [0  20] ;  % define time interval [t0  tf]
x0= [0   0.1]' ;  % initial conditions
[t,x]= ode23('robot', tint, x0);
plot(t,x)
```

Using this code, one obtains the figure shown.

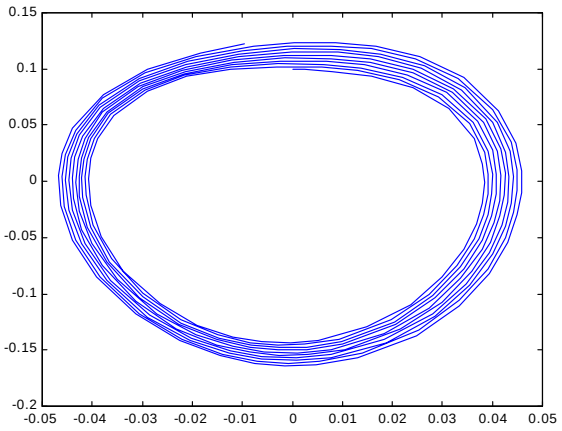



**Time plot of x(1), x(2) vs. time**

A Phase-Plane plot is a graph of x(2) vs. x(1). To make this plot, one uses

plot(x(:,1),x(:,2))

and obtains the figure shown. Note the use of the colon to select an entire column in an array.




**Phase-plane plot of x(2) vs. x(1)**

In which direction does the plot move- clockwise or counterclockwise?

Using MATLAB, one is able to perform very quickly the most exotic of analysis, simulation, and plotting tasks. An introduction to MATLAB and some excellent demos are available at the MATLAB website www.mathworks.com. The next two samples are taken from Shahian and Hassul, "Control System Design Using MATLAB," Prentice-Hall, 1993.

## 3.6 Program Examples

**Example 3.1**

The following is a simple script that will compute the step response of a second order system for values of $\zeta$ ranging from 0.1 to 1 and will create a mesh plot of the step responses.

```
n=1; y=zeros(200,1); i=1;
for del=0.1:0.1:1
        d=[1, 2*del, 1];
        t=[0:0.1:19.9]';
        y(:,i)=step(n,d,t);
        i=i+1;
end
mesh(fliplr(y),[-120 30])
```

The output of the program is shown in the Figure 3-1. The *fliplr* command flips a matrix from left to right. This along with the view angle change is done to re-create the plot on the cover of the Control System Toolbox User's Guide.

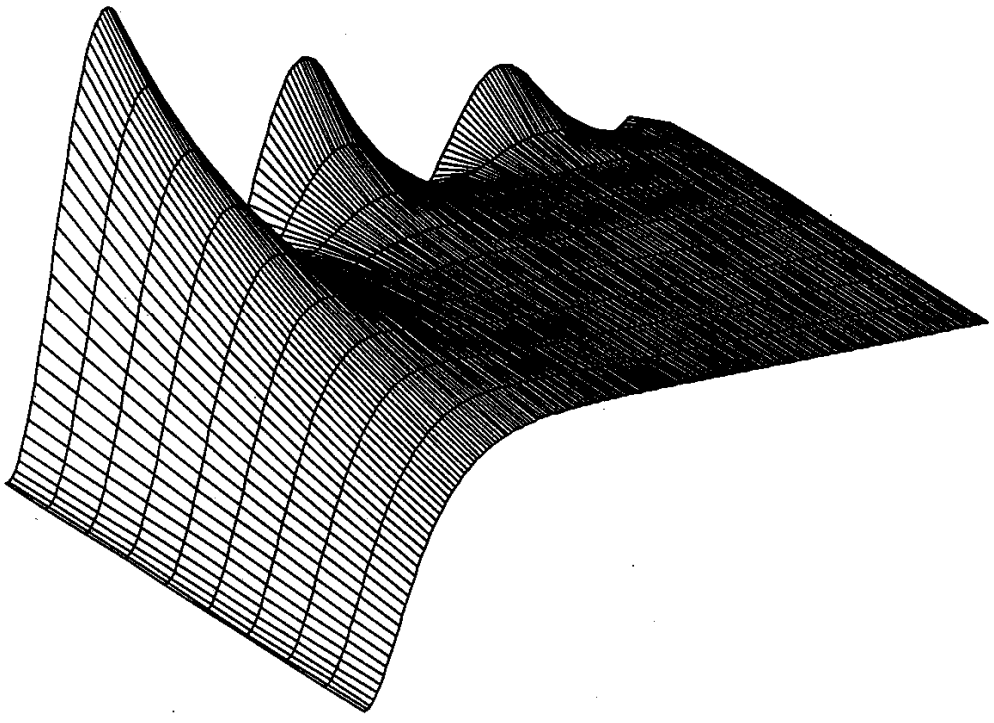


**Figure 3-1** Family of step responses.

We will demonstrate the process by creating the frequency response surface of a second order system. Consider the system

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Letting $s = \sigma + j\omega$ we get

$$G(\sigma + j\omega) = \frac{\omega_n^2}{(\sigma^2 - \omega^2 + 2\zeta\sigma\omega_n + \omega_n^2) + j\,(2\sigma\omega + 2\zeta\omega\omega_n)}$$

For $\zeta = 0.5$ and $\omega_n = 1$

$$G(\sigma + j\omega) = \frac{1}{(\sigma^2 - \omega^2 + \sigma + 1) + j\,(2\sigma\omega + \omega)}$$

This function of two variables will now be plotted for: $-3 \le \sigma \le 3$ and $-3 \le \omega \le 3$.

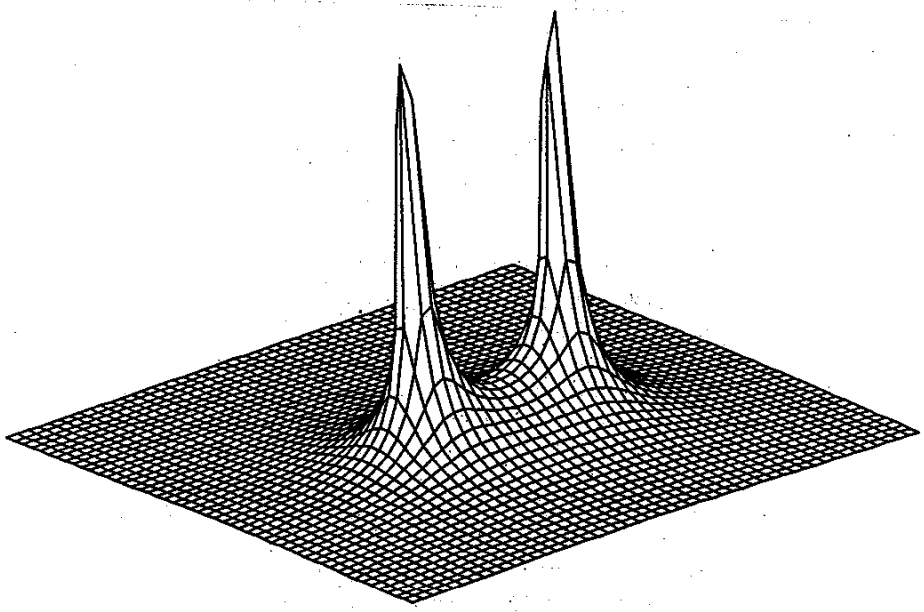


**Figure 2-8** Mesh plot of the frequency response surface. An example of 3-D plotting.

```
>> w=linspace(-3,3,50); s=linspace(-3,3,50);
>> [W,S]=meshdom(w,s);
>> re=S.^2-W.^2+S+1; im=2*S.*W+W; den=re+j*im;
>> z=ones(den)./abs(den); mesh(z)
```

Note the use of element-by-element operations for matrices. The plot is shown in Figure 2-8.

You can view mesh surfaces from specified viewpoints.

```
>> mesh(z,[azimuth  elevate])
```

allows you to specify the azimuth (horizontal rotation) and elevation angles (in degrees). Positive angle for the azimuth rotates the object clockwise, and positive value for the elevation views the object from above (90 degrees is directly overhead). The default viewing angles are [–37.5 30].