LQG Control with Servo Compensators

NDSU ECE 463/663

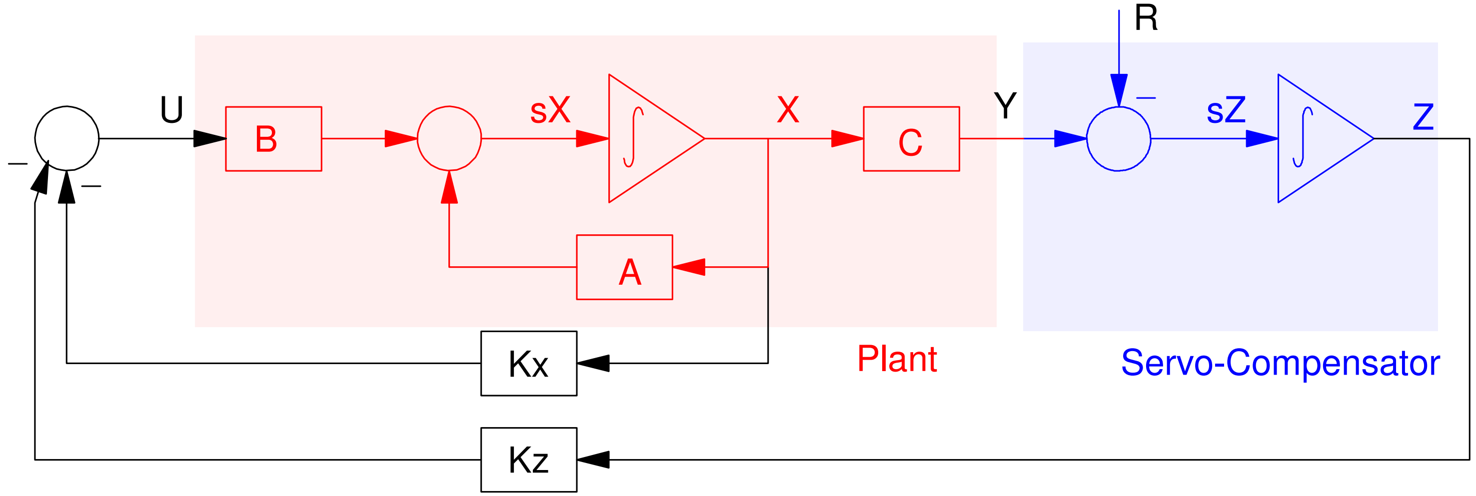
Lecture #27

Inst: Jake Glower

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Servo Compensator:

Servo-Compensators allow you to track a constant set-point:



$$s\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} A - BK_x & -BK_z \\ C & 0 \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} R$$

Problem

Heat Equation + Servo Compensator

$$s\begin{bmatrix} X \\ \cdots \\ Z \end{bmatrix} = \left[\begin{array}{cccc:c} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]\begin{bmatrix} X \\ \cdots \\ Z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} U + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ -1 \end{bmatrix} R$$

$$y = x_4 = \left[\begin{array}{cccc:c} 0 & 0 & 0 & 1 & 0 \end{array}\right]\begin{bmatrix} X \\ Z \end{bmatrix}$$

Use LQR techniques to find Kx and Kz

- No error for a step input (assured with the use of a servo compensator)
- A 2% settling time of 4 seconds, and
- <4% overshoot for a step input

Solution (take 1)

Ignore the servo states:

- They are just dummy states

Weight the output

- It's what you care about

$$y = x_4 = \begin{bmatrix} 0\;0\;0\;1 \,\vdots\, 0 \end{bmatrix} \begin{bmatrix} X \\ \cdots \\ Z \end{bmatrix}$$

$$y = C_x X$$

$$Q = C_x^T C_x$$

Problems:

Gives you an error: System is not observable

```
Q = 1 * Qy;
R = 1;
Kx = lqr(A, B, Q, R)
      !--error 998
      internal error, info=4.
```

When you get an error like that, the math is trying to tell you something. The challenge is trying to figure out what the math is trying to say....

Fix:

$$Q = C_x^T C + 10^{-3} I$$

Problem 2: The resulting system is *really* slow

```
Q = C5'*C5 + eye(5,5) * 1e-3

     0.001     0.        0.        0.        0.
     0.        0.001     0.        0.        0.
     0.        0.        0.001     0.        0.
     0.        0.        0.        1.001     0.
     0.        0.        0.        0.        0.001

Kx = lqr(A, B, Q, R)

     0.0641290    0.1298143    0.1958329     0.2527241    0.0316228

eig(A5 - B5*Kx)

   -3.5319727
   -2.3486196
   -0.9901871
   -0.1708104
   -0.0225392       ( dominant pole:  should be around -1 )
```

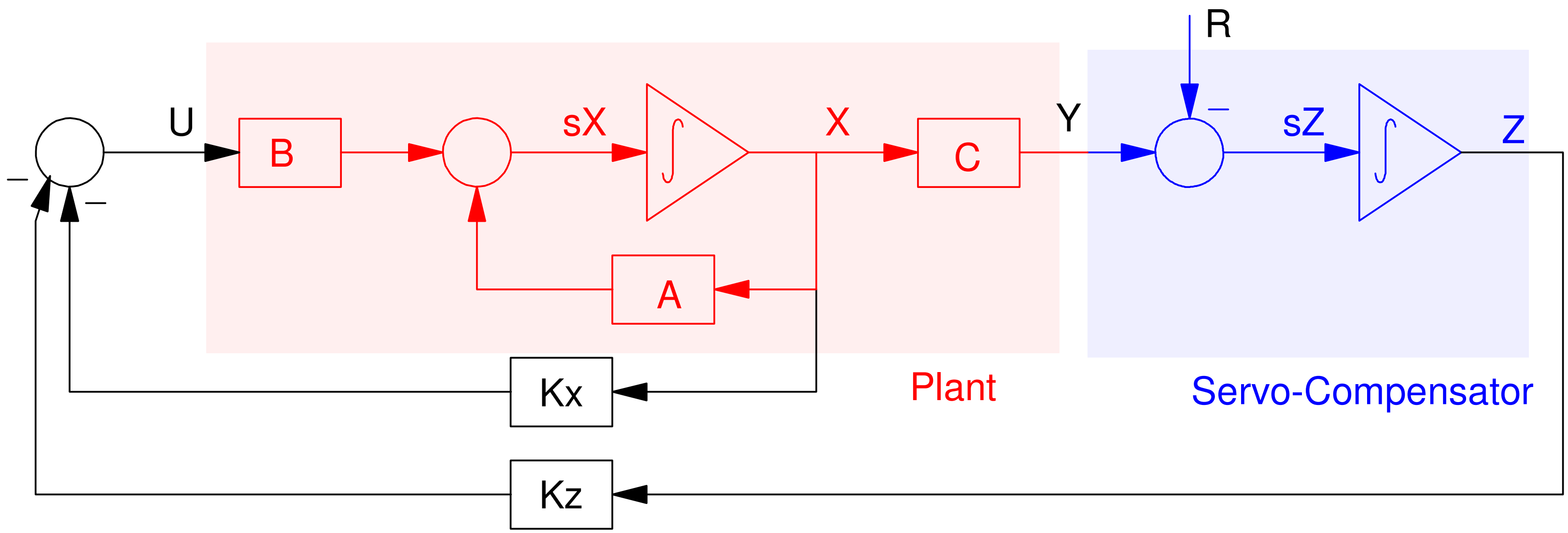
Problem 3: Increasing Q only makes things worse:

```
Q = 1000*C5'*C5 + eye(5,5) * 1e-3

    0.001    0.       0.       0.          0.
    0.       0.001    0.       0.          0.
    0.       0.       0.001    0.          0.
    0.       0.       0.       1000.001    0.
    0.       0.       0.       0.          0.001

Kx = lqr(A, B, Q, R)

    1.6015116    4.4849429    9.4975698    15.097135    0.0316228

eig(A5 - B5*Kx)

  -3.1945132
  -3.1689866
  -1.1185062 + 1.3690332i
  -1.1185062 - 1.3690332i
  -0.0009995         ( dominant pole:  worse than before )
```

Compensator Design (take 2)

The math is dumb: it thinks

- Z is the system output
- Y is the derivative of the system output



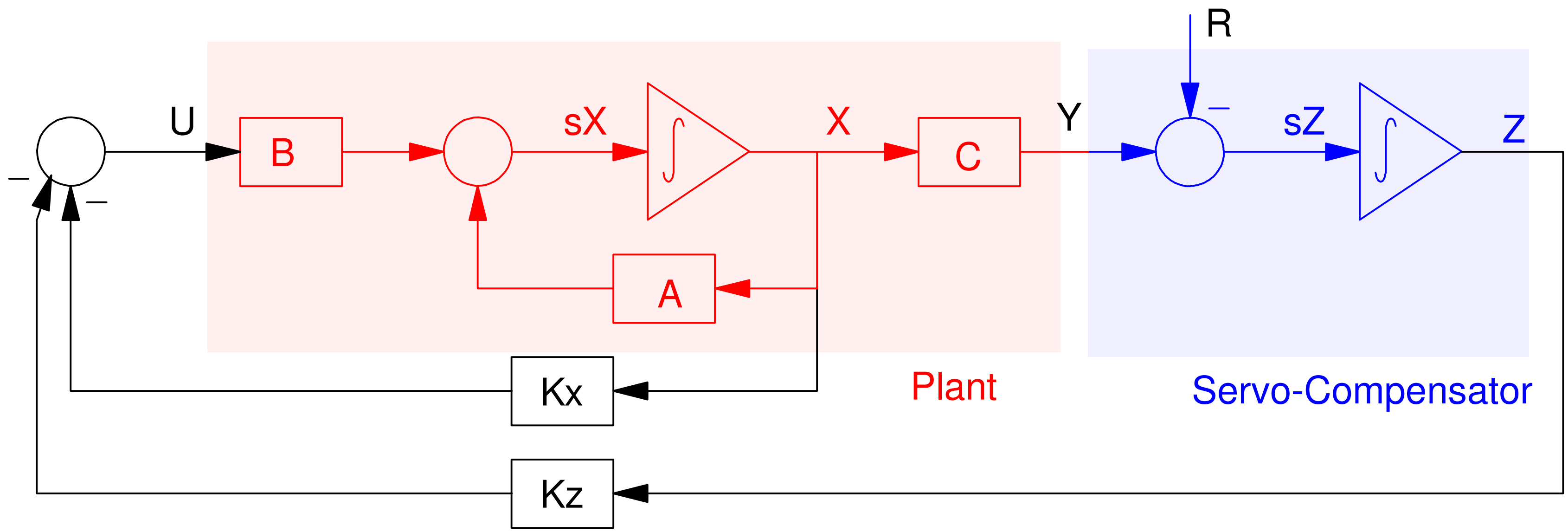
Procedure

Increasing the weighting on Z should speed up the system

- Stiffer spring

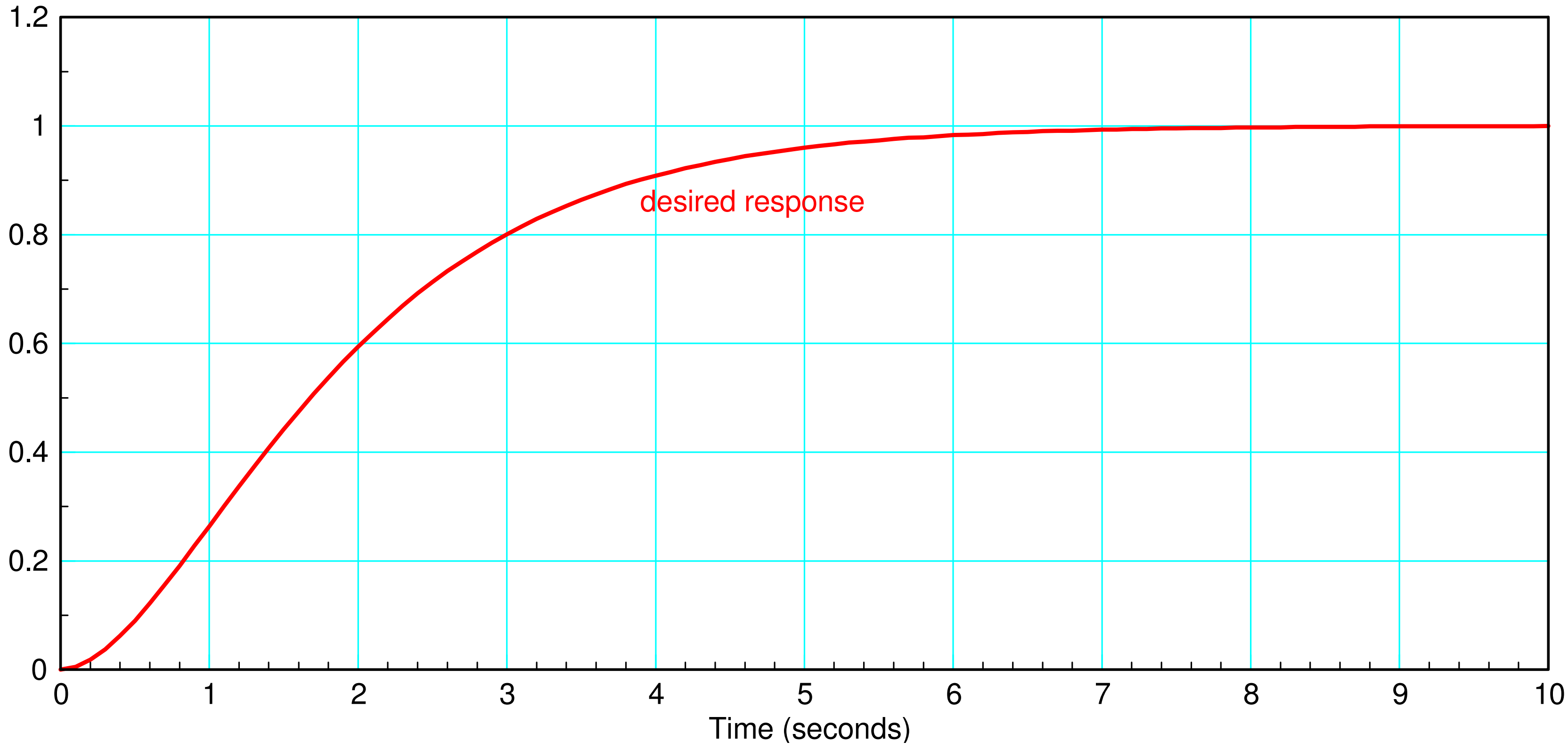
Increasing the weighting on Y should add more friction

- Stiffer shocks



Example: Find Kx and Kz so that the closed-loop system has

- No error for a step input
- 2% settling time of 4 seconds, and
- No overshoot for a step input



Start with Cz and Cx:

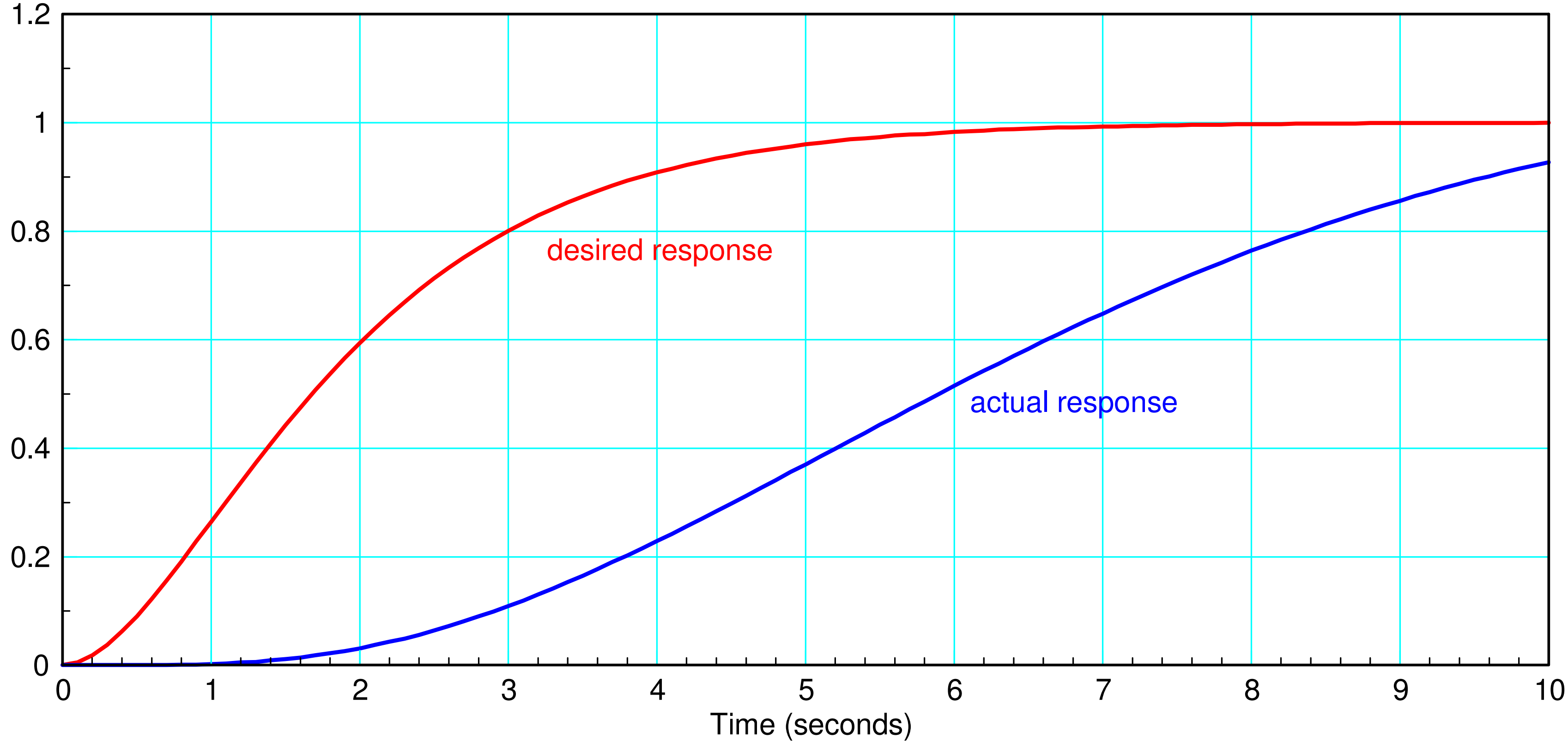
```
Cz = [0,0,0,0,1]

     0.    0.    0.    0.    1.

Cx = [0,0,0,1,0]

     0.    0.    0.    1.    0.
```

and the corresponding Q matrices:

```
Qz = Cz'*Cz;
Qx = Cx'*Cx
```

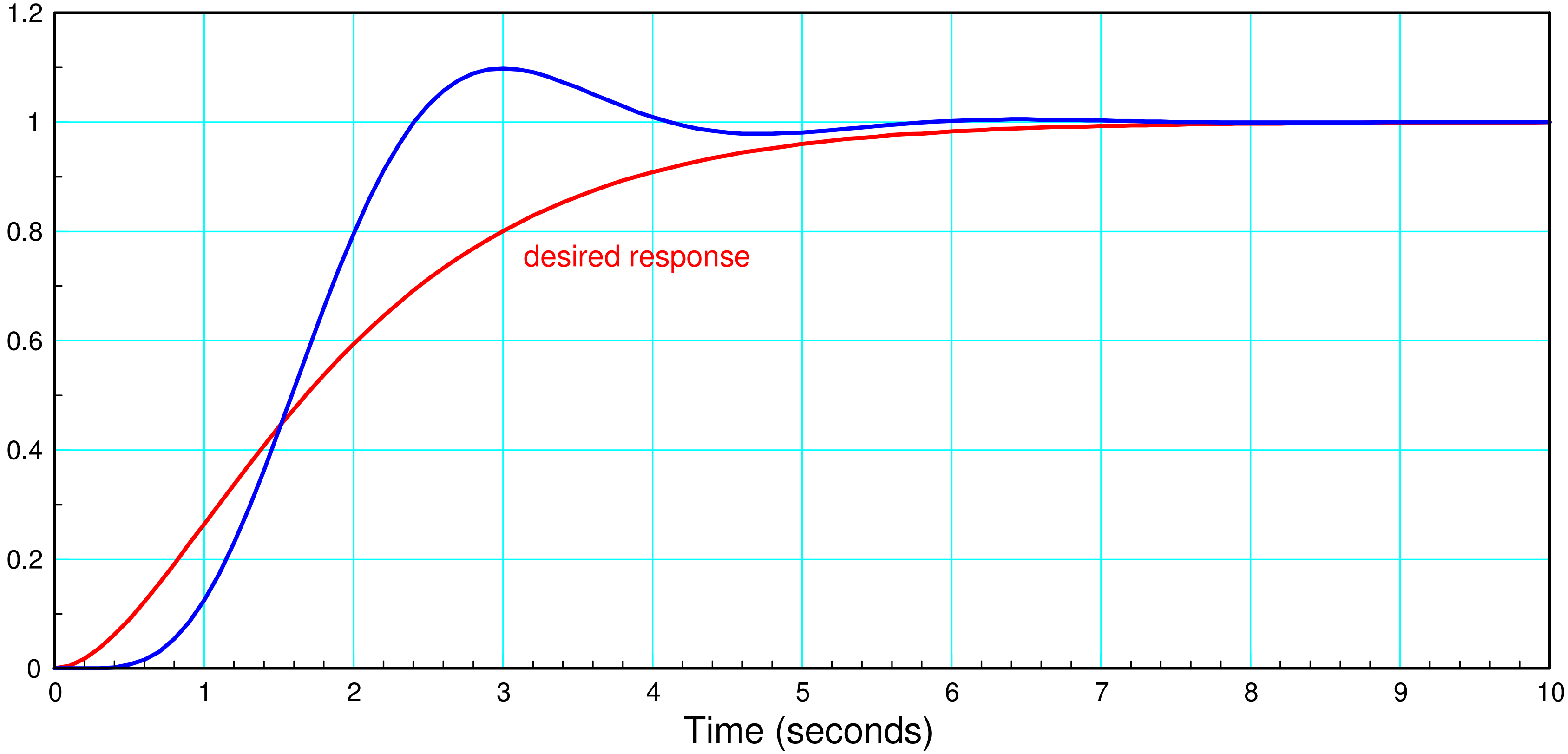
First guess, let Q = Qz

```
Q = Qz
Kx = lqr(A, B, Q, R)
```



Too slow: Increase Q to 10,000:

```
Q = Qz*1e4;
Kx = lqr(A, B, Q, R)
```



Add friction to drop overshoot:

```
Q = Qz*5000 + Qy*10000;
Kx = lqr(A, B, Q, R)
```