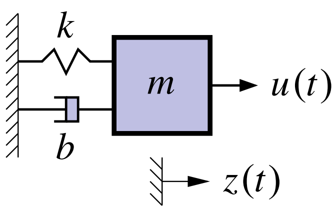
STATE-SPACE DYNAMIC SYSTEMS

2.1: Introduction to state-space systems

- Representation of the dynamics of an nth-order system as a first-order differential equation in an n-vector called the state.

  ➠ n first-order equations.

- Classic example: Second-order equation of motion.



$$m\ddot{z}(t) = u(t) - b\dot{z}(t) - kz(t)$$

$$➠ \quad \ddot{z}(t) = \frac{u(t) - b\dot{z}(t) - kz(t)}{m}.$$

- Define a (non-unique) state vector (note that $\dot{x}(t) = \mathrm{d}x(t)/\mathrm{d}t$, etc.)

$$x(t) = \begin{bmatrix} z(t) \\ \dot{z}(t) \end{bmatrix}, \quad \text{so, } \dot{x}(t) = \begin{bmatrix} \dot{z}(t) \\ \ddot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{z}(t) \\ -\dfrac{k}{m}z(t) - \dfrac{b}{m}\dot{z}(t) + \dfrac{1}{m}u(t) \end{bmatrix}.$$

- We can write this as $\dot{x}(t) = Ax(t) + Bu(t)$, where A and B are constant matrices.

$$\dot{x}(t) = \begin{bmatrix} \dot{z}(t) \\ \ddot{z}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}}_{A} \begin{bmatrix} z(t) \\ \dot{z}(t) \end{bmatrix} + \underbrace{\begin{bmatrix} \quad \\ \quad \end{bmatrix}}_{B} u(t).$$

- Complete the model by computing $z(t) = Cx(t) + Du(t)$, where C and D are constant matrices.

$$C = \begin{bmatrix} \quad & \quad \end{bmatrix}, \qquad D = \begin{bmatrix} \quad \end{bmatrix}.$$

- Fundamental form for deterministic, time-invariant, continuous-time linear state-space model:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$z(t) = Cx(t) + Du(t),$$

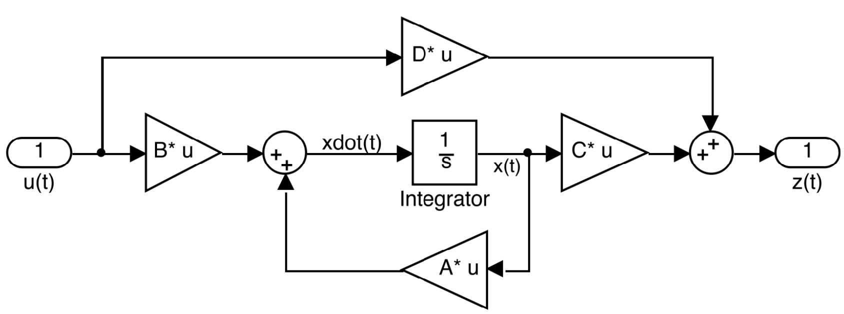
where $u(t)$ is input, $x(t)$ is the state, A, B, C, D are constant matrices.

  - Systems with noise inputs are considered in notes chapter 3.
  - Time-varying systems have A, B, C, D that change with time.

DEFINITION: The *state* of a system at time t_0 is a minimum amount of information at t_0 that, together with the input $u(t)$, $t \geq t_0$, uniquely determines the behavior of the system for all $t \geq t_0$.

- State variables provide access to what is going on *inside* the system.
- Convenient way to express equations of motion.
- Matrix format great for computers.
- Allows new analysis and synthesis tools.

SIMULATING IN SIMULINK: To investigate state-space systems, we can simulate them in Simulink. The block diagram below gives explicit access to the state and other internal signals. It is a direct implementation of the transfer function above, and the initial state may be set by setting the initial integrator values.



Example: The nearly constant position (NCP) model

- Consider a relatively immobile object that we would like to track using a Kalman filter.
- It gets bumped around by unknown forces.
- We let our model state be

$$x(t) = \begin{bmatrix} \xi(t) \\ \eta(t) \end{bmatrix},$$

  where $\xi(t)$ is the x-coordinate and $\eta(t)$ is the y-coordinate of position.

- Our model's state equation is then

$$\dot{x}(t) = 0x(t) + w(t),$$

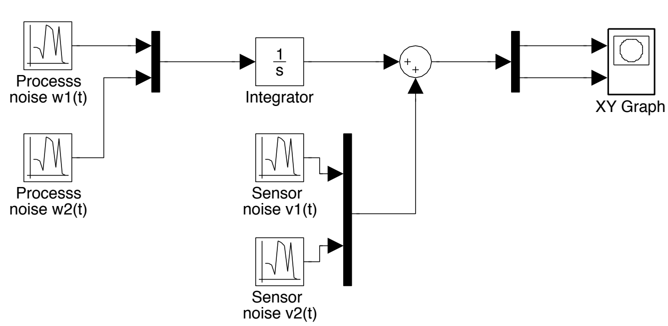
  where $w(t)$ is a random process-noise input (unlike known $u(t)$).

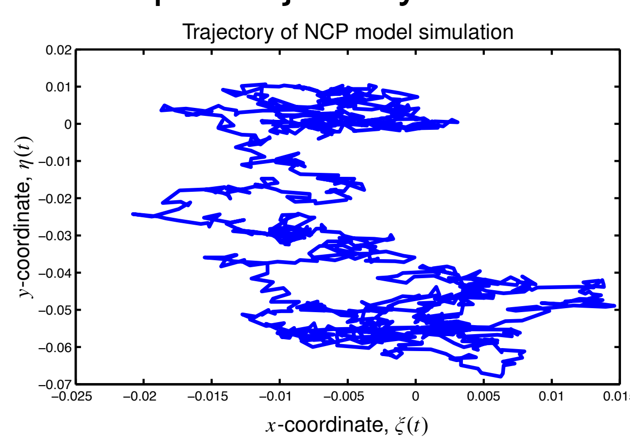
- One possible output equation is

$$z(t) = x(t) + v(t),$$

  where $v(t)$ is a random sensor-noise input.

- A possible Simulink implementation and output trajectory:





Example: The nearly constant velocity (NCV) model

- Another model we might consider is that of an object with momentum.
- The velocity is nearly constant, but gets perturbed by external forces.
- We let our model state be

$$x(t) = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \\ \eta(t) \\ \dot{\eta}(t) \end{bmatrix}.$$

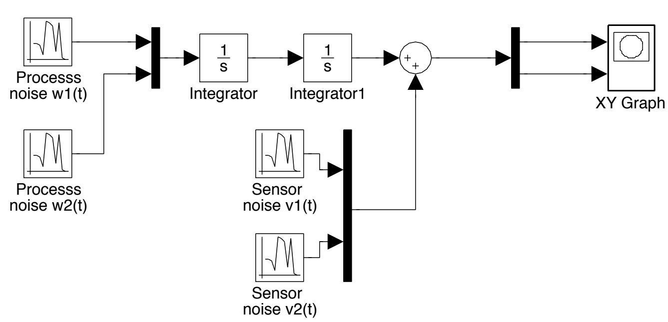
- Our model's state equation is then

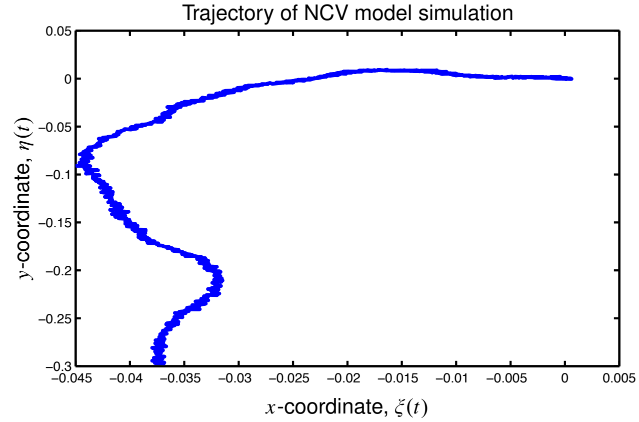
$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} w(t).$$

- One possible output equation is

$$z(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(t) + v(t).$$

- A possible Simulink implementation and output trajectory:





Example: The coordinated turn model

- A third model considers an object moving in a 2D plane with constant speed and angular rate Ω where $\Omega > 0$ is counter-clockwise motion and $\Omega < 0$ is clockwise motion.

$$\ddot{\xi}(t) = -\Omega\dot{\eta}(t) \qquad \text{and} \qquad \ddot{\eta}(t) = \Omega\dot{\xi}(t),$$

- We again let our model state be

$$x(t) = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \\ \eta(t) \\ \dot{\eta}(t) \end{bmatrix}.$$

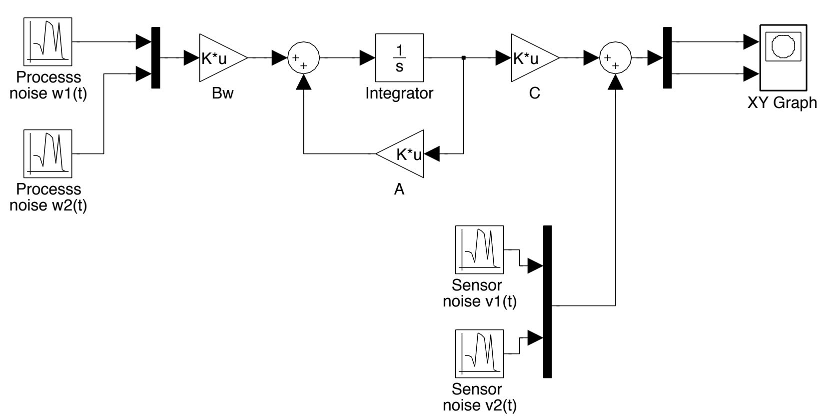
- Our model's state equation is then

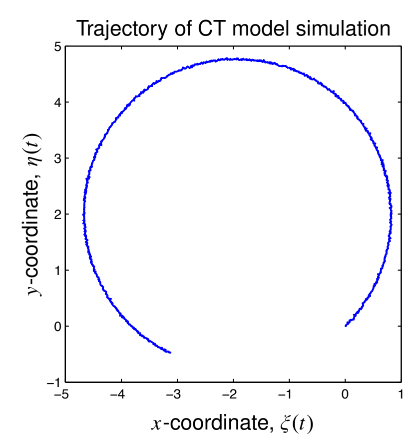
$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\Omega \\ 0 & 0 & 0 & 1 \\ 0 & \Omega & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} w(t).$$

- One possible output equation is

$$z(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(t) + v(t).$$

- A possible Simulink implementation and output trajectory:





2.2: Time (dynamic) response

- Develop more insight into the system response by looking at time-domain solution for $x(t)$.

Homogeneous part

- Start with $\dot{x}(t) = Ax(t)$ and some initial state $x(0)$.
- Take Laplace transform: $X(s) = (sI - A)^{-1}x(0)$.
- So, we have: $x(t) = \mathcal{L}^{-1}[(sI - A)^{-1}]x(0)$. But,

$$(sI - A)^{-1} = \frac{I}{s} + \frac{A}{s^2} + \frac{A^2}{s^3} + \cdots$$

so,

$$\mathcal{L}^{-1}[(sI - A)^{-1}] = I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \cdots$$

$$\triangleq e^{At} \qquad \text{matrix exponential}$$

$$x(t) = e^{At}x(0).$$

- e^{At} : "Transition matrix" or "state-transition matrix."
- In MATLAB,

```
x = expm(A*t)*x0;
```

- $e^{(A+B)t} = e^{At}e^{Bt}$ iff $AB = BA$. (*i.e.,* not in general).
- Will say more about e^{At} when we discuss the structure of A.
- Computation of $e^{At} = \mathcal{L}^{-1}[(sI - A)^{-1}]$ straightforward for 2×2.

EXAMPLE: Find e^{At} when $A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$.

- Solve

$$(sI-A)^{-1}=\begin{bmatrix} s & -1 \\ 2 & s+3 \end{bmatrix}^{-1}=\begin{bmatrix} s+3 & 1 \\ -2 & s \end{bmatrix}\frac{1}{(s+2)(s+1)}$$

$$=\begin{bmatrix} \dfrac{2}{s+1}-\dfrac{1}{s+2} & \dfrac{1}{s+1}-\dfrac{1}{s+2} \\ \dfrac{-2}{s+1}+\dfrac{2}{s+2} & \dfrac{-1}{s+1}+\dfrac{2}{s+2} \end{bmatrix}$$

$$e^{At}=\begin{bmatrix} 2e^{-t}-e^{-2t} & e^{-t}-e^{-2t} \\ -2e^{-t}+2e^{-2t} & -e^{-t}+2e^{-2t} \end{bmatrix}1(t)$$

- This is the best way to find e^{At} if A 2×2.

Forced solution

$$\dot{x}(t)=Ax(t)+Bu(t),\quad x(0)$$

$$x(t)=e^{At}x(0)+\underbrace{\int_0^t e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau}_{\text{convolution}}.$$

- Where did this come from?

1. $\dot{x}(t)-Ax(t)=Bu(t)$.
2. $e^{-At}[\dot{x}(t)-Ax(t)]=\dfrac{\mathrm{d}}{\mathrm{d}t}[e^{-At}x(t)]=e^{-At}Bu(t).$
3. $\displaystyle\int_0^t \frac{\mathrm{d}}{\mathrm{d}\tau}[e^{-A\tau}x(\tau)]\,\mathrm{d}\tau=e^{-At}x(t)-x(0)=\int_0^t e^{-A\tau}Bu(\tau)\,\mathrm{d}\tau.$

- Clearly, if $z(t)=Cx(t)+Du(t)$,

$$z(t)=\underbrace{Ce^{At}x(0)}_{\text{initial resp.}}+\underbrace{\int_0^t Ce^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau}_{\text{convolution}}+\underbrace{Du(t)}_{\text{feedthrough}}.$$

More on the matrix exponential

- Have seen the key role of e^{At} in the solution for $x(t)$. Impacts the system response, but need more insight.
- Consider what happens if the matrix A is *diagonalizable,* that is, there exists a matrix T such that $T^{-1}AT = \Lambda$ =diagonal. Then,

$$\begin{aligned} e^{At} &= I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \cdots \\ &= I + T\Lambda T^{-1}t + \frac{T\Lambda T^{-1}T\Lambda T^{-1}t^2}{2!} + \frac{T\Lambda T^{-1}T\Lambda T^{-1}T\Lambda T^{-1}t^3}{3!} + \cdots \\ &= T\left[I + \Lambda t + \frac{\Lambda^2t^2}{2!} + \frac{\Lambda^3t^3}{3!} + \cdots\right]T^{-1} = Te^{\Lambda t}T^{-1}, \end{aligned}$$

and

$$e^{\Lambda t} = \text{diag}\left(e^{\lambda_1 t},\ e^{\lambda_2 t},\ \ldots\ e^{\lambda_n t}\right).$$

- Much simpler form for the exponential, but how to find $T,\ \Lambda$?
- Write $T^{-1}AT = \Lambda$ as $T^{-1}A = \Lambda T^{-1}$ with

$$T^{-1} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix}, \qquad i.e., \text{ rows of } T^{-1}.$$

$w_i^TA = \lambda_i w_i^T$, so w_i is a *left eigenvector* of A and note that $w_i^Tv_j = \delta_{i,j}$.

- How does this help?

$$e^{At} = Te^{\Lambda t}T^{-1} = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \begin{bmatrix} e^{\lambda_1 t} & & & 0 \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ 0 & & & e^{\lambda_n t} \end{bmatrix} \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_{n^T} \end{bmatrix}$$

$$= \sum_{i=1}^{n} e^{\lambda_i t} v_i w_i^T.$$

- Very simple form, which can be used to develop intuition about dynamic response $\approx e^{\lambda_i t}$.

$$x(t) = e^{At}x(0) = Te^{\Lambda t}T^{-1}x(0) = \sum_{i=1}^{n} e^{\lambda_i t} v_i (w_i^T x(0)).$$

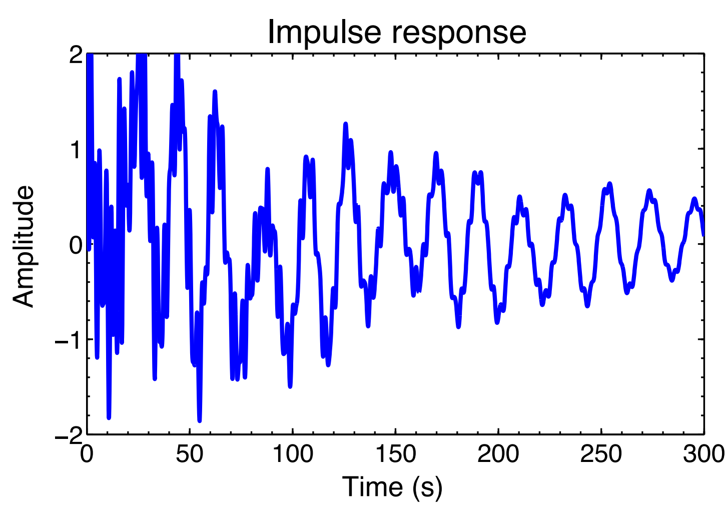
- Trajectory can be expressed as a linear combination of modes: $v_i e^{\lambda_i t}$.
- Left eigenvectors decompose $x(0)$ into modal coordinates $w_i^T x(0)$.
- $e^{\lambda_i t}$ propagates mode forward in time. Stability?
- v_i corresponds to "relative phasing" of state's part of the response.

EXAMPLE: Let's consider a specific system

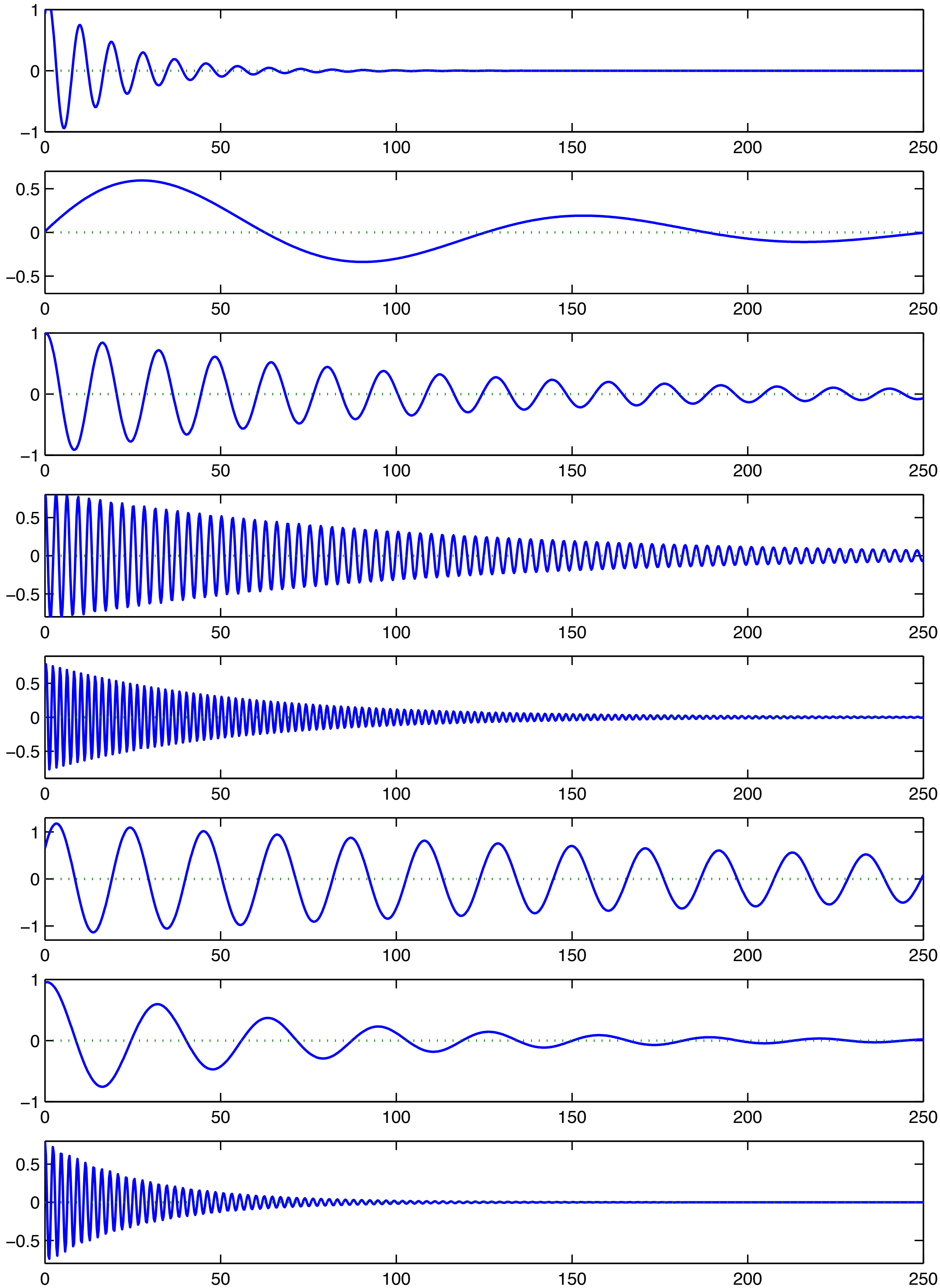
$$\dot{x}(t) = Ax(t)$$
$$z(t) = Cx(t),$$

with $x(t) \in \mathbb{R}^{16\times 1}$, $z(t) \in \mathbb{R}$ (16-state, single output).

- A lightly damped system.
- Typical output to initial conditions are shown:
- Waveform is very complicated. Looks almost random.

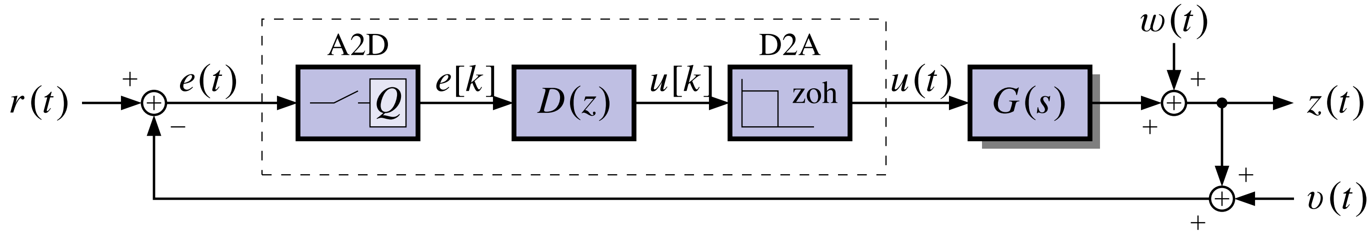


- However, the solution can be decomposed into much simpler modal components.


1
0
–1
0 50 100 150 200 250
0.5
0
–0.5
0 50 100 150 200 250
1
0
–1
0 50 100 150 200 250
0.5
0
–0.5
0 50 100 150 200 250
0.5
0
–0.5
0 50 100 150 200 250
1
0
–1
0 50 100 150 200 250
1
0
–1
0 50 100 150 200 250
0.5
0
–0.5
0 50 100 150 200 250

2.3: Discrete-time state-space systems

- Computer monitoring of real-time systems requires analog-to-digital (A2D) and digital-to-analog (D2A) conversion.



- Discrete-time systems can also be represented in state-space form

$$x_{k+1} = A_d x_k + B_d u_k$$

$$z_k = C_d x_k + D_d u_k.$$

- The subscript "d" is used here to emphasize that, in general, the "A", "B", "C" and "D" matrices are <u>different</u> for discrete-time and continuous-time systems, even if the underlying plant is the same.
- I will usually drop the "d" and expect you to interpret the system from its context.

Time (dynamic) response

- The full solution, found by induction from $x_{k+1} = Ax_k + Bu_k$, is

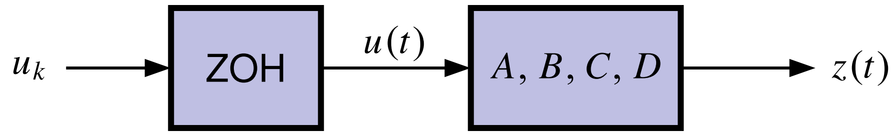
$$x_k = A^k x_0 + \underbrace{\sum_{j=0}^{k-1} A^{k-1-j} B u_j}_{\text{convolution}}.$$

- Clearly, if $z_k = Cx_k + Du_k$,

$$z_k = \underbrace{CA^k x_0}_{\text{initial resp.}} + \underbrace{\sum_{j=0}^{k-1} CA^{k-1-j} B u_j}_{\text{convolution}} + \underbrace{Du_k}_{\text{feedthrough}}.$$

Converting plant dynamics to discrete time.

- Combine the dynamics of the zero-order hold and the plant.



- The continuous-time dynamics of the plant are:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$z(t) = Cx(t) + Du(t).$$

- Evaluate $x(t)$ at discrete times. Recall

$$x(t) = \int_0^t e^{A(t-\tau)} Bu(\tau)\,\mathrm{d}\tau$$

$$x_{k+1} = x((k+1)T) = \int_0^{(k+1)T} e^{A((k+1)T-\tau)} Bu(\tau)\,\mathrm{d}\tau.$$

- With malice aforethought, break up the integral into two pieces. The first piece will become A_d times $x(kT)$. The second part will become B_d times $u(kT)$.

$$= \int_0^{kT} e^{A((k+1)T-\tau)} Bu(\tau)\,\mathrm{d}\tau + \int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)} Bu(\tau)\,\mathrm{d}\tau$$

$$= \int_0^{kT} e^{AT} e^{A(kT-\tau)} Bu(\tau)\,\mathrm{d}\tau + \int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)} Bu(\tau)\,\mathrm{d}\tau$$

$$= e^{AT} x(kT) + \int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)} Bu(\tau)\,\mathrm{d}\tau.$$

- In the remaining integral, note that $u(\tau)$ is constant from kT to $(k+1)T$, and equal to $u(kT)$.
- So, we let $\sigma = (k+1)T - \tau$; $\tau = (k+1)T - \sigma$; $\mathrm{d}\tau = -\mathrm{d}\sigma$.

$$x((k+1)T) = e^{AT}x(kT) + \left[\int_0^T e^{A\sigma}B\,\mathrm{d}\sigma\right]u(kT)$$

$$\text{or, } x_{k+1} = e^{AT}x_k + \left[\int_0^T e^{A\sigma}B\,\mathrm{d}\sigma\right]u_k.$$

- So, we have a discrete-time state-space representation from the continuous-time representation

$$x_{k+1} = A_d x_k + B_d u_k$$

where $A_d = e^{AT}$, $B_d = \int_0^T e^{A\sigma}B\,\mathrm{d}\sigma$.

- Similarly,

$$z_k = Cx_k + Du_k.$$

  - That is, $C_d = C$; $D_d = D$.

Calculating A_d, B_d, C_d and D_d

- C_d and D_d require no calculation since $C_d = C$ and $D_d = D$.
- A_d is calculated via the matrix exponential $A_d = e^{AT}$. This is different from taking the exponential of each element in AT.
- If MATLAB is handy, you can type in

```
Ad=expm(A*T)
```

- If MATLAB is not handy, then you need to work a little harder. Recall from earlier that $e^{At} = \mathcal{L}^{-1}[(sI-A)^{-1}]$. So,

$$e^{AT} = \mathcal{L}^{-1}[(sI-A)^{-1}]\Big|_{t=T},$$

which is probably the "easiest" way to work it out by hand.

- Now we focus on computing B_d. Recall that

$$\begin{aligned} B_d &= \int_0^T e^{A\sigma} B \,\mathrm{d}\sigma \\ &= \int_0^T \left(I + A\sigma + A^2 \frac{\sigma^2}{2} + \ldots \right) B \,\mathrm{d}\sigma \\ &= \left(IT + A\frac{T^2}{2!} + A^2 \frac{T^3}{3!} + \ldots \right) B \\ &= A^{-1}(e^{AT} - I)B \\ &= A^{-1}(A_d - I)B. \end{aligned}$$

- If A is invertible, this method works nicely; otherwise, we will need to perform the integral.
- Also, in MATLAB,

```
[Ad,Bd]=c2d(A,B,T)
```

2.4: Examples of discrete-time state-space models

The discrete-time NCP model

- We might consider a discrete-time version of the continuous-time nearly-constant-position model.
- Recall, in continuous time,

$$\dot{x}(t) = 0x(t) + w(t)$$

$$z(t) = x(t) + v(t).$$

- In discrete time,

$$x_{k+1} = e^{0T}x_k + \left(\int_0^T e^{0\sigma}\,\mathrm{d}\sigma\right)w_k$$

$$z_k = x_k + v_k,$$

where $e^{0T} = I$ and $\displaystyle\int_0^T I\,\mathrm{d}\sigma = TI$.

- Therefore,

$$x_{k+1} = x_k + Tw_k$$

$$z_k = x_k + v_k.$$

- Note, w_k is often scaled vis-à-vis $w(t)$ so that a commonly seen form of the discrete-time model is

$$x_{k+1} = x_k + w_k$$

$$z_k = x_k + v_k.$$

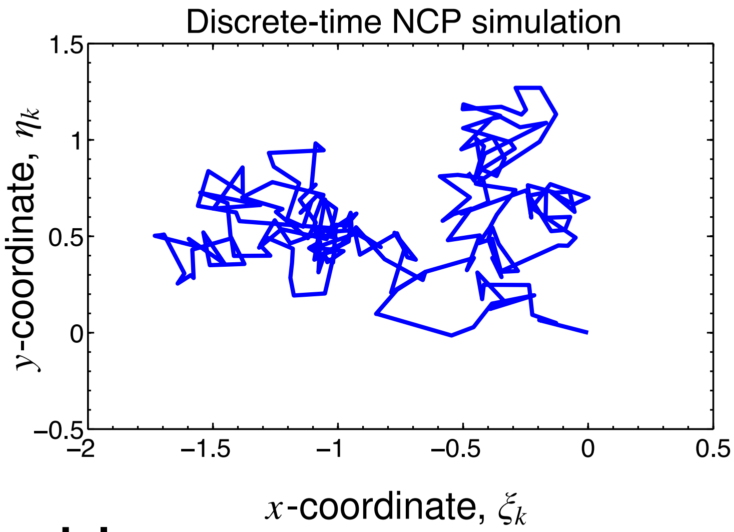
- We can use Simulink to simulate this discrete-time NCP model, much like the continuous-time NCP model.

- Or, we can also simulate it easily with a MATLAB script.

```
maxT=200;             % max. sim. time
x=zeros(2,maxT); % storage
x(:,1)=[0;0];       % initial posn.

for k=2:maxT,       % simulate model
  x(:,k)=x(:,k-1)+0.1*randn(2,1);
end

plot(x(1,:),x(2,:));
title('Discrete-time NCP sim.');
xlabel('x'); ylabel('y');
```



Example: The discrete-time NCV model

- Similarly, we might consider a discrete-time version of the continuous-time nearly-constant-velocity model.
- Recall,

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A_c} x(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{B_c} w(t).$$

- The discrete-time A matrix is $A = e^{A_c T}$

$$A = \mathcal{L}^{-1}\left\{(sI - A_c)^{-1}\right\}\Big|_{t=T} = \mathcal{L}^{-1}\left\{\begin{bmatrix} s & -1 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & -1 \\ 0 & 0 & 0 & s \end{bmatrix}^{-1}\right\}\Bigg|_{t=T}$$

$$= \mathcal{L}^{-1}\left\{\left[\begin{array}{cccc} 1/s & 1/s^2 & 0 & 0 \\ 0 & 1/s & 0 & 0 \\ 0 & 0 & 1/s & 1/s^2 \\ 0 & 0 & 0 & 1/s \end{array}\right]\right\}\Bigg|_{t=T}$$

$$= \left[\begin{array}{cccc} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{array}\right]\Bigg|_{t=T} = \left[\begin{array}{cccc} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{array}\right].$$

- This can be verified in MATLAB using the symbolic toolbox,

```
syms T
Ac = [0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];
expm(Ac*T)
```

- The discrete-time B matrix may be found as before,

$$B = \int_0^T e^{A_c\sigma} B_c \,\mathrm{d}\sigma = \left[\begin{array}{cc} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{array}\right].$$

- This can also be verified in MATLAB using the symbolic toolbox,

```
syms sigma T
Ac = [0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];
Bc = [0 0; 1 0; 0 0; 0 1];
z = expm(Ac*sigma);
B = int(z,0,T)*Bc;
```

- Alternately, we can let MATLAB do even more of the heavy lifting

```
syms T
Ac = [0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];
Bc = [0 0; 1 0; 0 0; 0 1];
[A,B] = c2d(Ac,Bc,T); % continuous to discrete
```

- Note that we often state the discrete-time NCV model in terms of a 4-vector w_k with rescaled components.
- So, the overall discrete-time NCV model is

$$x_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + w_k$$

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_k + v_k.$$

- Note, this is saying

$$\xi_k = \xi_{k-1} + T\dot{\xi}_{k-1} + \text{noise}$$

$$\eta_k = \eta_{k-1} + T\dot{\eta}_{k-1} + \text{noise}$$

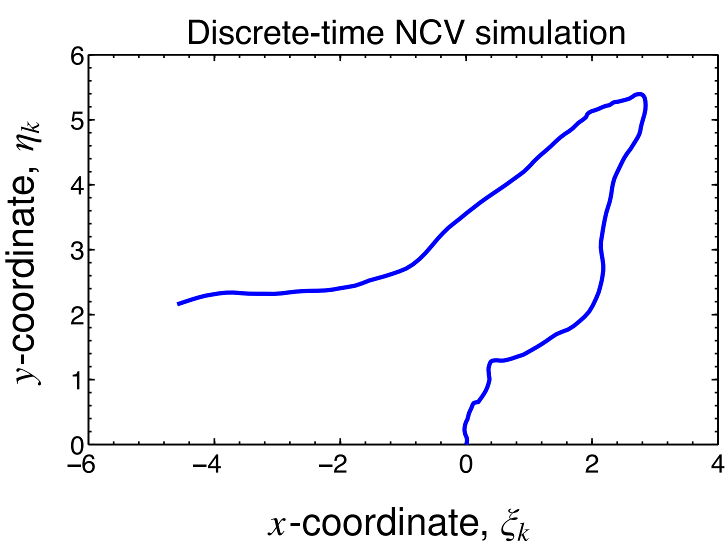
which is an NCV equation.

- We can simulate it easily with a MATLAB script.

```
maxT=200;             % max sim time
x = zeros(4,maxT); % storage
% initial position, velocity
x(:,1) = [0;0.1;0;0.1];
T = 0.1;              % sample period
A = [1 T 0 0; 0 1 0 0;
     0 0 1 T; 0 0 0 1];
B = [T^2/2 0; T 0; 0 T^2/2; 0 T];

for k=2:maxT,      % simulate model
  x(:,k)=A*x(:,k-1)+B*randn(2,1);
end

plot(x(1,:),x(3,:));
title('Discrete-time NCV sim.');
xlabel('x'); ylabel('y');
```



Example: The discrete-time coordinated-turn model

- Similarly, it can be shown that the discrete-time coordinated turn model is

$$x_k = \begin{bmatrix} 1 & \sin(\Omega T)/\Omega & 0 & (\cos(\Omega T)-1)/\Omega \\ 0 & \cos(\Omega T) & 0 & -\sin(\Omega T) \\ 0 & (1-\cos(\Omega T))/\Omega & 1 & \sin(\Omega T)/\Omega \\ 0 & \sin(\Omega T) & 0 & \cos(\Omega T) \end{bmatrix} x_{k-1}$$

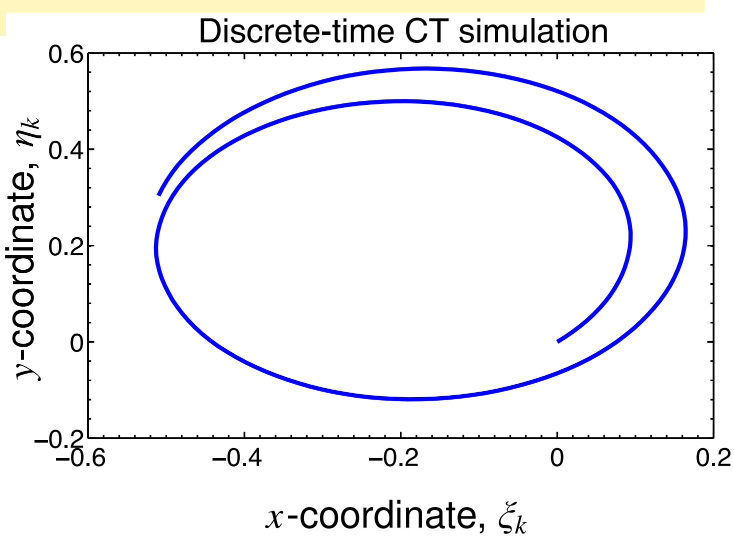
$$+ \begin{bmatrix} (1-\cos(\Omega T))/\Omega^2 & (\sin(\Omega T)-\Omega T)/\Omega^2 \\ \sin(\Omega T)/\Omega & (\cos(\Omega T)-1)/\Omega \\ (\Omega T - \sin(\Omega T))/\Omega^2 & (1-\cos(\Omega T))/\Omega^2 \\ (1-\cos(\Omega T))/\Omega & \sin(\Omega T)/\Omega \end{bmatrix} w_{k-1}.$$

- MATLAB code to implement this:

```
maxT = 200;                  % max simulation time
x = zeros(4,maxT);           % reserve storage
x(:,1) = [0;0.1;0;0.1]; % initial posn, velocity
T = 0.1; W = 0.5; WT = W*T; % Use W as Omega
A = [1 sin(WT)/W 0 (1-cos(WT))/W; 0 cos(WT) 0 -sin(WT); ...
     0 (1-cos(WT))/W 1 sin(WT)/W; 0 sin(WT) 0 cos(WT)];
B = [(1-cos(WT))/W^2, (sin(WT)-WT)/W^2; sin(WT)/W (cos(WT)-1)/W; ...
     (WT-sin(WT))/W^2, (1-cos(WT))/W^2; (1-cos(WT))/W sin(WT)/W];

for k=2:maxT, % simulate model
  x(:,k) = A*x(:,k-1) + ...
           B*0.01*randn(2,1);
end

plot(x(1,:),x(3,:));
title('Discrete-time CT sim.');
xlabel('x'); ylabel('y');
```



Comparing continuous-time and discrete-time models

- Consider again the first example of this section of notes

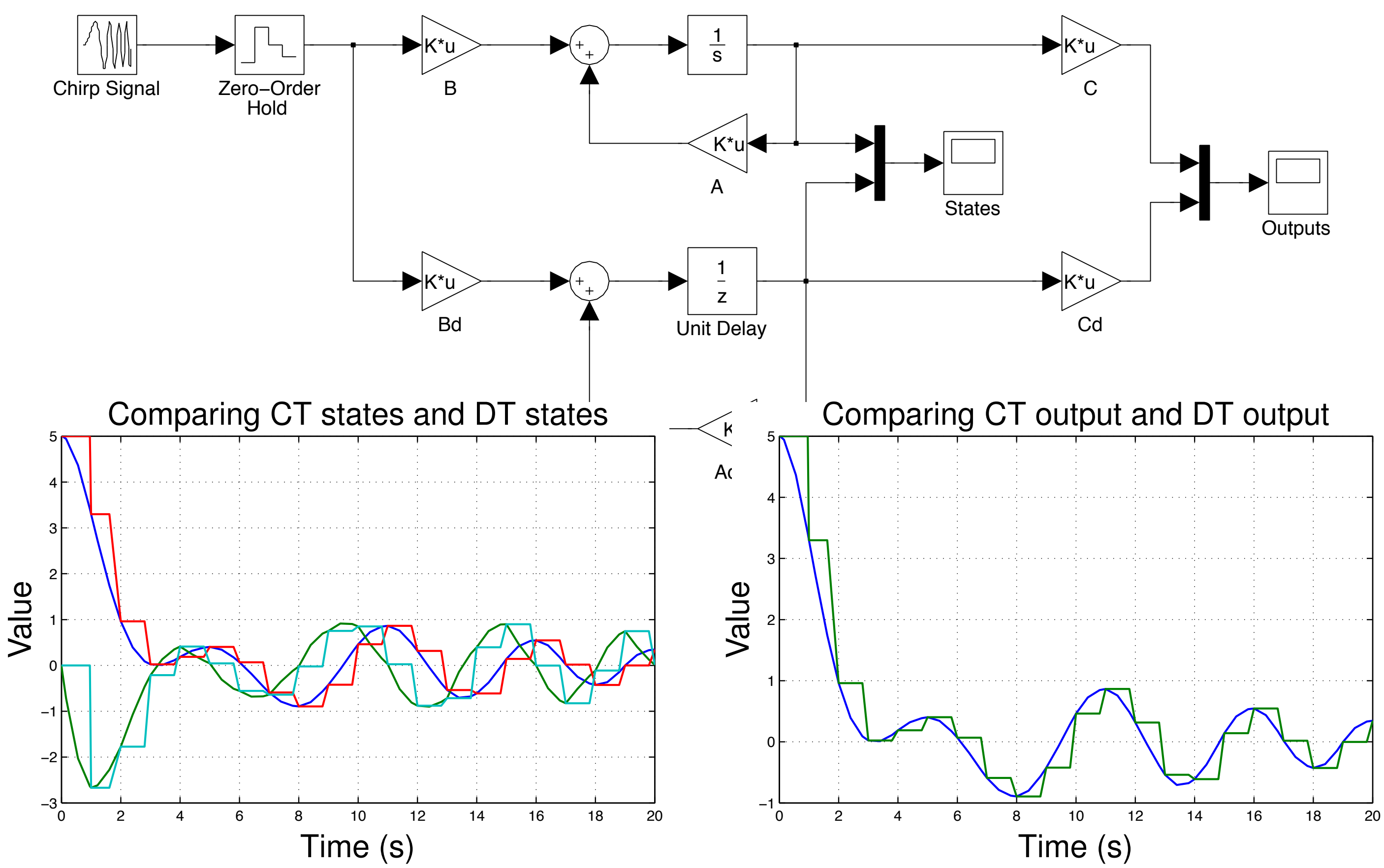
$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u(t)$$

$$z(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t).$$

- We expect agreement between continuous-time and discrete-time models *at the sampling instants*.
- For simplicity, let $k = b = m = T = 1$. We can find,

$$A_d = \begin{bmatrix} 0.6597 & 0.5335 \\ -0.5335 & 0.1262 \end{bmatrix} \quad \text{and} \quad B_d = \begin{bmatrix} 0.3403 \\ 0.5335 \end{bmatrix}.$$

- Simulate *both* systems with the same input ($u(t)$ constant over T)



2.5: Continuous-time observability and controllability

- If a system is observable, we can determine the initial condition of the state vector $x(0)$ via processing the input to the system $u(t)$ and the output of the system $z(t)$.
- Since we can simulate the system if we know $x(0)$ and $u(t)$ this also implies that we can determine $x(t)$ for $t \geq 0$.

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau.$$

- Therefore, it should not be surprising that a system must be observable for the Kalman filter to work.
- Consider the LCCODE

$$\dddot{z}(t) + a_1\ddot{z}(t) + a_2\dot{z}(t) + a_3z(t) = b_0\dddot{u}(t) + b_1\ddot{u}(t) + b_2\dot{u}(t) + b_3u(t).$$

- If we have a realization of this system in state-space form

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$z(t) = Cx(t) + Du(t),$$

and we have initial conditions $z(0)$, $\dot{z}(0)$, $\ddot{z}(0)$, how do we find $x(0)$?

$$\begin{aligned} z(0) &= Cx(0) + Du(0) \\ \dot{z}(0) &= C(\underbrace{Ax(0) + Bu(0)}_{\dot{x}(0)}) + D\dot{u}(0) \\ &= CAx(0) + CBu(0) + D\dot{u}(0) \\ \ddot{z}(0) &= CA^2x(0) + CABu(0) + CB\dot{u}(0) + D\ddot{u}(0). \end{aligned}$$

- In general,

$$z^{(k)}(0) = CA^kx(0) + CA^{k-1}Bu(0) + \cdots + CBu^{(k-1)}(0) + Du^{(k)}(0),$$

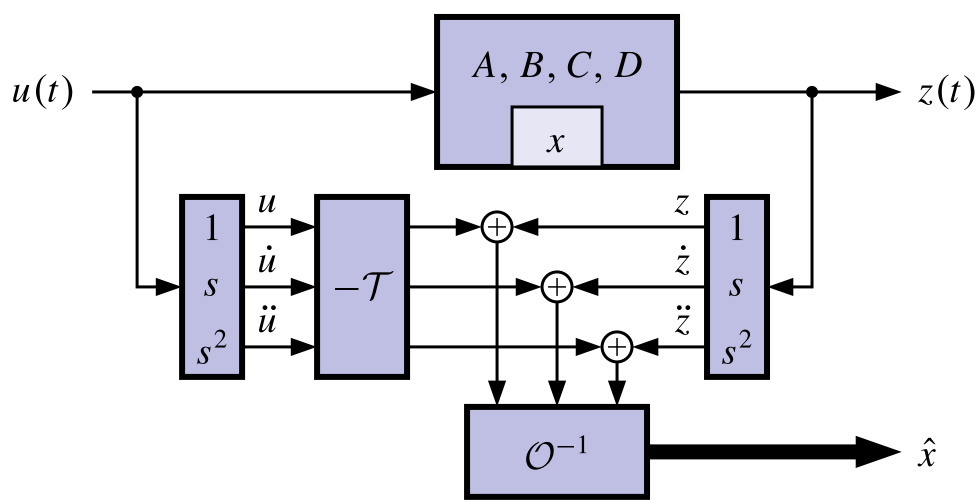
$$\begin{bmatrix} z(0) \\ \dot{z}(0) \\ \ddot{z}(0) \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}}_{\mathcal{O}(C,A)} x(0) + \underbrace{\begin{bmatrix} D & 0 & 0 \\ CB & D & 0 \\ CAB & CB & D \end{bmatrix}}_{\mathcal{T}} \begin{bmatrix} u(0) \\ \dot{u}(0) \\ \ddot{u}(0) \end{bmatrix},$$

where $\mathcal{T}$ is a (block) "Toeplitz matrix".

- Thus, if $\mathcal{O}(C, A)$ is invertible, then

$$x(0) = \mathcal{O}^{-1} \left\{ \begin{bmatrix} z(0) \\ \dot{z}(0) \\ \ddot{z}(0) \end{bmatrix} - \mathcal{T} \begin{bmatrix} u(0) \\ \dot{u}(0) \\ \ddot{u}(0) \end{bmatrix} \right\}.$$

- We say that $\{C, A\}$ is an observable pair if $\mathcal{O}$ is nonsingular.
- One possible approach to determining the system state, directly from the equations:

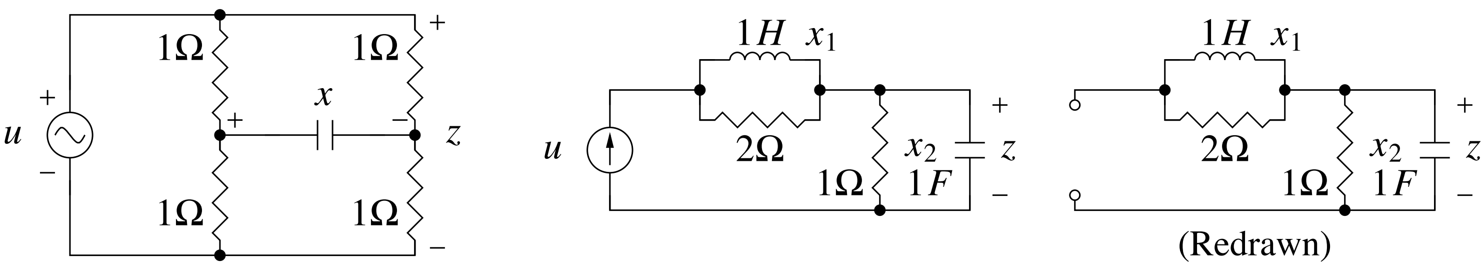


- The Kalman filter is a more practical observer that doesn't use differentiators.
- Regardless of the approach, it turns out that the system must be observable to be able to determine the initial state.

CONCLUSION: If $\mathcal{O}$ is nonsingular, then we can determine/estimate the initial state of the system $x(0)$ using only $u(t)$ and $z(t)$ (and therefore, we can estimate $x(t)$ for all $t \geq 0$).

ADVANCED TOPIC: If some states are unobservable but are stable, then an observer will still converge to the true state, even though the initial state $x(0)$ may not be uniquely determined.

EXAMPLE: Two unobservable networks



- In the first, if $u(t) = 0$ then $z(t) = 0 \quad \forall t$. Cannot determine $x(0)$.
- In the second, if $u(t) = 0$, $x_1(0) \neq 0$ and $x_2(0) = 0$, then $z(t) = 0$ and we cannot determine $x_1(0)$ (circuit redrawn for $u(t) = 0$).

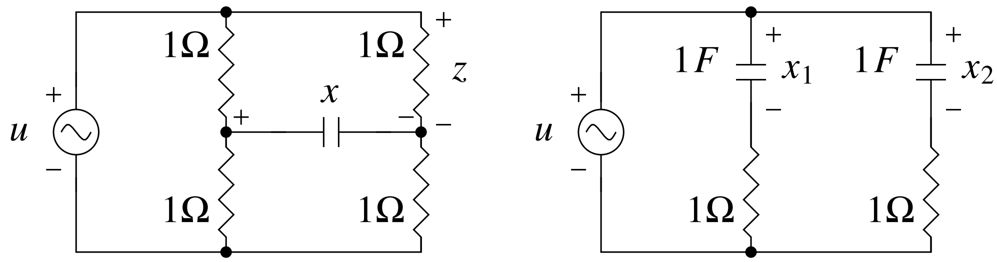
Continuous-time controllability: Can I get there from here?

- "Controllability" is a *dual* idea to observability. We won't go into as much depth here since it is not as important for our topic of study.
- Controllability asks the question, "can I move from any initial state to any desired state via suitable selection of the control input $u(t)$?"
- The answer boils down to a condition on a matrix called the controllability matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}.$$

TEST: If $\mathcal{C}$ is nonsingular, then the system is controllable.

EXAMPLE: Two uncontrollable networks.



- In the first one, if $x(0)=0$ then $x(t)=0 \quad \forall t$. Cannot influence state!
- In the second one, if $x_1(0)=x_2(0)$ then $x_1(t)=x_2(t) \quad \forall t$. Cannot independently alter state.
- Controllability is studied in more depth in *ECE5520: Multivariable Control Systems I*.

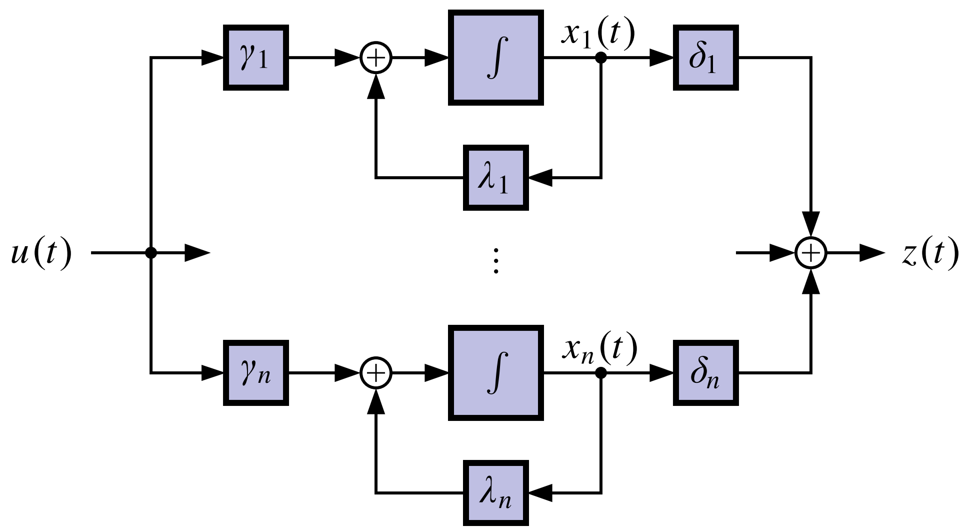
2.6: More insight; discrete-time controllability and observability

Diagonal systems, controllability and observability

- We can gain insight by considering a system in diagonal form

$$\dot{x}(t) = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} x(t) + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_n \end{bmatrix} u(t)$$

$$z(t) = \begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix} x(t) + \begin{bmatrix} 0 \end{bmatrix} u(t).$$



- When controllable? When observable?

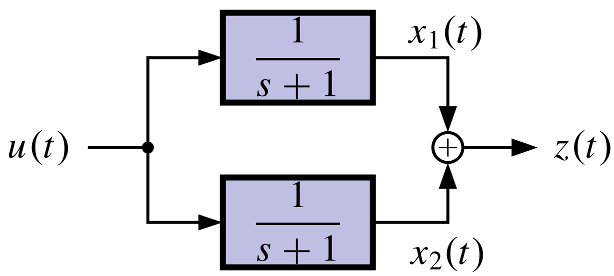
$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = \begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \\ \lambda_1\delta_1 & \lambda_2\delta_2 & \cdots & \lambda_n\delta_n \\ & & \ddots & \\ \lambda_1^{n-1}\delta_1 & \lambda_2^{n-1}\delta_2 & \cdots & \lambda_n^{n-1}\delta_n \end{bmatrix}$$

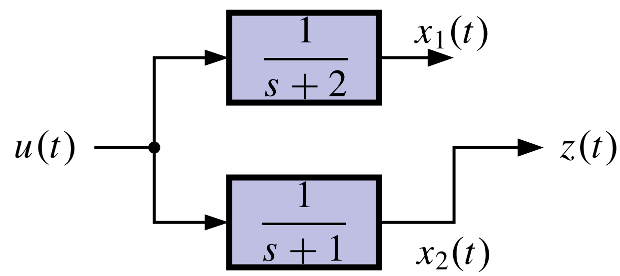
$$= \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ & & \ddots & \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1} \end{bmatrix}}_{\text{Vandermonde matrix } \mathcal{V}} \begin{bmatrix} \delta_1 & & & 0 \\ & \delta_2 & & \\ & & \ddots & \\ 0 & & & \delta_n \end{bmatrix}.$$

- Singular?

$$\det\{\mathcal{O}\} = (\delta_1 \cdots \delta_n) \det\{\mathcal{V}\} = (\delta_1 \cdots \delta_n) \prod_{i<j} (\lambda_j - \lambda_i).$$

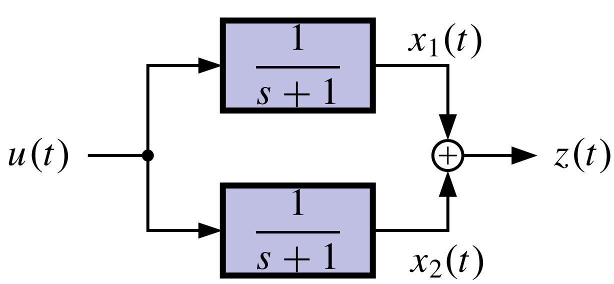
CONCLUSION: Observable $\Longleftrightarrow \lambda_i \neq \lambda_j,\ i \neq j$ and $\delta_i \neq 0\ i = 1, \cdots, n$.

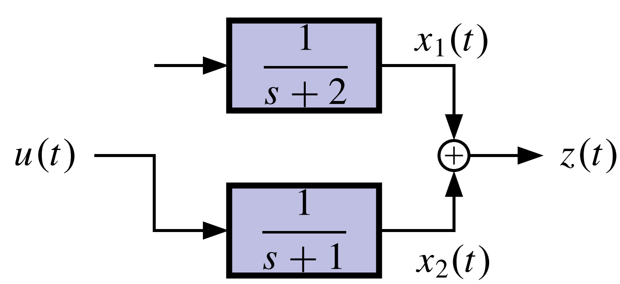




- If $\lambda_1 = \lambda_2$ then not observable. Can only "observe" the sum $x_1 + x_2$.
- If $\delta_k = 0$ then cannot observe mode k.
- What about controllability? Analysis is basically the same: just switch the roles of δs and γ s.

CONCLUSION: Controllable $\Longleftrightarrow \lambda_i \neq \lambda_j,\ i \neq j$ and $\gamma_i \neq 0\ i = 1, \cdots, n$.





- If $\lambda_1 = \lambda_2$ then not controllable. Can only "control" the sum $x_1 + x_2$.
- If $\gamma_k = 0$ then cannot control mode k.

Discrete-time controllability

- Similar concept for discrete-time. Form the discrete-time controllability matrix (where we use the discrete-time A and B matrices)

$$\mathcal{C} = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}.$$

- The matrix $\mathcal{C}$ is invertible iff the system is controllable.

Discrete-time observability

- Can we reconstruct the state x_0 from the output z_k and input u_k?

$$\begin{aligned} z_k &= Cx_k + Du_k \\ z_0 &= Cx_0 + Du_0 \\ z_1 &= C\left[Ax_0 + Bu_0\right] + Du_1 \\ z_2 &= C\left[A^2x_0 + ABu_0 + Bu_1\right] + Du_2 \\ &\vdots \\ z_{n-1} &= C\left[A^{n-1}x_0 + A^{n-2}Bu_0 + \cdots + Bu_{n-1}\right] + Du_{n-1}. \end{aligned}$$

- In vector form, we can write

$$\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{\mathcal{O}} x_0 + \underbrace{\begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & D \end{bmatrix}}_{\mathcal{T}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix}.$$

- So,

$$x_0 = \mathcal{O}^{-1}\left[\begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix} - \mathcal{T}\begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix}\right].$$

- If $\mathcal{O}$ is invertible, x_0 may be reconstructed with any z_k, u_k. We say that $\{C, A\}$ form an "observable pair."
- Do more measurements of $z_n, z_{n+1}, \ldots$ help in reconstructing x_0? No! (Caley–Hamilton theorem). So, if the original state is not "observable" with n measurements, then it will not be observable with more than n measurements either.
- Since we know u_k and the dynamics of the system, if the system is observable we can determine the entire state sequence x_k, $k \geq 0$ once we determine x_0

$$\begin{aligned} x_n &= A^n x_0 + \sum_{i=0}^{n-1} A^{n-1-i} B u_k \\ &= A^n \mathcal{O}^{-1}\left[\begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix} - \mathcal{T}\begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix}\right] + \mathcal{C}\begin{bmatrix} u_{n-1} \\ \vdots \\ u_0 \end{bmatrix}. \end{aligned}$$

- A perfectly good observer (no differentiators...), but still not nearly as good as the Kalman filters we will develop.

Appendix: Plett notation versus textbook notation

- For a continuous-time state-space model, I use:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + B_w(t)w(t)$$
$$z(t) = C(t)x(t) + D(t)u(t) + D_v(t)v(t).$$

- For a continuous-time state-space model, Simon uses:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + w(t)$$
$$y(t) = C(t)x(t) + v(t).$$

- For a continuous-time state-space model, Bar-Shalom uses:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + D(t)\tilde{v}(t)$$
$$z(t) = C(t)x(t) + \tilde{w}(t).$$

- For a discrete-time state-space model, I use:

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$
$$z_k = C_k x_k + D_k u_k + v_k.$$

- For a discrete-time state-space model, Simon uses:

$$x_{k+1} = F_k x_k + G_k u_k + \Lambda_k w_k$$
$$y_k = C_k x_k + v_k.$$

- For a discrete-time state-space model, Bar-Shalom uses:

$$x(k+1) = F(k)x(k) + G(k)u(k) + \Gamma(k)v(k)$$
$$z(k) = H(k)x(k) + w(k).$$