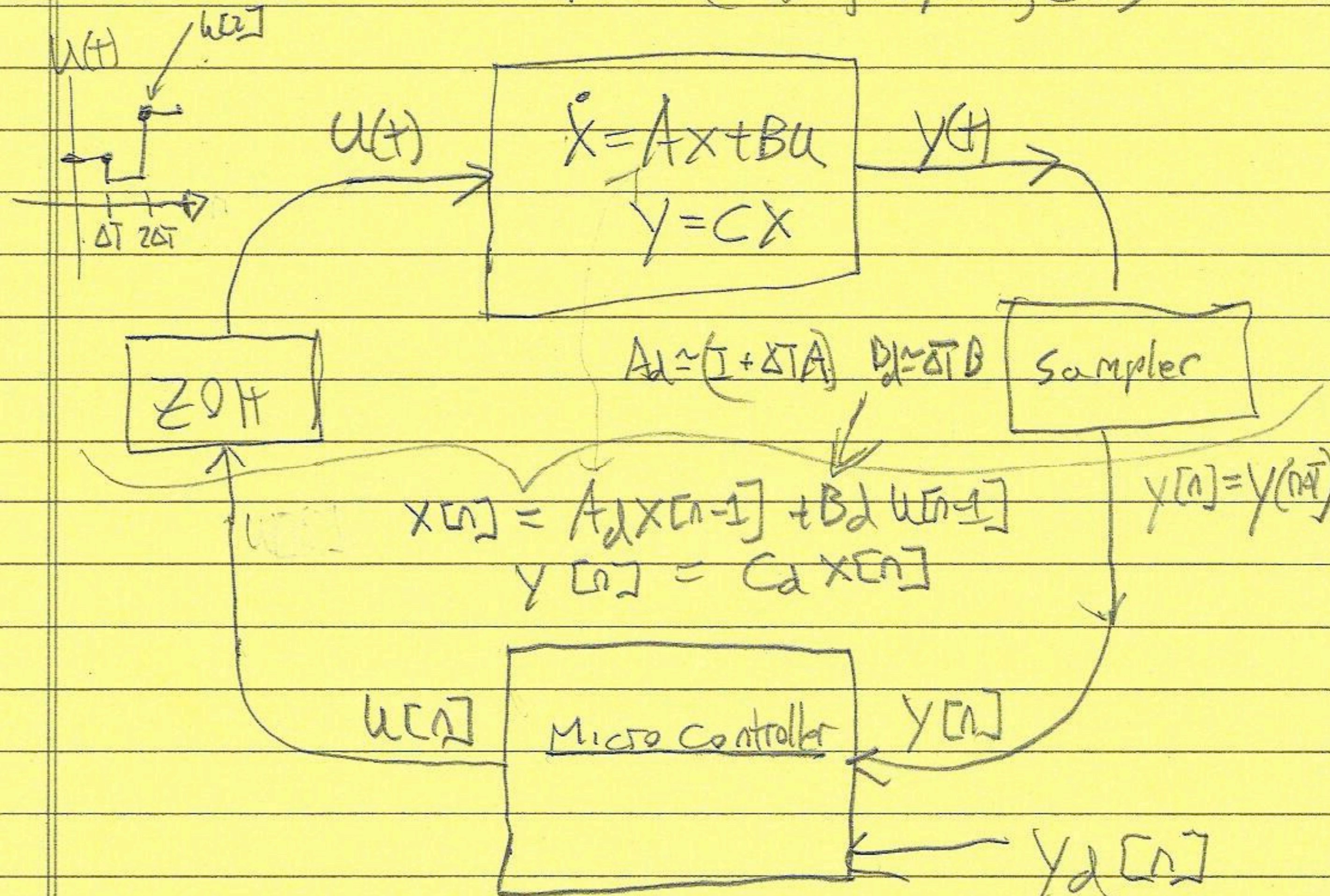
5/5/25 6.3100/2 ①

Additional Slides from Fall 2024 after page 9.

"Plant" (Maglev, Prop Arm, etc)



Micros algorithms for $u[n]$

PD. Control

$$u[n] = K_p\left(y_d[n] - y[n]\right) + K_d\left(\frac{y_d[n]-y[n]}{\Delta T} - \frac{y_d[n-1]-y[n-1]}{\Delta T}\right)$$

Depends on $\rightarrow$ ($y[n], y_d[n], y[n-1], y_d[n-1]$)

PID Control

$y[j], y_d[j] \quad j = n, n-1, n-2, \ldots \qquad + K_i \sum_{j=0}^{n} \Delta T \left(y_d[j] - y[j]\right)$

Depends on $\rightarrow$ ($y[n], y_d[n], \ldots\ldots, y[0], y_d[0]$)

Reminder

D.T. State-Space Model

$\dot{x} = Ax + Bu$
$y = Cx$

$$x[n] = A_d\, x[n-1] + B_d\, u[n-1]$$

$A_d = e^{A\Delta T} \approx I + \Delta T A$

$$y[n] = C\, x[n]$$

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

as $\Delta T \to 0$ $e^{A\Delta T} - I \to \Delta T A$

$\Delta T A A^{-1} B = \Delta T B$

Start with a CT State-Space Model

$\rightarrow$ Convert computationally to D.T.

State Feedback Control

$$u[n] = K_{r_d}\, y_d[n] - K_d\, x[n]$$

Closed-Loop System

$$x[n] = (A_d - B_d K_d)\, x[n-1] + B_d K_{rd}\, y_d[n-1]$$

If $y_d[n] = 0$

$$x[n] = (A_d - B_d K_d)^n\, x[0]$$

Pick K_d so $x[n] \to 0$ as fast as possible?

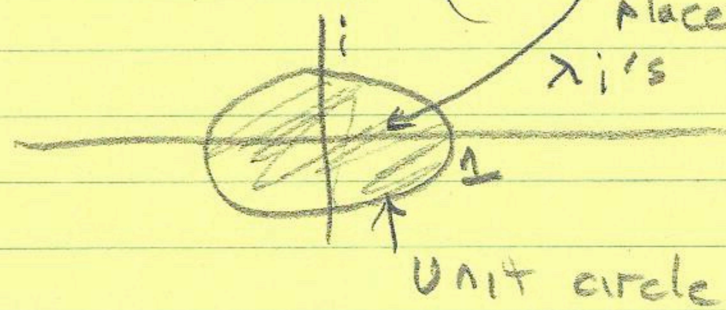
Reminder

Picking Kd

Pole Placement

$$\min_{K_d} \left(\max_i \left| \lambda_i (A_d - B_d K_d) \right| \right)$$

place λ_i's well inside unit circle



Hard to tie pole placement to metrics

LQR for D. T.

Determine Kd that minimizes

Diagonal LQR

$$\sum_{m=0}^{\infty} \left[\left(\sum_{i=1}^{\#states} q_i x_i^2[m] \right) + \left(\sum_{j=1}^{\#inputs} r_j u_j^2[m] \right) \right]$$

Pick q's & r's $\Rightarrow$ compute Kd's!

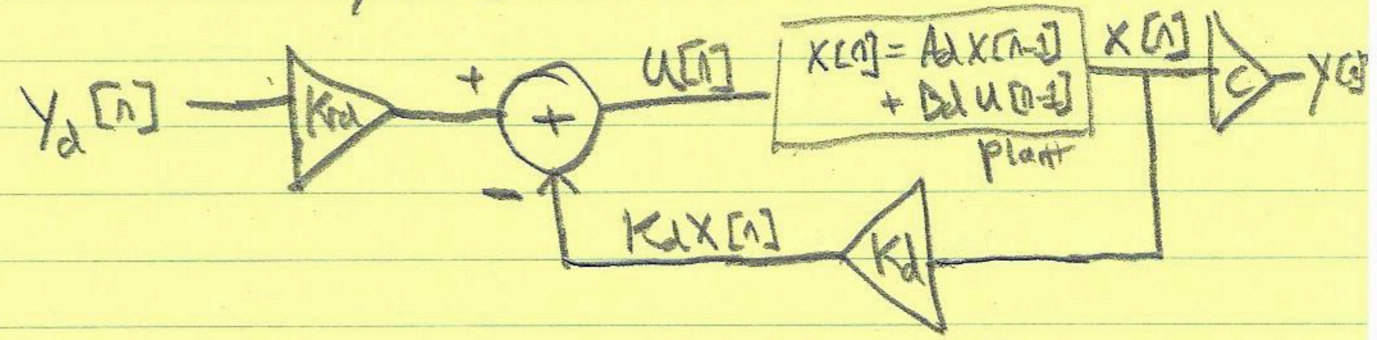
If ΔT is small enough

$K_{d_{lqr}} \simeq K_{lqr}$ for equal q's and r's

4

K_{rd}?

$$x[n] = A_d x[n-1] + B_d u[n-1]$$
$$y[n] = C x[n]$$



$$u[n] = K_{rd}\, y_d[n] - K_d x[n]$$

Closed-Loop

$$x[n] = (A_d - B_d K_d)\, x[n-1] + B_d K_{rd}\, y_d[n-1]$$

In steady-state (if exists) $\quad y_d[n] \to y_d[\infty]$

want $y_\infty = y_{d\infty}$ $\quad x_d[n] \to x[\infty]$

$$x_\infty = (A_d - B_d K_d)\, x_\infty + B_d K_{rd}\, y_\infty$$

$$(I - (A_d - B_d K_d))\, x_\infty = B_d K_{rd}\, y_\infty$$
$$x_\infty = (I - (A_d - B_d K_d))^{-1} B_d K_{rd}\, y_\infty$$

$$y_\infty = C x_\infty = C(I - (A_d - B_d K_d))^{-1} B\, \underline{K_d}\, y_{d\infty}$$
$$K_d = (C(I - (A_d - B_d K_d))^{-1} B)^{-1}$$

Observer Approach

suppose states hard to Measure

Another way to use past history $y[n], y[n-1], \ldots$
+
Incorporate a model!

Estimator (or Observer)

hat for estimate $\rightarrow$

$$\hat{x}[n] = A_d \hat{x}[n-1] + B_d u[n-1] + L_d \left(\underbrace{y[n-1]}_{\text{Measured } y} - \underbrace{\hat{y}[n-1]}_{\text{Estimated } y} \right)$$

Estimated state

Correction term

Estimated y
$\hat{y} = C\hat{x}[n-1]$

LQR Controller using estimate

$K, K_{rd} \Leftarrow$ LQR Gains

Use estimated state to compute u

Observer based u $\rightarrow$

$$\hat{u}^o[n] = K_r \, y_d[n] - K \hat{x}[n]$$

$[\leftarrow k \rightarrow]$ $\begin{bmatrix} \uparrow \\ \hat{x} \\ \downarrow \end{bmatrix}$

History is used to update observer

Observer & Controller only measure $y[n]$! Not $x[n]$

Observer Algorithm on Micro

Every ΔT

1) Use predicted state to compute control

$$u^{o} \Leftarrow K_{rd} y_d - K_d \hat{x}$$

Observer based u

Assume time to compute is << ΔT

2) Read In y from "plant" compare to predicted

$$\text{correction} \Leftarrow L_d (y - \hat{y})$$

3) Update predictions for next time

3.1) $\hat{x} \Leftarrow A\hat{x} + B_d u^{o} + \text{correction}$

3.2) $\hat{y} \Leftarrow C\hat{x}$

4) Output u^{o} to "plant."

Wait ΔT

$$\text{Analyze} \quad \overbrace{X^o[n] - \hat{X}[n]}^{\text{Estimation error}}$$ (7)

How good is the estimate

Actual State using observer-based control

(1) $X^o[n] = A_d X^o[n-1] + B_d u[n-1]$

(2) $\hat{X}[n] = A_d \hat{X}[n-1] + B_d u[n-1]$

$$+ \underbrace{L_d \left(C_d X^o[n-1] - C_d \hat{X}[n-1] \right)}_{L_d C_d \left(X^o[n-1] - \hat{X}[n-1] \right)}$$

Define state estimate error with obser control

$$X^o[n] - \hat{X}[n] \equiv \underbrace{e[n]}_{\text{state est. error}} \qquad \hat{X}[n] = X^o[n] - e[n]$$

(1) - (2)

$$\underbrace{X^o[n] - \hat{X}[n]}_{e[n]} = A_d \overbrace{\left(X^o[n-1] - \hat{X}[n-1] \right)}^{e[n-1]} - L_d C_d \underbrace{\left(X^o[n-1] - \hat{X}[n-1] \right)}_{e[n-1]}$$

$$e[n] = \left(A_d - L_d C_d \right) e[n-1]$$

State estimation error indep of Controller gains (K_{rd}, K_d)

actual state when using Observer control — estimated state

actual state when using Measured state control

How close is $X^{o}[n]$ to $X[n]$ 8

Using Observer-Based Control

(3) $X^{o}[n] = A_d X^{o}[n-1] + B_d\left(K_{rd} Y_d[n-1] - K_d \hat{X}[n-1]\right)$

Actual state with observer control — estimated state

estimated state

(3.5) $\hat{X}[n-1] = X^{o}[n-1] - e[n-1]$

Using (3.5) in (3) $\Rightarrow$ $X^{o}[n] = (A_d - B_d K_d) X^{o}[n-1]$
$+ B_d K_d e[n-1]$
$+ B_d K_{rd} Y_d[n-1]$

Using Measured States control

(4) $X[n] = (A_d - B_d K_d) X[n-1] + B_d K_{rd} Y_d[n-1]$

$u[n-1] = K_{rd} Y[n-1] - K_d X[n-1]$

Subtracting (3) - (4)

$(X^{o}[n] - X[n]) = (A_d - B_d K_d)(X^{o}[n-1] - X[n-1])$
$+ B_d K_d e[n]$

actual states

Observer state control "like" measured state control if

1) $(A_d - B_d K_d)^n \rightarrow 0$ fast with n

& if $x^0[0] \neq \hat{x}[0]$ then errors $\rightarrow 0$ fast when estimator is good

2) $e[n] \rightarrow 0$

(faster

6.3100: Dynamic System Modeling and Control Design

DT Observer

December 02, 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:

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

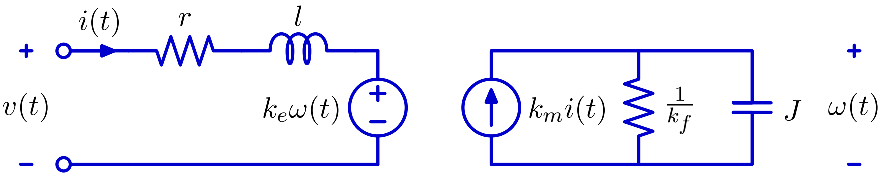
We also looked at the sensitivity of the controllers to **noise**.

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.

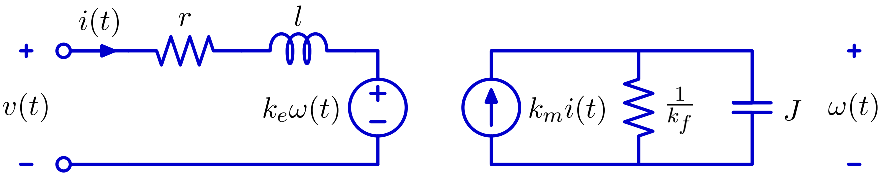
It excites a current $i(t)$, which generates a torque $k_m i(t)$ that tends to rotate the motor shaft.

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.

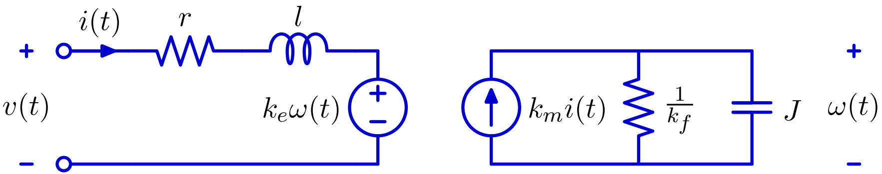


Electrical port

Mechanical port

Motor Speed Control: Mathematical Representation

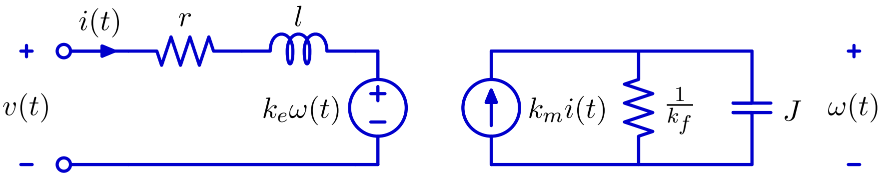
Simple circuit analysis provides a mathematical representation.



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

Motor Speed Control: Matrix Representation

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



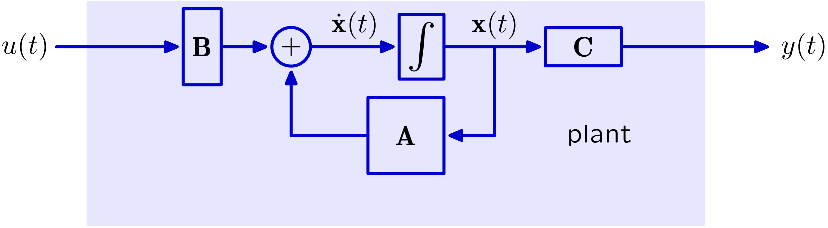
$$v(t) = ri(t) + l\frac{di(t)}{dt} + k_e\omega(t) \qquad k_m i(t) = k_f\omega(t) + J\frac{d\omega(t)}{dt}$$

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

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

State-Space Model

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

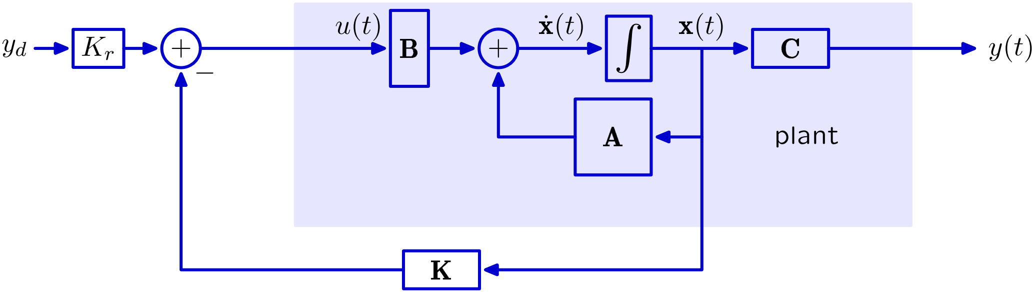


$$\frac{d}{dt}\underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} = \underbrace{\begin{bmatrix} -\frac{r}{l} & -\frac{k_e}{l} \\ \frac{k_m}{J} & -\frac{k_f}{J} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix}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

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

State-Space Model + 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**:

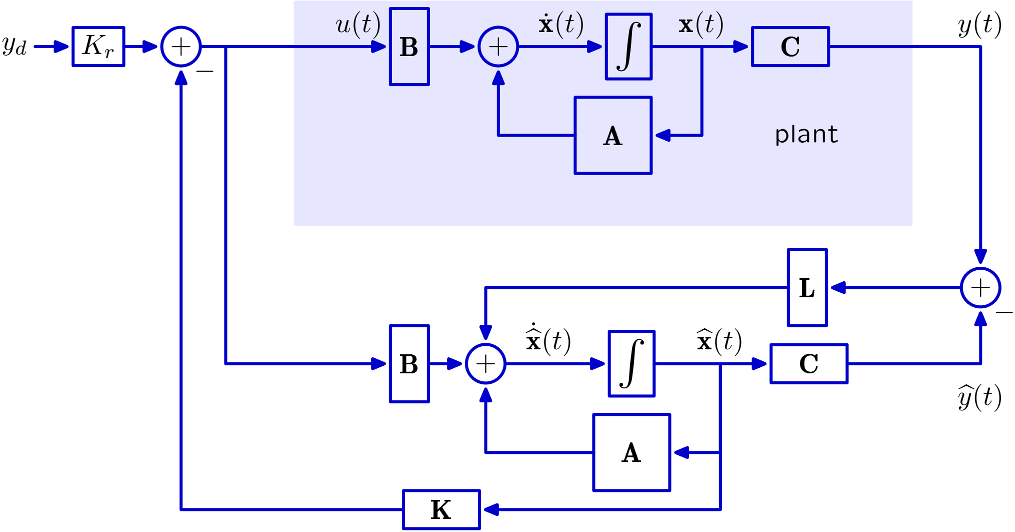
`K = lqr(A,B,Q,R)` where `Q = diag([penalty1,penalty2])` and `R = 1`

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

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

State-Space Model + Observer

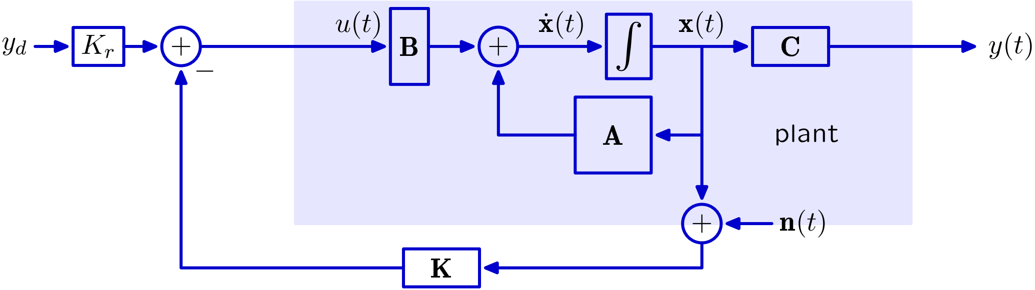
We also analyzed the performance of an observer-based controller.



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

Effects of Sensor Noise

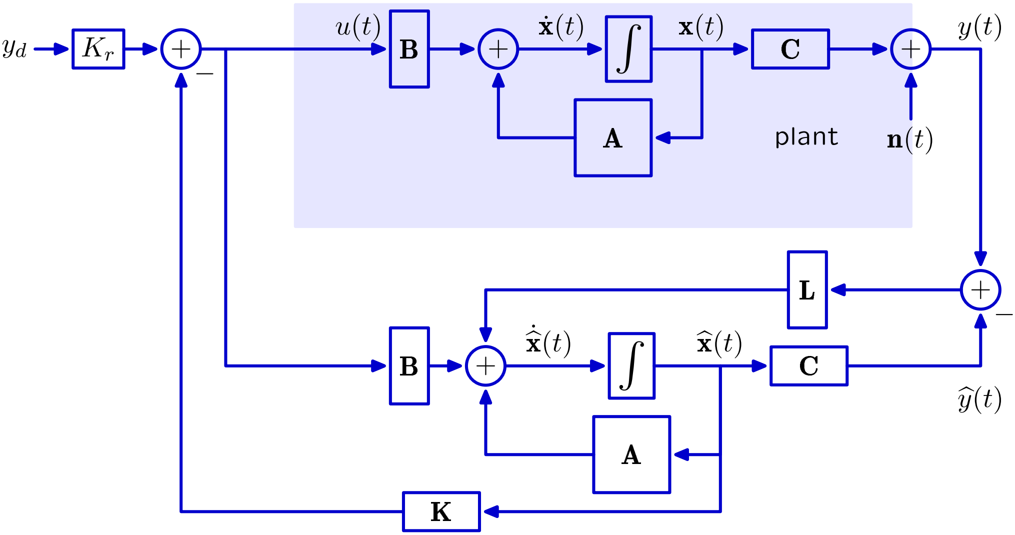
We looked at noise performance for both state-space and observer-based controllers.



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

Effects of Sensor Noise

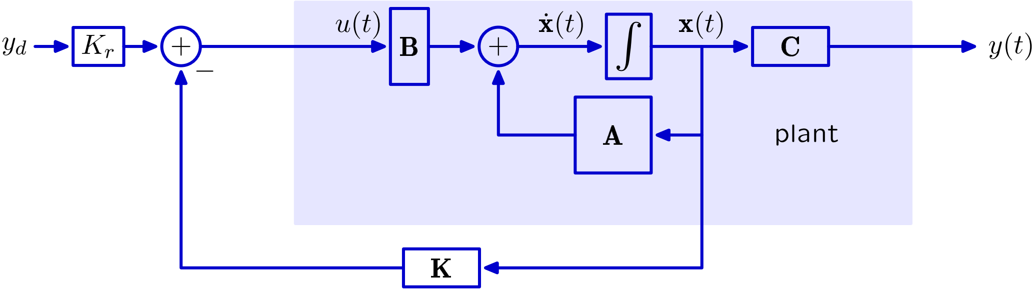
We looked at noise performance for both state-space and observer-based controllers.



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

Hybrid Representation

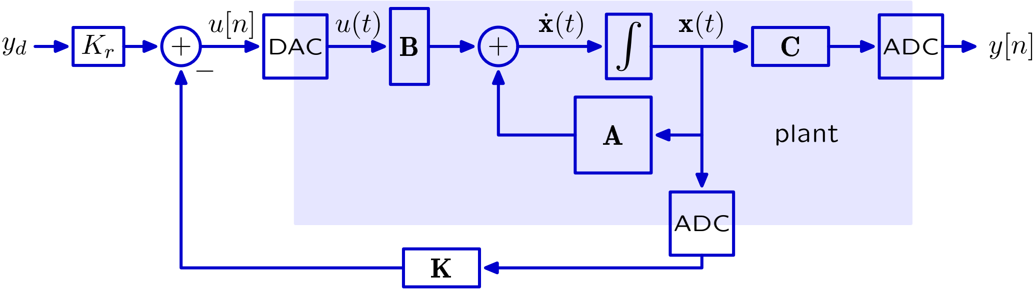
Using discrete-time control of a continuous-time plant.



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

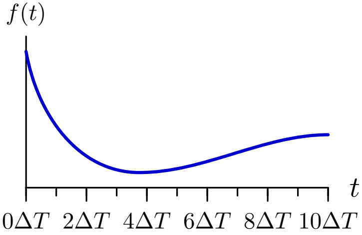
Analog-to-digital conversion entails two types of transformations.

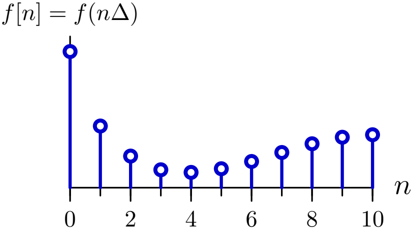
Sampling: process by which a function of real domain is transformed into a function of integer domain.

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

Sampling

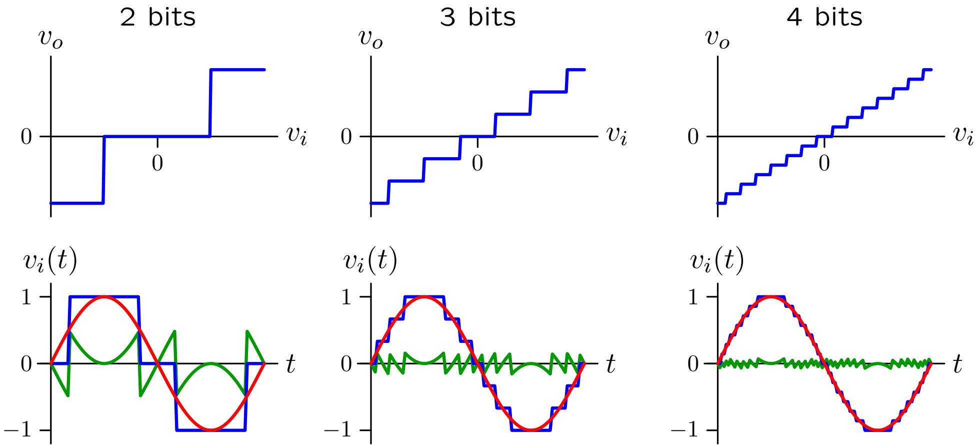
A function of real domain is transformed into a function of integer domain.





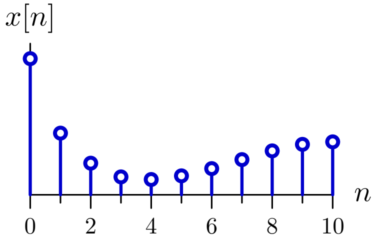
Quantization

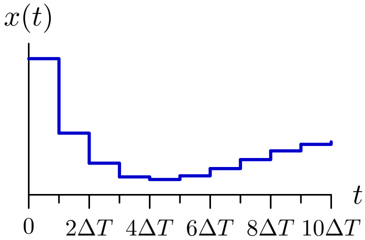
Quantization: process by which a continuous range of amplitudes is represented by a finite range of integers.



Digital-To-Analog Conversion

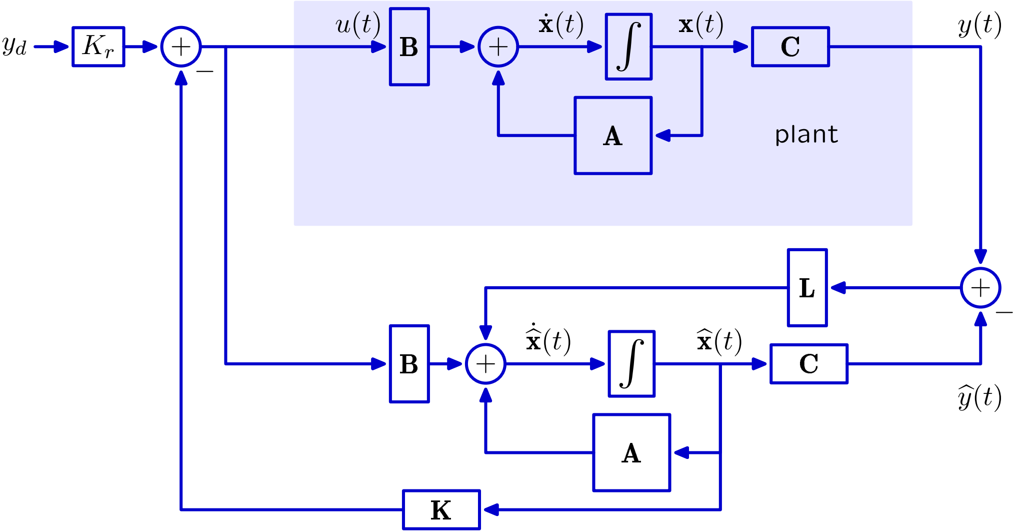
Digital-to-analog conversion **reconstructs** an analog signal from its digital representation. **zero-order hold**





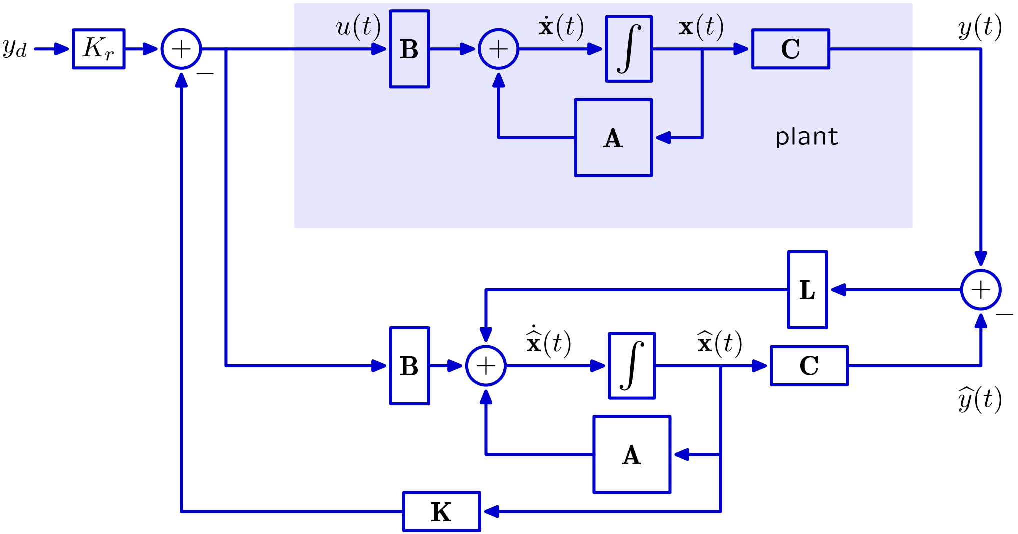
Hybrid Representations for Observer-Based Controllers

Even more changes are needed for hybrid control of observers.



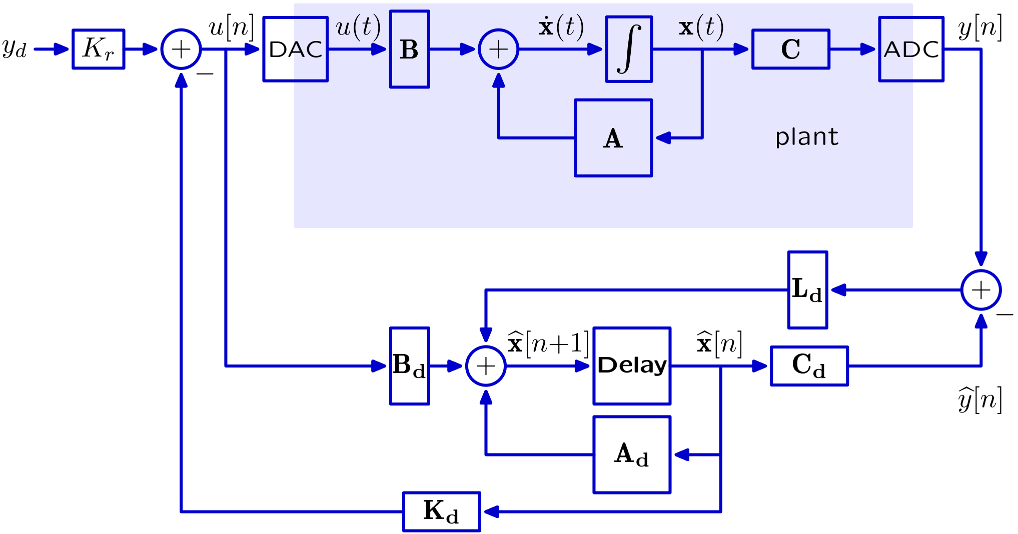
Check Yourself

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



Check Yourself

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



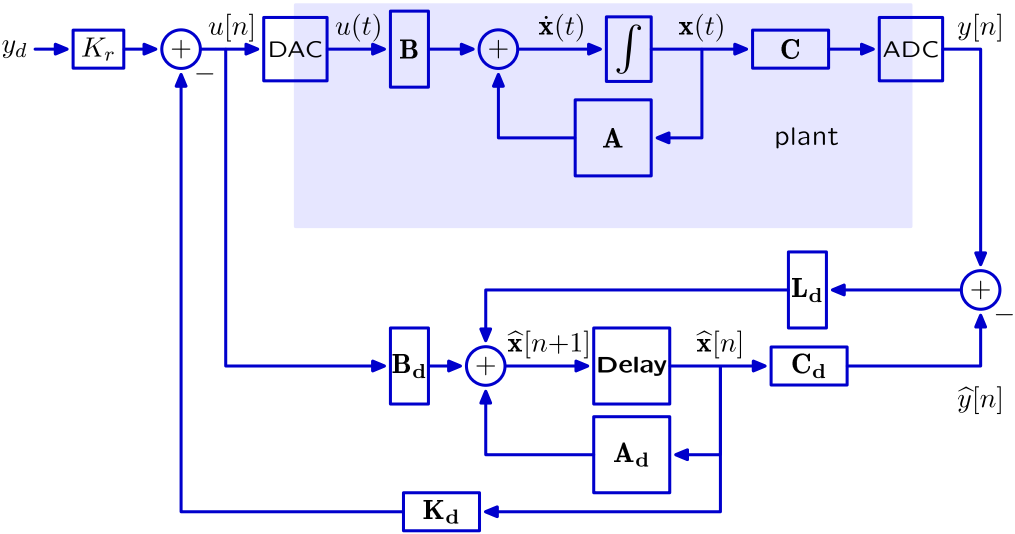
Signals outside plant must be discrete time: $y(t) \rightarrow y[n]$; $u[n] \rightarrow u(t)$.

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

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

Check Yourself

What must be changed to convert the controller 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: $\mathbf{x} = x$, $\mathbf{A} = a$, $\mathbf{B} = b$, 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:

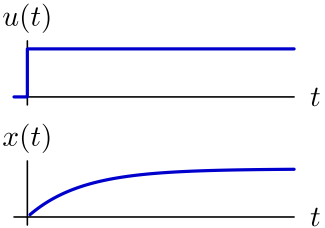
$$x(t) = \alpha e^{\beta t} + \gamma$$

Substituting into the plant equation:

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

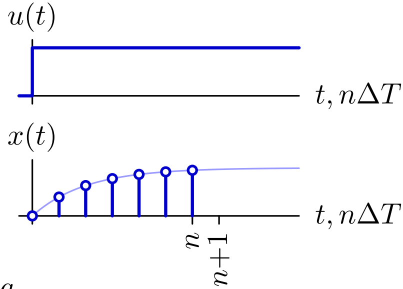
shows that $\beta = a$ and $\gamma = -bu(t)/a$ so that

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



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):

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

Discrete version of state evolution equation:

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

where

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

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

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.

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

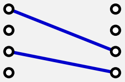
$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad\qquad \begin{bmatrix} e & 0 \\ 0 & 0 \end{bmatrix}$$

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

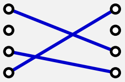
$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad \begin{bmatrix} e & e \\ 0 & e \end{bmatrix}$$

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

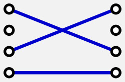
1.



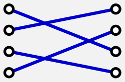
2.



3.



4.



5.

none

Matrix Exponentials

Let $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Find $e^{\mathbf{A}} = \mathbf{I} + \frac{1}{1!}\mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \frac{1}{4!}\mathbf{A}^4 + \cdots$

$$\begin{aligned}
e^{\mathbf{A}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^2 + \frac{1}{3!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^3 + \frac{1}{4!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^4 + \cdots \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{3!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{4!}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \cdots \\
&= \begin{bmatrix} 1 + \frac{1}{1} + \frac{1}{2!} + \frac{1}{3!} + \cdots & 0 \\ 0 & 1 + \frac{1}{1} + \frac{1}{2!} + \frac{1}{3!} + \cdots \end{bmatrix} \\
&= \begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix}
\end{aligned}$$

where $e = \displaystyle\sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$

Matrix Exponentials

Let $\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$

Find $e^{\mathbf{B}} = \mathbf{I} + \dfrac{1}{1!}\mathbf{B} + \dfrac{1}{2!}\mathbf{B}^2 + \dfrac{1}{3!}\mathbf{B}^3 + \dfrac{1}{4!}\mathbf{B}^4 + \cdots$

$$\begin{aligned} e^{\mathbf{B}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}^2 + \frac{1}{3!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}^3 + \frac{1}{4!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}^4 + \cdots \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{3!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{4!}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \cdots \\ &= \begin{bmatrix} 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Matrix Exponentials

Let $\mathbf{C} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$

Find $e^{\mathbf{C}} = \mathbf{I} + \frac{1}{1!}\mathbf{C} + \frac{1}{2!}\mathbf{C}^2 + \frac{1}{3!}\mathbf{C}^3 + \frac{1}{4!}\mathbf{C}^4 + \cdots$

$$e^{\mathbf{C}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^2 + \frac{1}{3!}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^3 + \frac{1}{4!}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^4 + \cdots$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{1!}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} + \frac{1}{2!}\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} + \frac{1}{3!}\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} + \frac{1}{4!}\begin{bmatrix} 1 & 4 \\ 0 & 1 \end{bmatrix} + \cdots$$

$$= \begin{bmatrix} 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots & 0 + \frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \cdots \\ 0 & 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \end{bmatrix}$$

$$= \begin{bmatrix} 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots & 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \\ 0 & 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \end{bmatrix}$$

$$= \begin{bmatrix} e & e \\ 0 & e \end{bmatrix}$$

Matrix Exponentials

Let $\mathbf{D} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$

Find $e^{\mathbf{D}}$

$$\mathbf{D} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \mathbf{A} + \mathbf{B}$$

$$e^{\mathbf{D}} = e^{\mathbf{A}+\mathbf{B}} = e^{\mathbf{A}} e^{\mathbf{B}} = \begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix} \times \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^2 & 0 \\ 0 & e \end{bmatrix}$$

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.

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

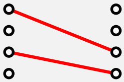
$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} e & 0 \\ 0 & 0 \end{bmatrix}$$

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

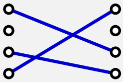
$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} e & e \\ 0 & e \end{bmatrix}$$

Which diagram below (if any) shows all of the valid matches? 1

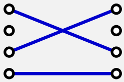
1.



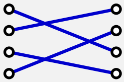
2.



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:

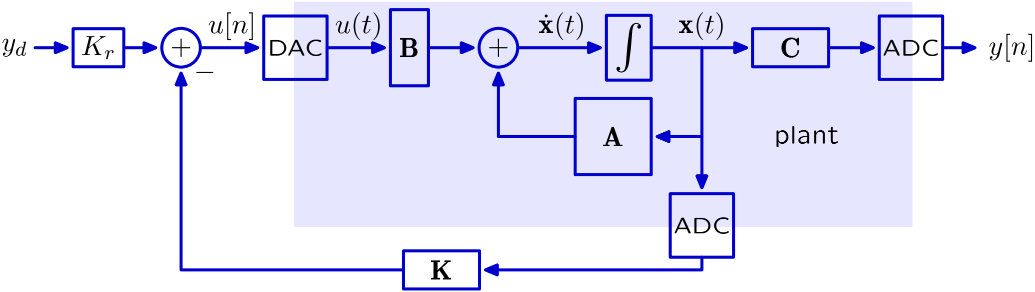
```
K=lqr(A,B,Q,R)
L=lqr(A.',B.',Q,R)
```

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.

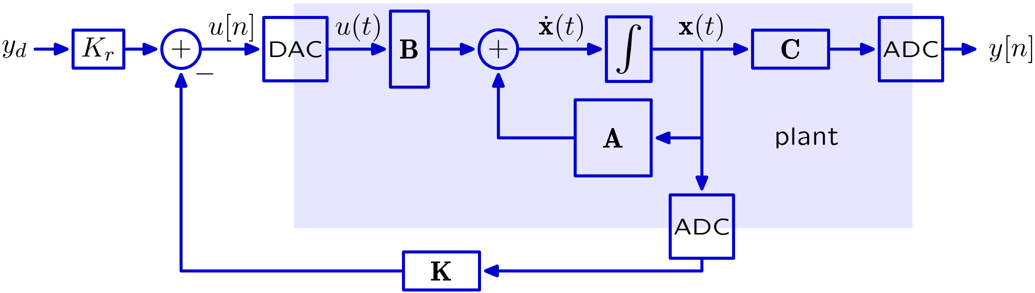


Which of the following values of $\mathbf{K}$ will work best if $\Delta T = 0.1$ ms?

```
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 a state-space controller for the motor model.



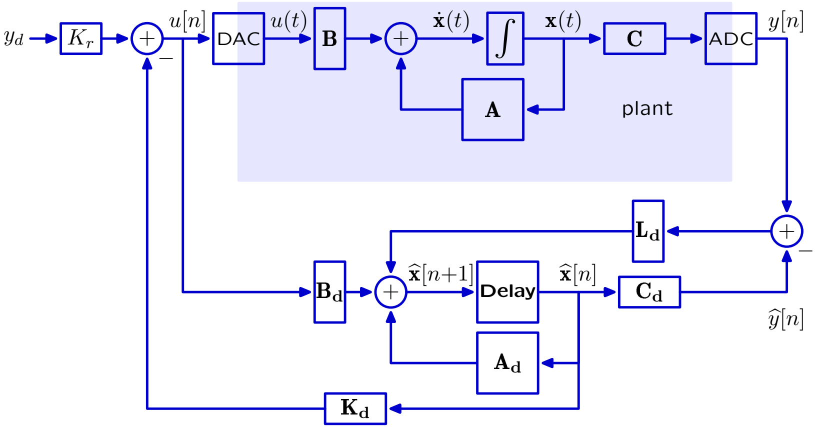
Since ΔT is small relative to the dynamics of the system, the path from $\mathbf{x}(t)$ to $u(t)$ is nearly instantaneous and can be approximated as simply adding a small amount of noise.

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.

Check Yourself

Consider an observer-based controller for the motor.

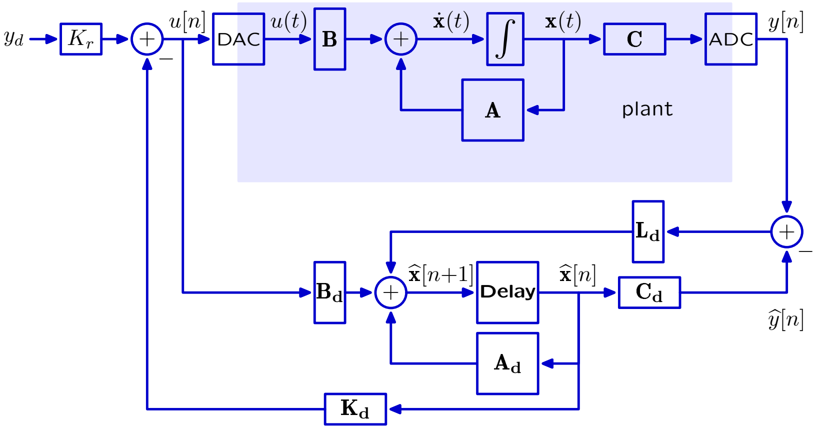


Which of the following values of $\mathbf{K_d}$ will work best?

```
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 $\mathbf{A}$ & $\mathbf{B}$.

→ `K5` doesn't make sense: cannot run `lqr` on DT matrices.

→ `K6` is the best choice.

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

→ `K3` doesn't make sense: cannot run `lqr` on DT matrices.

→ `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.