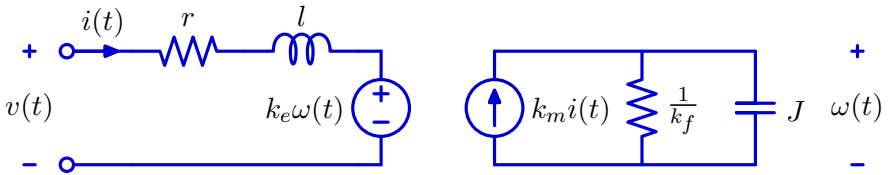
6.3100: Dynamic System Modeling and Control Design

Applications

merge with Applications_2

Motor Speed Control

Last time we designed several systems to control the speed of a motor.



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

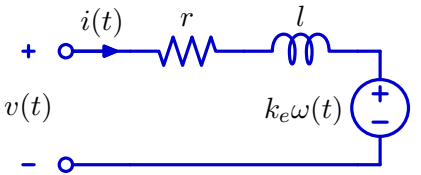
It excites a current $i(t)$, which generates a torque $k_m i(t)$ that tends to rotate the motor shaft.

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

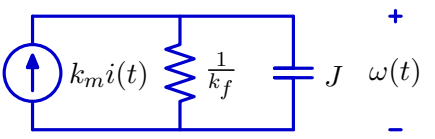
As the motor spins, it generates a back emf ($k_e\omega(t)$) that tends to reduce the electrical current $i(t)$ drawn by the motor.

Motor Speed Control: State-Space Model

We started with a circuit model of the motor.



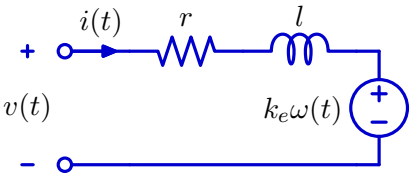
Electrical port

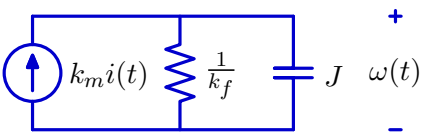


Mechanical port

Motor Speed Control: State-Space Model

We translated the ciruit model to a mathematical model.



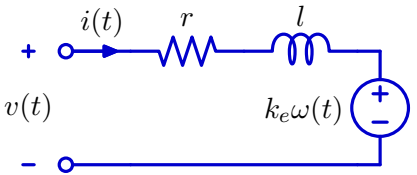


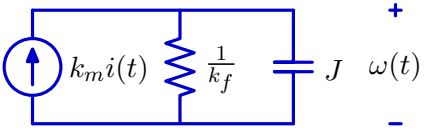
$$v(t) = ri(t) + l\frac{di(t)}{dt} + k_e\omega(t)$$

$$k_m i(t) = k_f\omega(t) + J\frac{d\omega(t)}{dt}$$

Motor Speed Control: State-Space Model

We express the result by a pair of matrix equations or by a block diagram.





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

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

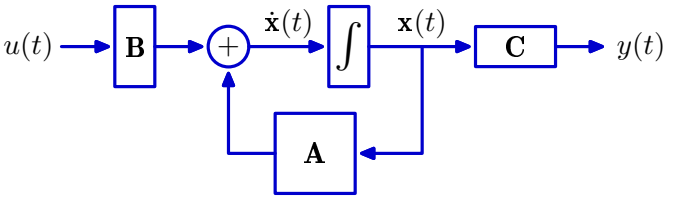
$$\frac{d}{dt}\underbrace{\mathbf{x}(t)} = \underbrace{\mathbf{A}}\underbrace{\mathbf{x}(t)} + \underbrace{\mathbf{B}}\underbrace{\mathbf{u}(t)}$$

$$\omega(t) = \begin{bmatrix} 0 & 1 \end{bmatrix}\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}$$

$$\underbrace{y(t)} = \underbrace{\mathbf{C}}\underbrace{\mathbf{x}(t)}$$

Motor Speed Control: State-Space Model

State-space model of the plant.

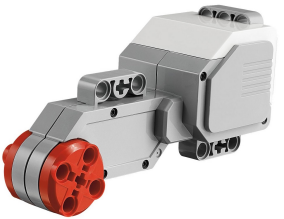


$$\frac{d}{dt}\underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} = \underbrace{\begin{bmatrix} -\frac{r}{l} & -\frac{k_e}{l} \\ \frac{k_m}{J} & -\frac{k_f}{J} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix}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

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

Parameters of the Model

Lego EV3 motor parameters.

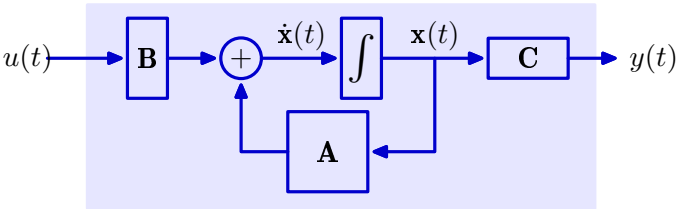


$$
\begin{aligned}
r &= 7\,\Omega \\
l &= 0.005\,\mathsf{H} \\
k_e &= 0.46\,\mathsf{volts/(radian/sec)} \\
k_m &= 0.3\,\mathsf{Nm/(radian/sec)} \\
k_f &= 0.00073\,\mathsf{Nm/(radian/sec)} \\
J &= 0.0015\,\mathsf{Nm/(radian/sec^2)}
\end{aligned}
$$

$$
A = \begin{bmatrix} -1400 & -92 \\ 200 & -0.5 \end{bmatrix} \quad B = \begin{bmatrix} 200 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}
$$

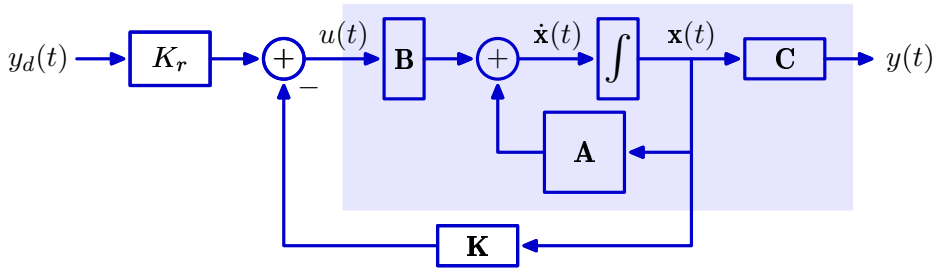
State-Space Controller

This motor 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,[poles])
```

or LQR:

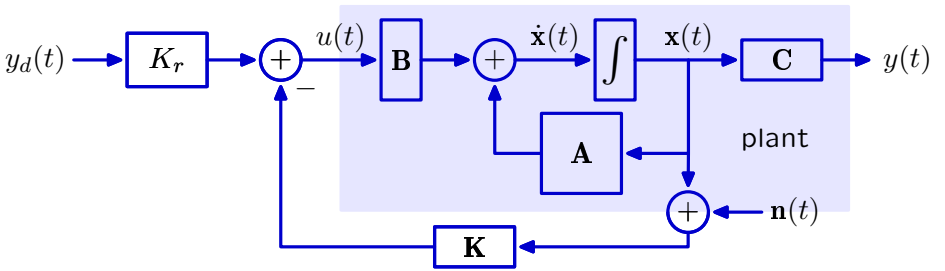
```
Q = diag([1,1,1,1])
R = 1
K = lqr(A,B,Q,R)
```

and

```
Kr = -1/(C*((A-BK)\B)
```

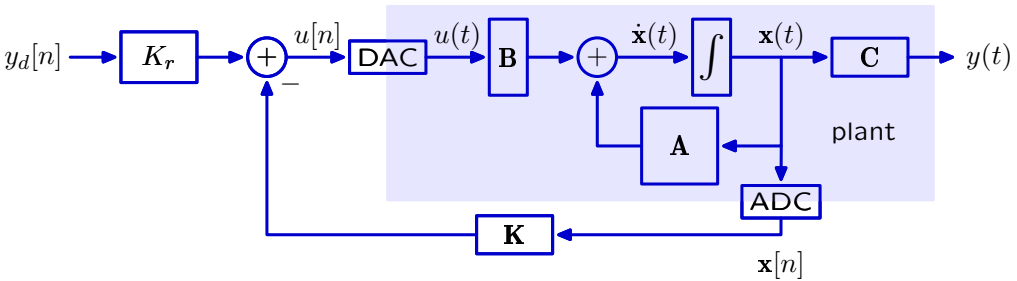
Effects of Sensor Noise

In the last lecture we analyzed how sensor noise affects performance.



Discrete-Time Control

Today we will look at a different issue that affects performance.



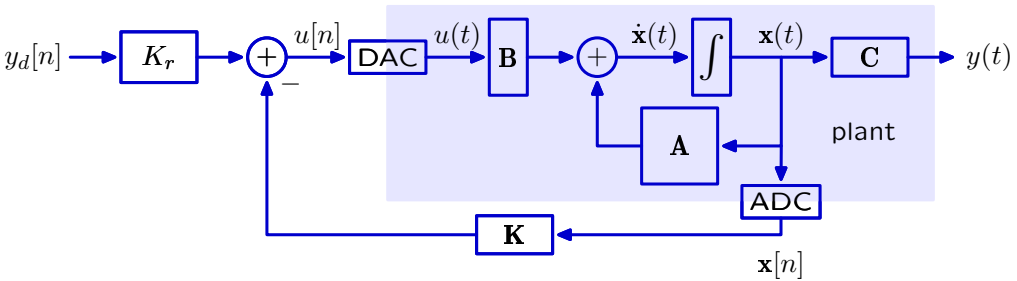
Hybrid representation: continuous-time plant with discrete-time control.

The state $\mathbf{x}(t)$ of the plant must be converted to discrete time to process in a digital controller (such as the Teensy).

The resulting discrete-time command $u[n]$ must be converted to continuous time for the plant.

Discrete-Time Control

Hybrid representation: continuous-time plant with discrete-time control.



graph(x(t)→ x[n])

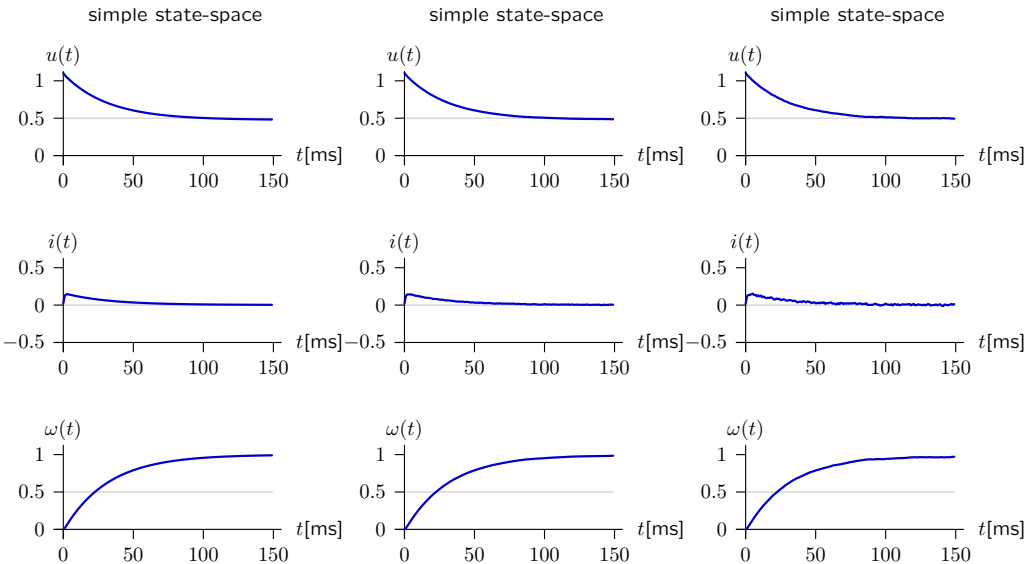
graph(u[n]→ u(t))

Instantanteous sampling: $x[n] = x(n\Delta T)$.

Zero-order hold: $u(t) = u\left[\lfloor \frac{t}{\Delta T} \rfloor\right]$.

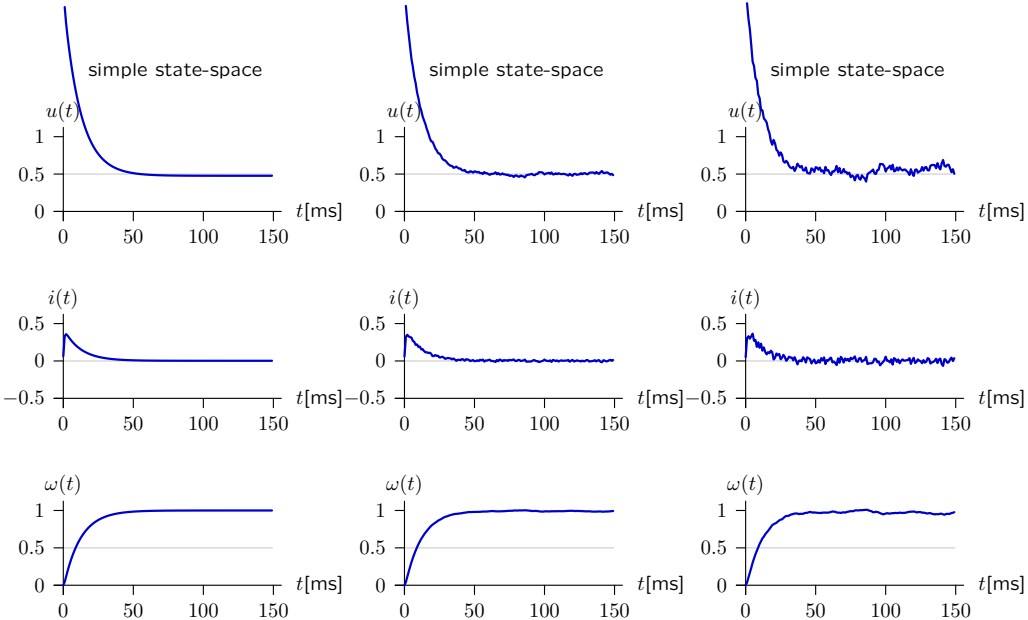
Effect of K on Noise Performance when R=1

Low (left), medium (center), and high (right) values of $\mathbf{n}(t)$.



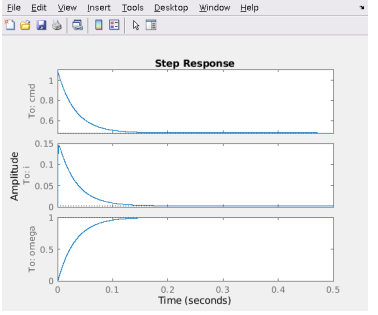
Effect of K on Noise Performance when R=0.1

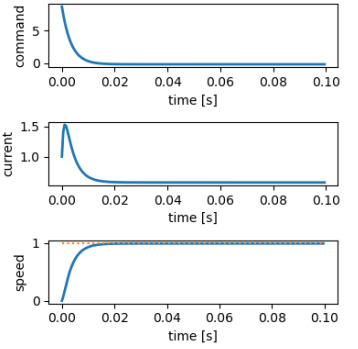
Low (left), medium (center), and high (right) values of $\mathbf{n}(t)$.

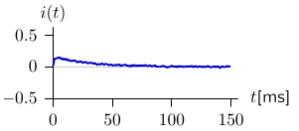


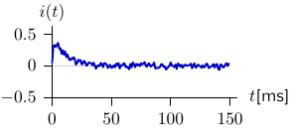
Speed/Noise Tradeoff

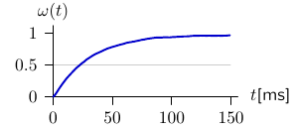
Higher gains can increase speed, but they also tend to increase noise.

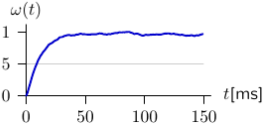





$i(t)$
0.5
0
−0.5
0
50
100
150
t[ms]

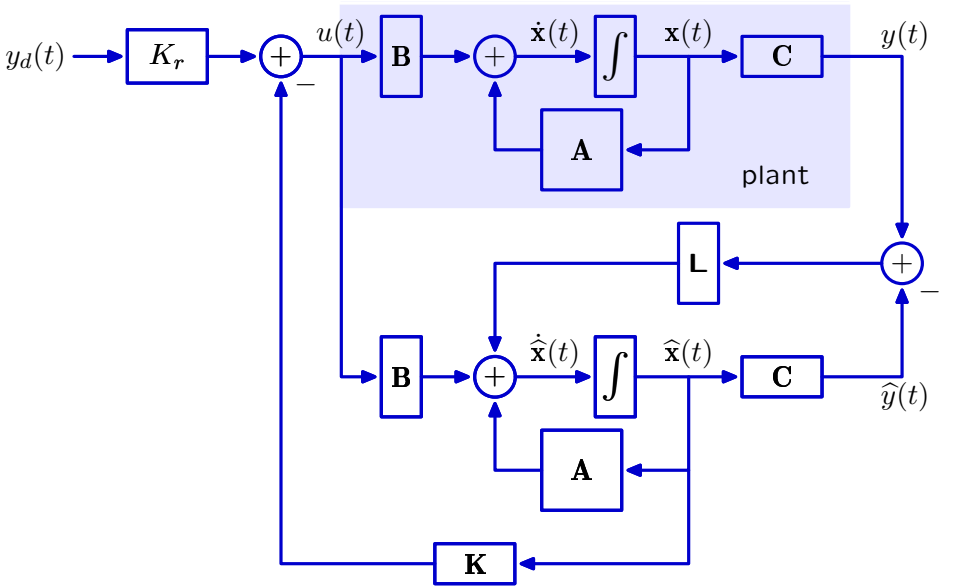

$i(t)$
0.5
0
−0.5
0
50
100
150
t[ms]


$\omega(t)$
1
0.5
0
0
50
100
150
t[ms]


$\omega(t)$
1
0.5
0
0
50
100
150
t[ms]

Design Tradeoffs with an Observer

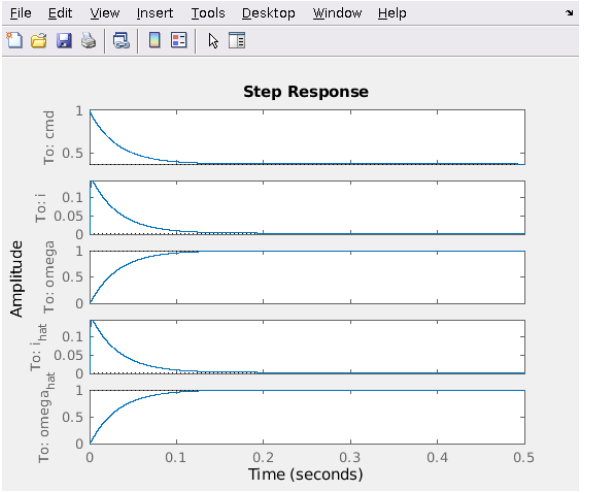
General form: Now we must choose both $\mathbf{K}$ and $\mathbf{L}$.



Choosing the Matrix L

Try LQR with `Q = diag([1,1])` and `R = 1` for both $\mathbf{K}$ and $\mathbf{L}$.

Result: `K = [0.1597 0.6305]` and `L = [-0.0024;0.0377]`

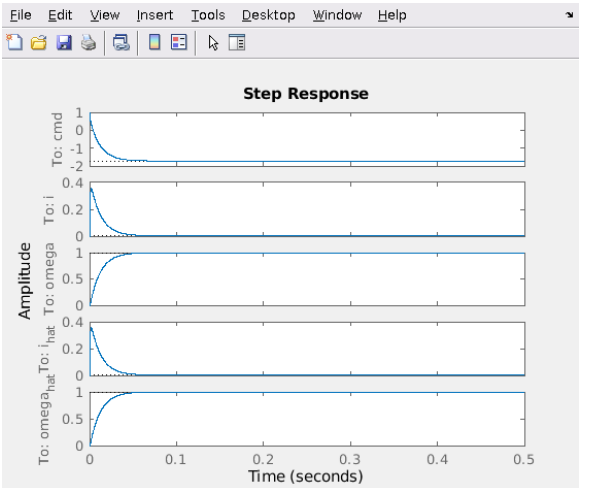


Is this good? Can we do better?

Choosing the Matrix L

Try more aggressive: `Q = diag([1,1])` and `R = 0.1` for both $\mathbf{K}$ and $\mathbf{L}$.

Result: `K = [1.0273 2.7185]` and `L = [-0.0237;0.3727]`

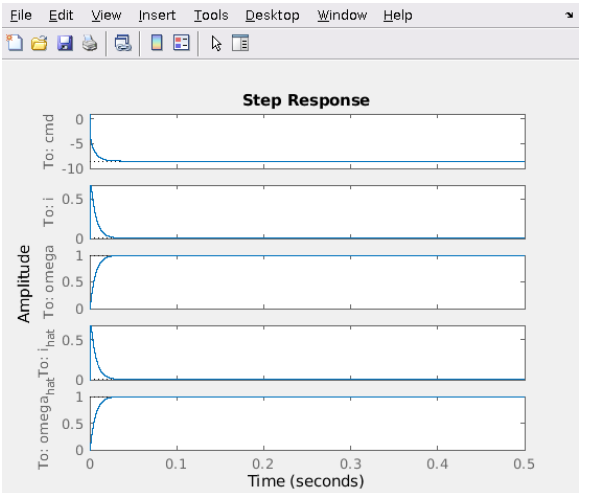


Similar to simple state-space controller: higher gain $\rightarrow$ faster response.

Choosing the Matrix L

Try more aggressive: `Q = diag([1,1])` and `R = 0.01` for both $\mathbf{K}$ and $\mathbf{L}$.

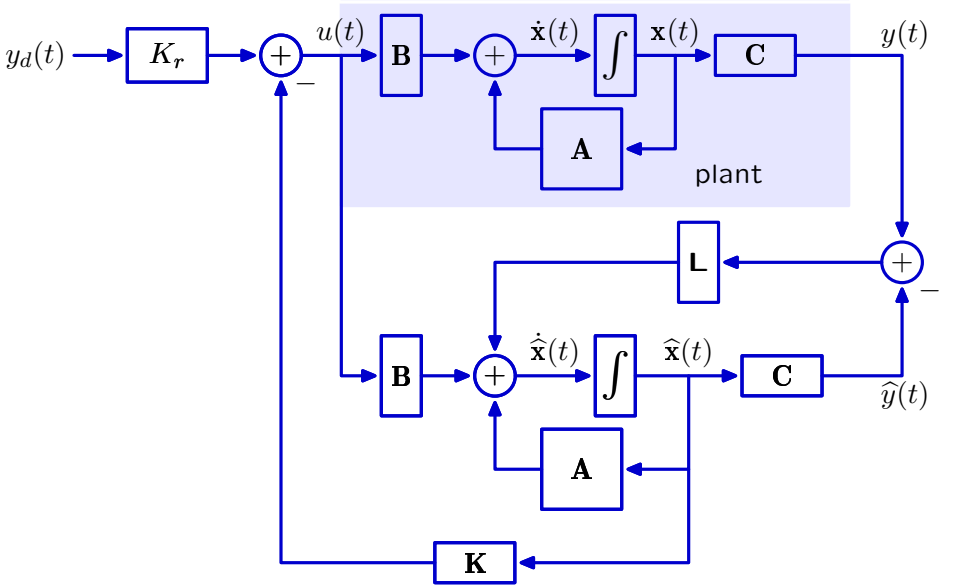
Result: `K = [5.9630 9.5199]` and `L = [-0.2135;3.3658]`



Even higher gain $\rightarrow$ even faster responses.

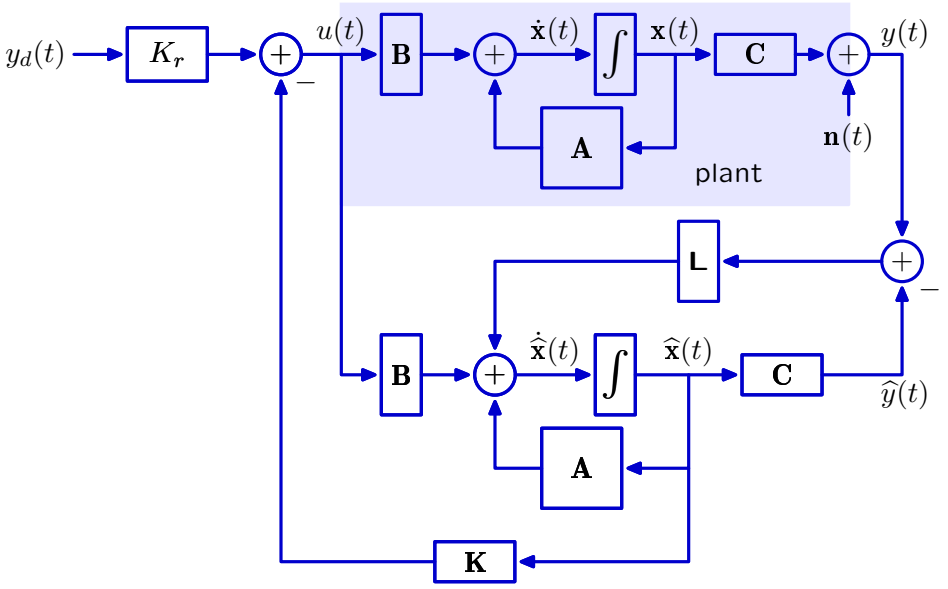
Effects of Sensor Noise

Model effects of noise on the observer system.

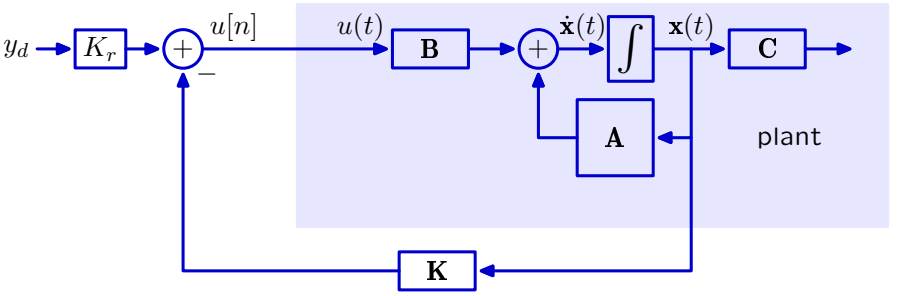


Effects of Sensor Noise

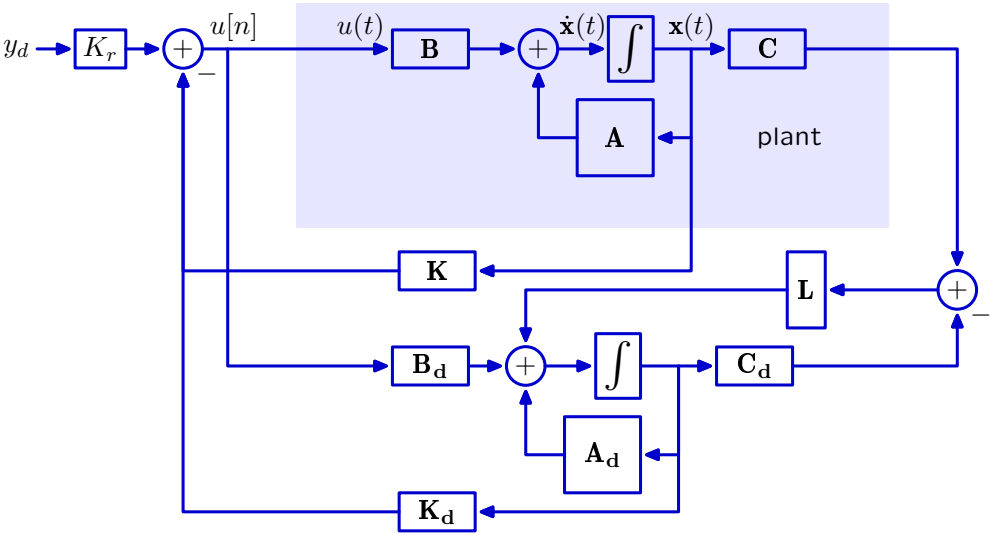
The important source of measurement noise is in the measurement of the output $y(t)$.



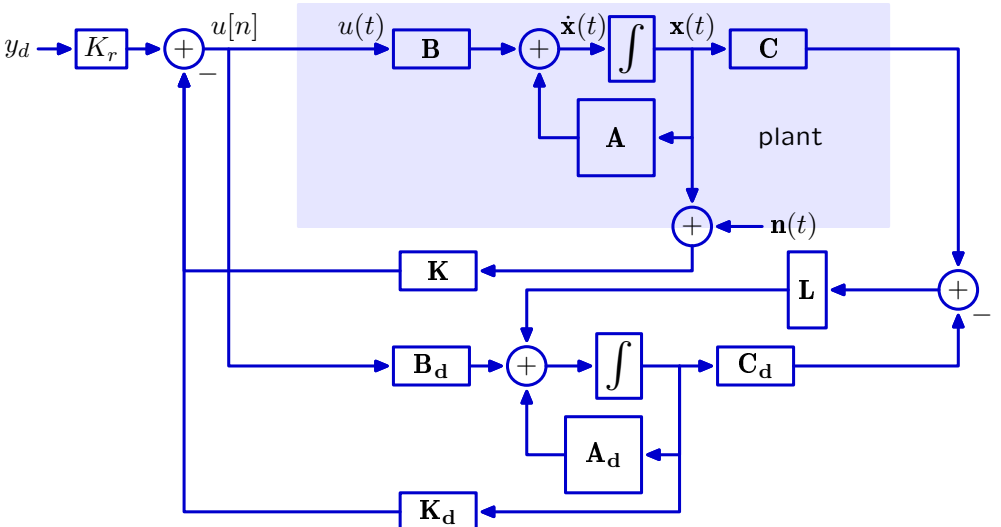
No Observer



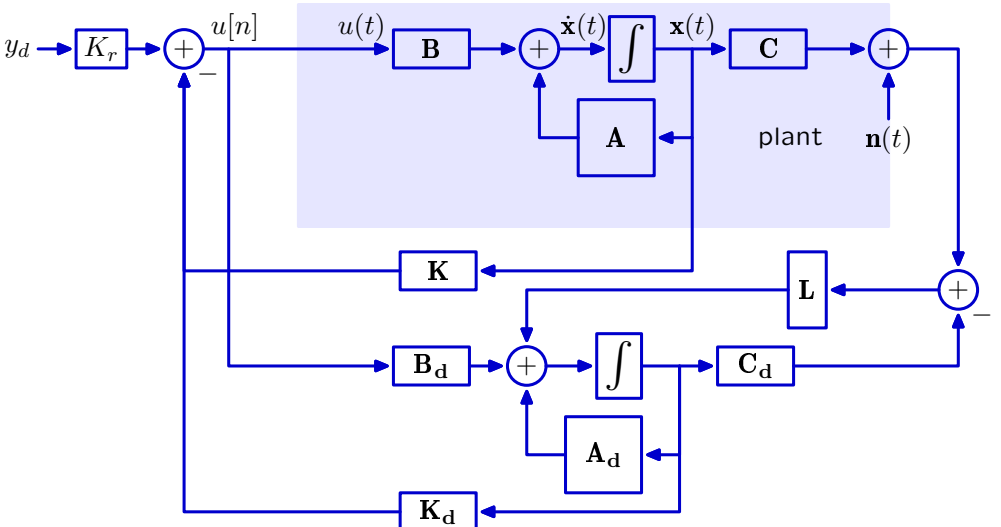
Observer


y_d
K_r
$u[n]$
$u(t)$
B
$\dot{\mathbf{x}}(t)$
$\mathbf{x}(t)$
C
A
plant
K
L
$\mathbf{B_d}$
$\mathbf{C_d}$
$\mathbf{A_d}$
$\mathbf{K_d}$

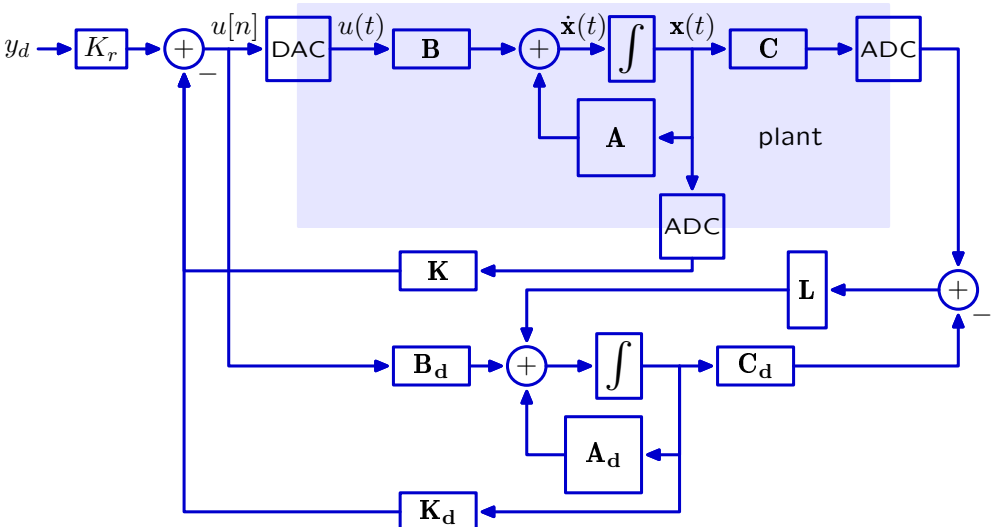
Observer with Noise in X



Observer, Noise in Y



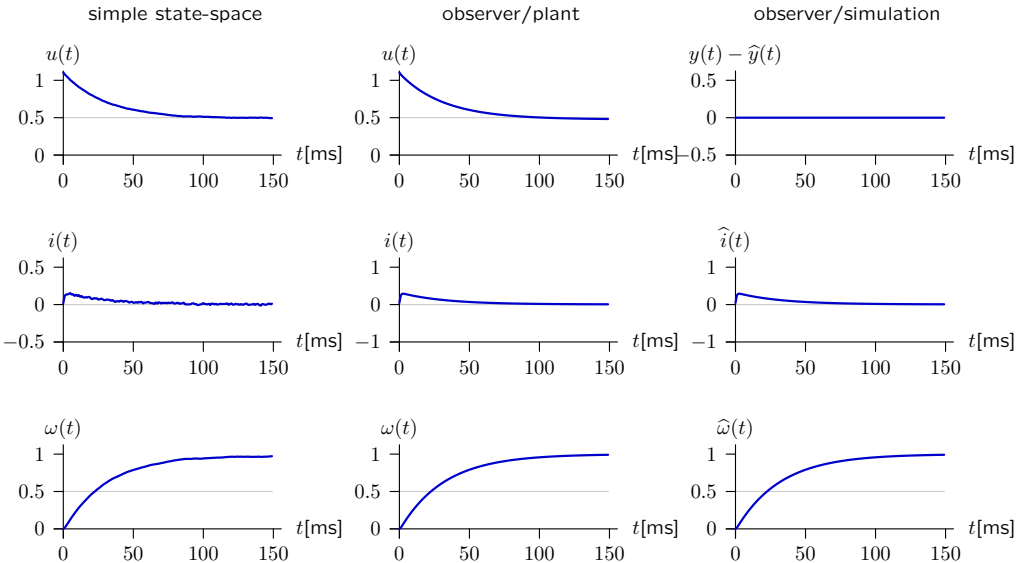
Observer, Discrete



Effects of K and L

Choose `K` based on `Q=diag([1,1])` and `R=1`.

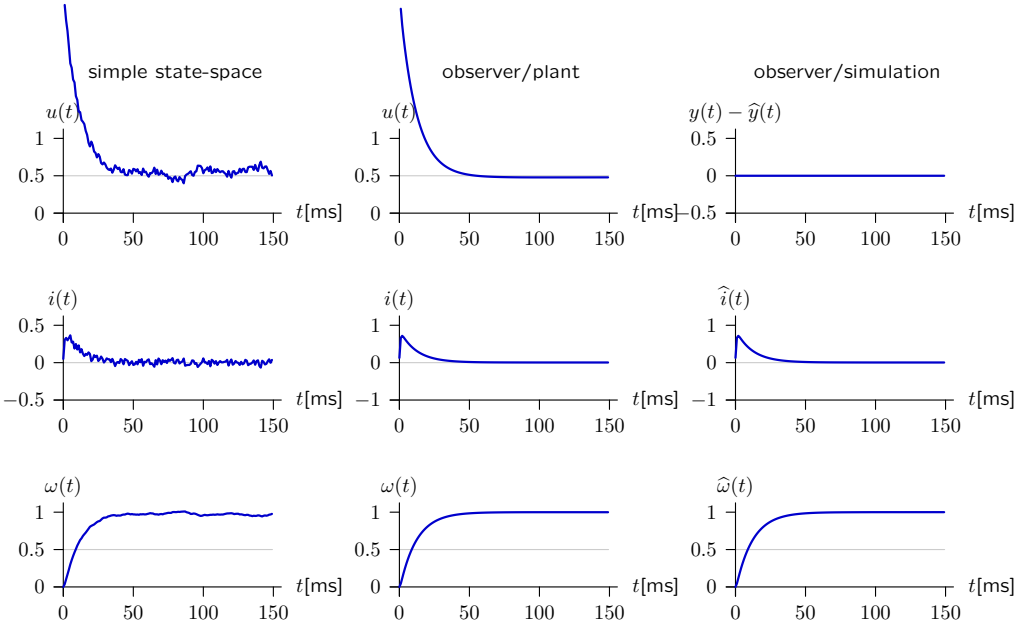
Choose `L` based on `Q=diag([1,1])` and `R=1`.



Effects of K and L

Choose `K` based on `Q=diag([1,1])` and `R=0.1`.

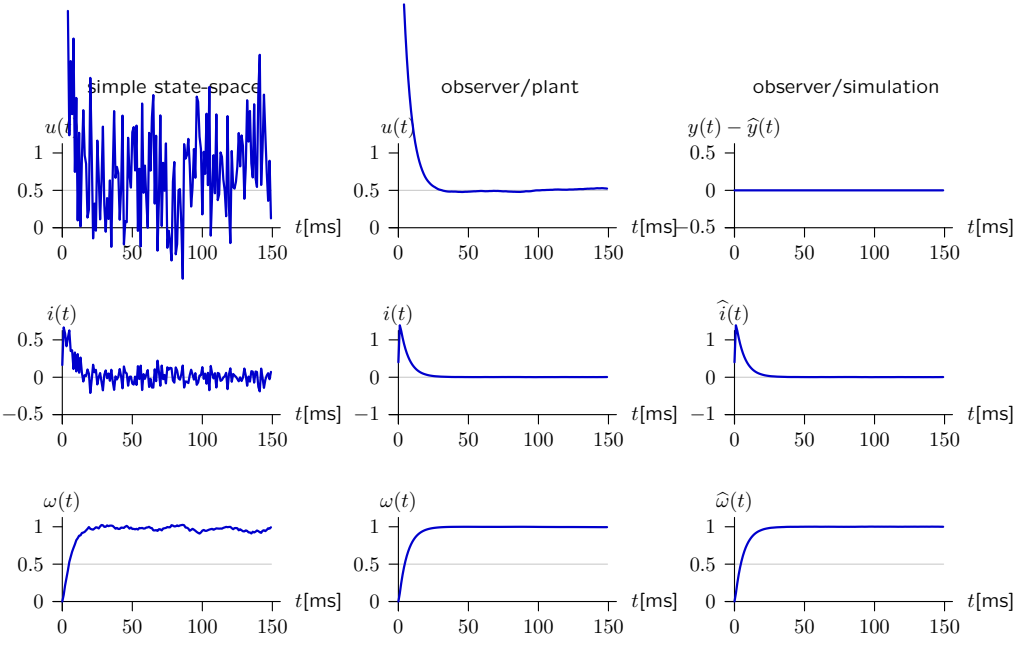
Choose `L` based on `Q=diag([1,1])` and `R=0.1`.



Effects of K and L

Choose `K` based on `Q=diag([1,1])` and `R=0.01`.

Choose `L` based on `Q=diag([1,1])` and `R=0.01`.


simple state-space
observer/plant
observer/simulation
u(t)
y(t) − ŷ(t)
i(t)
î(t)
ω(t)
ω̂(t)
t[ms]
0
50
100
150
1
0.5
0
−0.5
−1

Effects of K and L

Parameters.

$\mathbf{Q}$	$\mathbf{R}$	$\mathbf{K}$	eig$(\mathbf{A}-\mathbf{BK})$	$\mathbf{L}$	eig$(\mathbf{A^T}-\mathbf{C^T L^T})$
[1,1]	1	[0.1597,0.6305]	-1401,-31.6	[-0.0024;0.0377]	-1387,-13.8
[1,1]	0.1	[1.0273,2.7185]	-1522 -84.0	[-0.0237;0.3727]	-1387 -14.1
[1,1]	0.01	[5.9630,9.5199]	-2428,-164.9	[-0.2135;03.3658]	-1387 -17.1

Effects of K and L

Parameters.

$\mathbf{Q}$	$\mathbf{R}$	$\mathbf{K}$	eig$(\mathbf{A}-\mathbf{BK})$	$\mathbf{L}$	eig$(\mathbf{A}^{\mathbf{T}}-\mathbf{C}^{\mathbf{T}}\mathbf{L}^{\mathbf{T}})$
[1,1]	1	[0.1597,0.6305]	-1401,-31.6	[-0.0024;0.0377]	-1387,-13.8
[1,1]	0.1	[1.0273,2.7185]	-1522 -84.0	[-0.0237;0.3727]	-1387 -14.1
[1,1]	0.01	[5.9630,9.5199]	-2428,-164.9	[-0.2135;03.3658]	-1387 -17.1

For each entry, the dominant eigenvalue of $\mathbf{A}^{\mathbf{T}}-\mathbf{C}^{\mathbf{T}}\mathbf{L}^{\mathbf{T}}$ is greater than the dominant eigenvalue of $\mathbf{A}-\mathbf{BK}$.

The feedback from the observer to the plant happens **before** the observer has had a chance to synchronize with the plant!

We need faster eigenvalues of $\mathbf{A}^{\mathbf{T}}-\mathbf{C}^{\mathbf{T}}\mathbf{L}^{\mathbf{T}}$.

Effects of K and L

Parameters.

$\mathbf{Q}$	$\mathbf{R}$	$\mathbf{K}$	eig$(\mathbf{A}-\mathbf{BK})$	$\mathbf{L}$	eig$(\mathbf{A^T}-\mathbf{C^T L^T})$
[1,1]	1	[0.1597,0.6305]	-1401,-31.6	[-0.0024;0.0377]	-1387,-13.8
[1,1]	0.1	[1.0273,2.7185]	-1522 -84.0	[-0.0237;0.3727]	-1387 -14.1
[1,1]	0.01	[5.9630,9.5199]	-2428,-164.9	[-0.2135;03.3658]	-1387 -17.1
[1,100]	0.1	[4.0127,31.1396]	-1102 -1102	[-1.3463;21.0007]	-1387 -35
[1,100]	0.01	[[11.6545,99.4957]	-1865 -1865	[-5.3925;88.0755]	-1387 -102

Effects of K and L

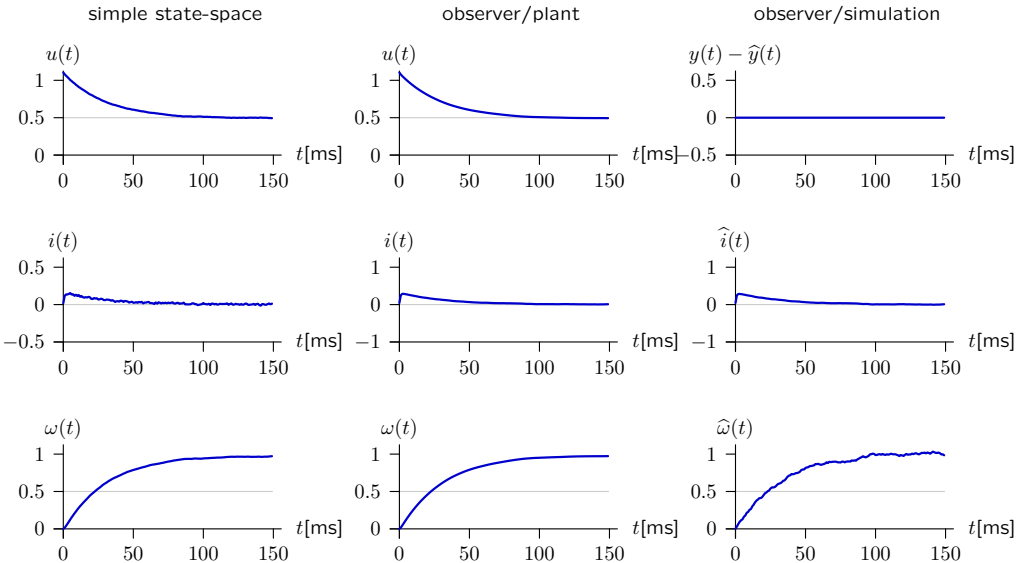
Use the red parameters.

$\mathbf{Q}$	$\mathbf{R}$	$\mathbf{K}$	eig$(\mathbf{A}-\mathbf{BK})$	$\mathbf{L}$	eig$(\mathbf{A^T}-\mathbf{C^T L^T})$
[1,1]	1	[0.1597,0.6305]	-1401,-31.6	[-0.0024;0.0377]	-1387,-13.8
[1,1]	0.1	[1.0273,2.7185]	-1522 -84.0	[-0.0237;0.3727]	-1387 -14.1
[1,1]	0.01	[5.9630,9.5199]	-2428,-164.9	[-0.2135;03.3658]	-1387 -17.1
[1,100]	0.1	[4.0127,31.1396]	-1102 -1102	[-1.3463;21.0007]	-1387 -35
[1,100]	0.01	[[11.6545,99.4957]	-1865 -1865	[-5.3925;88.0755]	-1387 -102

Effects of K and L

Choose `K` based on `Q=diag([1,1])` and `R=1`.

Choose `L` based on `Q=diag([1,100])` and `R=0.01`.



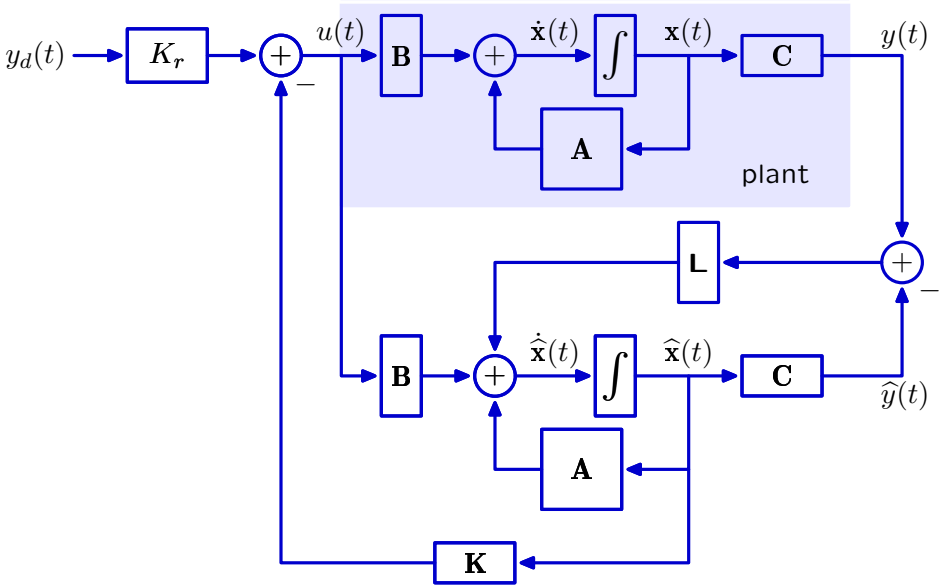
Effects of K and L

As with the simple state-space controller, higher gains can increase speed.

Extra care is needed when designing an observer. Make sure that the observer stabilizes to the plant **before** the observer is used to provide feedback to the plant.

State-Space Controller

Assume that we will implement the controller with a **microprocessor**.



Express the controller algorithm in pseudo-code.

State-Space Controller

Express the controller algorithm in pseudo-code.

Assume the `step` function (below) is executed once every `DeltaT` seconds.

```
global xhat
void step(){
  y = get_output_y()
  xhat = xhat + DeltaT * ((A-B*K)*xhat + B*Kr*yd + L*(y-C*xhat))
  put_command_u(Kr*yd-K*xhat)
}
```