6.3100: Dynamic System Modeling and Control Design

Hybrid CT-DT Control

December 04, 2024

Hybrid Representations

To date, we have analyzed both continuous-time and discrete-time models of plants (motor, arm, maglev) and controllers (PID, lead, SS).

Choosing continuous versus discrete was largely a matter of convenience:

- continuous frequencies (Hz) versus discrete frequencies (radians)
- unit circle versus left and right half-planes
- eigenfunctions e^{st} versus z^n

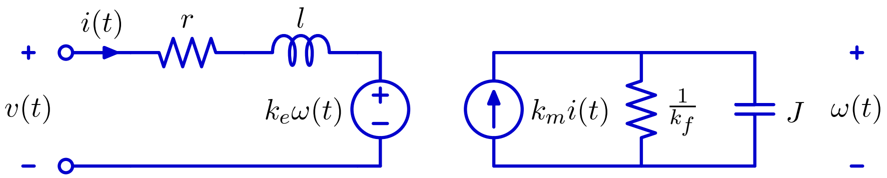
No clearly better choice because all of our systems have been hybrids:

- plant: continuous time
- controller (Teensy): discrete time

Today: Analyze interactions of discrete and continuous parts
in terms of a specific concrete context: motor speed control.

Motor Speed Control

Circuit model of a DC motor.



The voltage $v(t)$ represents the electrical input to the motor.

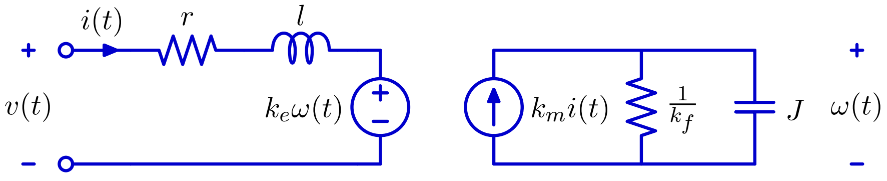
It excites a current $i(t)$, which generates a torque $k_m i(t)$ that tends to rotate the motor shaft.

The torque is resisted by the moment of inertia J and by friction (k_f).

As the motor spins, it generates a back emf $k_e\omega(t)$ that tends to reduce the electrical current $i(t)$ drawn by the motor.

Motor Speed Control: Two-Port Model

Motors have two ports: one is electrical, the other is mechanical.

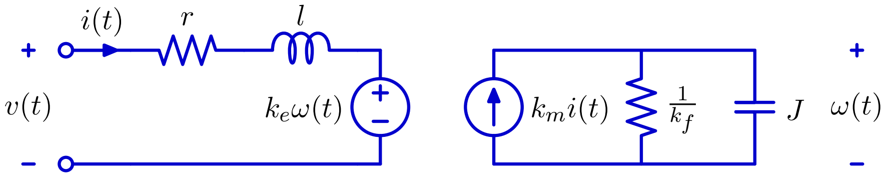


Electrical port

Mechanical port

Motor Speed Control: Mathematical Representation

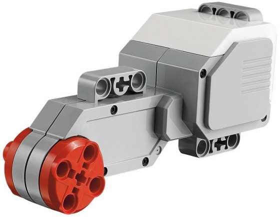
Simple circuit analysis provides a mathematical representation.



$$v(t) = ri(t) + l\frac{di(t)}{dt} + k_e\omega(t) \qquad k_m i(t) = k_f\omega(t) + J\frac{d\omega(t)}{dt}$$

Parameters of the Model

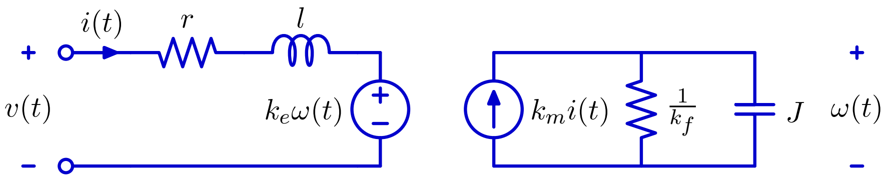
Lego EV3 motor parameters.



$$
\begin{aligned}
r &= 7\,\Omega \\
l &= 0.005\,\mathsf{H} \\
k_e &= 0.46\,\text{volts/(radian/sec)} \\
k_m &= 0.3\,\text{Nm/(radian/sec)} \\
k_f &= 0.00073\,\text{Nm/(radian/sec)} \\
J &= 0.0015\,\text{Nm/(radian/sec}^2)
\end{aligned}
$$

Continuous-Time State-Space Representation

Analyze in continuous time since the equations are in continuous time.



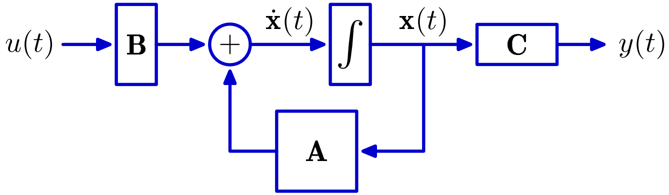
$$v(t) = ri(t) + l\frac{di(t)}{dt} + k_e\omega(t) \qquad k_m i(t) = k_f\omega(t) + J\frac{d\omega(t)}{dt}$$

$$\frac{d}{dt}\underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} = \underbrace{\begin{bmatrix} -\frac{r}{l} & -\frac{k_e}{l} \\ \frac{k_m}{J} & -\frac{k_f}{J} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} \frac{1}{l} \\ 0 \end{bmatrix}}_{\mathbf{B}} \underbrace{v(t)}_{\mathbf{u}(t)}$$

$$\underbrace{\omega(t)}_{y(t)} = \underbrace{[0 \quad 1]}_{\mathbf{C}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)}$$

Motor Speed Control: State-Space Model

State-space block diagram.

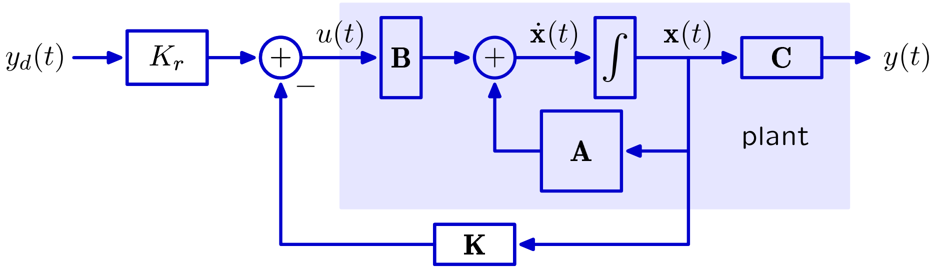


$$\underbrace{\frac{d}{dt}\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\frac{d}{dt}\mathbf{x}(t)} = \underbrace{\begin{bmatrix} -\frac{r}{l} & -\frac{k_e}{l} \\ \frac{k_m}{J} & -\frac{k_f}{J} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} \frac{1}{l} \\ 0 \end{bmatrix}}_{\mathbf{B}} \underbrace{v(t)}_{\mathbf{u}(t)}$$

$$\underbrace{\omega(t)}_{y(t)} = \underbrace{[0 \quad 1]}_{\mathbf{C}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)}$$

Full-State Feedback

We wish to design a feedback system to make the output speed $y(t) = \omega(t)$ track the desired speed $y_d(t)$.



We can find $\mathbf{K}$ by using **pole placement**:

```
K = place(A,B,[pole1,pole2])
```

or **LQR**:

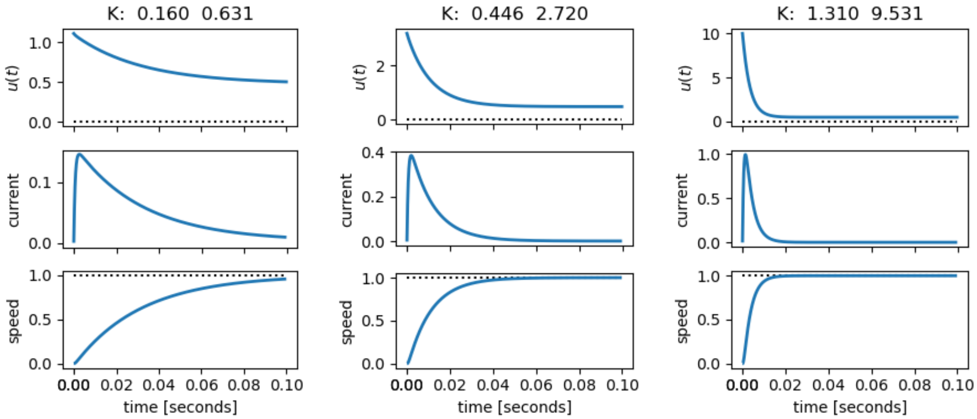
`K = lqr(A,B,Q,R)` where `Q = diag([penalty1,penalty2])` and `R = 1`

Then `Kr` is set to minimize steady-state errors.

```
Kr = -1/(C*inv(A-B*K)*B)
```

Effects of Increasing Gain

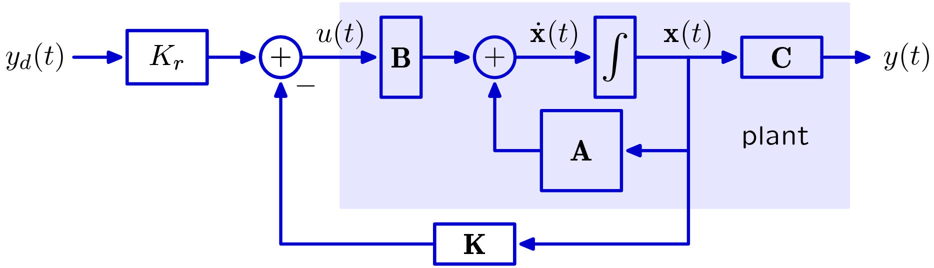
Columns show step responses for increasing gains $\mathbf{K}$ as $\mathbf{Q}$ is increased from $[1, 1]$ to $[1, 10]$ to $[1, 100]$ while R is constant at $R = 1$.



As expected, increasing gains speed convergence.

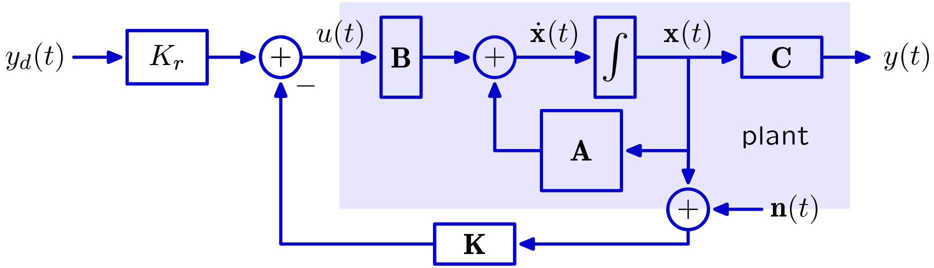
Effects of Sensor Noise

Feedback control can be significantly degraded by noise.



Effects of Sensor Noise

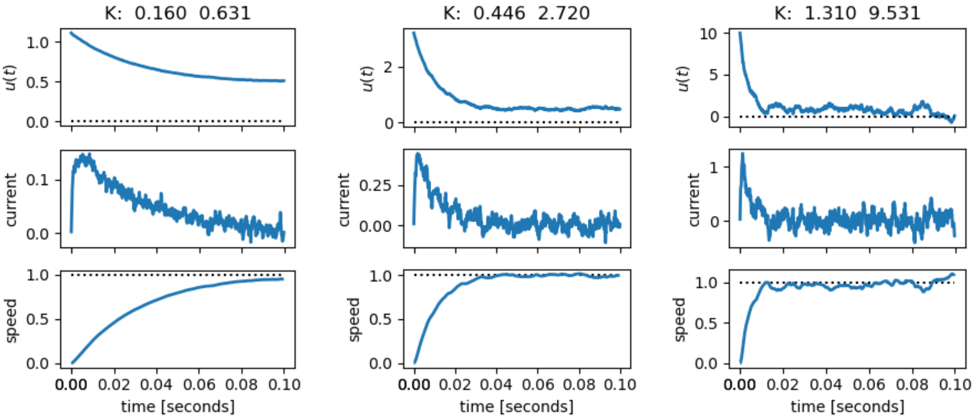
Feedback control can be significantly degraded by noise.



In the last lecture, we introduced a model for sensor noise.

Full-State Feedback Results With Noise

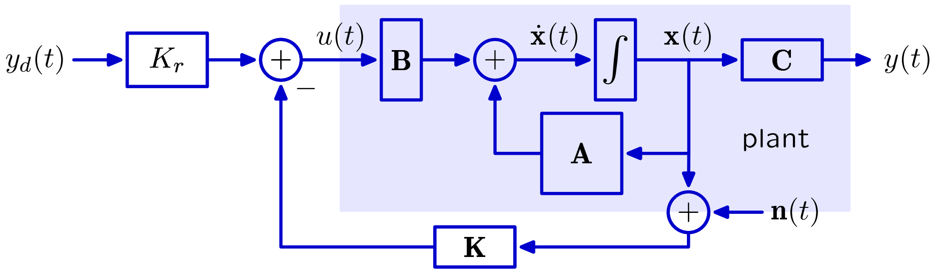
Same amount of added noise in each column. Gains increase left to right.



Increasing the gains speeds convergence but also increases noise in the effort and states, especially important for the output (speed of rotation).

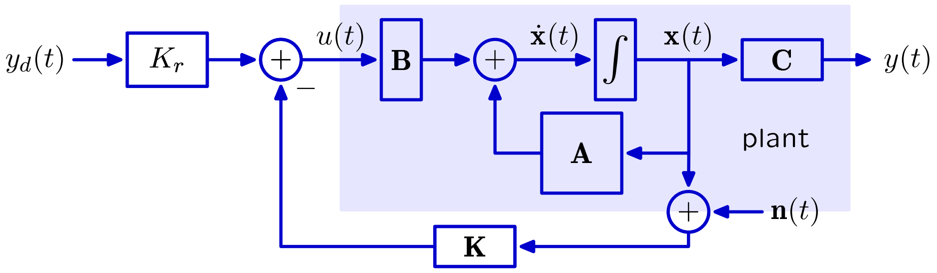
Check Yourself

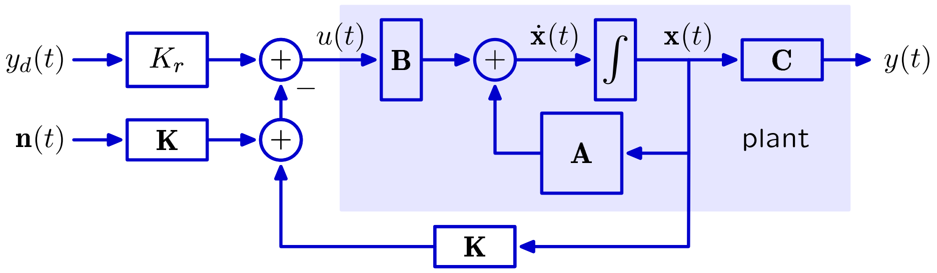
Why does increasing the gains increase noise sensitivity?



Check Yourself

Why does increasing the gains increase noise sensitivity?





Noise $\mathbf{n}(t)$ could equivalently be treated as an input **scaled by K**.

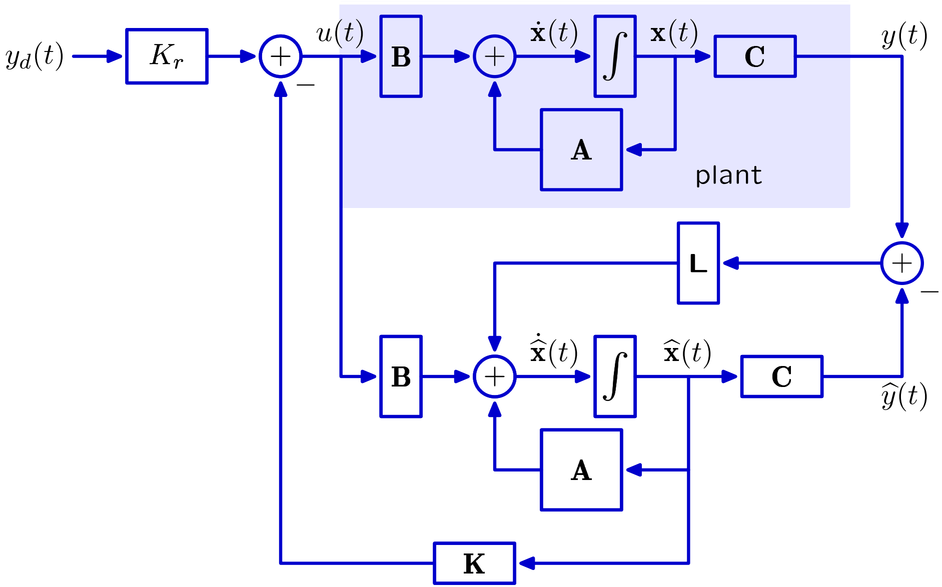
Sensors

Full-state sensing is both a strength and a weakness. It provides a maximum amount of feedback information, but it requires sensing all possible states.

What if information about internal states is not available?

Feedback with an Observer

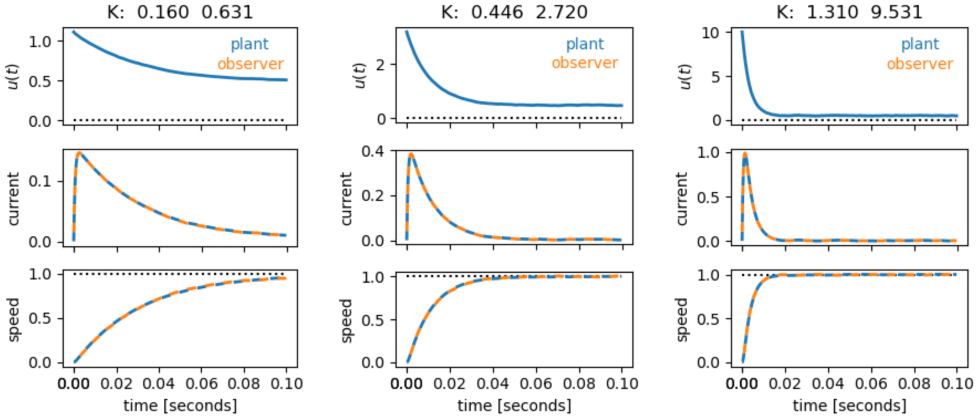
An observer provides estimates of states that are difficult to measure.



Observer Results With Noise

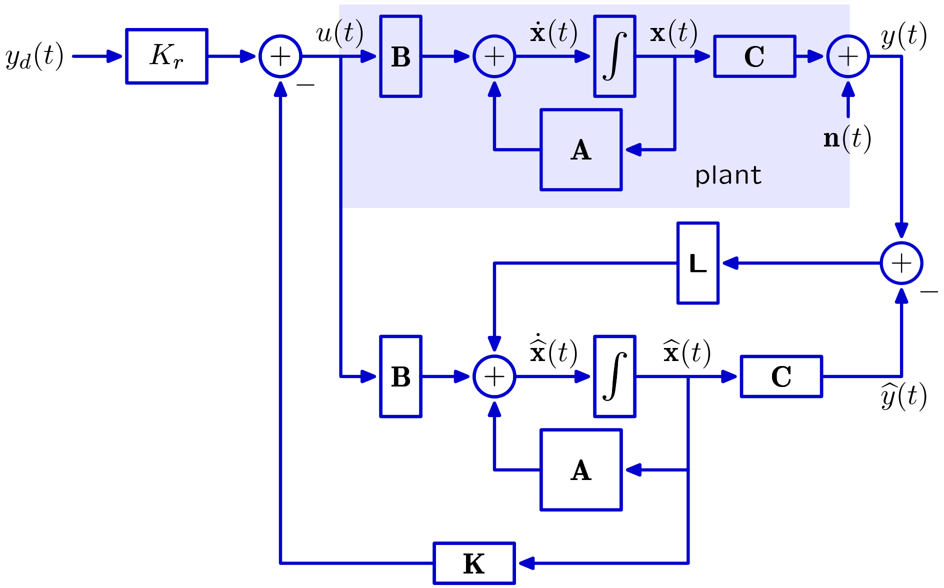
As in full-state feedback, increasing the gain of a controller with observer speeds convergence.

Gains $\mathbf{K}$ increase from left to right.



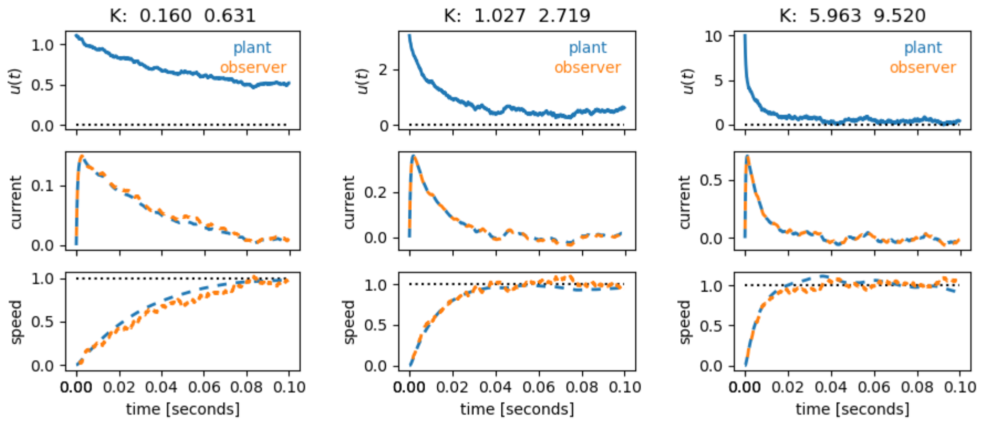
Effects of Sensor Noise

Sensor noise at the output is an important source of noise.



Observer Results With Noise

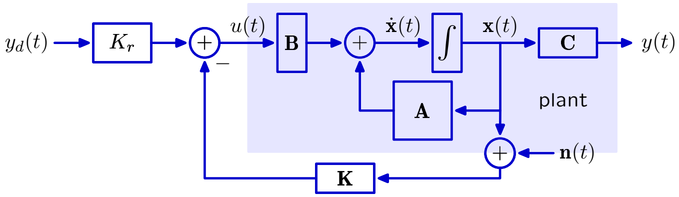
Same level of noise in each column, with gains increasing from left to right.

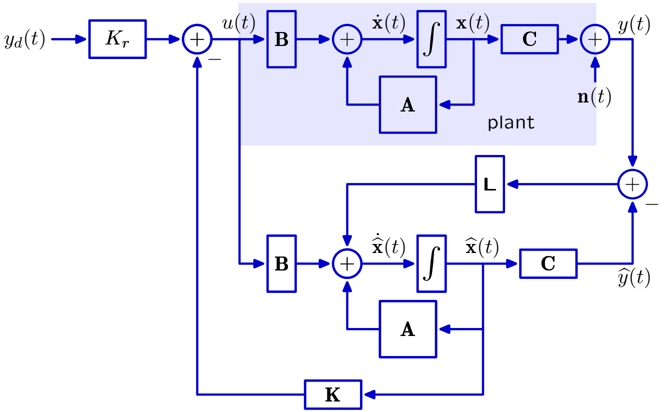


An observer effectively reduces noise, even with high gain feedback.

Controller Analysis

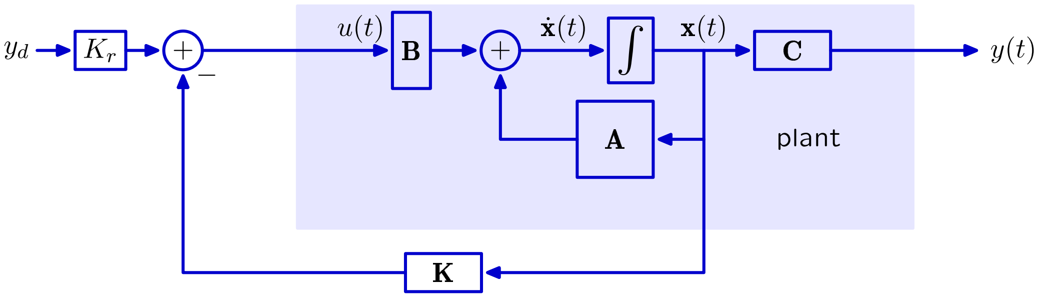
We analyzed the controller in CT even though it is intrinsically DT.





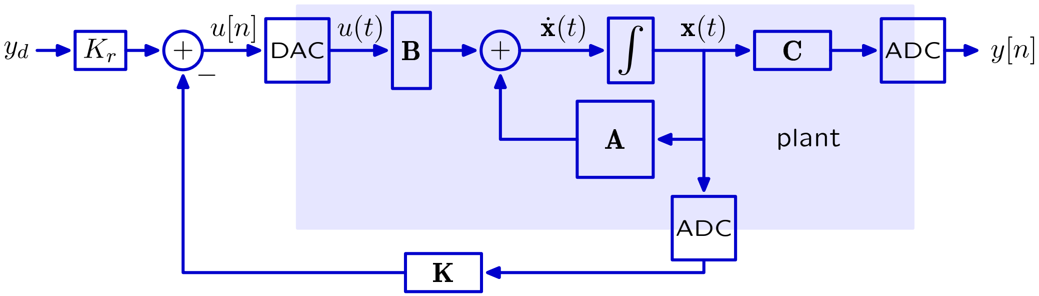
Hybrid Representation

To use a microprocessor to control a continous time (physical) plant, we must convert between CT and DT representations of signals.



Hybrid Representation

To use a microprocessor to control a continous time (physical) plant, we must convert between CT and DT representations of signals.



We use an analog-to-digital converter (ADC) to create a discrete-time representation of the state and output,.

We use a digital-to-analog converter (DAC) to reconstruct a continuous-time representation of the command $u[n]$.

Analog-To-Digital Conversion

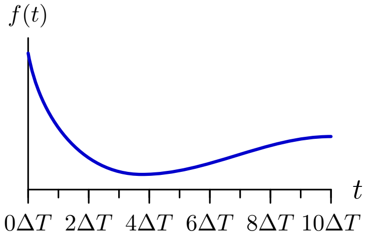
Analog-to-digital conversion entails two types of transformations.

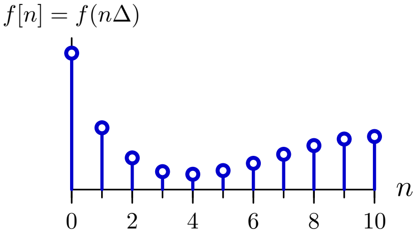
Sampling: process by which a function of real domain is transformed into a function of integer domain.

Quantization: process by which a continuous range of amplitudes is represented by a finite range of integers.

Sampling

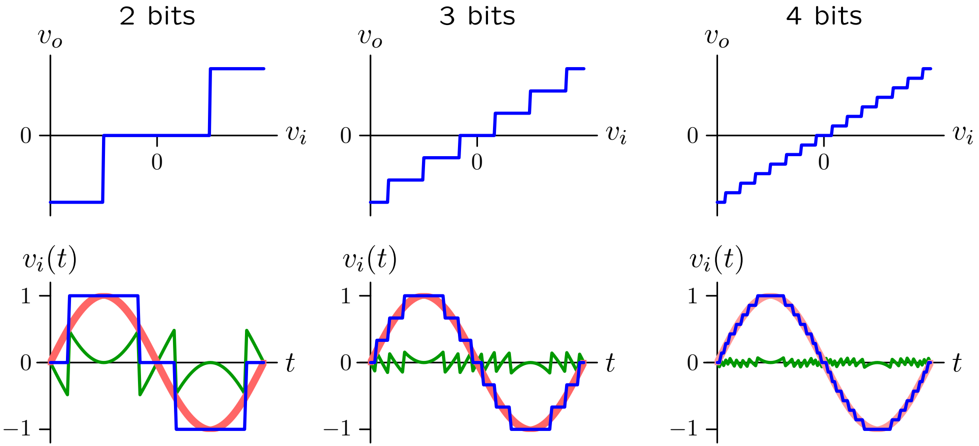
A function of real domain is transformed into a function of integer domain.





Quantization

Quantization: process by which a continuous range of amplitudes is represented by a finite range of integers.



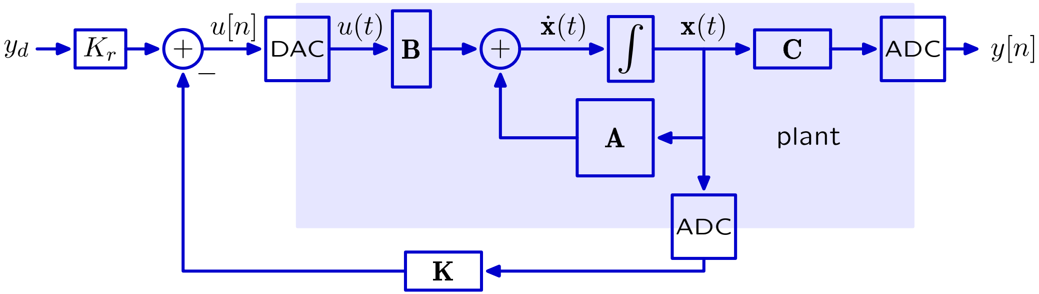
Continuous-Time Signal $v_i(t)$

Quantized Signal

Quantization Error (difference)

Hybrid Representation

To use a microprocessor to control a continous time (physical) plant, we must convert between CT and DT representations of signals.

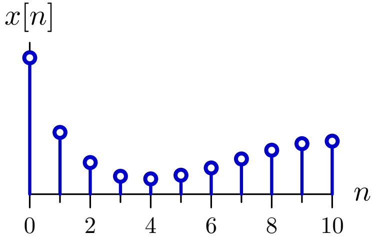


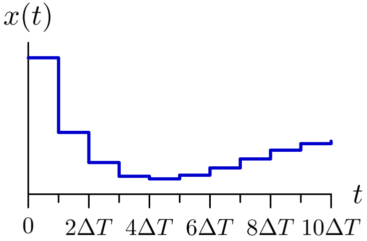
We use an analog-to-digital converter (ADC) to create a discrete-time representation of the state and output,.

We use a digital-to-analog converter (DAC) to reconstruct a continuous-time representation of the command $u[n]$.

Digital-To-Analog Conversion

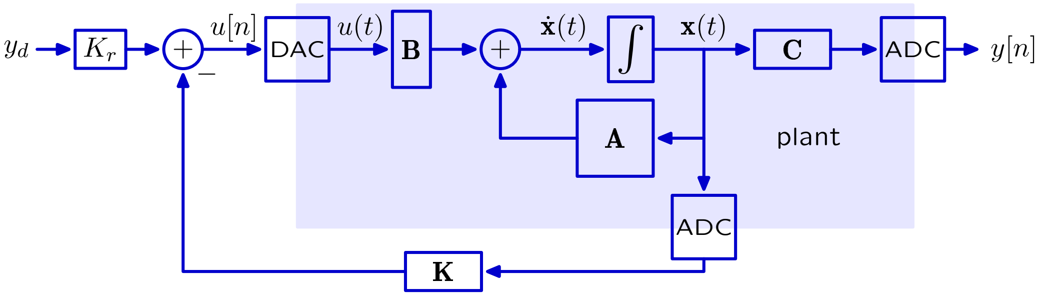
Digital-to-analog conversion **reconstructs** an analog signal from its digital representation. **zero-order hold**





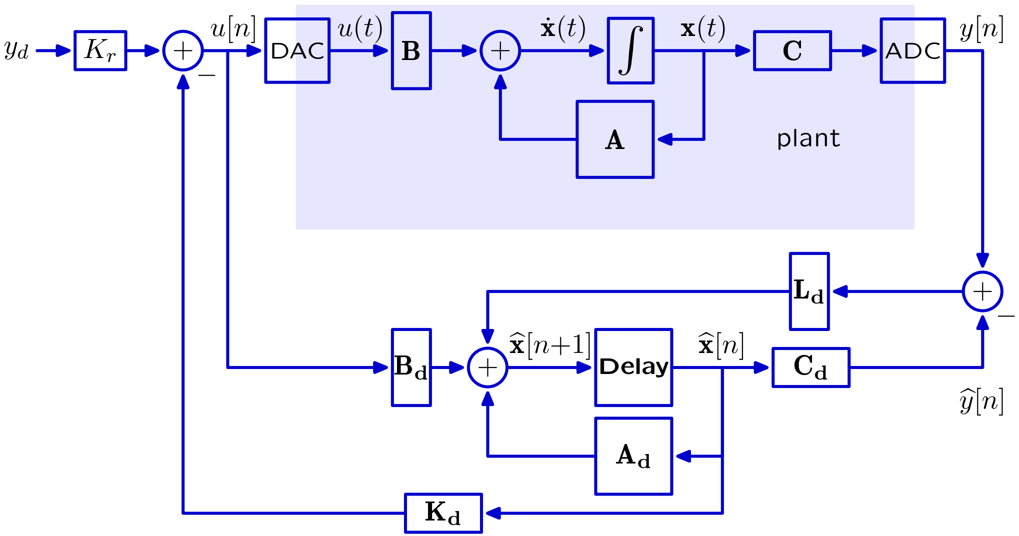
Hybrid Representation

Explicitly convert $y(t) \to y[n]$, $\mathbf{x}(t) \to \mathbf{x}[n]$ and $u[n] \to u(t)$.



Hybrid Representation

More changes are needed to convert an observer system to discrete time.



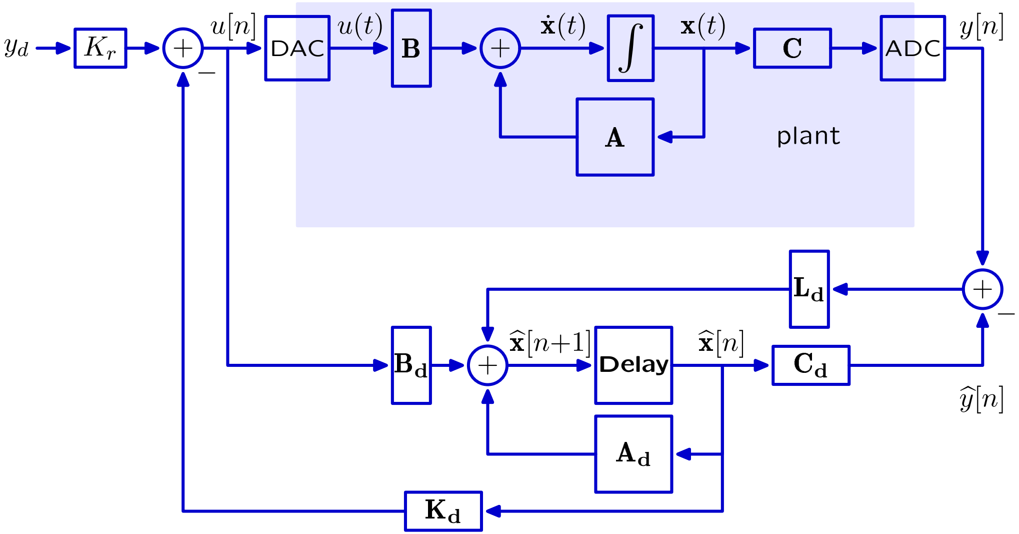
Signals outside plant (e.g., $u[n]$ and $y[n]$ must be discrete time.

Integrator in plant $\rightarrow$ delay in the observer: $\widehat{\mathbf{x}}(t) \rightarrow \widehat{\mathbf{x}}[n]$; $\dot{\widehat{\mathbf{x}}}(t) \rightarrow \widehat{\mathbf{x}}[n+1]$

Control matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{L}$, and $\mathbf{K}$ must be converted to discrete versions.

Hybrid Representation

More changes are needed to convert an observer system to discrete time.



How can we convert $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ to $\mathbf{A_d}$, $\mathbf{B_d}$, $\mathbf{C_d}$?

Discrete-Time State Evolution

Start by considering the scalar case.

The continuous-time state evolution equation is

$$\dot{x}(t) = ax(t) + bu(t)$$

Since $u[n]$ only changes on step boundaries, $u(t)$ is constant between steps.

Then $x(t)$ has homogeneous and particular parts:

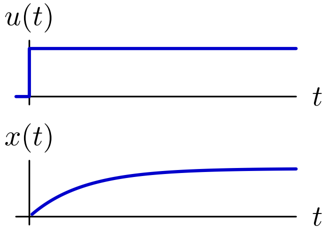
$$x(t) = \alpha e^{\beta t} + \gamma$$

Substituting into the plant equation:

$$\dot{x}(t) = \beta\alpha e^{\beta t} = ax(t) + bu(t) = a(\alpha e^{\beta t} + \gamma) + bu(t)$$

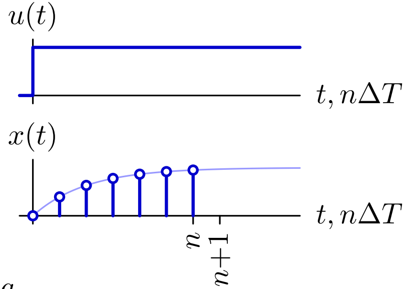
shows that $\beta = a$ and $\gamma = -bu(t)/a$ so that

$$x(t) = \alpha e^{at} - bu(t)/a$$



Discrete-Time State Evolution

The discrete-time state evolution equation computes $x[n{+}1] = x((n{+}1)\Delta T)$ from $x[n] = x(n\Delta T)$.



$$x(t) = \alpha e^{at} - bu(t)/a$$

$$x[n] = \alpha e^{an\Delta T} - bu[n]/a$$

$$x[n{+}1] = \underbrace{\alpha e^{a(n+1)\Delta T}}_{e^{a\Delta T}\ \underbrace{\alpha e^{an\Delta T}}_{x[n]+bu[n]/a}} - bu[n]/a$$

$$x[n{+}1] = e^{a\Delta T}x[n] + \left(e^{a\Delta T}{-}1\right)\frac{b}{a}u[n]$$

Discrete-Time State Evolution

Use linear algebra to compute the analogous matrix expression.

State update equation (scalar form):

$$x[n{+}1] = e^{a\Delta T}x[n] + \left(e^{a\Delta T} - 1\right)\frac{b}{a}u(t)$$

State update equation (matrix form):

$$\mathbf{x}[n{+}1] = e^{\mathbf{A}\Delta T}\,\mathbf{x}[n] + \left(e^{\mathbf{A}\Delta T}{-}\mathbf{I}\right)\mathbf{A}^{-\mathbf{1}}\mathbf{B}\,u[n]$$

Discrete version of state evolution equation:

$$\mathbf{x}[n{+}1] = \mathbf{A_d}\mathbf{x}[n] + \mathbf{B_d}u[n]$$

where

$$\mathbf{A_d} = e^{\mathbf{A}\Delta T}$$

$$\mathbf{B_d} = \left(e^{\mathbf{A}\Delta T}{-}\mathbf{I}\right)\mathbf{A}^{-\mathbf{1}}\mathbf{B}$$

The exponential function in the scalar form is replaced by a matrix exponential function in the matrix form.

Discrete-Time State Evolution

Comparison of discrete and continuous time plant descriptors.

Continuous Time

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)$$

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

Discrete Time

$$\dot{\mathbf{x}}[n{+}1] = \mathbf{A_d}\mathbf{x}[n] + \mathbf{B_d}u[n]$$

$$y[n] = \mathbf{C_d}\mathbf{x}[n] + \mathbf{D_d}u[n]$$

where

$$\mathbf{A_d} = e^{\mathbf{A}\Delta T}$$

$$\mathbf{B_d} = \left(e^{\mathbf{A}\Delta T} - \mathbf{I}\right)\mathbf{A}^{\mathbf{-1}}\mathbf{B}$$

$$\mathbf{C_d} = \mathbf{C}$$

$$\mathbf{D_d} = \mathbf{D}$$

Discrete-Time Gain Matrices

For continuous-time observers, we find the state feedback matrix $\mathbf{K}$ by solving a continuous-time minimization problem:

$$\min_{\mathbf{K}} \left(\int_0^\infty \mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau)d\tau + \int_0^\infty \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)d\tau \right)$$

For discrete-time observers, we find the state feedback matrix $\mathbf{K_d}$ by solving a discrete-time minimization problem:

$$\min_{\mathbf{K_d}} \left(\sum_{m=0}^{\infty} \mathbf{x}^T[m]\mathbf{Q}\mathbf{x}[m] + \sum_{m=0}^{\infty} \mathbf{u}^T[m]\mathbf{R}\mathbf{u}[m] \right)$$

These algorithms are different!

For continuous-time systems:

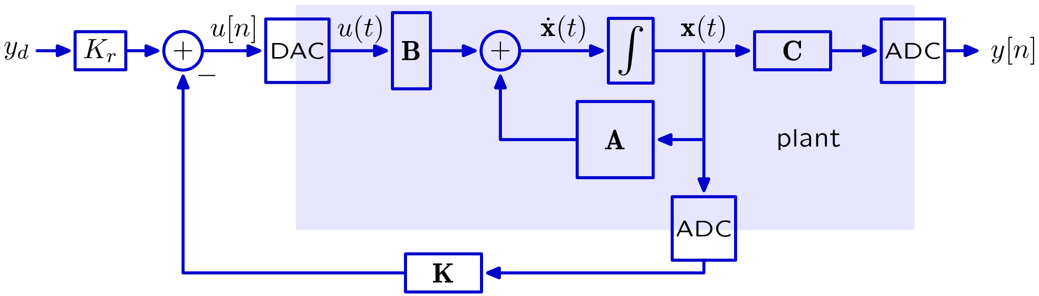
```
K=lqr(A,B,Q,R)
L=lqr(A.',B.',Q,R).'
```

For discrete-time systems:

```
Kd=dlqr(Ad,Bd,Q,R)
Ld=dlqr(Ad.',Bd.',Q,R).'
```

Check Yourself

Consider a state-space controller for the motor model.

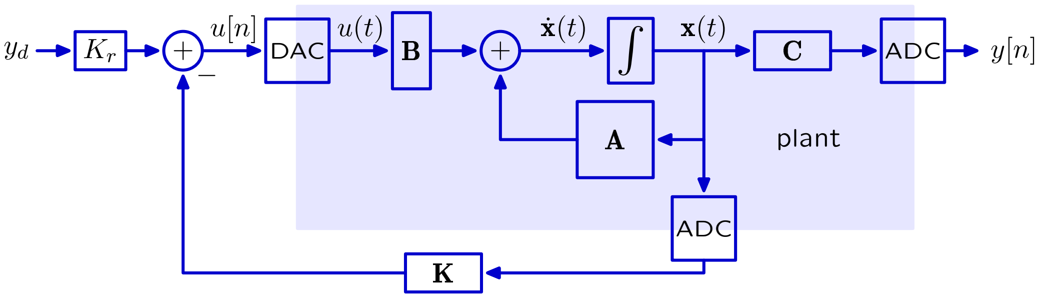


Which of the following values of $\mathbf{K}$ will work best if $\Delta T << \tau$?

```
K1 =  lqr(A,B,Q,R)
K2 =  dlqr(A,B,Q,R)
K3 =  lqr(I+A*DeltaT,B*DeltaT,Q,R)
K4 =  dlqr(I+A*DeltaT,B*DeltaT,Q,R)
K5 =  lqr(expm(A*DeltaT),(expm(A*DeltaT)-I)*A\B,Q,R)
K6 =  dlqr(expm(A*DeltaT),(expm(A*DeltaT)-I)*A\B,Q,R)
```

Check Yourself

Consider a state-space controller for the motor model.



Since ΔT is small relative to the dynamics of the system, the net effect of the intervening ADC and DAC is to add a very small delay.

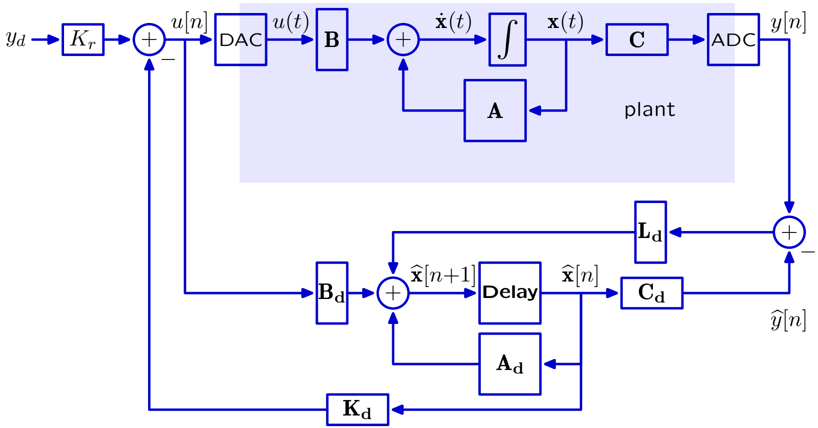
The hybrid system is approximately continuous.
$\rightarrow$ `K1` is a reasonable approximation.

`K2` doesn't makes sense: cannot run `dlqr` on a CT state evolution matrix.

The other options don't make sense because the system is essentially CT.

Check Yourself

Consider an observer-based controller for the motor.

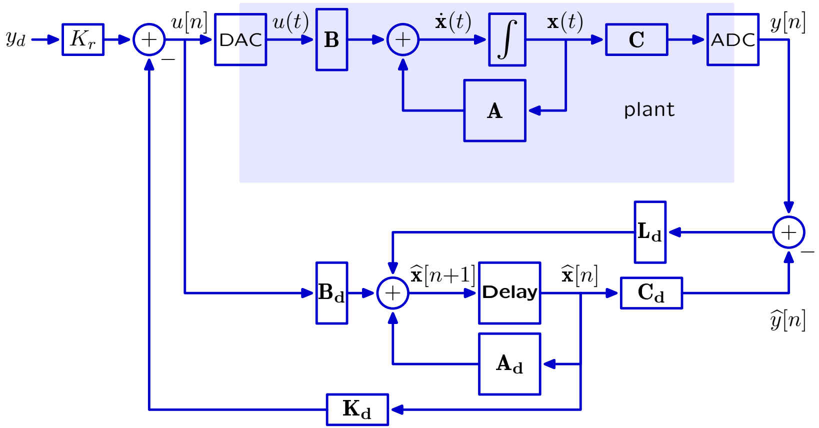


Which of the following values of $\mathbf{K_d}$ will work best?

```
K1 =  lqr(A,B,Q,R)
K2 =  dlqr(A,B,Q,R)
K3 =  lqr(I+A*DeltaT,B*DeltaT,Q,R)
K4 =  dlqr(I+A*DeltaT,B*DeltaT,Q,R)
K5 =  lqr(expm(A*DeltaT),(expm(A*DeltaT)-I)*A\B,Q,R)
K6 =  dlqr(expm(A*DeltaT),(expm(A*DeltaT)-I)*A\B,Q,R)
```

Check Yourself

Consider an observer-based controller for the motor.



`K5` and `K6` are based on discrete approximations to the CT matrices $\mathbf{A}$ & $\mathbf{B}$.

$\rightarrow$ `K5` doesn't make sense: cannot run `lqr` on DT matrices.

$\rightarrow$ `K6` is the best choice.

`K3` and `K4` are based on 1st-order approximations to the DT matrices.

$\rightarrow$ `K3` doesn't make sense: cannot run `lqr` on DT matrices.

$\rightarrow$ `K4` is a very good choice.

Summary

Microcontrollers (such as the Teensy) are increasingly used to control systems because of their low cost and high performance.

Using a microcontroller with a physical plant creates a hybrid system with part described in continuous time and part described in discrete time.

Optimization algorithms (such as pole placement and LQR) have been developed for both continuous- and discrete-time systems.

Next Time

Micro-robots with Professor Kevin Chen