DIFFERENCE EQUATIONS TO STATE SPACE

Any explicit LTI difference equation (§5.1) can be converted to state-space form. In state-space form, many properties of the system are readily obtained. For example, using standard utilities (such as in Matlab), there are functions for computing the *modes* of the system (its poles), an equivalent *transfer-function* description, *stability* information, and whether or not modes are ``observable" and/or ``controllable" from any given input/output point.

Every nth order scalar (ordinary) difference equation may be reformulated as a *first order vector* difference equation. For example, consider the second-order difference equation

$$y(n) = u(n) + 2u(n-1) + 3u(n-2) - \frac{1}{2}y(n-1) - \frac{1}{3}y(n-2). \tag{G.7}$$

We may define a vector first-order difference equation--the ``state space representation"--as discussed in the following sections.

CONVERTING TO STATE-SPACE FORM BY HAND

Converting a digital filter to state-space form is easy because there are various ``canonical forms" for state-space models which can be written by inspection given the strictly proper transfer-function coefficients.

The canonical forms useful for transfer-function to state-space conversion are *controller canonical form* (also called *control* or *controllable canonical form*) and *observer canonical form* (or *observable canonical form*) [28, p. 80], [37]. These names come from the field of *control theory* [28] which is concerned with designing feedback laws to control the dynamics of real-world physical systems. State-space models are used extensively in the control field to model physical systems.

The name ``controller canonical form'' reflects the fact that the input signal can ``drive'' all modes (poles) of the system. In the language of control theory, we may say that all of the system poles are *controllable* from the input $u(n)$. In observer canonical form, all modes are guaranteed to be *observable*. Controllability and observability of a state-space model are discussed further in §G.7.3 below.

The following procedure converts any causal LTI digital filter into state-space form:

1. Determine the filter transfer function $H(z) = B(z)/A(z)$.
2. If $H(z)$ is not strictly proper ($b_0 \neq 0$), ``pull out'' the delay-free path to obtain a feed-through gain b_0 in parallel with a strictly proper transfer function.
3. Write down the state-space representation by inspection using controller canonical form for the strictly proper transfer function. (Or use the matlab function `tf2ss`.)

We now elaborate on these steps for the general case:

1. The general causal IIR filter

$$y(n) = b_0\,u(n) + b_1\,u(n-1) + \cdots + b_{N_b}\,u(n-N_b) \tag{G.8}$$

$$-\,a_1\,y(n-1) - \cdots - a_{N_a}\,y(n-N_a) \tag{G.9}$$

   has transfer function

$$H(z) \triangleq \frac{Y(z)}{U(z)} = \frac{b_0 + b_1\,z^{-1} + \cdots + b_{N_b}\,z^{-N_b}}{1 + a_1\,z^{-1} + \cdots + a_{N_a}\,z^{-N_a}}. \tag{G.10}$$

2. By convention, state-space descriptions handle any delay-free path from input to output via the direct-path coefficient D in Eq. (G.1). This is natural because the delay-free path does not affect the state of the system.

   A causal filter contains a delay-free path if its impulse response $h(n)$ is nonzero at time zero, *i.e.*, if $h(0) \neq 0$.[G.5] In such cases, we must ``pull out'' the delay-free path in order to implement it in parallel, setting $D = h(0) = b_0$ in the state-space model.

   In our example, one step of long division yields

$$H(z) = b_0 + \frac{(b_1 - b_0 a_1)\,z^{-1} + \cdots + (b_N - b_0 a_N)\,z^{-N}}{1 + a_1\,z^{-1} + \cdots + a_{N_a}\,z^{-N_a}},$$

$$\triangleq b_0 + \frac{\beta_1\,z^{-1} + \cdots + \beta_N\,z^{-N}}{1 + a_1\,z^{-1} + \cdots + a_{N_a}\,z^{-N_a}}, \tag{G.11}$$

   where $N \triangleq \max(N_a, N_b)$, with $a_i \triangleq 0$ for $i > N_a$, and $b_i \triangleq 0$ for $i > N_b$.

3. The controller canonical form is then easily written as follows:

$$A = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{N-1} & -a_N \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_N \end{bmatrix} \qquad D = b_0 \tag{G.12}$$

An alternate controller canonical form is obtained by applying the similarity transformation (see §G.8 below) which simply reverses the order of the state variables. Any permutation of the state variables would similarly yield a controllable form. The transpose of a controllable form is an observable form.

One might worry that choosing controller canonical form may result in unobservable modes. However, this will not happen if $B(z)$ and $A(z)$ have no common factors. In other words, if there are no pole-zero cancellations in the transfer function $H(z) = B(z)/A(z)$, then either controller or observer canonical form will yield a controllable and observable state-space model.

We now illustrate these steps using the example of Eq. (G.7):

1. The transfer function can be written, by inspection, as

$$H(z) = \frac{1 + 2z^{-1} + 3z^{-2}}{1 + \frac{1}{2}z^{-1} + \frac{1}{3}z^{-2}}. \tag{G.13}$$

2. We need to convert Eq. (G.13) to the form

$$H(z) = b_0 + \frac{\beta_1 z^{-1} + \beta_2 z^{-2}}{1 + \frac{1}{2}z^{-1} + \frac{1}{3}z^{-2}}. \tag{G.14}$$

   Obtaining a common denominator and equating numerator coefficients with Eq. (G.13) yields

$$\begin{aligned} b_0 &= 1, \\ \beta_1 &= 2 - \frac{1}{2} = \frac{3}{2}, \text{ and} \\ \beta_2 &= 3 - \frac{1}{3} = \frac{8}{3}. \end{aligned} \tag{G.15}$$

   The same result is obtained using long division (or synthetic division).

3. Finally, the controller canonical form is given by

$$\begin{aligned} A &\triangleq \begin{bmatrix} -\frac{1}{2} & -\frac{1}{3} \\ 1 & 0 \end{bmatrix} \\ B &\triangleq \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ C^T &\triangleq \begin{bmatrix} 3/2 & 8/3 \end{bmatrix} \\ D &\triangleq 1. \end{aligned} \tag{G.16}$$

CONVERTING SIGNAL FLOW GRAPHS TO STATE-SPACE FORM BY HAND

The procedure of the previous section quickly converts any *transfer function* to state-space form (specifically, controller canonical form). When the starting point is instead a signal flow graph, it is usually easier to go directly to state-space form by *labeling each delay-element output as a state variable* and writing out the state-space equations by inspection of the flow graph.

For the example of the previous section, suppose we are given Eq. (G.14) in *direct-form II* (DF-II), as shown in Fig.G.1. It is important that the filter representation be *canonical with respect to delay*, *i.e.*, that the number of delay elements equals the order of the filter. Then the third step (writing down controller canonical form by inspection) may replaced by the following more general procedure:

1. Assign a state variable to the output of each delay element (indicated in Fig.G.1).
2. Write down the state-space representation by inspection of the flow graph.

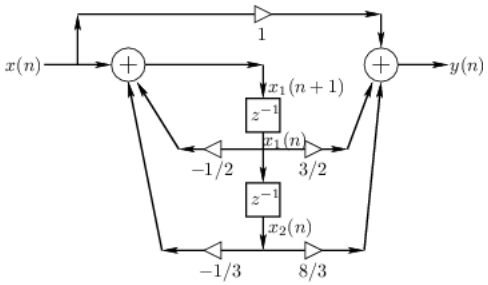

Figure: Direct-form-II representation of the difference equation in Eq. (G.7), after pulling out the direct path as given in Equations (G.14-G.15).

The state-space description of the difference equation in Eq. (G.7) is given by Eq. (G.16). We see that controller canonical form follows immediately from the direct-form-II digital filter realization, which is fundamentally an all-pole filter followed by an all-zero (FIR) filter (see §9.1.2). By starting instead from the *transposed direct-form-II* (TDF-II) structure, the *observer canonical form* is obtained [28, p. 87]. This is because the zeros effectively precede the poles in a TDF-II realization, so that they may introduce nulls in the input spectrum, but they cannot cancel output from the poles (*e.g.*, from initial conditions). Since the other two digital-filter direct forms (DF-I and TDF-I--see Chapter 9 for details) are not canonical with respect to delay, they are not used as a basis for deriving state-space models.

CONTROLLABILITY AND OBSERVABILITY

Since the output $y(n)$ in Fig.G.1 is a linear combination of the input and states $x_i(n)$, one or more poles can be *canceled* by the zeros induced by this linear combination. When that happens, the canceled modes are said to be *unobservable*. Of course, since we started with a transfer function, any pole-zero cancellations should be dealt with at that point, so that the state space realization will always be *controllable and observable*. If a mode is uncontrollable, the input cannot affect it; if it is unobservable, it has no effect on the output. Therefore, there is usually no reason to include unobservable or uncontrollable modes in a state-space model.[G.6]

A physical example of uncontrollable and unobservable modes is provided by the plucked vibrating string of an *electric guitar* with one (very thin) magnetic pick-up. In a vibrating string, considering only one plane of vibration, each quasi-harmonic[G.7] overtone corresponds to a *mode of vibration* [86] which may be modeled by a pair of complex-conjugate poles in a digital filter which models a particular point-to-point transfer function of the string. All modes of vibration having a *node* at the plucking point are *uncontrollable* at that point, and all modes having a node at the pick-up are *unobservable* at that point. If an ideal string is plucked at its midpoint, for example, all even numbered harmonics will not be excited, because they all have vibrational nodes at the string midpoint. Similarly, if the pick-up is located one-fourth of the string length from the bridge, every fourth string harmonic will be ``nulled" in the output. This is why plucked and struck strings are generally excited near one end, and why magnetic pick-ups are located near the end of the string.

A basic result in control theory is that a system in state-space form is *controllable* from a scalar input signal $u(n)$ if and only if the matrix

$$\left[B,\, AB,\, A^2B,\, \ldots,\, A^{N-1}B\right]$$

has full rank (*i.e.*, is invertible). Here, B is $N \times 1$. For the general $N \times q$ case, this test can be applied to each of the q columns of B, thereby testing controllability from each input in turn. Similarly, a state-space system is *observable* from a given output if and only if

$$\begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{N-1} \end{bmatrix}$$

is nonsingular (*i.e.*, invertible), where C is $1 \times N$. In the p-output case, C can be considered the row corresponding to the output for which observability is being checked.

A SHORT-CUT TO CONTROLLER CANONICAL FORM

When converting a transfer function to state-space form by hand, the step of pulling out the direct path, like we did in going from Eq. (G.13) to Eq. (G.14), can be bypassed [28, p. 87].

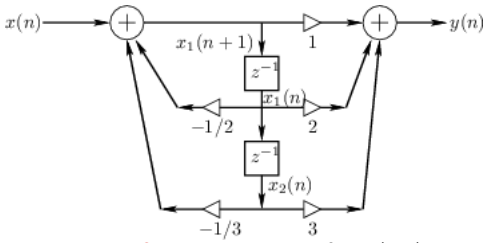

Figure: Direct-form-II realization of Eq. (G.7). Note that this form has a delay-free path from input to output.

Figure G.2 gives the standard direct-form-II structure for a second-order IIR filter. Unlike Fig.G.1, it includes a direct path from the input to the output. The filter coefficients are all given directly by the transfer function, Eq. (G.13).

This form can be converted directly to state-space form by carefully observing all paths from the input and state variables to the output. For example, $x_1(n)$ reaches the output through gain 2 on the right, but also via gain $-1/2 \cdot 1$ on the left and above. Therefore, its contribution to the output is $(2 - 1/2)x_1(n) = (3/2)x_1(n)$, as obtained in the DF-II realization with direct-path pulled out shown in Fig.G.1. The state variable $x_2(n)$ reaches the output with gain $3 - 1/3 \cdot 1 = 8/3$, again as we obtained before. Finally, it must also be observed that the gain of the direct path from input to output is 1.

MATLAB DIRECT-FORM TO STATE-SPACE CONVERSION

Matlab and Octave support state-space models with functions such as

- `tf2ss` - transfer-function to state-space conversion
- `ss2tf` - state-space to transfer-function conversion

Note that while these utilities are documented primarily for use with continuous-time systems, they are also used for discrete-time systems.

Let's repeat the previous example using Matlab:

```
>> num = [1 2 3]; % transfer function numerator
>> den = [1 1/2 1/3]; % denominator coefficients
>> [A,B,C,D] = tf2ss(num,den)

A =
   -0.5000   -0.3333
    1.0000         0

B =
     1
     0

C =  1.5000    2.6667

D =  1
```

```
>> [N,D] = ss2tf(A,B,C,D)

N = 1.0000    2.0000    3.0000

D = 1.0000    0.5000    0.3333
```

The `tf2ss` and `ss2tf` functions are documented online at The Mathworks *help desk* as well as within Matlab itself (say `help tf2ss`). In Octave, say `help tf2ss` or `help -i tf2ss`.

STATE SPACE SIMULATION IN MATLAB

Since matlab has first-class support for matrices and vectors, it is quite simple to implement a state-space model in Matlab using no support functions whatsoever, *e.g.*,

```
% Define the state-space system parameters:
A = [0 1; -1 0]; % State transition matrix
B = [0; 1];  C = [0 1];  D = 0; % Input, output, feed-around

% Set up the input signal, initial conditions, etc.
x0 = [0;0];             % Initial state (N=2)
Ns = 10;                % Number of sample times to simulate
u = [1, zeros(1,Ns-1)]; % Input signal (an impulse at time 0)
y = zeros(Ns,1);        % Preallocate output signal for n=0:Ns-1

% Perform the system simulation:
x = x0;                 % Set initial state
for n=1:Ns-1            % Iterate through time
  y(n) = C*x + D*u(n); % Output for time n-1
  x = A*x + B*u(n);    % State transitions to time n
end
y' % print the output y (transposed)
% ans =
%  0   1   0  -1   0   1   0  -1   0   0
```

The restriction to indexes beginning at 1 is unwieldy here, because we want to include time $n = 0$ in the input and output. It can be readily checked that the above examples has the transfer function

$$H(z) = \frac{z^{-1}}{1 + z^{-2}},$$

so that the following matlab checks the above output using the built-in `filter` function:

```
NUM = [0 1];
DEN = [1 0 1];
y = filter(NUM,DEN,u)
% y =
%   0   1   0  -1   0   1   0  -1   0   1
```

To eliminate the unit-sample delay, *i.e.*, to simulate $H(z) = 1/(1 + z^{-2})$ in state-space form, it is necessary to use the D (feed-around) coefficient:

```
[A,B,C,D] = tf2ss([1 0 0], [1 0 1])
% A =
%    0   1
%   -1  -0
%
% B =
%   0
%   1
%
% C =
%   -1   0
%
% D = 1

x = x0;                 % Reset to initial state
for n=1:Ns-1
  y(n) = C*x + D*u(n);
  x = A*x + B*u(n);
end
y'
% ans =
%  1   0  -1   0   1   0  -1   0   1   0
```

Note the use of trailing zeros in the first argument of `tf2ss` (the transfer-function numerator-polynomial coefficients) to make it the same length as the second argument (denominator

coefficients). This is *necessary* in `tf2ss` because the same function is used for both the continous- and discrete-time cases. Without the trailing zeros, the numerator will be extended by zeros on the *left*, *i.e.*, ``right-justified" relative to the denominator.

OTHER RELEVANT MATLAB FUNCTIONS

Related Signal Processing Toolbox functions include

- `tf2sos` -- Convert digital filter transfer function parameters to second-order sections form. (See §9.2.)
- `sos2ss` -- Convert second-order filter sections to state-space form.[G.8]
- `tf2zp` -- Convert transfer-function filter parameters to zero-pole-gain form.
- `ss2zp` -- Convert state-space model to zeros, poles, and a gain.
- `zp2ss` -- Convert zero-pole-gain filter parameters to state-space form.

In Matlab, say `lookfor state-space` to find your state-space support utilities (there are many more than listed above). In Octave, say `help -i ss2tf` and keep reading for more functions (the above list is complete, as of this writing).

MATLAB STATE-SPACE FILTER CONVERSION EXAMPLE

Here is the example of §F.6 repeated using matlab.[G.9] The difference equation

$$y(n) = u(n-1) + u(n-2) + 0.5\,y(n-1) - 0.1\,y(n-2) + 0.01\,y(n-3)$$

corresponds to the transfer function

$$H(z) = \frac{B(z)}{A(z)} = \frac{z^{-1} + z^{-2}}{1 - 0.5\,z^{-1} + 0.1\,z^{-2} - 0.01\,z^{-3}},$$

so that in matlab the filter is represented by the vectors

```
NUM = [0  1   1    0   ]; % NUM and DEN should be same length
DEN = [1 -0.5 0.1 -0.01];
```

The `tf2ss` function converts from ``transfer-function" form to state-space form:

```
 [A,B,C,D] = tf2ss(NUM,DEN)
A =
   0.00000   1.00000   0.00000
   0.00000   0.00000   1.00000
   0.01000  -0.10000   0.50000

B =
  0
  0
  1

C =
  0  1  1

D = 0
```

Next Section:
Similarity Transformations
Previous Section:
Poles of a State Space Filter