Canonical Forms and Similarity Transforms

Canonical Forms

Suppose you want to represent the transfer function

$$Y = \left(\frac{a_3 s^3 + a_2 s^2 + a_1 s + a_0}{s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0} \right) U$$

in state-space form

$$sX = AX + BU$$

$$Y = CX + DU$$

In the transfer function, there are six constraints. In state-space, there are 16 degrees of freedom. What this means is there are in infinite number of ways to represent the same system in state-space form. Some of these forms have names.

Controller Canonical Form

Define a dummy variable, X

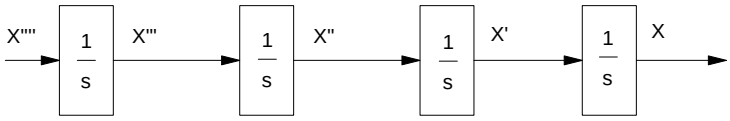
$$X = \left(\frac{1}{s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0} \right) U$$

$$Y = (a_3 s^3 + a_2 s^2 + a_1 s + a_0) X$$

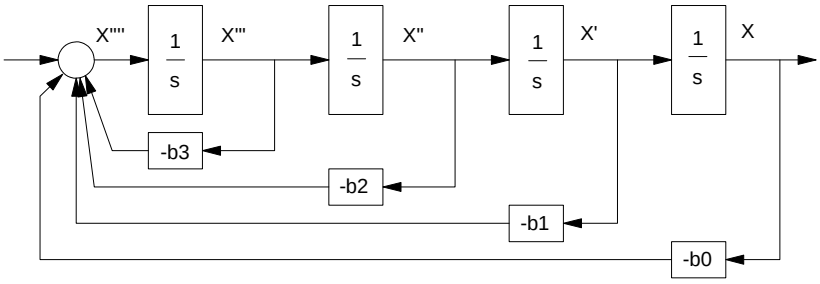
Solve for the highest derivative of X

$$s^4 X = U - b_3 s^3 X - b_2 s^2 X - b_1 s X - b_0 X$$

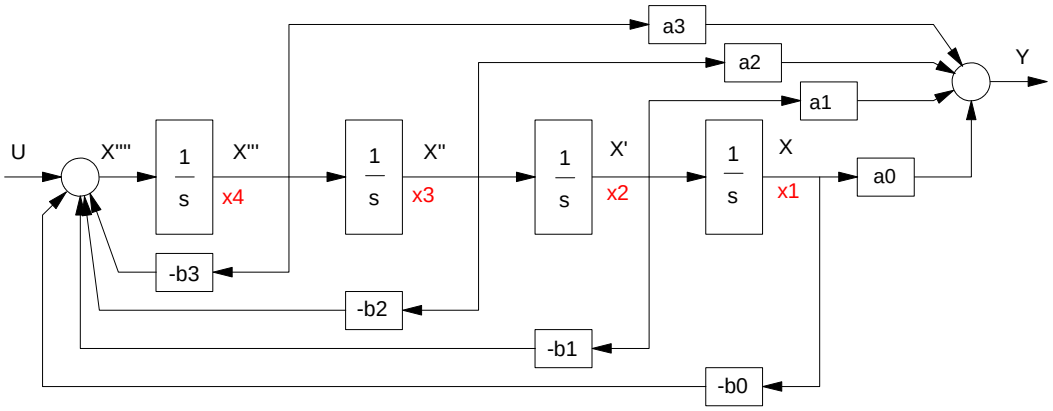
Given the 4th derivative of X, integrate four times to get X



From the dynamics, X''' is a linear combination of the input (U) and the derivatives of X:



Now that you have X and its derivatives, create Y



Block Diagram for Controller Canonical Form

This results in the following state-space form, called *controller canonical form*

$$s\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -b_0 & -b_1 & -b_2 & -b_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} U$$

$$Y = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix} X + [0]U$$

Controller canonical form has some nice properties:

- The transfer function can be found by inspection: the numerator and denominator polynomials appear in the A and C matrices
- You can control (set to any value) all of the states with input, U.

Controller canonical form also has some of the *worst* numerical properties.

Observer Canonical Form

Given a system

$$sX = AX + BU$$

$$Y = CX + DU$$

the transfer function from U to Y is

$$Y = \left(C(sI - A)^{-1}B + D \right) U$$

For a single-input single-output (SISO) system this is also

$$Y = \left(C(sI - A)^{-1}B + D \right)^T U$$

$$Y = \left(B^T(sI - A^T)^{-1}C^T + D^T \right) U$$

Another perfectly valid representation for a system is to let

$$A^T \rightarrow A$$

$$B^T \rightarrow C$$

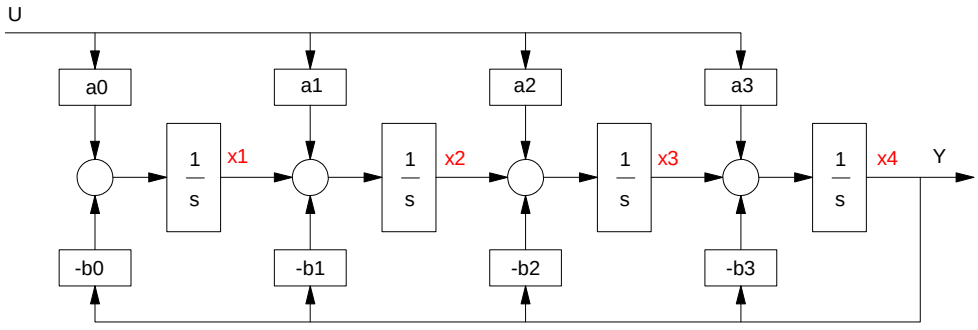
$$C^T \rightarrow B$$

For example, the 4th-order system from before becomes

$$s\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -b_0 \\ 1 & 0 & 0 & -b_1 \\ 0 & 1 & 0 & -b_2 \\ 0 & 0 & 1 & -b_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} U$$

$$Y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} X + [0]U$$

This is called *observer canonical form:* from the output (Y) you can determine all of the states through differentiation. The block-diagram representation for this system is:



Block Diagram for Observer Canonical Form

Cascade Form

If you have real poles, you can write the transfer function as

$$Y = \left(\frac{a_4+a_3(s+p_4)+a_2(s+p_3)(s+p_4)+a_1(s+p_2)(s+p_3)(s+p_4)}{(s+p_1)(s+p_2)(s+p_3)(s+p_4)} \right) U$$

For this system, you could write it as four cascaded 1st-order systems

$$x_1 = \left(\frac{1}{s+p_1} \right) U$$

$$x_2 = \left(\frac{1}{s+p_2} \right) X_1$$

$$x_3 = \left(\frac{1}{s+p_3} \right) X_2$$

$$x_4 = \left(\frac{1}{s+p_4} \right) X_3$$

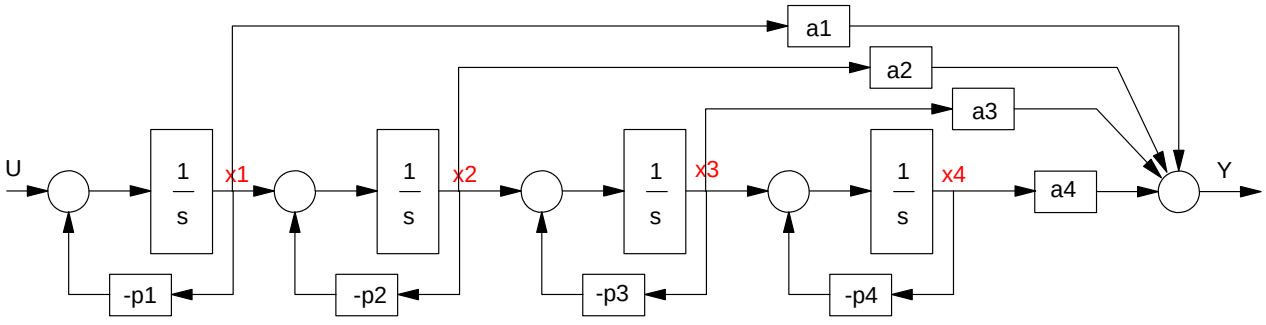
$$Y = a_4 x_4 + a_3 x_3 + a_2 x_2 + a_1 x_1$$

The state-space model is

$$s\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -p_1 & 0 & 0 & 0 \\ 1 & -p_2 & 0 & 0 \\ 0 & 1 & -p_3 & 0 \\ 0 & 0 & 1 & -p_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} U$$

$$Y = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} X$$

The block-diagram representation for this is:



Block Diagram for Cascade Canonical Form

Jordan (Diagonal) Canonical Form

If you use partial fraction expansion

$$Y = \left(\frac{a_3 s^3 + a_2 s^2 + a_1 s + a_0}{s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0} \right) U$$

becomes

$$Y = \left(\left(\frac{c_1}{s+p_1} \right) + \left(\frac{c_2}{s+p_2} \right) + \left(\frac{c_3}{s+p_3} \right) + \left(\frac{c_4}{s+p_4} \right) \right) U$$

Treat this as four coupled systems

$$x_1 = \left(\frac{c_1}{s+p_1} \right) U$$

$$x_2 = \left(\frac{c_2}{s+p_2} \right) U$$

$$x_3 = \left(\frac{c_3}{s+p_3} \right) U$$

$$x_4 = \left(\frac{c_4}{s+p_4} \right) U$$

with

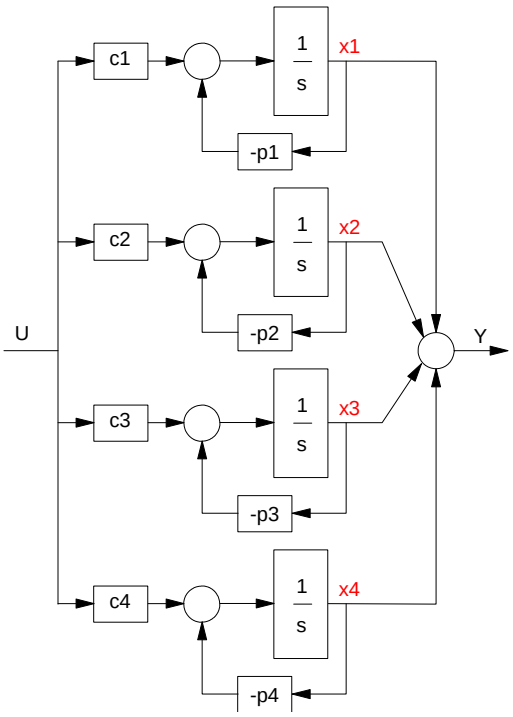
$$Y = x_1 + x_2 + x_3 + x_4$$

In state-space

$$s \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -p_1 & 0 & 0 & 0 \\ 0 & -p_2 & 0 & 0 \\ 0 & 0 & -p_3 & 0 \\ 0 & 0 & 0 & -p_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} U$$

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} X$$

with the state-space model being



Block Diagram for Jordan (Diagonal) Canonical Form

Similarity Transforms

In state-space, a dynamic system is written as

$$sX = AX + BU$$

$$Y = CX + DU$$

with the transfer function from U to Y being

$$Y = \left(C(sI - A)^{-1}B + D \right) U$$

From the previous lecture, there are several ways to put a system into state-space form, resulting in

- Controller canonical form
- Observer canonical form
- Cascade form
- Jordan form

to name a few. What is the relationship between each of these forms?

Similarity Transforms:

Let Z be a change of variable defined as

$$X = TZ$$

or

$$Z = T^{-1}X$$

where T is an NxN non-singular matrix called the *similarity transform*. Substituting for X

$$sTZ = ATZ + BU$$

$$Y = CTZ + DU$$

or

$$sZ = T^{-1}ATZ + T^{-1}BU$$

$$Y = CTZ + DU$$

You can convert from one canonical form to another using a similarity transform, T. The transformed system is

$$(A, B, C, D) \Rightarrow (T^{1}AT, T^{-1}B, CT, D)$$

The challenge is in finding the similarity transform.

Case 1: Converting to and from Jordan Form

This is the easiest transform. Almost by definition, the transformation matrix is the Eigenvector matrix

For example, convert the following system to Jordan form:

$$sX = \begin{bmatrix} -2.1 & 1 & 0 & 0 \\ 1 & -2.1 & 1 & 0 \\ 0 & 1 & -2.1 & 1 \\ 0 & 0 & 1 & -1.1 \end{bmatrix} X + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} U$$

$$Y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} X + [0] U$$

In Matlab:

```
>> A = [-2.1,1,0,0;1,-2.1,1,0;0,1,-2.1,1;0,0,1,-1.1]

   -2.1000    1.0000         0         0
    1.0000   -2.1000    1.0000         0
         0    1.0000   -2.1000    1.0000
         0         0    1.0000   -1.1000

>> B = [1;0;0;0]

     1
     0
     0
     0

>> C = [0,0,0,1]

     0     0     0     1

>> D = 0;
>> [M,N] = eig(A)

M = eigenvectors

   -0.4285   -0.6565    0.5774    0.2280
    0.6565    0.2280    0.5774    0.4285
   -0.5774    0.5774   -0.0000    0.5774
    0.2280   -0.4285   -0.5774    0.6565


N = eigenvalues

   -3.6321         0         0         0
         0   -2.4473         0         0
         0         0   -1.1000         0
         0         0         0   -0.2206
```

The similarity transform, T, is simply the eigenvector matrix:

```
>> T = M;
```

The system in state-variable Z becomes:

```
>> Az = inv(T)*A*T

   -3.6320         0         0         0
         0   -2.4470         0         0
         0         0   -1.1000         0
         0         0         0   -0.2210



>> Bz = inv(T)*B

   -0.4285
   -0.6565
    0.5774
    0.2280

>> Cz = C*T

    0.2280   -0.4285   -0.5774    0.6565

>> Dz = D

     0
```

or

$$sZ = \begin{bmatrix} -3.632 & & & \\ & -2.4470 & & \\ & & -1.1 & \\ & & & -0.22 \end{bmatrix} Z + \begin{bmatrix} -0.4285 \\ -0.6565 \\ 0.5774 \\ 0.2280 \end{bmatrix} U$$

$$Y = \begin{bmatrix} 0.2280 & -0.4285 & -0.5774 & 0.6565 \end{bmatrix} Z + [0] U$$

which is Jordan canonical form.

Note that the transfer function doesn't change:

```
>> Gx = ss(A,B,C,D);

>> zpk(Gx)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)

>> Gz = ss(Az,Bz,Cz,Dz);

>> zpk(Gz)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)
```

Case 2: Converting to Output and its Derivatives

Suppose you'd like the states to be the output and its derivatives. Let the states be

$$Z = \begin{bmatrix} y \\ y' \\ y'' \\ y''' \end{bmatrix}$$

or

$$Z = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^4 \end{bmatrix} X = T^{-1}X$$

then

```
>> T = inv([C; C*A; C*A*A;  C*A*A*A])

    1.6510    7.0300    5.3000    1.0000
    1.3100    3.2000    1.0000         0
    1.1000    1.0000         0         0
    1.0000         0         0         0

>> Az = inv(T)*A*T

         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
   -2.1570  -13.2140  -17.1600   -7.4000

>> Bz = inv(T)*B

         0
    0.0000
         0
    1.0000

>> Cz = C*T

     1     0     0     0

>> Dz = D

     0
```

This representation is convenient since each state is the output and its derivatives. Note again that the eigenvalues don't change with a similarity transform

```
>> eig(A)'

   -3.6321   -2.4473   -1.1000   -0.2206

>> eig(Az)'

   -3.6321   -2.4473   -1.1000   -0.2206
```

nor does the transfer function

```
>> Gx = ss(A,B,C,D);
```

```
>> zpk(Gx)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)


>> Gz = ss(Az,Bz,Cz,Dz);

>> zpk(Gz)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)
```

Case 3: Converting to a difference in states:

Let the states be

$$Z = \begin{bmatrix} x_1 - x_2 \\ x_2 - x_3 \\ x_3 - x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} X = T^{-1} X$$

In Matlab

```
>> Ti = [1,-1,0,0;0,1,-1,0;0,0,1,-1;0,0,0,1]

     1    -1     0     0
     0     1    -1     0
     0     0     1    -1
     0     0     0     1

>> T = inv(Ti)

     1     1     1     1
     0     1     1     1
     0     0     1     1
     0     0     0     1

>> Az = inv(T)*A*T

   -3.1      0   -1.0   -1.0
    1.0   -2.1    1.0      0
      0    1.0   -2.1      0
      0      0    1.0   -0.1

>> Bz = inv(T)*B

     1
     0
     0
     0

>> Cz = C*T

     0     0     0     1

>> Dz = D

     0
```

Again, the eigenvalues don't change with a similarity transform

```
>> eig(A)'

   -3.6321   -2.4473   -1.1000   -0.2206

>> eig(Az)'

   -3.6321   -2.4473   -1.1000   -0.2206
```

and neither does the transfer function

```
>> Gx = ss(A,B,C,D);

>> zpk(Gx)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)

>> Gz = ss(Az,Bz,Cz,Dz);

>> zpk(Gz)

                  1
-------------------------------------
(s+3.632) (s+2.447) (s+1.1) (s+0.2206)
```

Conclusion

There are an infinite many ways to represent a system in state-space. All related by a similarity transform.

Each transformed system has the same eigenvalues: how you represent the system doesn't affect how the energy in the system moves about.

It may be difficult to determine what the similarity transform is that relates two similar systems.