The target-tracking application

- One application of KFs is for target tracking. We seek to be able to predict of the future location of a dynamic system (the target) based on KF estimates and measurements.
- Ideally, we would know all matrices and signals of its continuous-time linear state-space model:

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

- However, in the target-tracking appliction, we don't generally know the input signal $u(t)$, and probably don't know the state-space matrices that describe the target's dynamics very well either.
- So, we need to approximate target dynamics based on some assumed behaviors.
  - Maybe the target is generally stationary, or tends to move in a straight line, or in circles...

Example: The nearly-constant-position (NCP) model

- Consider a relatively immobile object, which we would like to track, that gets bumped around by unknown forces.
- If we consider motion in 2-d, 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 an unknown (random) process-noise input (unlike known $u(t)$).
  - $A = 0_{2\times 2}$, $B = 0$. The size of B depends on unknown $u(t)$, so isn't well defined.
- One possible output equation is: $z(t) = x(t) + v(t)$, where $v(t)$ is an unknown random sensor-noise input.
  - $C = I_{2\times 2}$, $D = 0$. Again, the dimensions of D are not well defined.

Example: The nearly-constant-velocity (NCV) model

- We now consider an object having momentum: its velocity is nearly constant, but gets perturbed by external forces.
- We let our model state and state equation be:

$$x(t) = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \\ \eta(t) \\ \dot{\eta}(t) \end{bmatrix}, \qquad \dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \widetilde{w}(t)}_{w(t)},$$

  where $\widetilde{w}(t)$ is a 2×1 vector of random values.
- One possible output equation is: $z(t) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C} x(t) + v(t).$

Example: The coordinated-turn model

- Now, consider an object moving in a 2-d plane with constant speed and angular rate Ω where $\Omega > 0$ is counter-clockwise motion and $\Omega < 0$ is clockwise motion: $\ddot{\xi}(t) = -\Omega\dot{\eta}(t)$ and $\ddot{\eta}(t) = \Omega\dot{\xi}(t)$.
- We keep the same model state, and modify the state equation to be:

$$x(t) = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \\ \eta(t) \\ \dot{\eta}(t) \end{bmatrix}, \qquad \dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\Omega \\ 0 & 0 & 0 & 1 \\ 0 & \Omega & 0 & 0 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \widetilde{w}(t)}_{w(t)}.$$

- One possible output equation is again: $z(t) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C} x(t) + v(t)$.

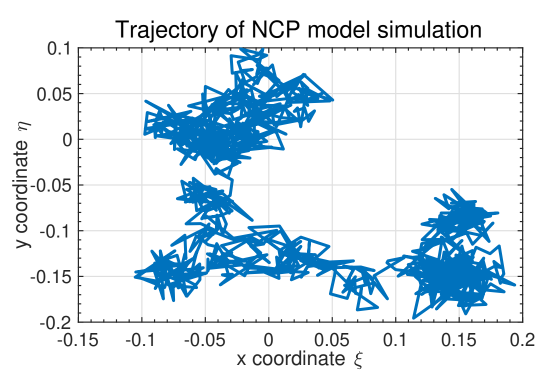
Simulating continuous-time systems

- Octave has two functions in its "control" package that help us simulate continuous-time state-space models easily.
- The first is "`ss`", which creates a state-space object from A, B, C, and D matrices.
- The second is "`lsim`", which simulates a state-space object for given input conditions and an optional initial state.
- To simulate a model:
  - We first define the A, B, C, and D matrices in the Octave workplace.
  - Then, we create a state-space model using "`ss`".
  - We define a time vector and an input sequence in the Octave workplace.
  - Then, we simulate the model for that input sequence using "`lsim`".
  - The output of the simulation is the signal $z(t)$.
- The following slides show some examples.

Time (dynamic) response

- Let's first consider simulating an NCP model.
- The figure on the right is an output from simulating the Octave code below.

```
%% Simulate the NCP model
A = zeros(2); Bw = eye(2);
C = eye(2);   D = zeros(2);
ncp = ss(A,Bw,C,D);
t = (0:999)*0.1;
w = 0.05*randn(2,1000);
v = 0.01*randn(2,1000);
z = lsim(ncp,w',t)+v';
plot(z(:,1),z(:,2))
```

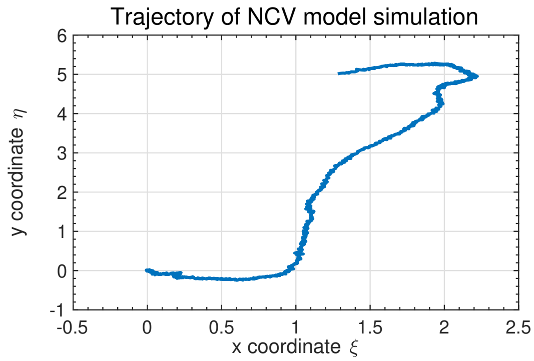


- Trajectory starts at $(0, 0)$ and randomly moves from there as w pushes object.

NCV-model example

- Now, let's consider an NCV model.
- The figure on the right is an output from simulating the Octave code below.

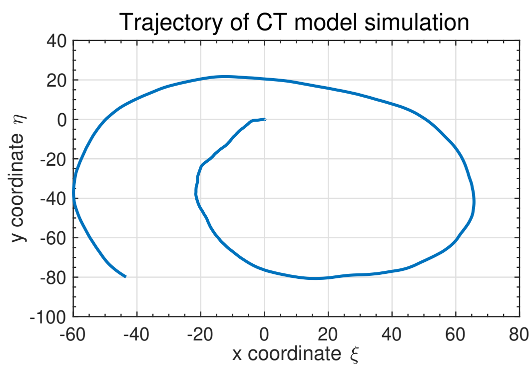
```
%% Simulate the NCV model
A  = [0 1 0 0; 0 0 0 0; ...
      0 0 0 1; 0 0 0 0];
Bw = [0 0; 1 0; 0 0; 0 1];
C  = [1 0 0 0; 0 0 1 0];
D  = zeros(2);
ncv = ss(A,Bw,C,D);
t = (0:999)*0.1;
w = 0.05*randn(2,1000);
v = 0.01*randn(2,1000);
z = lsim(ncv,w',t)+v';
plot(z(:,1),z(:,2))
```



Coordinated-turn example

- Now, let's consider a CT model.
- The figure on the right is an output from simulating the Octave code below.

```
%% Simulate the CT model
W  = 0.01; % Value of Omega
A  = [0 1 0 0; 0 0 0 -W; ...
      0 0 0 1; 0 W 0 0];
Bw = [0 0; 1 0; 0 0; 0 1];
C  = [1 0 0 0; 0 0 1 0];
D  = zeros(2);
ct = ss(A,Bw,C,D);
t = (0:999)*0.1;
w = 0.01*randn(2,1000);
v = 0.001*randn(2,1000);
z = lsim(ct,w',t)+v';
plot(z(:,1),z(:,2))
```



Summary

- Target tracking is an important application of KF.
  - We seek to be able to estimate the present position or predict of the future location of a dynamic system (the target) based on KF estimates and measurements.
- In this application, we don't know the deterministic input signal $u(t)$, so we consider it to be zero.
- We also don't generally know the state-space matrices that describe the target's dynamics, so we adopt approximate models based on some assumed behaviors.
- You learned about the nearly-constant-position (NCP), nearly-constant-velocity (NCV), and coordinated-turn (CT) models in this lesson.
- You also saw how to simulate them in Octave code.