6.3100: Dynamic System Modeling and Control Design

Discrete-Time Observer

May 6, 2024

Overview

In past lectures, we analyzed behaviors of **continuous-time observers**.

We analyzed **two types of convergence**:

– convergence of the observer and plant states:

$$\widehat{\mathbf{x}}(\mathbf{t}) \to \mathbf{x}(t)$$

– and convergence of the plant output to the desired value:

$$y(t) \to y_d(t)$$

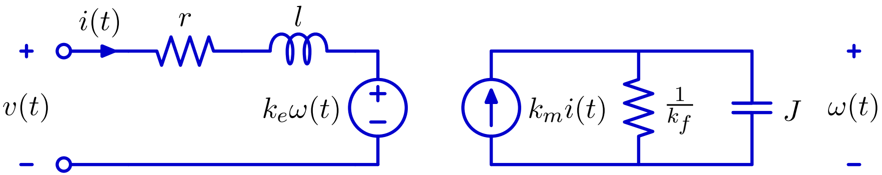
We also looked at the sensitivity of the controllers to **noise**.

While we have focused on continuous-time controllers, modern controllers increasingly work in **discrete time** – in large part because of the availability of low cost, high performance microprocessors.

Today: analyze systems that **combine continuous time** representations of a plant **with discrete time** implementations of its control.

Motor Speed Control

We will use the motor speed control system as an example.



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

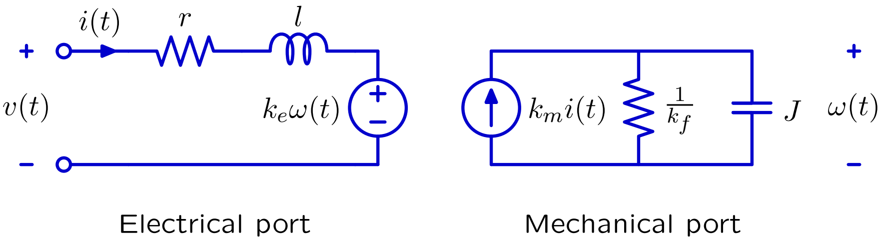
It excites a current $i(t)$, which generates a torque $k_m i(t)$ that tends to rotate the motor shaft.

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.

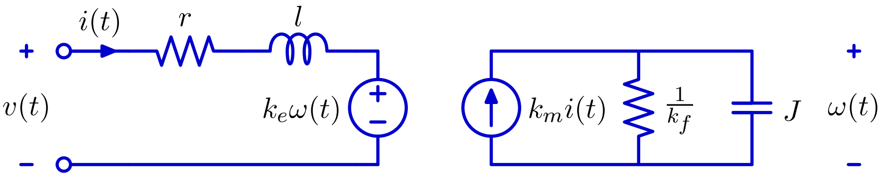
Motor Speed Control: Two-Port Model

Motors have two ports: one is electrical and one is mechanical.



Motor Speed Control: Mathematical Representation

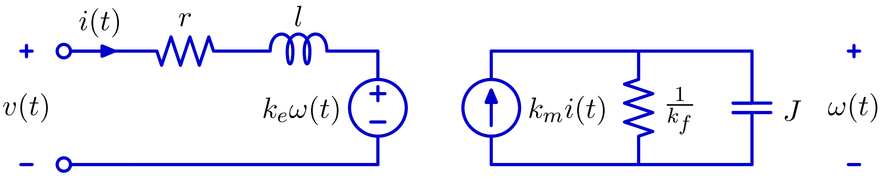
Simple circuit analysis provides a mathematical representation.



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

Motor Speed Control: Matrix Representation

The equations are conveniently represented by a pair of matrix equations.



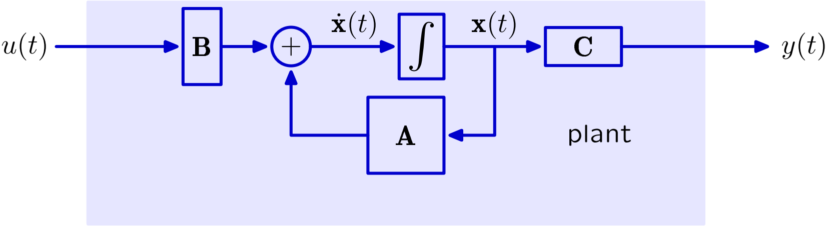
$$v(t) = ri(t) + l\frac{di(t)}{dt} + k_e\omega(t) \qquad k_m i(t) = k_f\omega(t) + J\frac{d\omega(t)}{dt}$$

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

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

State-Space Model

The matrix equations provide a complete representation of the **plant**.

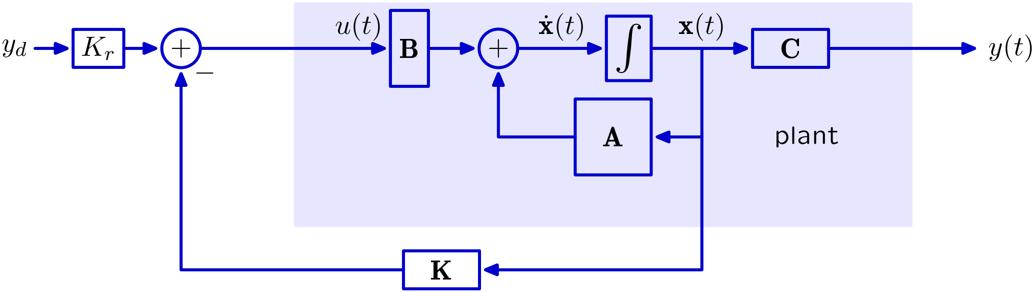


$$\underbrace{\frac{d}{dt}\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\frac{d}{dt}\mathbf{x}(t)} = \underbrace{\begin{bmatrix} -\frac{r}{l} & -\frac{k_e}{l} \\ \frac{k_m}{J} & -\frac{k_f}{J} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)}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

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

State-Space Model + State-Space Controller

This motor model was then put into a feedback loop that was designed to make the output speed $y(t) = \omega(t)$ track the desired speed $y_d(t)$



where $\mathbf{K}$ is found using **pole placement**:

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

or **LQR**:

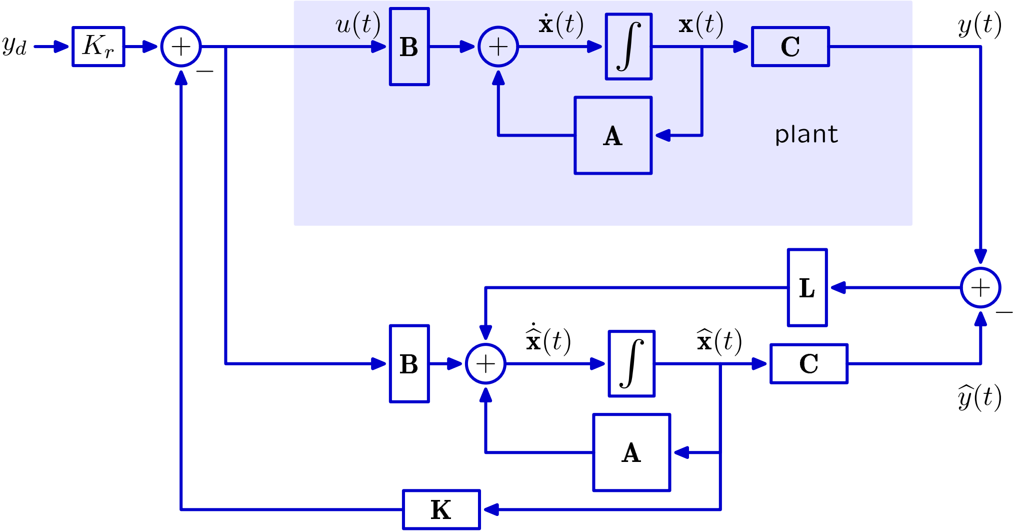
`K = lqr(A,B,Q,R)` where `Q = diag([penalty1,penalty2])` and `R = 1`

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

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

State-Space Model + Observer

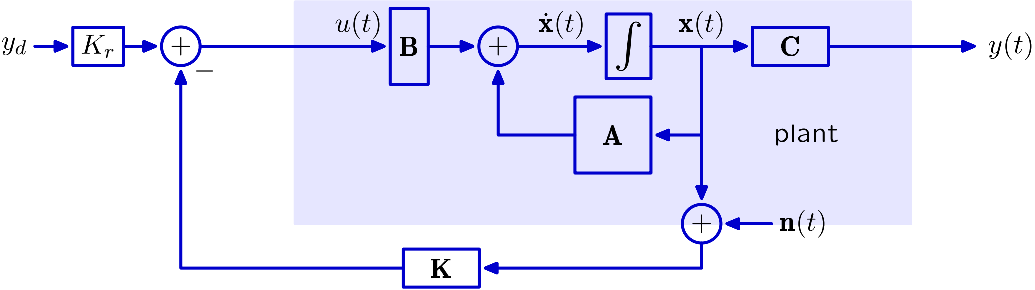
We also analyzed the performance of an observer-based controller.



More effective control without having to measure the states of the plant.

Effects of Sensor Noise

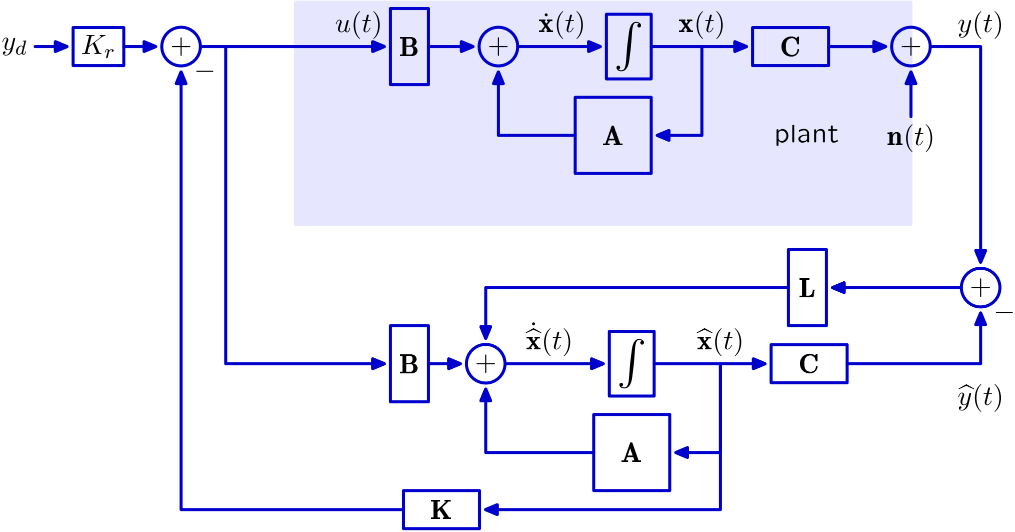
We looked at noise performance for both state-space and observer-based controllers.



We focused on sensing (measurement) noise at the interface between the plant and the controller.

Effects of Sensor Noise

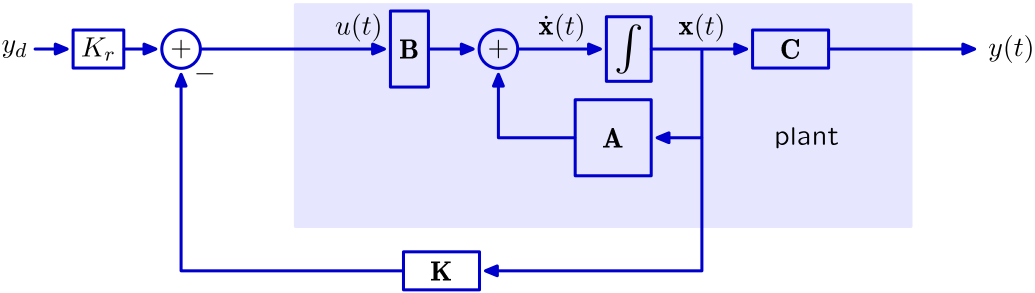
We looked at noise performance for both state-space and observer-based controllers.



We focused on sensing (measurement) noise at the plant's output.

Hybrid Representation

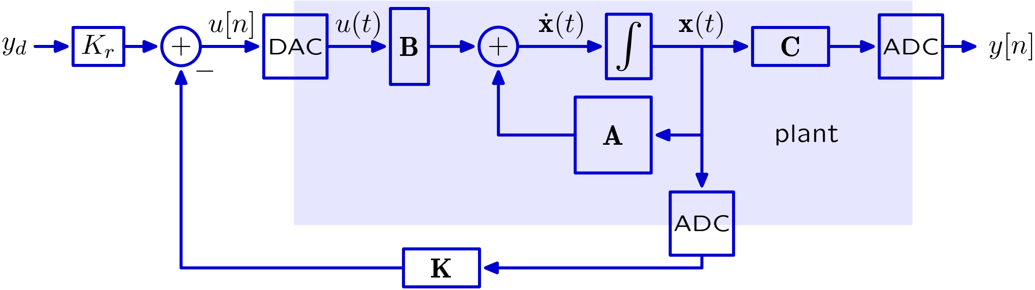
Using discrete-time control of a continuous-time plant.



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

Hybrid Representation

Using discrete-time control of a continuous-time plant.



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

We use an analog-to-digital converter to create a discrete-time representation of the state and a digital-to-analog converter to reconstruct a continuous-time representation of the command $u(t)$.

Analog-To-Digital Conversion

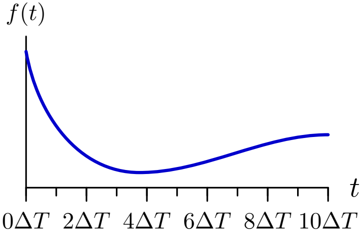
Analog-to-digital conversion entails two types of transformations.

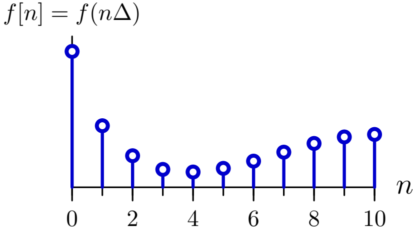
Sampling: process by which a function of real domain is transformed into a function of integer domain.

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

Sampling

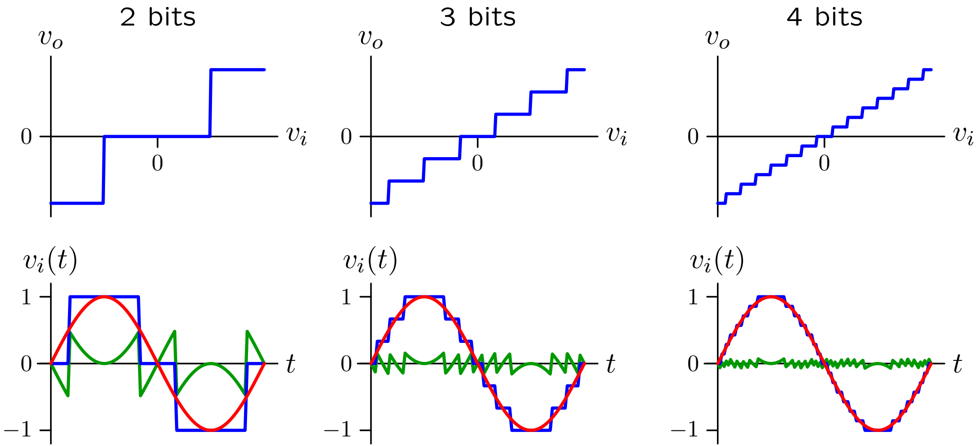
A function of real domain is transformed into a function of integer domain.





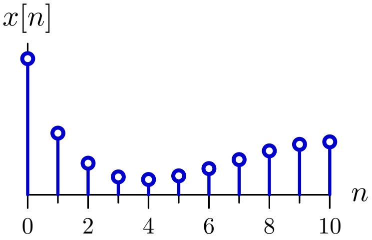
Quantization

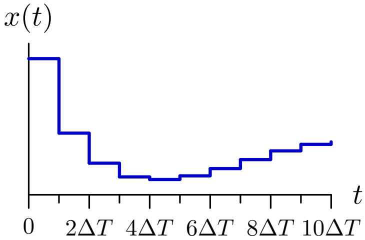
Quantization: process by which a continuous range of amplitudes is represented by a finite range of integers.



Digital-To-Analog Conversion

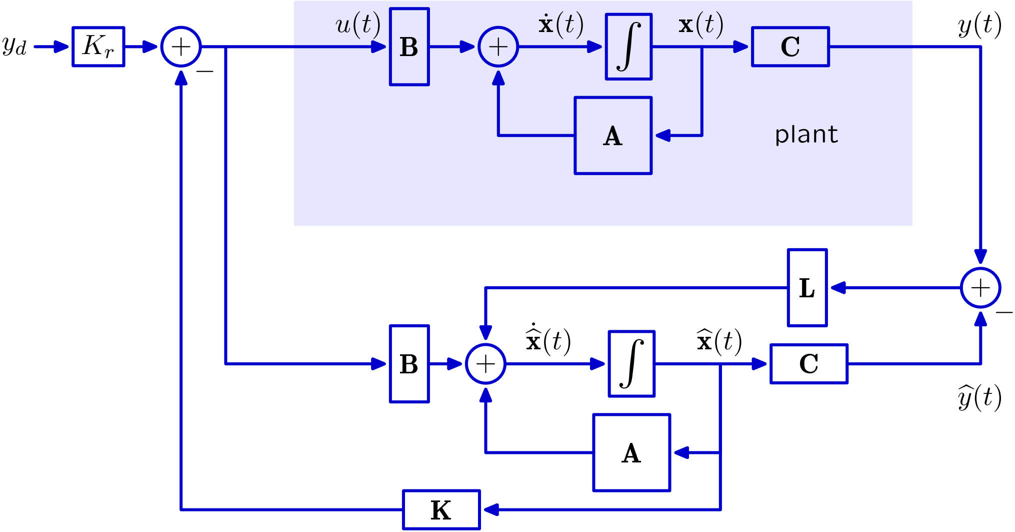
Digital-to-analog conversion **reconstructs** an analog signal from its digital representation. **zero-order hold**





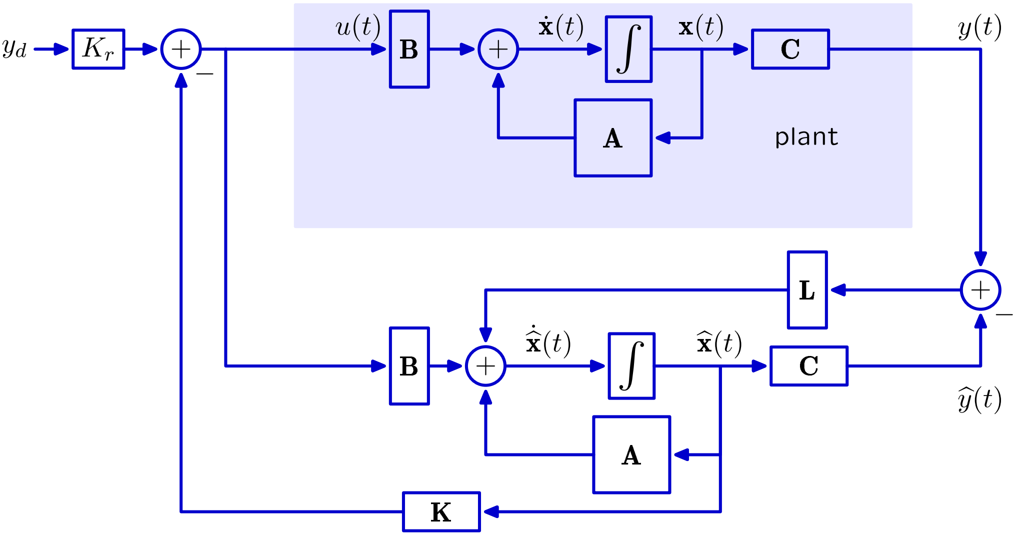
Hybrid Representations for Observer-Based Controllers

Even more changes are needed for hybrid control of observers.



Check Yourself

What must be changed to convert the controller to discrete time?



Discrete-Time State Evolution

Start by considering the scalar case: $\mathbf{x} = x$, $\mathbf{A} = a$, $\mathbf{B} = b$, and $\mathbf{C} = c$.

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:

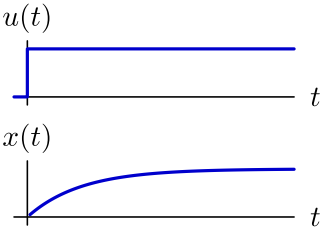
$$x(t) = \alpha e^{\beta t} + \gamma$$

Substituting into the plant equation:

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

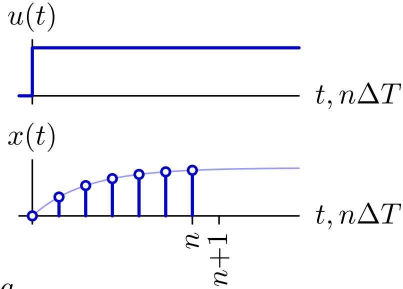
shows that $\beta = a$ and $\gamma = -bu(t)/a$ so that

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



Discrete-Time State Evolution

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



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

$$x(n\Delta T) = \alpha e^{an\Delta T} - bu(t)/a \quad \rightarrow \quad \alpha = \frac{x(n\Delta T) + bu(t)/a}{e^{an\Delta T}}$$

$$x((n+1)\Delta T) = \alpha e^{a(n+1)\Delta T} - bu(t)/a = \frac{x(n\Delta T) + bu(t)/a}{e^{an\Delta T}} e^{a(n+1)\Delta T} - bu(t)/a$$

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

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

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.

Check Yourself

Without using a computer, determine which (if any) of the matrices on the right is the exponential of the matrix on the left.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad \begin{bmatrix} 2e & 0 \\ 0 & e \end{bmatrix}$$

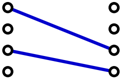
$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad\qquad \begin{bmatrix} e & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad\qquad \begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix}$$

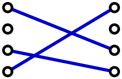
$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad \begin{bmatrix} e & e \\ 0 & e \end{bmatrix}$$

Which diagram below (if any) shows all of the valid matches?

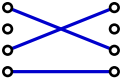
1.



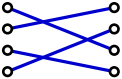
2.



3.



4.



5.

none

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:

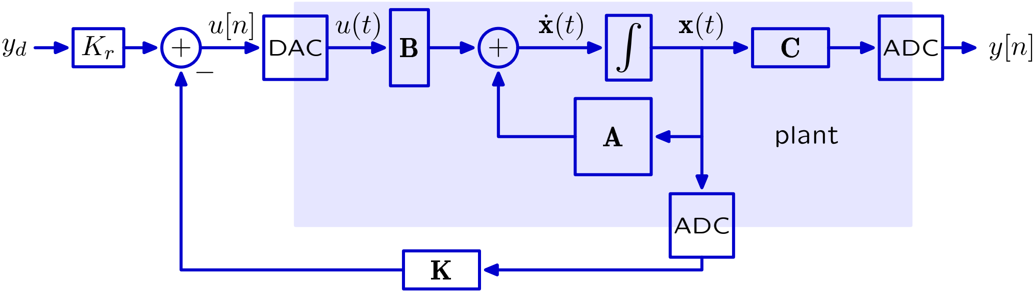
```
K=lqr(A,B,Q,R)
L=lqr(A.',B.',Q,R)
```

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.

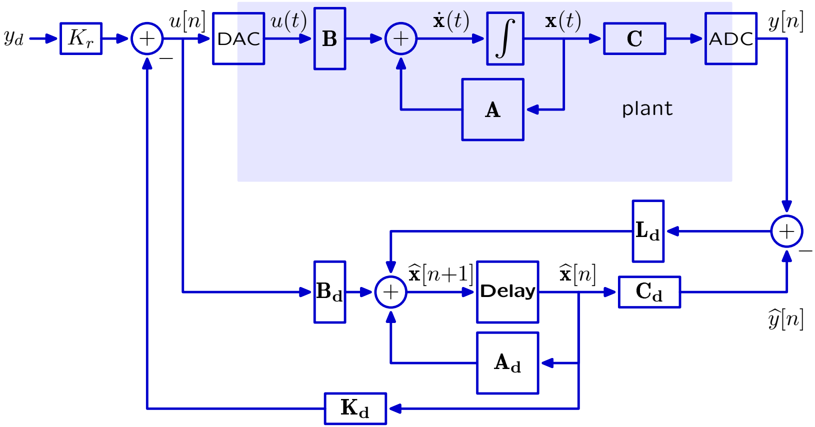


Which of the following values of $\mathbf{K}$ will work best if $\Delta T = 0.1$ ms?

```
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.



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)
```

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.