ELEC9403 Real Time Computing & Control

Linear Systems - Modelling - Revision

March, 2010

1 Introduction

Linear systems may be represented in a number of different ways. Figure 1 shows the relationship between various representations.

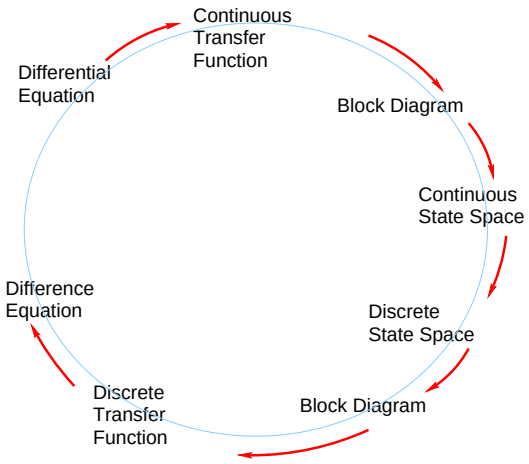

Figure 1: Relationships between different linear system representations

We will explore the relationships by following the development of different models for a representative linear system.

2 Differential Equations and Transfer Functions

Consider the linear system represented by the differential equation:

$$\frac{d^3y}{dt^3} + 0.3\frac{d^2y}{dt^2} + 0.4\frac{dy}{dt} + 0.5y = -2\frac{du}{dt} + u$$

In the time domain, the relationship between y and u is determined by convolution, but this operation is complex for higher order differential equations. By taking Laplace Transforms we can produce a frequency domain representation of the system:

$$s^3Y + 0.3s^2Y + 0.4sY + 0.5Y = -2sU + U$$

Note that in taking Laplace Transforms we have set all the initial conditions to zero. This is appropriate if we are intent on forming a transfer function representation, but would not be correct if we need to calculate responses which might be dependent on initial conditions. In the frequency domain, convolution operations are replaced by algebraic ones, and the transfer function is:

$$\frac{Y(s)}{U(s)} = \frac{-2s+1}{s^3+0.3s^2+0.4s+0.5}$$

3 Block Diagrams

In forming a block diagram, we need to separate the transfer function into polynomials. There are an infinite number of ways in which this can be done. A simple example would be to write:

$$s^3+0.3s^2+0.4s+0.5)Y = (-2s+1)U$$

However, it is much more useful to introduce another variable and write:

$$\begin{aligned} Y &= (-2s+1)e \\ e &= \frac{U}{s^3+0.3s^2+0.4s+0.5} \end{aligned}$$

Rearranging the second equation yields:

$$s^3e = U - 0.3s^2e - 0.4se - 0.5e$$

This helps us define a chain of integrators whose output is e, and we are in a position to draw a block diagram representing the process (Figure 2).

This is a *canonical* representation of the process, because the coefficient blocks correspond exactly to the coefficients in the original differential equation. A little block diagram algebra reveals that this is not necessarily always the case. This particular form is known as a *Controller Canonical Form*.

4 Continuous State Space Models

The outputs of the integrators have been labelled x_1, x_2 and x_3. These are the *state variables*. State variables represent storage or memory of some sort, and integrators are typical storage devices. We can write a first order differential equation for each state variable, simply by looking at the block diagram.

$$\frac{d}{dt}x_1 = x_2$$

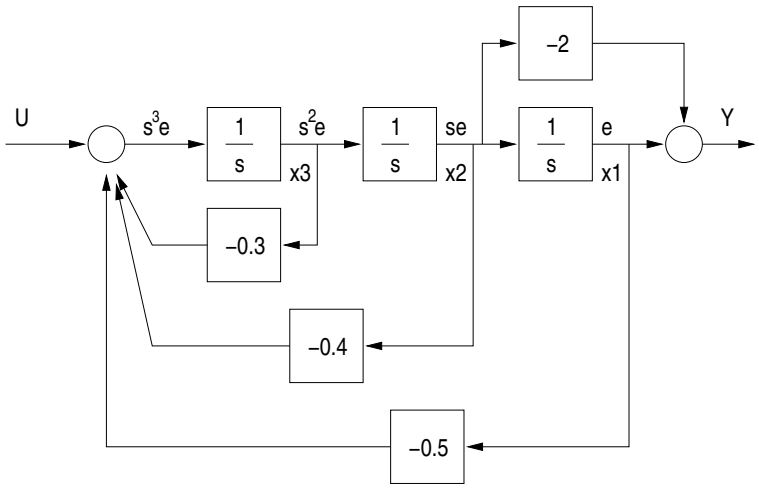

Figure 2: Block Diagram for our process

$$\begin{aligned}\frac{d}{dt}x_2 &= x_3 \\ \frac{d}{dt}x_3 &= u - 0.5x_1 - 0.4x_2 - 0.3x_3\end{aligned}$$

Together with the equation for Y (the output equation), these equations give the state-space representation for the system.

$$y = x_1 - 2x_2$$

In matrix form, this gives:

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.5 & -0.4 & -0.3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & -2 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\end{aligned}$$

This is in the familiar form (with suitable definition of $x,\ A,\ B,\ C$):

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}$$

4.1 Linearization

A system is said to be linear if it obeys the principle of superposition.

Is $\dot{x} = Ax + Bu$ linear? In accordance with assumptions, we must have:

$$\dot{x}_I = Ax_I + Bu_I$$

and

$$\dot{x}_{II} = Ax_{II} + Bu_{II}$$

Adding these two equations we obtain

$$\dot{x}_I + \dot{x}_{II} = Ax_I + Ax_{II} + Bu_I + Bu_{II}$$

This last equation can be written

$$\frac{d}{dt}(x_I + x_{II}) = A(x_I + x_{II}) + B(u_I + u_{II})$$

and as this equation tells us that the sum $x_I + x_{II}$ satisfies the given state space equation, we conclude that superposition applies.

Assume that the differential equations for the system under study may be written in the form

$$\dot{x} = f(x, u)$$

and the function $f()$ is nonlinear. The nominal state vector will be called x^0 and the corresponding actuation u^0. x^0 and u^0 are functions of time. x^0 satisfies the vector equation

$$\dot{x}^0 = f(x^0, u^0)$$

Now assume that we are slightly off the reference trajectory due to the fact that the actuation is not exactly u^0. Define x and u as

$$\begin{aligned} x &= x^0 + \delta x \\ u &= u^0 + \delta u \end{aligned}$$

x and u must also satisfy the given nonlinear system equation, i.e.

$$\frac{d}{dt}(x^0 + \delta x) = \dot{x}^0 + \delta\dot{x} = f(x^0 + \delta x, u^0 + \delta u)$$

Consider the ith component of this vector equation, and expand f_i in a Taylor series around the reference trajectory

$$\begin{aligned} \dot{x}_i^0 + \delta\dot{x}_i^0 \approx\ & f_i(x^0, u^0) + \frac{\partial f_i}{\partial x_1}\delta x_1 + \ldots + \frac{\partial f_i}{\partial x_n}\delta x_n \\ & + \frac{\partial f_i}{\partial u_1}\delta u_1 + \ldots + \frac{\partial f_i}{\partial u_m}\delta u_m \end{aligned}$$

The existence of the partial derivatives is assumed, and all will be computed along the reference trajectory. In view of the original nonlinear differential equation

$$\begin{aligned}\delta\dot{x}_i &\approx \left(\frac{\partial f_i}{\partial x_1}\right)^0 \delta x_1 + \ldots + \left(\frac{\partial f_i}{\partial x_n}\right)^0 \delta x_n \\ &\quad + \left(\frac{\partial f_i}{\partial u_1}\right)^0 \delta u_1 + \ldots + \left(\frac{\partial f_i}{\partial u_m}\right)^0 \delta u_m\end{aligned}$$

where $()^0$ implies computation of the partials along the reference trajectory. Introducing *Jacobian Matrices*

$$\begin{aligned}A &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \\ B &= \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}\end{aligned}$$

the perturbation for of the model may be written

$$\delta\dot{x} \approx A\delta x + B\delta u$$

Although the system differential equations are nonlinear, the differential equations describing perturbations around the nominal trajectory are linear to first order accuracy.

5 Discrete State Space Models

Discrete models represent *solutions* to continuous models. They can be derived from the standard convolution solution to a first order dfferential equation (i.e. the state-space model):

$$x(t) = e^{At}x(0) + \int_0^t e^{At-\tau}x(\tau)d\tau$$

In general, it becomes very difficult to evaluate the convolution integral, but the problem can be alleviated by making an assumption about how the actuation is applied to the process. A D/A converter will normally hold its value until that is changed at the next sample instant. For this very practical reason, it is usual (and also very convenient from the ease of evaluating the integral) to assume that the actuation is applied through a zero-order hold. Between sample instants, this is equivalent to $\dot{u} = 0$, so that the state-space equations can be written:

$$\frac{d}{dt}\begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ u \end{bmatrix}$$

If we consider this to be $\dot{\chi} = \mathcal{A}\chi$ it has a solution

$$\chi(t) = e^{\mathcal{A}t}$$

where t is the sampling interval. For our sample process, we would have:

$$\dot{\chi} = \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & -0.4 & -0.3 & 1 \\ \hline 0 & 0 & 0 & 0 \end{array}\right] \chi$$

We can use a numerical analysis package such as MATLAB or SCILAB to compute the matrix exponential, yielding (for $t = 1$):

$$\chi(t) = e^{\mathcal{A}}\chi(0) = \left[\begin{array}{cccc} 0.924 & 0.920 & 0.435 & 0.151 \\ -0.218 & 0.750 & 0.790 & 0.435 \\ -0.395 & -0.533 & 0.513 & 0.790 \\ 0 & 0 & 0 & 1 \end{array}\right] \chi(0)$$

This is then written in recursive form as a discrete state space equation:

$$\begin{aligned} x(k+1) &= \left[\begin{array}{ccc} 0.924 & 0.920 & 0.435 \\ -0.218 & 0.750 & 0.790 \\ -0.395 & -0.533 & 0.513 \end{array}\right] x(k) + \left[\begin{array}{c} 0.151 \\ 0.435 \\ 0.790 \end{array}\right] u(k) \\ y(k) &= \left[\begin{array}{ccc} 1 & -2 & 0 \end{array}\right] x(k) \end{aligned}$$

Note that the 4th row of the matrix exponential is equivalent to saying $u(k+1) = u(k)$, i.e. the zero order hold is invariant across the continuous-to-discrete transformation.

6 Difference Equations

The link between state space and trasnfer function representations may again be traced through a block diagram. The important thing to realise is that the operator is now the delay operator q^{-1} rather than the Laplace operator s. The delay operator defines the following relationship:

$$q^{-1}x(k) = x(k-1)$$

However, it quickly becomes necessary to use a numerical package to perform transformations between state space and transfer function models. The transfer function for our system is:

$$\frac{y(k)}{u(k)} = \frac{-0.719q^{-1} + 0.623q^{-2} + 0.931q^{-3}}{1 - 2.188q^{-1} + 2.347q^{-2} - 0.741q^{-3}}$$

Computationally, this requires the evaluation of a *difference equation* which requires storage of a short history of values of the measurement and actuation. The difference equation is:

$$y(k) = -0.719u(k-1)+0.623u(k-2)+0.931u(k-2)+2.188y(k-1)-2.347y(k-2)+0.741y(k-3)$$

Note that a dfference equation may be regarded as a *recursive solution* to the differential equation.

7 SCILAB/SCICOS Operations

SCILAB understands about polynomials, transfer functions, state space and all the rest. The operations inherent in the above sections may be carried out using the folloiwng SCILAB commands. An on-line help facility provides clear documentation for each command.

First, define a polynomial operator, s:

```
s = poly(0,'s')
```

We can define a continuous linear system, either by entering the state space matrices or by entering the transfer function, for example:

```
ctf = (-2*s + 1)/(s^3 + 0.3*s^2 + 0.4*s + 0.5)
cg = syslin('c', ctf)
```

We can obtain the state space system by:

```
css = tf2ss(cg)
```

This may give state space matrices which are not in canonical form. The alternative is to enter the state space matrices derived from the block diagram, where the matrices a, b and c have been previously entered:

```
css = syslin('c',a,b,c)
```

The calculation of the matrix exponential to get the discrete state space system is invoked (for a step interval of 1) by:

```
dss = dscr(css,1)
```

The discrete transfer function may then be obtained:

```
dg = ss2tf(dss)
```

SCICOS is a block diagram simulator, which may be used to obtain responses for simulations based on any of the system representations (transfer function or state space, continuous or discrete) or a suitable mixture of blocks using different representations.

8 Class Exercises

1. Verify the various model representations appearing in this paper using SCILAB.
2. Obtain step responses for various system representations, ensuring that all the representations are "equivalent".

9 Assignment

Using manual tuning, obtain a representation of a discrete 3-term controller for controlling this process. Use a difference form for the 3-term controller.

Check the operation of the controller with

1. a continuous transfer function simulation of the process
2. a discrete state-space simulation of the process