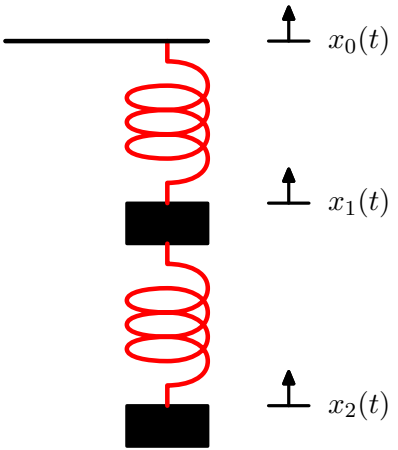
6.3100: Dynamic System Modeling and Control Design

Controlling a System with an Observer

May 1, 2023

Two-Spring System

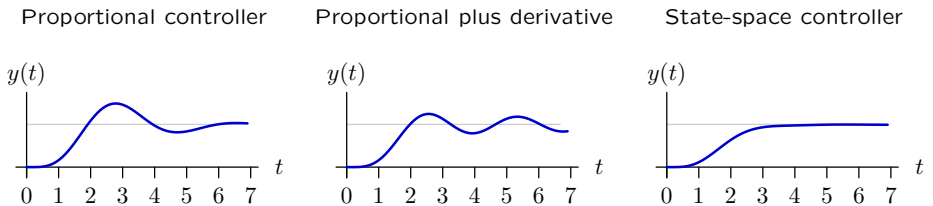
Last time, we developed classical and state-space controllers for a two-spring system.



The goal was to move the input $u(t) = x_0(t)$ so as to position the bottom mass $y(t) = x_2(t)$ at some desired location $y_d(t)$.

Comparison of Control Schemes

We found that the state-space control system allowed much better control of **overshoot** than the classical control systems.



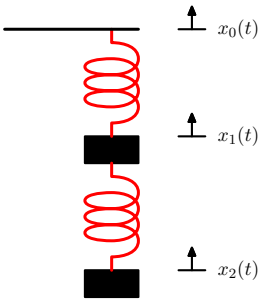
We reasoned that better performance resulted because the state-space controller has access to the motions of **both** masses.

We also outlined a framework for designing an **observer** to provide information about the motion of the center mass without actually measuring that motion.

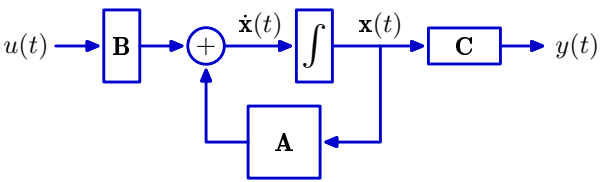
Today we will work through the **implementation** of this design.

State-Space Model

To apply the state-space approach, we must express the equations of motion in the following matrix form.

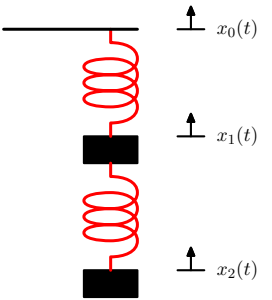


$$f_{m1} = m\ddot{x}_1(t) = k\Big(x_0(t) - x_1(t)\Big) - k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_1(t)$$
$$f_{m2} = m\ddot{x}_2(t) = k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_2(t)$$

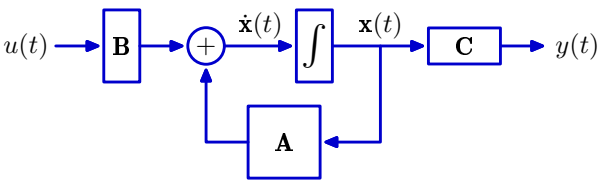


Check Yourself

Find **A**, **B**, and **C**.



$$f_{m1} = m\ddot{x}_1(t) = k\Big(x_0(t) - x_1(t)\Big) - k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_1(t)$$
$$f_{m2} = m\ddot{x}_2(t) = k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_2(t)$$



State-Space Description

Equations of motion:

$$f_{m1} = m\ddot{x}_1(t) = k\Big(x_0(t) - x_1(t)\Big) - k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_1(t)$$

$$f_{m2} = m\ddot{x}_2(t) = k\Big(x_1(t) - x_2(t)\Big) - b\dot{x}_2(t)$$

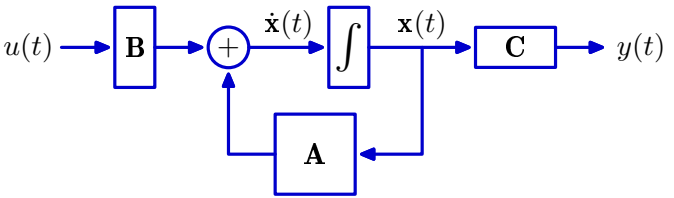
Four state variables (two displacements and their velocities):

$$\frac{d}{dt}\begin{bmatrix} v_1(t) \\ x_1(t) \\ v_2(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{b}{m} & -\frac{2k}{m} & 0 & \frac{k}{m} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{k}{m} & -\frac{b}{m} & -\frac{k}{m} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_1(t) \\ x_1(t) \\ v_2(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{k}{m} \\ 0 \\ 0 \\ 0 \end{bmatrix} x_0(t) - \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} g$$

$$y(t) = [0\ 0\ 0\ 1] \begin{bmatrix} v_1(t) \\ x_1(t) \\ v_2(t) \\ x_2(t) \end{bmatrix}$$

State-Space Model

To apply the state-space approach, we must express the equations of motion in the following matrix form.

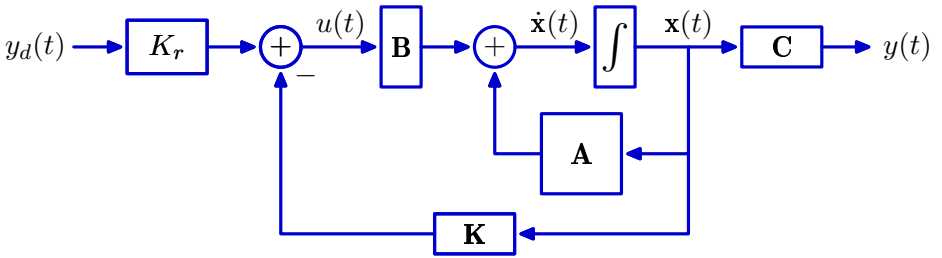


$$\mathbf{A} = \begin{bmatrix} -\frac{b}{m} & -\frac{2k}{m} & 0 & \frac{k}{m} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{k}{m} & -\frac{b}{m} & -\frac{k}{m} \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \frac{k}{m} \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{C} = [0\ 0\ 0\ 1]$$

$$\mathbf{x}(t) = \begin{bmatrix} v_1(t) \\ x_1(t) \\ v_2(t) \\ x_2(t) \end{bmatrix} \qquad u(t) = x_0(t) \qquad y(t) = x_2(t)$$

State-Space Controller

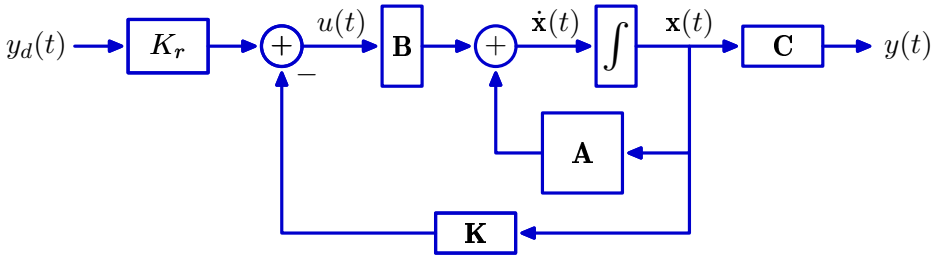
A state-space controller can then be expressed as follows.



How do we find $\mathbf{K}$ and K_r?

State-Space Controller

A state-space controller can then be expressed as follows.



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

```
K = place(A,B,[poles])
```

or LQR:

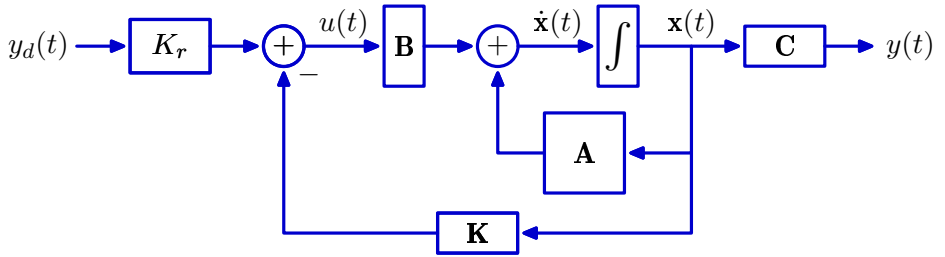
```
Q = diag([1,1,1,1])
R = 1
K = lqr(A,B,Q,R)
```

and

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

State-Space Controller

A state-space controller can then be expressed as follows.



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

```
K = place(A,B,[poles])
```

or LQR:

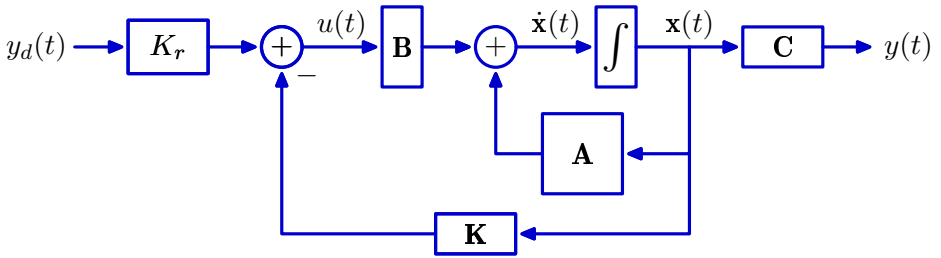
```
Q = diag([1,1,1,1])
R = 1
K = lqr(A,B,Q,R)
```

and

```
Kr = -1/(C*((A-BK)\B))    <-- where does this come from?
```

State-Space Controller

Assume that we will implement the controller with a microprocessor.



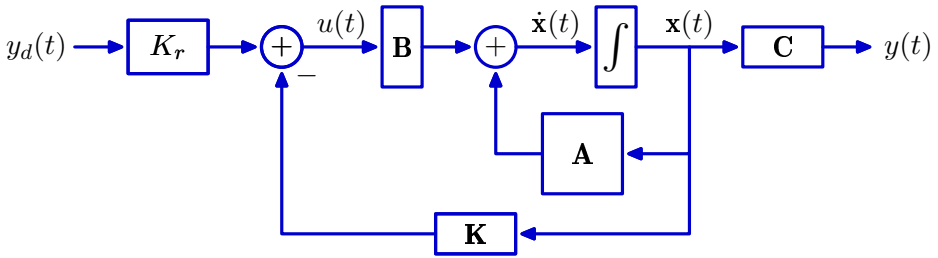
Express the controller algorithm in pseudo-code.

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

```
void step(){
  PUT YOUR CODE HERE
}
```

State-Space Controller

Assume that we will implement the controller with a microprocessor.



Express the controller algorithm in pseudo-code.

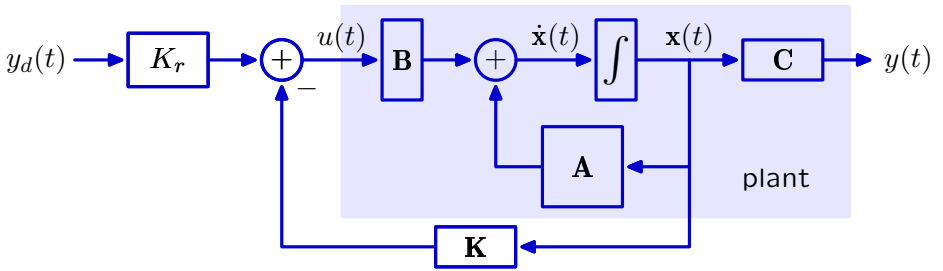
Assume that the `step` function (below) is executed once every ΔT seconds.

```
void step(){
    v1,x1,v2,x2 = get_state_x();
    put_command_u(Kr*yd - (K1*v1 + K2*x1 + K3*v2 + K4*x2));
}
```

Where are $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$? Shouldn't the controller need these?

State-Space Controller

Assume that we will implement the controller with a microprocessor.



Where are $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$?

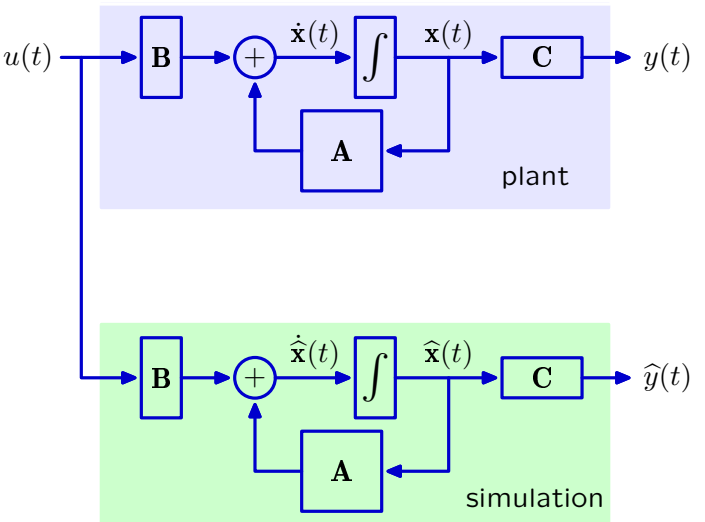
$\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are components of our model of the plant (blue shading above). They are used to design $\mathbf{K}$ and K_r, but do not appear explicitly in a simple state-space controller.

Observers

By contrast, observer-based controllers **explicitly** depend on $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$.

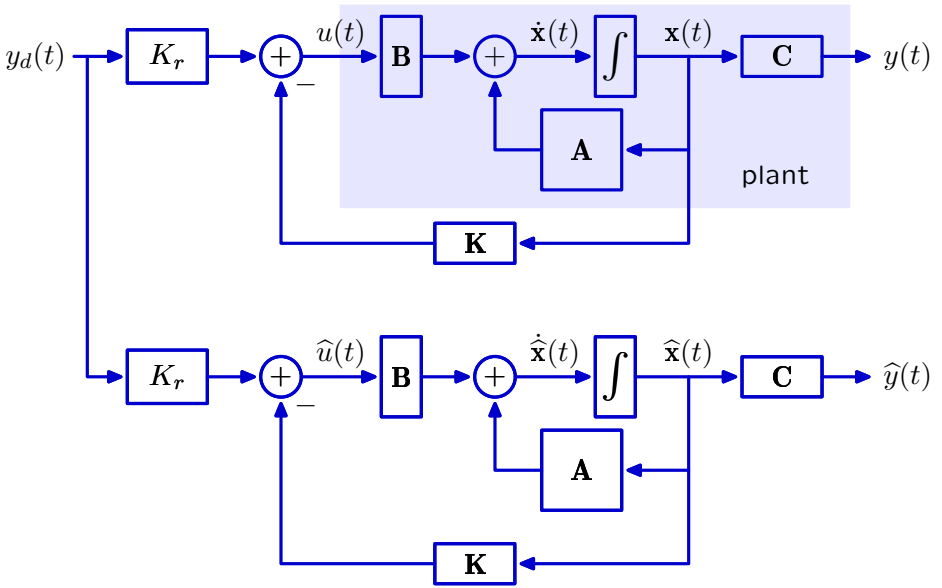
Observers (Recap)

An **observer** is a **simulation** of the plant that is used by the controller – i.e., the simulation is part of the controller!



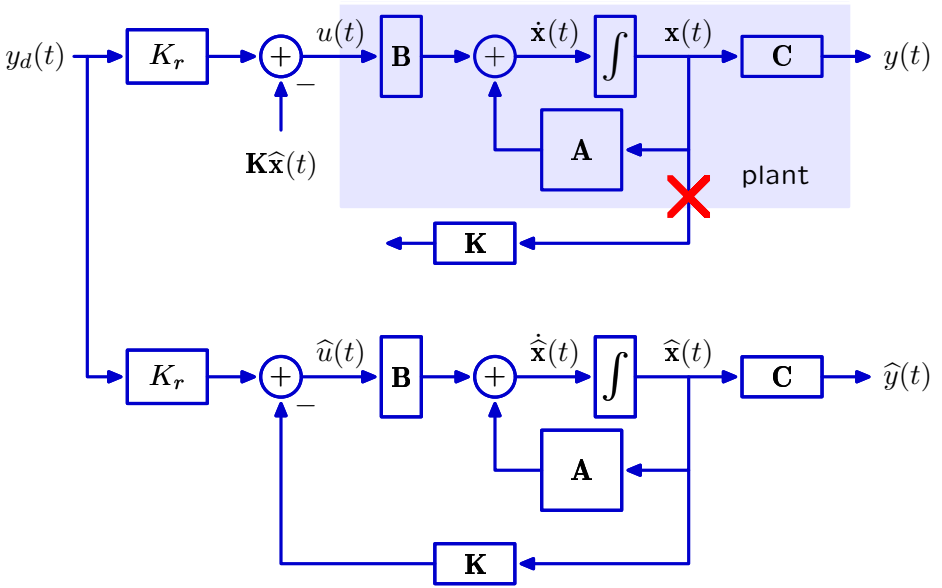
Observers (Recap)

We can build state-space controllers for both the plant and the simulation.



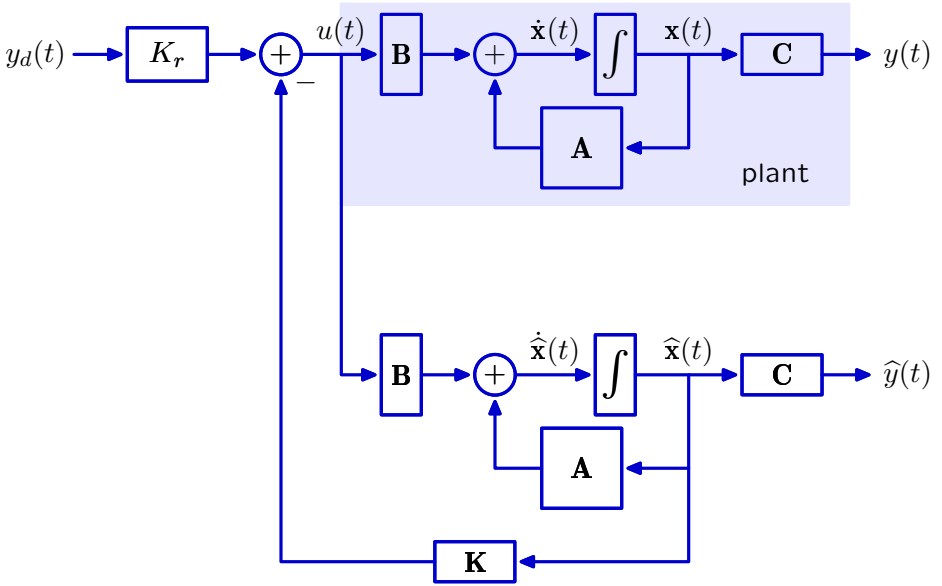
Observers (Recap)

If our model of the plant (**A, B, C**) is perfect, then $\widehat{\mathbf{x}}(t) = \mathbf{x}(t)$ and we can replace $\mathbf{K}\mathbf{x}(t)$ with $\mathbf{K}\widehat{\mathbf{x}}(t)$. This substitution also makes $u(t) = \widehat{u}(t)$.



Observers (Recap)

The resulting structure provides feedback from all **simulated** states $\widehat{\mathbf{x}}(t)$. Unfortunately even small differences between the plant and simulation can lead to large differences between $\mathbf{x}(t)$ and $\widehat{\mathbf{x}}(t)$.

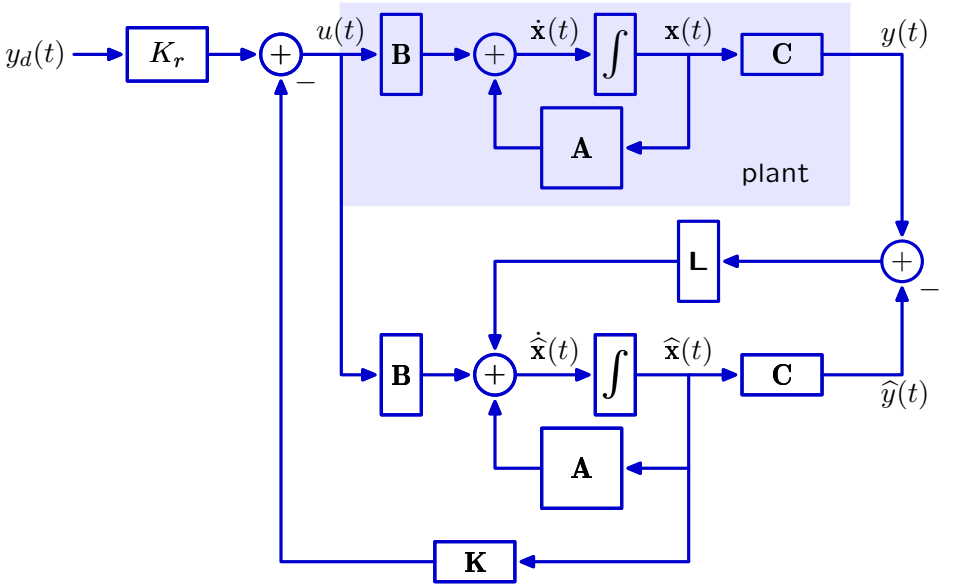


Observers (Recap)

Fortunately, we can use **feedback** to correct simulation errors!

Calculate the difference between $y(t)$ and $\widehat{y}(t)$.

Then use that signal (times $\mathbf{L}$) to correct $\dot{\widehat{\mathbf{x}}}(t)$.



Observers (Recap)

Dynamics:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{K}\widehat{\mathbf{x}}(t) + \mathbf{B}K_r y_d(t)$$
$$\dot{\widehat{\mathbf{x}}}(t) = \mathbf{A}\widehat{\mathbf{x}}(t) - \mathbf{B}\mathbf{K}\widehat{\mathbf{x}}(t) + \mathbf{B}K_r y_d(t) + \mathbf{L}\Big(y(t) - \widehat{y}(t)\Big)$$

Matrix form:

$$\begin{bmatrix}\dot{\mathbf{x}}(t)\\ \dot{\widehat{\mathbf{x}}}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{A} & -\mathbf{B}\mathbf{K}\\ \mathbf{L}\mathbf{C} & \mathbf{A}-\mathbf{L}\mathbf{C}-\mathbf{B}\mathbf{K}\end{bmatrix}\begin{bmatrix}\mathbf{x}(t)\\ \widehat{\mathbf{x}}(t)\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\ \mathbf{B}\end{bmatrix}K_r y_d(t)$$

$$y(t) = \begin{bmatrix}\mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}(t)\\ \widehat{\mathbf{x}}(t)\end{bmatrix}$$

Choose $\mathbf{K}$ to optimize the eigenvalues of $\mathbf{A} - \mathbf{B}\mathbf{K}$.
Choose $\mathbf{L}$ to optimize the eigenvalues of $\mathbf{A}^T - \mathbf{C}^T\mathbf{L}^T$.

```
K = place(A,B,[poles])
L = place(A.',C.',[poles]).'
```

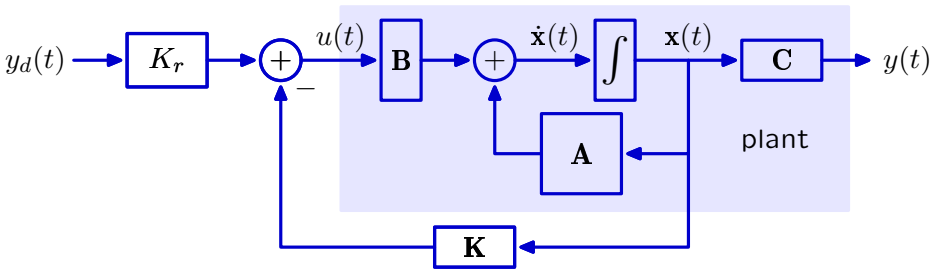
or

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

Noise Performance

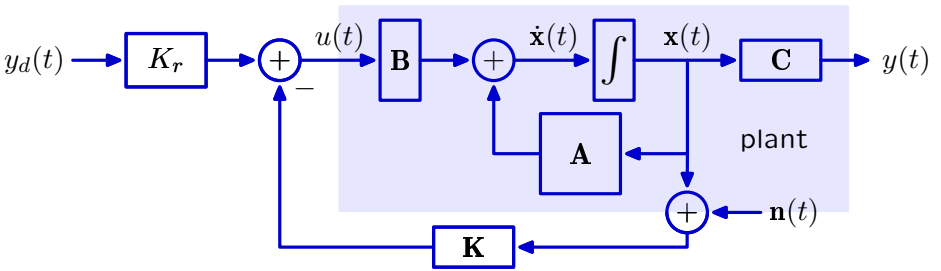
Feedback control can be significantly degraded by noise that is introduced by the sensors that provide information about the plant to the controller.

Suggest a model for the effects of sensor noise on the following state-space control system. Assume that sensor noise is additive.



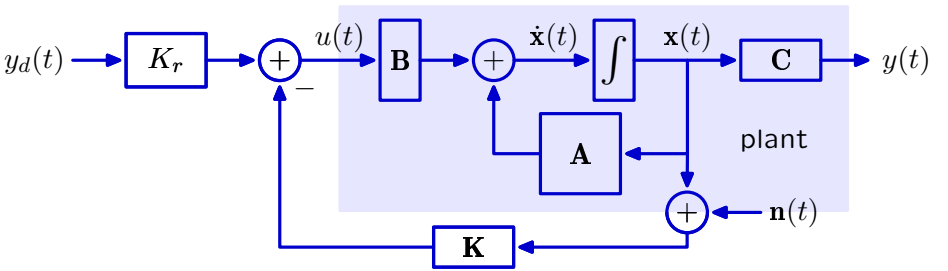
Effects of Sensor Noise

Sensor noise can contaminate each of the state measurements.



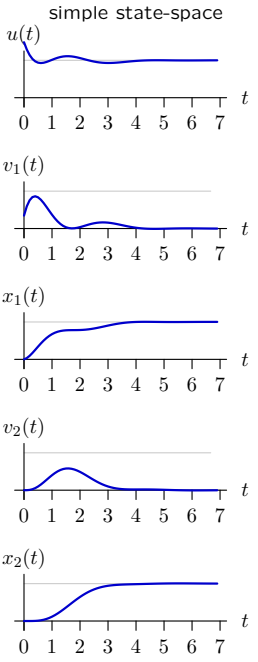
Effects of Sensor Noise

How will this noise affect performance of the control system?



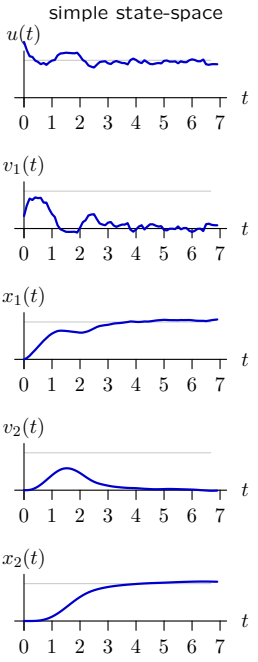
Results 1

Compare with additive noise: amplitude = 0



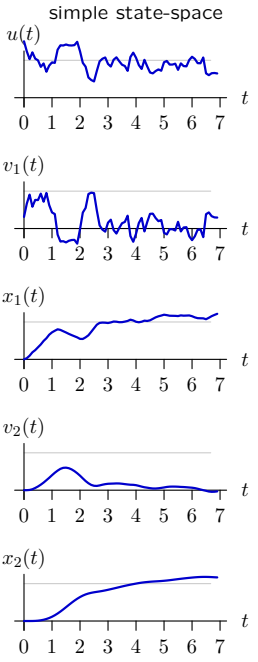
Results 2

Compare with additive noise: amplitude = 0.3



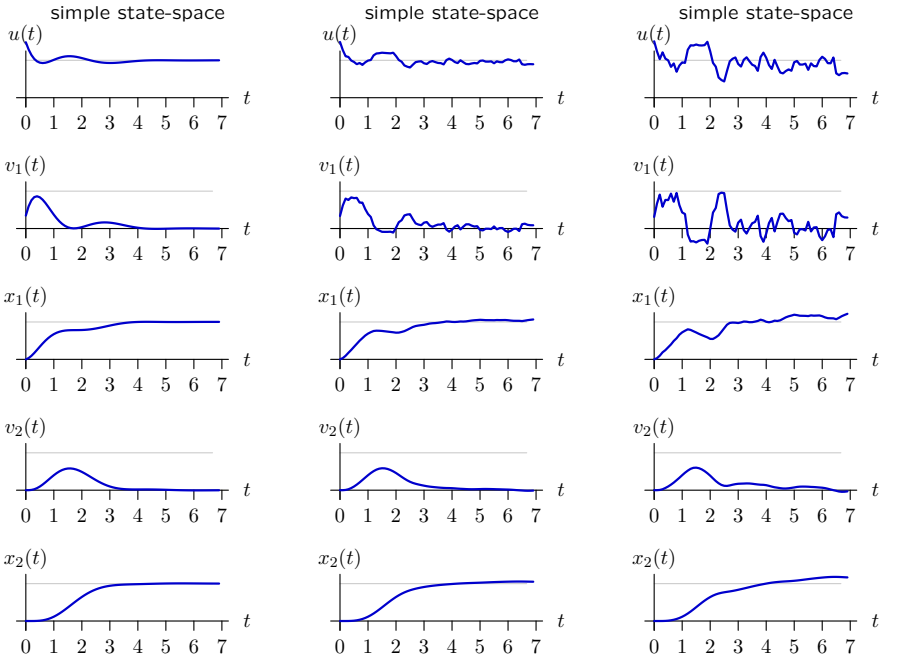
Results 3

Compare with additive noise: amplitude = 1



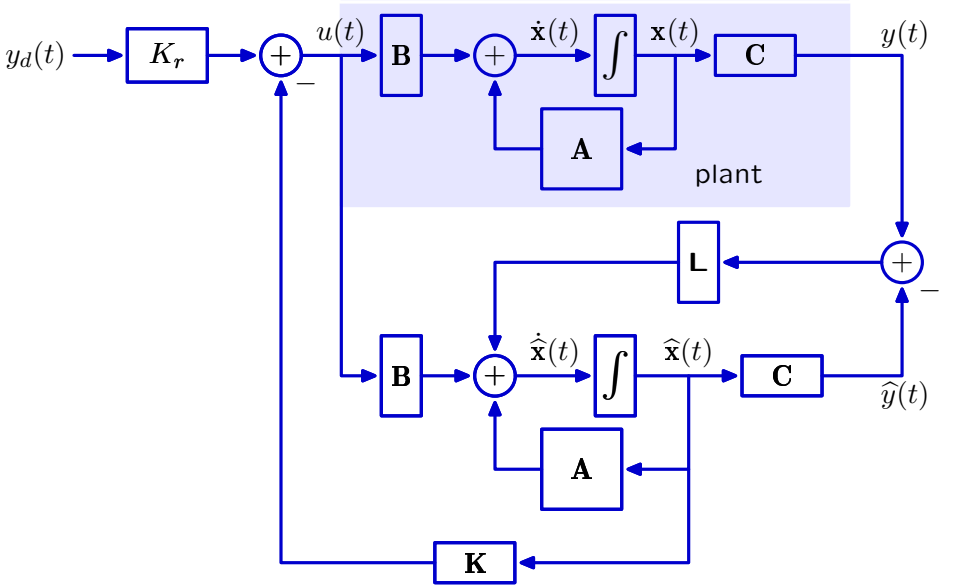
Results 4

Additive noise: 0, 0.3, and 1:



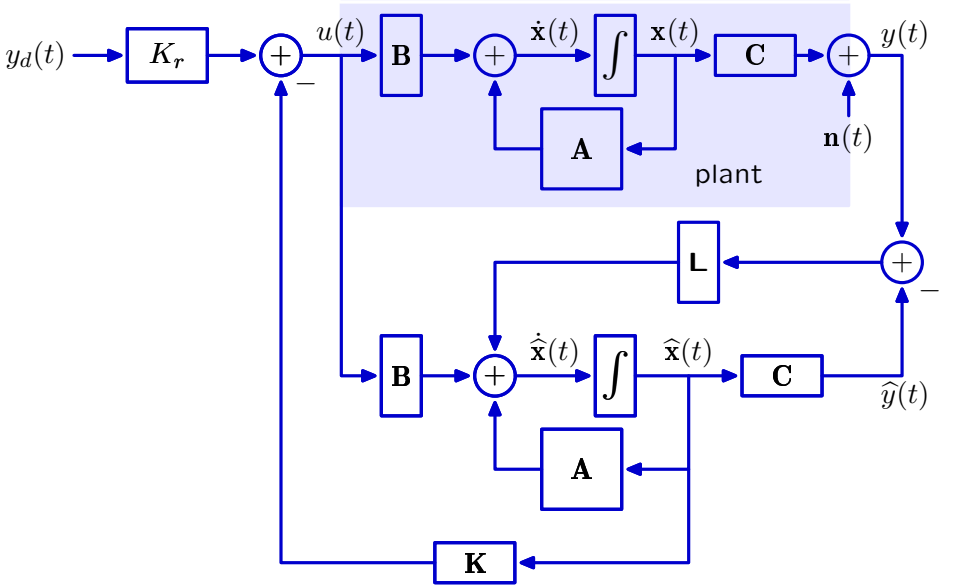
Effects of Sensor Noise

How should we model sensor noise with an observer?



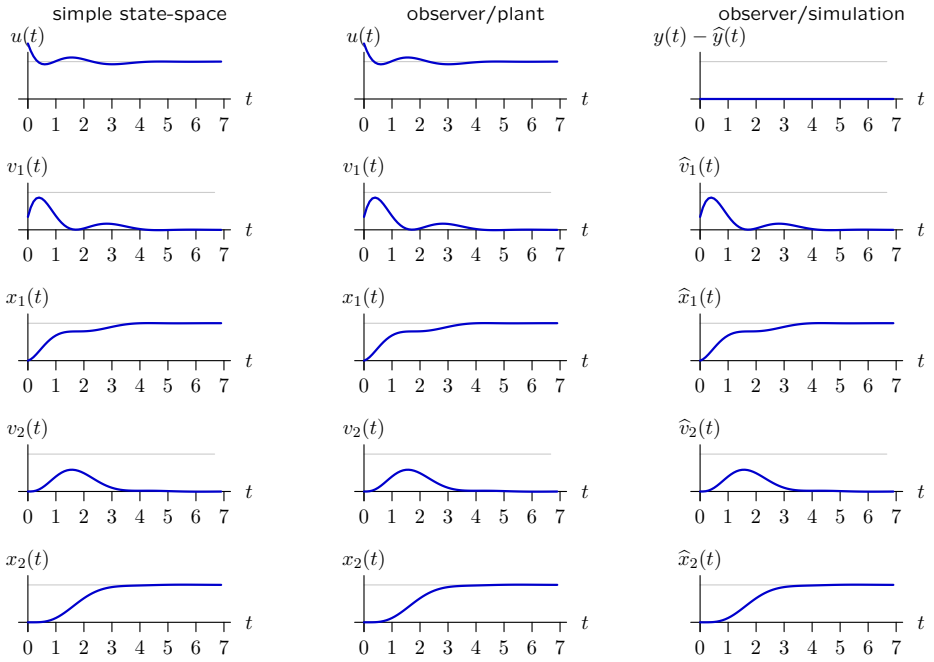
Effects of Sensor Noise

How will this noise affect performance of the control system?



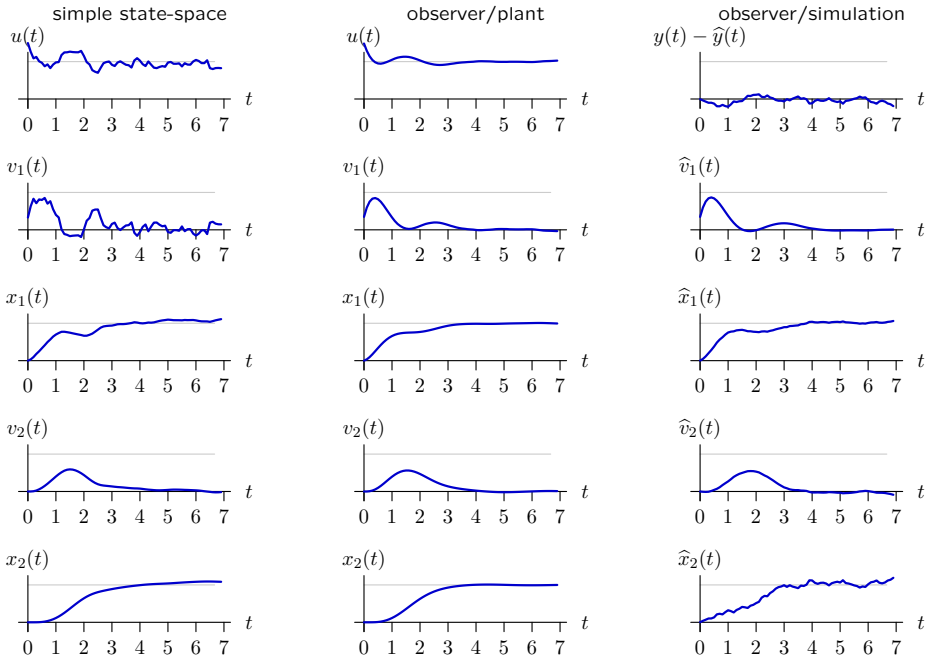
Results 5

Compare a simple state-space controller with an observer-based controller.



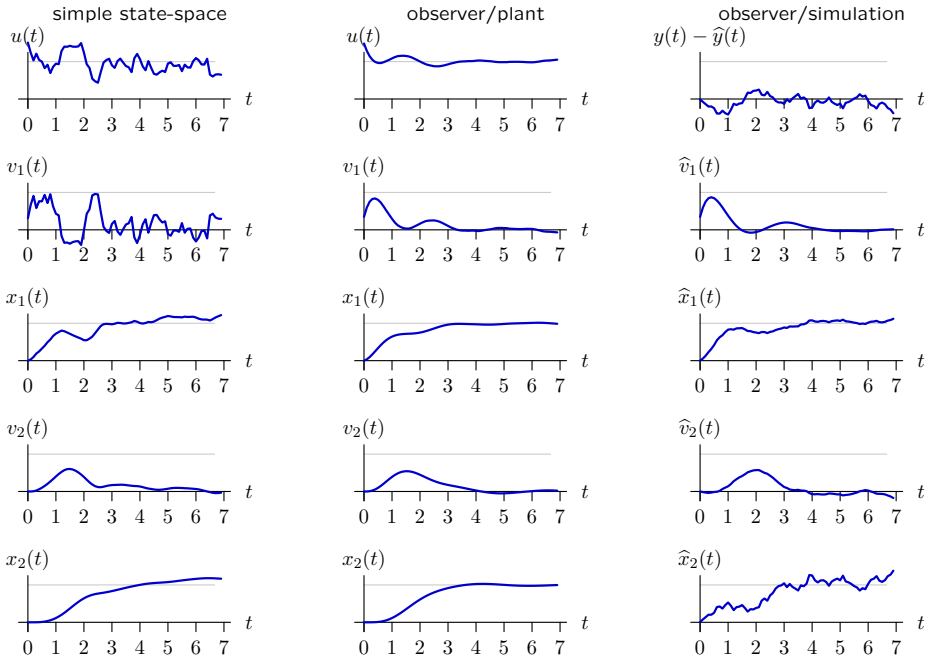
Results 6

Compare a simple state-space controller with an observer-based controller.



Results 7

Compare a simple state-space controller with an observer-based controller.



Effects of Sensor Noise

How will this noise affect performance of the control system?