MUS420
Introduction to Linear State Space Models

Julius O. Smith III (jos@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

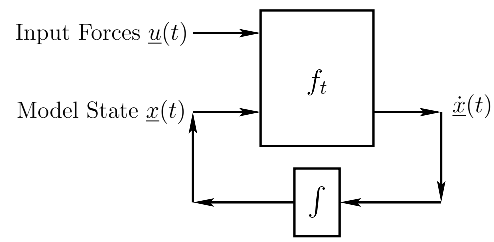
February 5, 2019

Outline

- State Space Models
- Linear State Space Formulation
- Markov Parameters (Impulse Response)
- Transfer Function
- Difference Equations to State Space Models
- Similarity Transformations
- Modal Representation (Diagonalization)
- Matlab Examples

State Space Models

Equations of motion for any physical system may be conveniently formulated in terms of its *state* $\underline{x}(t)$:



$$\dot{\underline{x}}(t) = f_t[\underline{x}(t), \underline{u}(t)]$$

where

$\underline{x}(t)$ = *state* of the system at time t
$\underline{u}(t)$ = vector of *external inputs* (typically driving forces)
f_t = general function mapping the current state $\underline{x}(t)$ and inputs $\underline{u}(t)$ to the state time-derivative $\dot{\underline{x}}(t)$

- The function f_t may be time-varying, in general
- This potentially nonlinear time-varying model is extremely general (but causal)
- Even the *human brain* can be modeled in this form

State-Space History

1. Classic *phase-space* in physics (Gibbs 1901)
   System state = point in *position-momentum space*
2. Digital computer (1950s)
3. Finite State Machines (Mealy and Moore, 1960s)
4. Finite Automata
5. State-Space Models of Linear Systems
6. Reference:
   Linear system theory: The state space approach
   L.A. Zadeh and C.A. Desoer
   Krieger, 1979

Key Property of State Vector

The key property of the state vector $\underline{x}(t)$ in the state space formulation is that it *completely determines the system at time* t

- Future states depend only on the current state $\underline{x}(t)$ and on any inputs $\underline{u}(t)$ at time t and beyond
- All past states and the entire input history are “summarized” by the current state $\underline{x}(t)$
- State $\underline{x}(t)$ includes all “memory” of the system

Force-Driven Mass Example

Consider $f = ma$ for the force-driven mass:

- Since the mass m is constant, we can use *momentum* $p(t) = m\,v(t)$ in place of velocity (more fundamental, since momentum is *conserved*)
- $x(t_0)$ and $p(t_0)$ (or $v(t_0)$) define the *state* of the mass m at time t_0
- In the absence of external forces $f(t)$, all future states are *predictable* from the state at time t_0:

$$\begin{aligned} p(t) &= p(t_0) \quad \text{(conservation of momentum)} \\ x(t) &= x(t_0) + \frac{1}{m}\int_{t_0}^{t} p(\tau)\,d\tau, \quad t \geq t_0 \end{aligned}$$

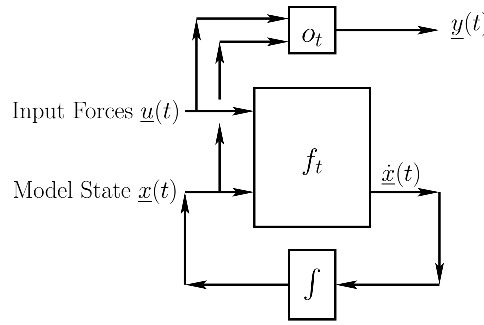
- External forces $f(t)$ *drive the state* to arbitrary points in state space:

$$\begin{aligned} p(t) &= p(t_0) + \int_{t_0}^{t} f(\tau)\,d\tau, \quad t \geq t_0, \quad p(t) \stackrel{\Delta}{=} m\,v(t) \\ x(t) &= x(t_0) + \frac{1}{m}\int_{t_0}^{t} p(\tau)\,d\tau, \quad t \geq t_0 \end{aligned}$$

Forming Outputs

Any system *output* is some function of the state, and possibly the input (directly):

$$\underline{y}(t) \stackrel{\Delta}{=} o_t[\underline{x}(t), \underline{u}(t)]$$



Usually the output is a *linear combination* of state variables and possibly the current input:

$$\underline{y}(t) \stackrel{\Delta}{=} \mathbf{C}\underline{x}(t) + \mathbf{D}\underline{u}(t)$$

where $\mathbf{C}$ and $\mathbf{D}$ are constant matrices of linear-combination coefficients

Numerical Integration

Recall the general state-space model in continuous time:

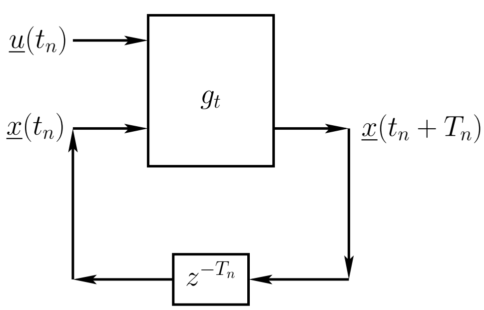
$$\dot{\underline{x}}(t) = f_t[\underline{x}(t), \underline{u}(t)]$$

An approximate discrete-time numerical solution is

$$\underline{x}(t_n + T_n) = \underline{x}(t_n) + T_n\, f_{t_n}[\underline{x}(t_n), \underline{u}(t_n)]$$

for $n = 0, 1, 2, \ldots$. (*Forward Euler*)

Let $g_{t_n}[\underline{x}(t_n), \underline{u}(t_n)] \stackrel{\Delta}{=} \underline{x}(t_n) + T_n\, f_{t_n}[\underline{x}(t_n), \underline{u}(t_n)]$:



- This is a simple example of *numerical integration* for solving the ODE
- ODE can be nonlinear and/or time-varying
- The sampling interval T_n may be fixed or adaptive

State Definition

We need a *state variable* for the amplitude of each *physical degree of freedom*

Examples:

- Ideal Mass:

  $$\text{Energy} = \frac{1}{2}mv^2 \;\Rightarrow\; \text{state variable} = v(t)$$

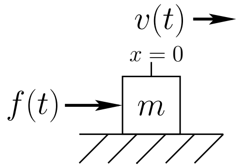
  Note that in 3D we get three state variables (v_x, v_y, v_z)
- Ideal Spring:

  $$\text{Energy} = \frac{1}{2}kx^2 \;\Rightarrow\; \text{state variable} = x(t)$$

- Inductor: Analogous to mass, so *current*
- Capacitor: Analogous to spring, so *charge* (or voltage = charge/capacitance)
- Resistors and dashpots need no state variables assigned—they are *stateless* (no "memory")

State-Space Model of a Force-Driven Mass

For the simple example of a mass m driven by external force f along the x axis:



- There is only one energy-storage element (the mass), and it stores energy in the form of *kinetic energy*
- Therefore, we should choose the state variable to be velocity $v = \dot{x}$ (or momentum $p = mv = m\dot{x}$)
- Newton's $f = ma$ readily gives the state-space formulation:

$$\boxed{\dot{v} \;=\; \frac{1}{m} f}$$

or

$$\boxed{\dot{p} \;=\; f}$$

- This is a first-order system (no vector needed)

Force-Driven Mass Reconsidered

Why not include *position* $x(t)$ as well as velocity $v(t)$ in the state-space model for the force-driven mass?

$$\begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} \;=\; \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f(t)$$

We might expect this because we know from before that the complete physical state of a mass consists of its velocity v *and* position x!

Force-Driven Mass Reconsidered and Dismissed

- *Position x does not affect stored energy*

$$E_m = \frac{1}{2} m \, v^2$$

- Velocity $v(t)$ is the only *energy-storing degree of freedom*
- Only velocity $v(t)$ is needed as a state variable
- The initial position $x(0)$ can be kept "on the side" to enable computation of the complete state in position-momentum space:

$$x(t) \;=\; x(0) + \int_0^t v(\tau)\, d\tau$$

- In other words, the position can be derived from the velocity history without knowing the force history
- Note that the external force $f(t)$ can only drive $\dot{v}(t)$. It cannot drive $\dot{x}(t)$ directly:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} \;=\; \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f(t)$$

State Variable Summary

- State variable $=$ *physical amplitude* for some *energy-storing degree of freedom*
- **Mechanical Systems:**
  State variable for each
  - *ideal spring* (linear or rotational)
  - *point mass* (or moment of inertia)

  times the number of dimensions in which it can move
- **RLC Electric Circuits:**
  State variable for each *capacitor* and *inductor*
- **In Discrete-Time:**
  State variable for each *unit-sample delay*
- **Continuous- or Discrete-Time:**
  Dimensionality of state-space $=$ *order* of the system (LTI systems)

Discrete-Time Linear State Space Models

For linear, time-invariant systems, a discrete-time *state-space model* looks like a *vector first-order finite-difference* model:

$$\begin{aligned} \underline{x}(n+1) &= \mathbf{A}\,\underline{x}(n) + \mathbf{B}\,\underline{u}(n) \\ \underline{y}(n) &= \mathbf{C}\,\underline{x}(n) + \mathbf{D}\,\underline{u}(n) \end{aligned}$$

where

- $\underline{x}(n) \in \mathbb{R}^N =$ *state vector* at time n
- $\underline{u}(n) = p \times 1$ vector of inputs
- $\underline{y}(n) = q \times 1$ output vector
- $\mathbf{A} = N \times N$ *state transition matrix*
- $\mathbf{B} = N \times p$ *input coefficient matrix*
- $\mathbf{C} = q \times N$ *output coefficient matrix*
- $\mathbf{D} = q \times p$ *direct path coefficient matrix*

The state-space representation is especially powerful for

- *multi-input, multi-output* (MIMO) linear systems
- *time-varying* linear systems
  (every matrix can have a time subscript n)

Zero-State Impulse Response (Markov Parameters)

Linear State-Space Model:

$$\begin{aligned} \underline{y}(n) &= \mathbf{C}\underline{x}(n) + \mathbf{D}\underline{u}(n) \\ \underline{x}(n+1) &= \mathbf{A}\underline{x}(n) + \mathbf{B}\underline{u}(n) \end{aligned}$$

The zero-state *impulse response* of a state-space model is easily found by direct calculation: Let $\underline{x}(0) \stackrel{\Delta}{=} \underline{0}$ and $\underline{u} = \mathbf{I}_p\delta(n) = \text{diag}(\delta(n),\ldots,\delta(n))$. Then

$$\begin{aligned} \mathbf{h}(0) &= \mathbf{C}\underline{x}(0)\mathbf{B} + \mathbf{D}\,\mathbf{I}_p\delta(0) = \mathbf{D} \\ \underline{x}(1) &= \mathbf{A}\,\underline{x}(0) + \mathbf{B}\,\mathbf{I}_p\delta(0) = \mathbf{B} \\ \mathbf{h}(1) &= \mathbf{CB} \\ \underline{x}(2) &= \mathbf{A}\,\underline{x}(1) + \mathbf{B}\,\delta(1) = \mathbf{AB} \\ \mathbf{h}(2) &= \mathbf{CAB} \\ \underline{x}(3) &= \mathbf{A}\,\underline{x}(1) + \mathbf{B}\,\delta(1) = \mathbf{A}^2\mathbf{B} \\ \mathbf{h}(3) &= \mathbf{CA}^2\mathbf{B} \\ &\vdots \\ \mathbf{h}(n) &= \boxed{\mathbf{CA}^{n-1}\mathbf{B}}, \quad n > 0 \end{aligned}$$

Zero-State Impulse Response (Markov Parameters)

Thus, the "impulse response" of the state-space model can be summarized as

$$\boxed{\mathbf{h}(n) = \begin{cases} \mathbf{D}, & n = 0 \\ \mathbf{CA}^{n-1}\mathbf{B}, & n > 0 \end{cases}}$$

- Initial state $\underline{x}(0)$ assumed to be $\underline{0}$
- Input "impulse" is $\underline{u} = \mathbf{I}_p\delta(n) = \text{diag}(\delta(n),\ldots,\delta(n))$
- Each "impulse-response sample" $\mathbf{h}(n)$ is a $p \times q$ matrix, in general
- The impulse-response terms $\mathbf{CA}^n\mathbf{B}$ for $n \geq 0$ are called *Markov parameters*

Linear State-Space Model Transfer Function

- Recall the linear state-space model:

$$\begin{aligned} \underline{y}(n) &= \mathbf{C}\,\underline{x}(n) + \mathbf{D}\,\underline{u}(n) \\ \underline{x}(n+1) &= \mathbf{A}\,\underline{x}(n) + \mathbf{B}\,\underline{u}(n) \end{aligned}$$

  and its "impulse response"

$$\mathbf{h}(n) = \begin{cases} \mathbf{D}, & n = 0 \\ \mathbf{CA}^{n-1}\mathbf{B}, & n > 0 \end{cases}$$

- The *transfer function* is the z transform of the impulse response:

$$\begin{aligned} \mathbf{H}(z) \stackrel{\Delta}{=} \sum_{n=0}^{\infty}\mathbf{h}(n)z^{-n} &= \mathbf{D} + \sum_{n=1}^{\infty}\left(\mathbf{CA}^{n-1}\mathbf{B}\right)z^{-n} \\ &= \mathbf{D} + z^{-1}\mathbf{C}\left[\sum_{n=0}^{\infty}\left(z^{-1}\mathbf{A}\right)^n\right]\mathbf{B} \end{aligned}$$

  The closed-form sum of a matrix geometric series gives

$$\boxed{\mathbf{H}(z) = \mathbf{D} + \mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B}}$$

  (a $p \times q$ *matrix* of rational polynomials in z)

- If there are p inputs and q outputs, then $\mathbf{H}(z)$ is a $p \times q$ *transfer-function matrix* (or "matrix transfer function")
- Given transfer-function coefficients, many digital filter *realizations* are possible (different computing structures)

Example: ($p = 3$, $q = 2$)

$$\mathbf{H}(z) = \begin{bmatrix} z^{-1} & \dfrac{1-z^{-1}}{1-0.5z^{-1}} & 1+z^{-1} \\ \dfrac{2+3z^{-1}}{1-0.1z^{-1}} & \dfrac{1+z^{-1}}{1-z^{-1}} & \dfrac{(1-z^{-1})^2}{(1-0.1z^{-1})(1-0.2z^{-1})} \end{bmatrix}$$

System Poles

Above, we found the transfer function to be

$$\mathbf{H}(z) = \mathbf{D} + \mathbf{C}\,(z\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}$$

The poles of $\mathbf{H}(z)$ are the same as those of

$$H_p(z) \stackrel{\Delta}{=} (z\mathbf{I} - \mathbf{A})^{-1}$$

By *Cramer's rule* for matrix inversion, the denominator polynomial for $(z\mathbf{I} - \mathbf{A})^{-1}$ is given by the *determinant*:

$$d(z) \stackrel{\Delta}{=} |z\mathbf{I} - \mathbf{A}|$$

where $|\mathbf{Q}|$ denotes the *determinant* of the square matrix $\mathbf{Q}$ (also written as $\det(\mathbf{Q})$.)

- In linear algebra, the polynomial $d(z) = |z\mathbf{I} - \mathbf{A}|$ is called the *characteristic polynomial* for the matrix $\mathbf{A}$
- The roots of the characteristic polynomial are called the *eigenvalues* of $\mathbf{A}$
- Thus, the *eigenvalues* of the state transition matrix $\mathbf{A}$ are the system *poles*
- Each *mode of vibration* gives rise to a pole pair

Initial-Condition Response

Going back to the time domain, we have the linear discrete-time state-space model

$$\begin{aligned} \underline{y}(n) &= \mathbf{C}\,\underline{x}(n) + \mathbf{D}\,\underline{u}(n) \\ \underline{x}(n+1) &= \mathbf{A}\,\underline{x}(n) + \mathbf{B}\,\underline{u}(n) \end{aligned}$$

and its "impulse response"

$$\mathbf{h}(n) = \begin{cases} \mathbf{D}, & n = 0 \\ \mathbf{C}\mathbf{A}^{n-1}\mathbf{B}, & n > 0 \end{cases}$$

Given zero inputs and initial state $\underline{x}(0) \neq \underline{0}$, we get

$$\underline{y}_x(n) = \mathbf{C}\mathbf{A}^n \underline{x}(0), \quad n = 0, 1, 2, \ldots.$$

By *superposition* (for LTI systems), the *complete response* of a linear system is given by the sum of its *forced response* (such as the impulse response) and its *initial-condition response*

Difference Equation to State Space Form

A digital filter is often specified by its *difference equation* (Direct Form I). Second-order example:

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

Every nth order *difference equation* can be reformulated as a *first order vector* difference equation called the "state space" (or "state variable") representation:

$$\begin{aligned} \underline{x}(n+1) &= \mathbf{A}\,\underline{x}(n) + \mathbf{B}\,u(n) \\ y(n) &= \mathbf{C}\,\underline{x}(n) + \mathbf{D}\,u(n) \end{aligned}$$

For the above example, we have, as we'll show,

$$\mathbf{A} \stackrel{\Delta}{=} \begin{bmatrix} -\frac{1}{2} & -\frac{1}{3} \\ 1 & 0 \end{bmatrix} \quad \text{(state transition matrix)}$$

$$\mathbf{B} \stackrel{\Delta}{=} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{(matrix routing input to state variables)}$$

$$\mathbf{C} \stackrel{\Delta}{=} \begin{bmatrix} 3/2 \\ 8/3 \end{bmatrix} \quad \text{(output linear-combination matrix)}$$

$$\mathbf{D} \stackrel{\Delta}{=} 1 \quad \text{(direct feedforward coefficient)}$$

Converting to State-Space Form by Hand

1. First, determine the filter transfer function $\mathbf{H}(z)$. In the example, the transfer function can be written, by inspection, as

$$\mathbf{H}(z) = \frac{1 + 2z^{-1} + 3z^{-2}}{1 + \frac{1}{2}z^{-1} + \frac{1}{3}z^{-2}}$$

2. If $\mathbf{h}(0) \neq 0$, we must "pull out" the parallel delay-free path:

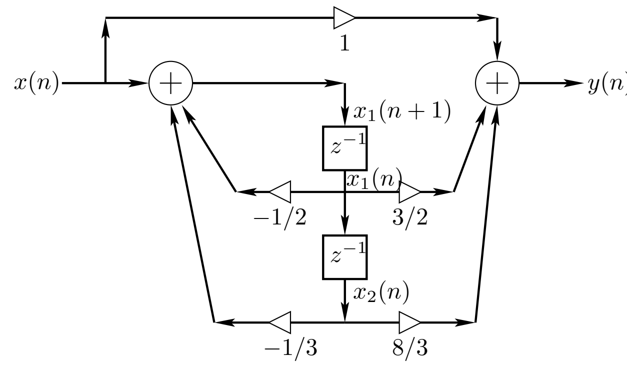
$$\mathbf{H}(z) = d_0 + \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + \frac{1}{2}z^{-1} + \frac{1}{3}z^{-2}}$$

Obtaining a common denominator and equating numerator coefficients yields

$$\begin{aligned} d_0 &= 1 \\ b_1 &= 2 - \frac{1}{2} = \frac{3}{2} \\ b_2 &= 3 - \frac{1}{3} = \frac{8}{3} \end{aligned}$$

The same result is obtained using long or synthetic division

3. Next, draw the strictly causal part in *direct form II*, as shown below:



It is important that the filter representation be *canonical with respect to delay*, i.e., the number of delay elements equals the order of the filter

4. Assign a state variable to the output of each delay element (see figure)

5. Write down the state-space representation by inspection. (Try it and compare to answer above.)

Matlab Conversion from Direct-Form to State-Space Form

Matlab has extensive support for state-space models, such as

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

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

Let's repeat the previous example using Matlab:

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
```

Matlab Documentation

The `tf2ss` and `ss2tf` functions are documented at
`http://www.mathworks.com/access/helpdesk/help/toolbox/signal/tf2ss.shtml`
as well as within Matlab itself (e.g., `help tf2ss`).

Related Signal Processing Toolbox functions include

- `tf2sos` — Convert digital filter transfer function parameters to second-order sections form.
- `sos2ss` — Convert second-order filter sections to state-space form.
- `tf2zp` — Convert transfer function filter parameters to zero-pole-gain form.
- `zp2ss` — Convert zero-pole-gain filter parameters to state-space form.

Similarity Transformations

A *similarity transformation* of a state-space system is a *linear change of state variable coordinates*:

$$\underline{x}(n) \stackrel{\Delta}{=} \mathbf{E}\underline{\tilde{x}}(n)$$

where

- $\underline{x}(n)$ = original state vector
- $\underline{\tilde{x}}(n)$ = state vector in *new coordinates*
- $\mathbf{E}$ = any *invertible* (one-to-one) matrix (linear transformation)

Substituting $\underline{x}(n) = \mathbf{E}\underline{\tilde{x}}(n)$ gives

$$\begin{aligned}\mathbf{E}\underline{\tilde{x}}(n+1) &= \mathbf{A}\,\mathbf{E}\underline{\tilde{x}}(n) + \mathbf{B}\underline{u}(n)\\ \underline{y}(n) &= \mathbf{C}\mathbf{E}\underline{\tilde{x}}(n) + \mathbf{D}\underline{u}(n)\end{aligned}$$

Premultiplying the first equation above by $\mathbf{E}^{-1}$ gives

$$\begin{aligned}\underline{\tilde{x}}(n+1) &= \left(\mathbf{E}^{-1}\mathbf{A}\mathbf{E}\right)\underline{\tilde{x}}(n) + \left(\mathbf{E}^{-1}\mathbf{B}\right)\underline{u}(n)\\ \underline{y}(n) &= (\mathbf{C}\mathbf{E})\,\underline{\tilde{x}}(n) + \mathbf{D}\underline{u}(n)\end{aligned}$$

Define the *transformed system matrices* by

$$\begin{aligned}\tilde{\mathbf{A}} &= \mathbf{E}^{-1}\mathbf{A}\mathbf{E}\\ \tilde{\mathbf{B}} &= \mathbf{E}^{-1}\mathbf{B}\\ \tilde{\mathbf{C}} &= \mathbf{C}\mathbf{E}\\ \tilde{\mathbf{D}} &= \mathbf{D}\end{aligned}$$

We can now write

$$\begin{aligned}\underline{\tilde{x}}(n+1) &= \tilde{\mathbf{A}}\underline{\tilde{x}}(n) + \tilde{\mathbf{B}}\underline{u}(n)\\ \underline{y}(n) &= \tilde{\mathbf{C}}\underline{\tilde{x}}(n) + \mathbf{D}\underline{u}(n)\end{aligned}$$

The transformed system describes the *same system* in new state-variable coordinates

Let's verify that the transfer function has not changed:

$$\begin{aligned}\tilde{\mathbf{H}}(z) &= \tilde{\mathbf{D}} + \tilde{\mathbf{C}}(z\mathbf{I} - \tilde{\mathbf{A}})^{-1}\tilde{\mathbf{B}}\\ &= \mathbf{D} + (\mathbf{C}\mathbf{E})\left(z\mathbf{I} - \mathbf{E}^{-1}\mathbf{A}\mathbf{E}\right)^{-1}(\mathbf{E}^{-1}\mathbf{B})\\ &= \mathbf{D} + \mathbf{C}\left[\mathbf{E}\left(z\mathbf{I} - \mathbf{E}^{-1}\mathbf{A}\mathbf{E}\right)\mathbf{E}^{-1}\right]^{-1}\mathbf{B}\\ &= \mathbf{D} + \mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} = \mathbf{H}(z)\end{aligned}$$

- Since the eigenvalues of $\mathbf{A}$ are the poles of the system, it follows that the eigenvalues of $\tilde{\mathbf{A}} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}$ are the same. In other words, eigenvalues are unaffected by a similarity transformation.
- The transformed Markov parameters, $\tilde{\mathbf{C}}\tilde{\mathbf{A}}^n\tilde{\mathbf{B}}$, are also unchanged since they are given by the inverse z transform of the transfer function $\tilde{\mathbf{H}}(z)$. However, it is also easy to show this by direct calculation.

State Space Modal Representation

Diagonal state transition matrix = *modal representation*:

$$\begin{bmatrix} x_1(n+1)\\ x_2(n+1)\\ \vdots\\ x_{N-1}(n+1)\\ x_N(n+1)\end{bmatrix} = \begin{bmatrix}\lambda_1 & 0 & 0 & \cdots & 0\\ 0 & \lambda_2 & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \lambda_{N-1} & 0\\ 0 & 0 & 0 & 0 & \lambda_N\end{bmatrix}\begin{bmatrix} x_1(n)\\ x_2(n)\\ \vdots\\ x_{N-1}(n)\\ x_N(n)\end{bmatrix} + \begin{bmatrix} b_1\\ b_2\\ \vdots\\ b_{N-1}\\ b_N\end{bmatrix} u(n)$$

$$y(n) = \mathbf{C}\underline{x}(n) + \mathbf{D}u(n)$$

(always possible when there are no repeated poles)

The N complex modes are *decoupled*:

$$\begin{aligned}x_1(n+1) &= \lambda_1 x_1(n) + b_1 u(n)\\ x_2(n+1) &= \lambda_2 x_2(n) + b_2 u(n)\\ &\vdots\\ x_N(n+1) &= \lambda_N x_N(n) + b_N u(n)\\ y(n) &= c_1 x_1(n) + c_2 x_2(n) + \cdots + c_N x_N(n) + \mathbf{D}u(n)\end{aligned}$$

That is, the diagonal state-space system consists of N *parallel one-pole systems*:

$$\begin{aligned}\mathbf{H}(z) &= \mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\\ &= \boxed{\mathbf{D} + \sum_{i=1}^{N}\frac{c_i b_i z^{-1}}{1 - \lambda_i z^{-1}}}\end{aligned}$$

Finding the (Diagonalized) Modal Representation

The ith *eigenvector* $\underline{e}_i$ of a matrix $\mathbf{A}$ has the defining property

$$\mathbf{A}\underline{e}_i = \lambda_i \underline{e}_i,$$

where λ_i is the associated *eigenvalue*. Thus, the eigenvector $\underline{e}_i$ is *invariant* under the linear transformation $\mathbf{A}$ to within a (generally complex) scale factor λ_i.

An $N \times N$ matrix $\mathbf{A}$ typically has N eigenvectors.[1] Let's make a similarity-transformation matrix $\mathbf{E}$ out of the N eigenvectors:

$$\mathbf{E} = \left[\, \underline{e}_1 \;\; \underline{e}_2 \;\; \cdots \;\; \underline{e}_N \,\right]$$

Then we have

$$\mathbf{AE} = \left[\, \lambda_1\underline{e}_1 \;\; \lambda_2\underline{e}_2 \;\; \cdots \;\; \lambda_N\underline{e}_N \,\right] \stackrel{\Delta}{=} \mathbf{E\Lambda}$$

where $\mathbf{\Lambda} \stackrel{\Delta}{=} \text{diag}(\underline{\lambda})$ is a diagonal matrix having $\underline{\lambda} \stackrel{\Delta}{=} \left[\, \lambda_1 \;\; \lambda_2 \;\; \cdots \;\; \lambda_N \,\right]^T$ along its diagonal. Premultiplying by $\mathbf{E}^{-1}$ gives

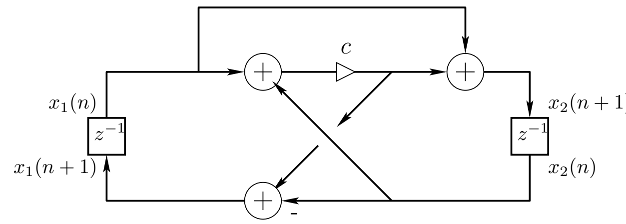
$$\boxed{\mathbf{E}^{-1}\mathbf{AE} = \mathbf{\Lambda}}$$

Thus, $\mathbf{E} = \left[\, \underline{e}_1 \;\; \underline{e}_2 \;\; \cdots \;\; \underline{e}_N \,\right]$ is a similarity transformation that *diagonalizes the system*.

[1] When there are repeated eigenvalues, there may be only one linearly independent eigenvector for the repeated group. We will not consider this case and refer the interested reader to a Web search on "generalized eigenvectors," e.g., `http://en.wikipedia.org/wiki/Generalized_eigenvector` .

State-Space Analysis Example: The Digital Waveguide Oscillator

Let's use state-space analysis to determine the frequency of oscillation of the following system:



The second-order digital waveguide oscillator.

Note the assignments of unit-delay *outputs* to state variables $x_1(n)$ and $x_2(n)$.

We have

$$x_1(n{+}1) = c[x_1(n){+}x_2(n)]{-}x_2(n) = c\,x_1(n){+}(c{-}1)x_2(n)$$

and

$$x_2(n{+}1) = x_1(n){+}c[x_1(n){+}x_2(n)] = (1{+}c)x_1(n){+}c\,x_2(n)$$

In matrix form, the state transition can be written as

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \underbrace{\begin{bmatrix} c & c-1 \\ c+1 & c \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix}$$

or, in vector notation,

$$\underline{x}(n+1) = \mathbf{A}\,\underline{x}(n)$$

The poles of the system are given by the eigenvalues of $\mathbf{A}$, which are the roots of its characteristic polynomial. That is, we solve

$$|\lambda_i \mathbf{I} - \mathbf{A}| = 0$$

for λ_i, $i = 1, 2, \ldots, N$, or, for our $N = 2$ problem,

$$0 = \begin{vmatrix} \lambda_i - c & 1 - c \\ -c - 1 & \lambda_i - c \end{vmatrix} = (\lambda_i{-}c)^2{+}(1{-}c)(1{+}c) = \lambda_i{}^2{-}2\lambda_i c{+}1$$

Using the quadratic formula, the two solutions are found to be

$$\lambda_i = c \pm \sqrt{c^2 - 1} = c \pm j\sqrt{1 - c^2}$$

Defining $c = \cos(\theta)$, we obtain the simple formula

$$\lambda_i = \cos(\theta) \pm j\sin(\theta) = \boxed{e^{\pm j\theta}}$$

It is now clear that the system is a real sinusoidal oscillator for $-1 \leq c \leq 1$, oscillating at normalized radian frequency $\omega_c T \stackrel{\Delta}{=} \theta \stackrel{\Delta}{=} \arccos(c) \in [-\pi, \pi]$.

We determined the frequency of oscillation $\omega_c T$ from the eigenvalues λ_i of $\mathbf{A}$. To study this system further, we can *diagonalize* $\mathbf{A}$. For that we need the eigenvectors as well as the eigenvalues.

Eigenstructure of A

The defining property of the eigenvectors $\underline{e}_i$ and eigenvalues λ_i of $\mathbf{A}$ is the relation

$$\boxed{\mathbf{A}\underline{e}_i = \lambda_i \underline{e}_i}, \quad i = 1, 2,$$

which expands to

$$\begin{bmatrix} c & c-1 \\ c+1 & c \end{bmatrix} \begin{bmatrix} 1 \\ \eta_i \end{bmatrix} = \begin{bmatrix} \lambda_i \\ \lambda_i \eta_i \end{bmatrix}.$$

- The first element of $\underline{e}_i$ is normalized arbitrarily to 1
- We have two equations in two unknowns λ_i and η_i:

$$\begin{aligned} c + \eta_i(c-1) &= \lambda_i \\ (1+c) + c\eta_i &= \lambda_i \eta_i \end{aligned}$$

(We already know λ_i from above, but this analysis will find them by a different method.)

- Substitute the first into the second to eliminate λ_i:

$$\begin{aligned} 1 + c + c\eta_i &= [c + \eta_i(c-1)]\eta_i = c\eta_i + \eta_i^2(c-1) \\ \Rightarrow\; 1 + c &= \eta_i^2(c-1) \\ \Rightarrow\; \eta_i &= \pm\sqrt{\frac{c+1}{c-1}} \end{aligned}$$

- We have found both eigenvectors:

$$\underline{e}_1 = \begin{bmatrix} 1 \\ \eta \end{bmatrix}, \quad \underline{e}_2 = \begin{bmatrix} 1 \\ -\eta \end{bmatrix}, \quad \text{where } \eta \stackrel{\Delta}{=} \sqrt{\frac{c+1}{c-1}}$$

  They are linearly independent provided $\eta \neq 0 \Leftrightarrow c \neq -1$ and finite provided $c \neq 1$.

- The eigenvalues are then

$$\lambda_i = c + \eta_i(c-1) = c \pm \sqrt{\frac{c+1}{c-1}(c-1)^2} = c \pm \sqrt{c^2-1}$$

- Assuming $|c| < 1$, they can be written as

$$\lambda_i = c \pm j\sqrt{1-c^2}$$

- With $c \in (-1,1)$, define $\theta = \arccos(c)$, i.e., $c \stackrel{\Delta}{=} \cos(\theta)$ and $\sqrt{1-c^2} = \sin(\theta)$.
- The eigenvalues become

$$\begin{aligned} \lambda_1 &= c + j\sqrt{1-c^2} = \cos(\theta) + j\sin(\theta) = e^{j\theta} \\ \lambda_2 &= c - j\sqrt{1-c^2} = \cos(\theta) - j\sin(\theta) = e^{-j\theta} \end{aligned}$$

  as expected.

We again found the explicit formula for the frequency of oscillation:

$$\omega_c = \frac{\theta}{T} = f_s \arccos(c),$$

where f_s denotes the sampling rate. Or,

$$\boxed{c = \cos(\omega_c T)}$$

The coefficient range $c \in (-1,1)$ corresponds to frequencies $f \in (-f_s/2, f_s/2)$.

We have shown that the example system oscillates sinusoidally at any desired digital frequency ω_c when $c = \cos(\omega_c T)$, where T denotes the sampling interval.

The Diagonalized Example System

We can now diagonalize our system using the similarity transformation

$$\mathbf{E} = \left[\, \underline{e}_1 \;\; \underline{e}_2 \,\right] = \begin{bmatrix} 1 & 1 \\ \eta & -\eta \end{bmatrix}$$

where $\eta = \sqrt{\frac{c+1}{c-1}}$.

We have only been working with the state-transition matrix $\mathbf{A}$ up to now.

The system has no inputs so it must be excited by initial conditions (although we could easily define one or two inputs that sum into the delay elements).

We have two natural choices of output which are the state variables $x_1(n)$ and $x_2(n)$, corresponding to the choices $\mathbf{C} = [1,0]$ and $\mathbf{C} = [0,1]$:

$$\begin{aligned} y_1(n) &\stackrel{\Delta}{=} x_1(n) = [1,0]\,\underline{x}(n) \\ y_2(n) &\stackrel{\Delta}{=} x_2(n) = [0,1]\,\underline{x}(n) \end{aligned}$$

Thus, a convenient choice of the system $\mathbf{C}$ matrix is the 2×2 identity matrix.

For the *diagonalized system* we obtain

$$\begin{aligned} \tilde{\mathbf{A}} &= \mathbf{E}^{-1}\mathbf{A}\mathbf{E} = \begin{bmatrix} e^{j\theta} & 0 \\ 0 & e^{-j\theta} \end{bmatrix} \\ \tilde{\mathbf{B}} &= \mathbf{E}^{-1}\mathbf{B} = \mathbf{0} \\ \tilde{\mathbf{C}} &= \mathbf{C}\mathbf{E} = \mathbf{E} = \begin{bmatrix} 1 & 1 \\ \eta & -\eta \end{bmatrix} \\ \tilde{\mathbf{D}} &= 0 \end{aligned}$$

where $\theta = \arccos(c)$ and $\eta = \sqrt{\frac{c+1}{c-1}}$ as derived above.

We may now view our state-output signals in terms of

the modal representation:

$$\begin{aligned} y_1(n) &= [1,0]\underline{x}(n) = [1,0]\begin{bmatrix} 1 & 1 \\ \eta & -\eta \end{bmatrix}\underline{\tilde{x}}(n) \\ &= [1,1]\underline{\tilde{x}}(n) = \lambda_1^n\,\tilde{x}_1(0) + \lambda_2^n\,\tilde{x}_2(0) \end{aligned}$$

$$\begin{aligned} y_2(n) &= [0,1]\underline{x}(n) = [0,1]\begin{bmatrix} 1 & 1 \\ \eta & -\eta \end{bmatrix}\underline{\tilde{x}}(n) \\ &= [\eta,-\eta]\underline{\tilde{x}}(n) = \eta\lambda_1^n\tilde{x}_1(0) - \eta\lambda_2^n\,\tilde{x}_2(0) \end{aligned}$$

The output signal from the first state variable $x_1(n)$ is

$$\begin{aligned} y_1(n) &= \lambda_1^n\,\tilde{x}_1(0) + \lambda_2^n\,\tilde{x}_2(0) \\ &= e^{j\omega_c nT}\tilde{x}_1(0) + e^{-j\omega_c nT}\tilde{x}_2(0) \end{aligned}$$

The *initial condition* $\underline{x}(0) = [1,0]^T$ corresponds to modal initial state

$$\underline{\tilde{x}}(0) = \mathbf{E}^{-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{-1}{2e}\begin{bmatrix} -e & -1 \\ -e & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

For this initialization, the output y_1 from the first state variable x_1 is simply

$$y_1(n) = \frac{e^{j\omega_c nT} + e^{-j\omega_c nT}}{2} = \boxed{\cos(\omega_c nT)}$$

Similarly $y_2(n)$ is proportional to $\sin(\omega_c nT)$ ("phase quadrature" output), with amplitude η.