

6.3100: Dynamic System Modeling and Control Design

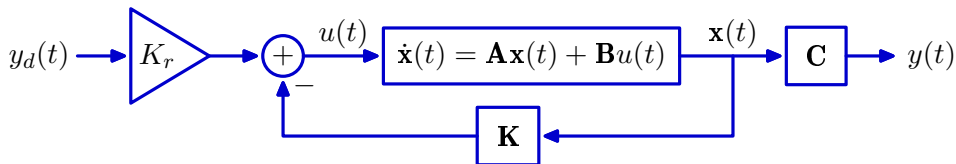
Linear Quadratic Regulator

November 13, 2024

Modern Control

State-Space Approach

- Describe a system by its **states**.
- Describe dynamics of a system by **first-order** relations among states.
- Collect the states and relations in a single first-order **matrix** equation.



Plant: state matrix \mathbf{A} , input vector \mathbf{B} , and output vector \mathbf{C} :

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)$$

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

Feedback is characterized by a feedback vector \mathbf{K} and input scaler K_r :

$$u(t) = K_r y_d(t) - \mathbf{K}\mathbf{x}(t)$$

Combine to obtain **closed-loop** characterization:

$$\dot{\mathbf{x}}(t) = \left(\mathbf{A} - \mathbf{B}\mathbf{K} \right) \mathbf{x}(t) + \mathbf{B}\mathbf{K}_r y_d(t) \equiv \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c y_d(t)$$

From State-Space to Transfer Function

Find the transfer function representation from the state-space description.

Start with the state equation:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c u(t)$$

Consider the input $u(t)$ and state $\mathbf{x}(t)$ at a particular complex frequency s :

$$u(t) = U(s)e^{st} \text{ and } \mathbf{x}(t) = \mathbf{X}(s)e^{st}$$

Find $H(s)$ at the same complex frequency.

$$s\mathbf{X}(s)e^{st} = \mathbf{A}_c \mathbf{X}(s)e^{st} + \mathbf{B}_c U(s)e^{st}$$

$$s\mathbf{X}(s) = \mathbf{A}_c \mathbf{X}(s) + \mathbf{B}_c U(s)$$

$$(s\mathbf{I} - \mathbf{A}_c)\mathbf{X}(s) = \mathbf{B}_c U(s)$$

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A}_c)^{-1} \mathbf{B}_c U(s)$$

$$Y(s) = \mathbf{C}_c \mathbf{X}(s) = \mathbf{C}_c (s\mathbf{I} - \mathbf{A}_c)^{-1} \mathbf{B}_c U(s)$$

$$H(s) = \frac{Y(s)}{U(s)} = \mathbf{C}_c (s\mathbf{I} - \mathbf{A}_c)^{-1} \mathbf{B}_c$$

State-Space Analysis of Natural Frequencies

Are there frequencies s for which large outputs result when input $u(t)=0$?

$$H(s) = \frac{Y(s)}{X(s)} = \mathbf{C}_c(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}_c = \mathbf{C}_c \frac{\text{adj}(s\mathbf{I}-\mathbf{A})}{|s\mathbf{I}-\mathbf{A}|} \mathbf{B}_c$$

If $|s\mathbf{I}-\mathbf{A}| = 0$, $H(s)$ is unbounded and therefore $|Y(s)| \rightarrow \infty$.

The natural frequencies are the solutions to the **characteristic equation**:

$$|s\mathbf{I}-\mathbf{A}_c| = 0$$

Step Response

Find the step response $\mathbf{x}_s(t)$ of the following matrix system equation:

$$\dot{\mathbf{x}}(t) = \mathbf{P}\mathbf{x}(t) + \mathbf{Q}u(t)$$

Assume the initial value of the step response $\mathbf{x}_s(0) = \mathbf{0}$, and $u(t)=1$ for $t>0$.

Homogeneous equation: $\dot{\mathbf{x}}_h(t) = \mathbf{P}\mathbf{x}_h(t)$

$$\mathbf{x}_h(t) = e^{\mathbf{P}t}\mathbf{\Psi}$$

Particular solution: $\mathbf{x}_p(t) = \mathbf{\Phi}$

$$\dot{\mathbf{x}}_p(t) = \mathbf{0} = \mathbf{P}\mathbf{\Phi} + \mathbf{Q}$$

$$\mathbf{\Phi} = -\mathbf{P}^{-1}\mathbf{Q} \quad (\text{provided that } \mathbf{P} \text{ is not singular})$$

Initial condition: $\mathbf{x}(0) = \mathbf{\Psi} - \mathbf{P}^{-1}\mathbf{Q} = \mathbf{0}$

$$\mathbf{\Psi} = \mathbf{P}^{-1}\mathbf{Q}$$

Step response:

$$\begin{aligned}\mathbf{x}_s(t) &= (e^{\mathbf{P}t} - \mathbf{I})\mathbf{P}^{-1}\mathbf{Q} \\ &= \mathbf{P}^{-1}(e^{\mathbf{P}t} - \mathbf{I})\mathbf{Q}\end{aligned}$$

Exponential functions play important role in solving matrix diff eq's.

Computing Matrix Exponentials

A matrix exponential can always be found from its series expansion:

$$e^{\mathbf{P}} = \mathbf{I} + \mathbf{P} + \mathbf{P}^2/2! + \mathbf{P}^3/3! + \mathbf{P}^4/4! + \dots$$

To avoid computing infinite sums, we can diagonalize the matrix \mathbf{P} .

Start with the eigenvector/eigenvalue property:

$$\mathbf{v}_i \longrightarrow \boxed{\mathbf{P}} \longrightarrow \lambda_i \mathbf{v}_i$$

where λ_i is the i^{th} eigenvalue and \mathbf{v}_i is the i^{th} eigenvector (a column vector).

Assemble the eigenvectors into an eigenvector matrix:

$$\mathbf{V} = \left[\mathbf{v}_1 \mid \mathbf{v}_2 \mid \mathbf{v}_3 \mid \dots \mid \mathbf{v}_n \right]$$

and the eigenvalues into an eigenvalue matrix:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \dots \end{bmatrix}$$

If \mathbf{P} is full rank and if none of the eigenvalues are repeated

$$\mathbf{P} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

$$e^{\mathbf{P}} = \mathbf{V}e^{\mathbf{\Lambda}}\mathbf{V}^{-1}$$

Controller Design

Many methods to optimize performance of **classical controllers** choose gains to move closed-loop poles to locations that are favorable for

- stability,
- disturbance rejection,
- noise immunity, etc.

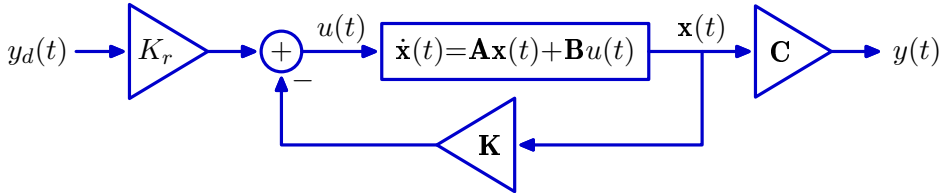
Example: the root-locus method allows us to see all of the closed-loop pole positions that can be accessed by changing a gain K .

More powerful design methods exist for state-space controllers.

For example, we can use the **pole placement** algorithm to set the closed-loop pole positions ANYWHERE in the complex plane!

Pole Placement

The pole placement algorithm determines gains \mathbf{K} and K_r to locate the closed-loop poles of a state-space model **anywhere** in the complex plane.



The closed-loop poles of a state-space model are equal to the roots of its characteristic polynomial:

$$\left| s\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K}) \right| = 0$$

Fundamental theorem of algebra: an n^{th} order polynomial as n roots.

Factor theorem: each root determines a first-order factor.

→ characteristic polynomial can be written as a product of first-order terms:

$$\left| s\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K}) \right| = \prod_{i=1}^n (s - s_i) = 0$$

LHS: n^{th} order polynomial in s (pole locations)

RHS: same polynomial, but coeff's in terms of desired pole locations s_i .

Pole Placement

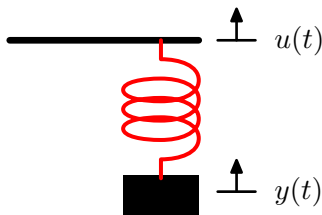
With full-state feedback, the gain \mathbf{K} can be adjusted to produce **ANY** set of n closed-loop poles! → **much more powerful than classical methods!**

The design problem shifts ...

- from finding gains to optimize pole locations (classical view)
- to finding pole locations to optimize performance (modern view).

Unfortunately, the relation between pole locations and performance is not simple. For example, we often have **multiple objectives**.

Example: Optimizing Performance



Plant:

$$\underbrace{k(u(t)-y(t))}_{F} - \underbrace{b\dot{y}(t)}_{ma} = m\ddot{y}(t)$$

Express differential equation as a first-order matrix differential equation:

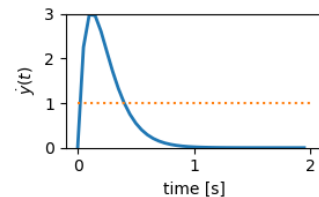
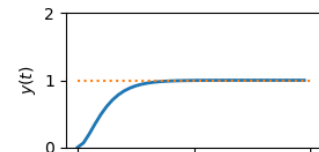
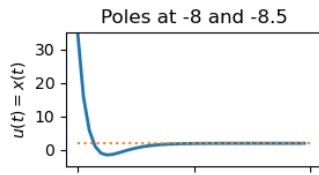
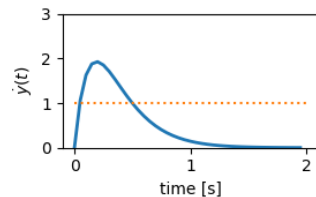
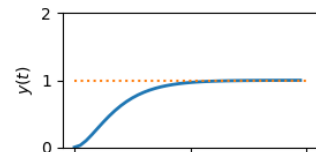
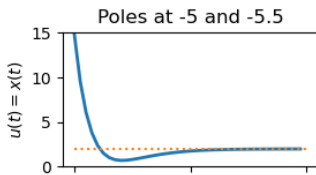
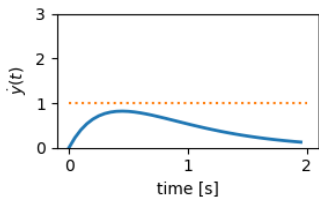
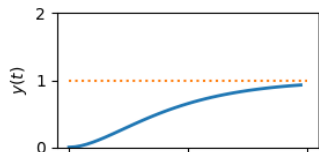
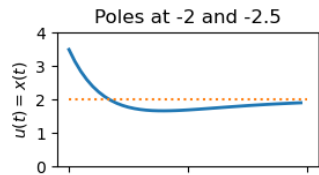
$$\underbrace{\frac{d}{dt} \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} 0 \\ k/m \end{bmatrix}}_{\mathbf{B}} u(t)$$

Decide where to put the two closed-loop poles s_i :

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K})| = \prod_{i=1}^n (s - s_i) = 0$$

Example: Optimizing Performance

We can place the poles anywhere – which places are best?



Large negative poles \rightarrow fast response. ✓

Large negative poles \rightarrow large effort. ✗

Example: Optimizing Performance

How do we find the “best” pole locations?

Which is better: small effort or fast response?

We'd like both – but that's not generally feasible.

Multiple Objectives

Find best way to go from 77 Mass Ave to Lexington Green (12 miles)? ¹



walk

BLUEbikes

bike



subway/bus

Uber



auto

¹adapted from Brian Douglas (MATLAB) and Christopher Lum (U Washington).

Multiple Objectives

Find best way to go from 77 Mass Ave to Lexington Green (12 miles)?

- How much does it cost?
- How long does it take?

	mode	dollars	time
	walk	\$0.00	3h 50m
BLUEbikes	bike	\$10.00	1h 4m
	subway/bus	\$2.40	1h 2m
Uber	auto	\$38.33	46m



It's easy to optimize either objective. But we'd like a solution that costs nothing and takes no time → not **feasible**.

How do we find the best feasible solution? What does "best" even mean?

Prioritizing Mixed Objectives with a Cost Function

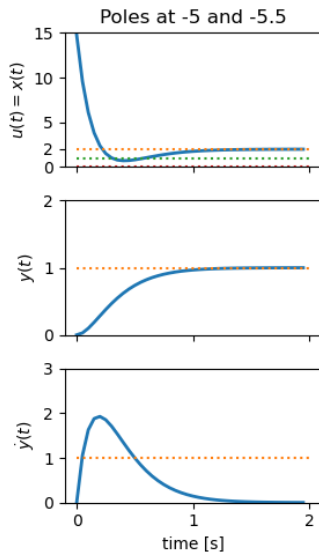
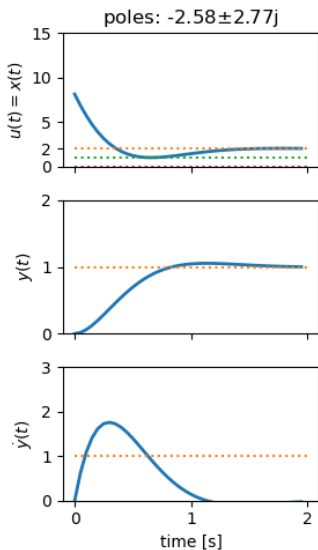
More generally, we can define a **cost function** to assign a real-valued penalty to all possible scenarios.

- cost: 1 point per dollar + 1/10 point per minute

	mode	dollars	time	cost
	walk	\$0.00	3h 50m	23
BLUEbikes	bike	\$10.00	1h 4m	16.4
	subway/bus	\$2.40	1h 2m	8.6
Uber	auto	\$38.33	46m	42.93

Cost Functions for the Mass-Spring-Dashpot

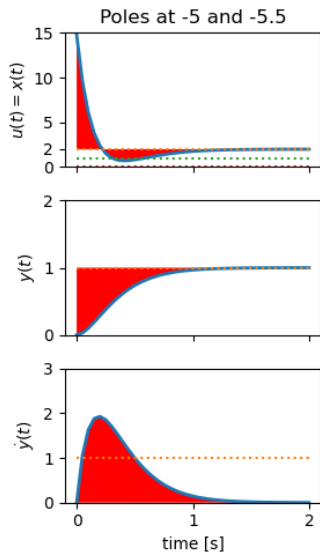
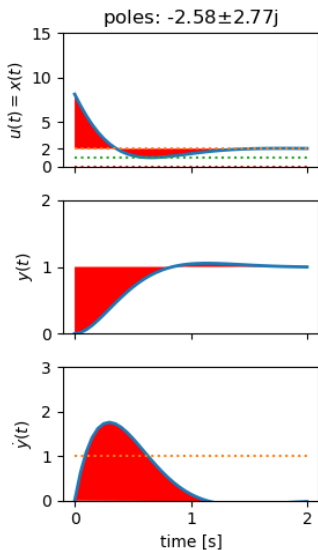
We could assign costs based on $x(t)$ or peak value of $y(t)$.



A better cost function might consider entire time functions ($x(t)$ and $y(t)$).

Cost Functions for the Mass-Spring-Dashpot

Mean squares: integrate squared errors: $\int (\text{desired} - \text{measured})^2 dt$.



Squaring penalizes both positive and negative errors, and it's mathematically tractable.

Quadratic Cost Functions

Define a cost function J that depends on the integral of the squares of the elements of $\mathbf{x}(t)$ and $\mathbf{u}(t)$:

$$J = \int_0^{\infty} \left(\mathbf{x}^T(t) \mathbf{Q} \mathbf{x}(t) + \mathbf{u}^T(t) \mathbf{R} \mathbf{u}(t) \right) dt$$

where \mathbf{Q} and \mathbf{R} are matrix constants that we can choose so as to weight errors in each component of $\mathbf{x}(t)$ and $\mathbf{u}(t)$ differently.

The goal will be to find the gain matrices \mathbf{K} and K_r to minimize J .

Linear Quadratic Regulator (LQR¹)

We want to find the gain matrix \mathbf{K} that minimizes the cost function

$$J = \int_0^{\infty} \left(\mathbf{x}^T(t) \mathbf{Q} \mathbf{x}(t) + \mathbf{u}^T(t) \mathbf{R} \mathbf{u}(t) \right) dt$$

where $\mathbf{u}(t)$ and $\mathbf{x}(t)$ are related

- by the state transition equation: $\dot{\mathbf{x}}(t) = \mathbf{A} \mathbf{x}(t) + \mathbf{B} \mathbf{u}(t)$ and
- by the feedback constraint (for homogeneous case): $\mathbf{u}(t) = -\mathbf{K} \mathbf{x}(t)$.

The “optimal” \mathbf{K} can be shown to be given by

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S}$$

where \mathbf{S} is the symmetric $n \times n$ solution to the **algebraic Riccati equation**:

$$\mathbf{A}^T \mathbf{S} + \mathbf{S} \mathbf{A} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{Q} = \mathbf{0}$$

¹ quadratic regulation of a linear system

LQR Solution

Fortunately there are efficient algorithms for solving the LQR problem.

Given the state-space matrices \mathbf{A} and \mathbf{B} and the LQR weights \mathbf{Q} and \mathbf{R} , the following Python code

```
> from control import lqr  
> K,S,E = lqr(A,B,Q,R)
```

and MATLAB code

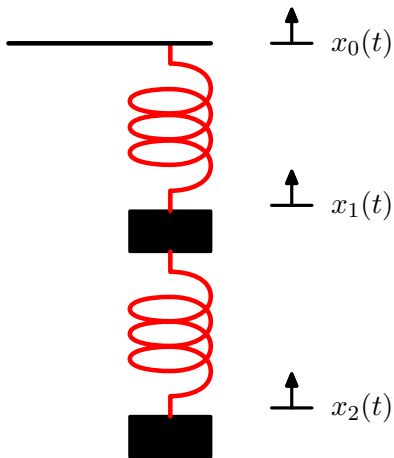
```
> K,S,E = lqr(A,B,Q,R);
```

finds the optimal solutions and returns

- K : state feedback gains,
- S : solution to the algebraic Riccati equation, and
- E : eigenvalues of the resulting closed-loop system.

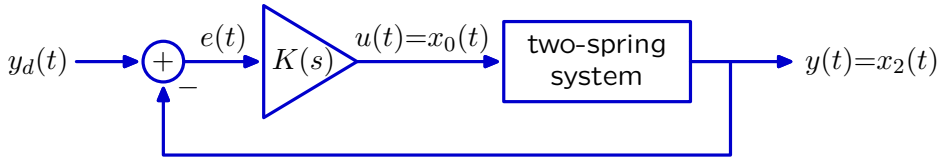
Example: Two-Spring System

A **plant** consists of two springs and two masses. Use the input $u(t) = x_0(t)$ to move the bottom mass to the desired location $x_2(t) = y_d(t)$.



Classical Control

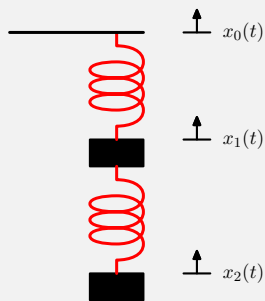
A classical controller for this problem has the following form.



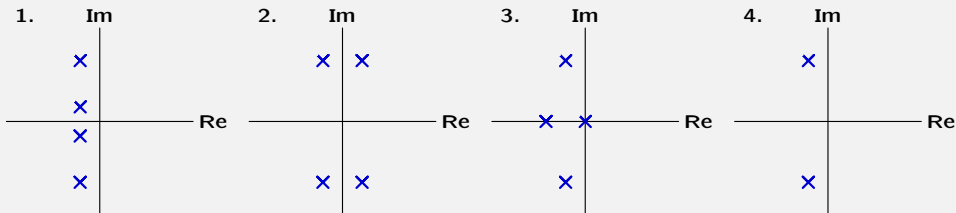
To solve this classical control problem, we must

- find the equations of motion for the plant (the two-spring system) and
- express those equations in terms of a transfer function.

Check Yourself

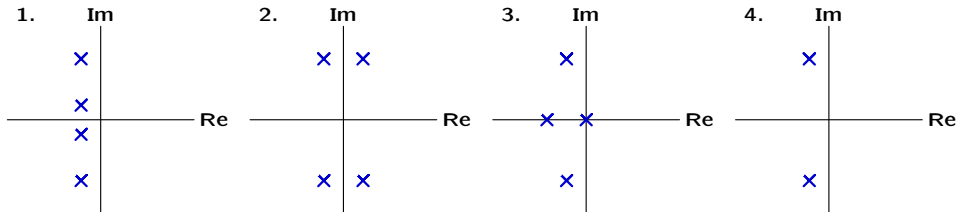


Which plot (if any) shows the poles of the two-spring system?



Check Yourself

Which plot (if any) shows the poles of the two-spring system?

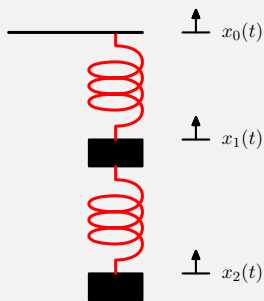


All of the elements in the two-spring system are passive – they cannot create energy. Therefore none of the poles can be in the right half-plane. Therefore the answer cannot be 2. Intrinsic losses also prevent solution 3.

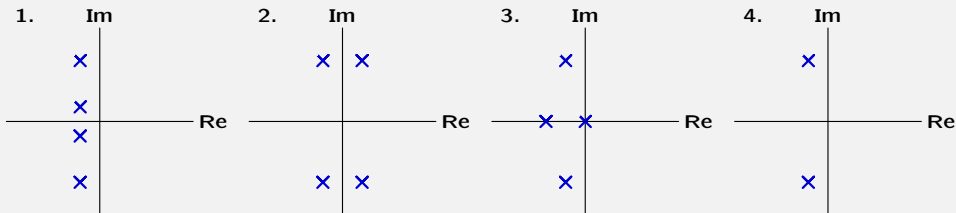
There are 4 poles. The force exerted by a spring is determined by the displacements of its endpoints. The force on a mass is proportional to acceleration, which is the second derivative of displacement and therefore scales as s^2 in frequency. Since there are two masses, the number of characteristic equation will be quartic in s .

Thus the answer is 1.

Check Yourself

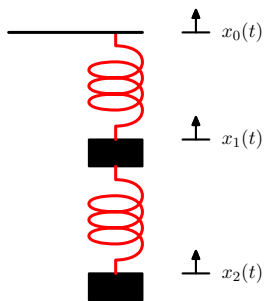


Which plot (if any) shows the poles of the two-spring system? 1



Two-Spring System

Equations of motion.



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

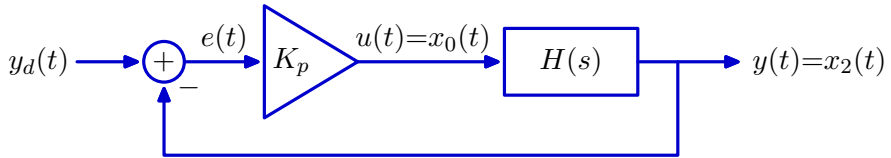
$$f_{m2} = m\ddot{x}_2(t) = k(x_1(t) - x_2(t)) - b\dot{x}_2(t)$$

Transfer function:

$$H(s) = \frac{X_2(s)}{X_0(s)} = \frac{k^2}{(s^2m + sb + 2k)(s^2m + sb + k) - k^2}$$

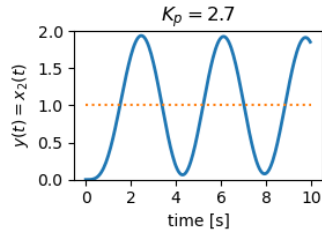
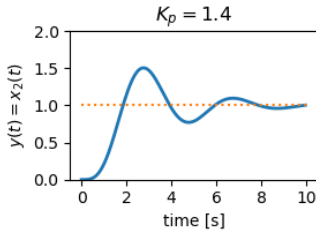
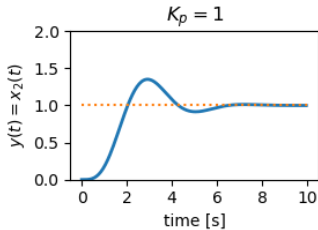
Classical Control

Try proportional control.



The feedback system is stable for only a small range of gains: $K_p < 2.7$

Step responses (mass $m = 1$, stiffness $k = 2$, damping $b = 1.4$):

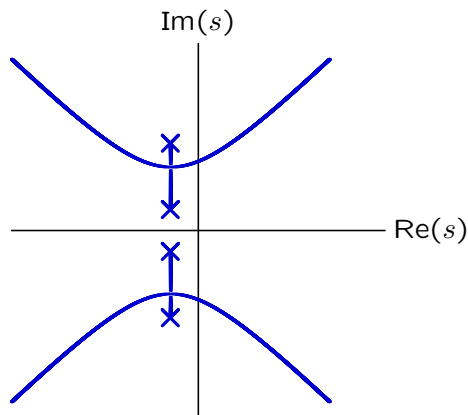


Slow convergence and large oscillatory overshoots.

Why such poor behavior?

Classical Control

Root Locus: As K_p increases, the lower and higher frequency poles converge with no change in damping, then split and approach asymptotic trajectories at angles of $\pm\pi/4$ and $\pm 3\pi/4$. Unstable when poles enter right half-plane.

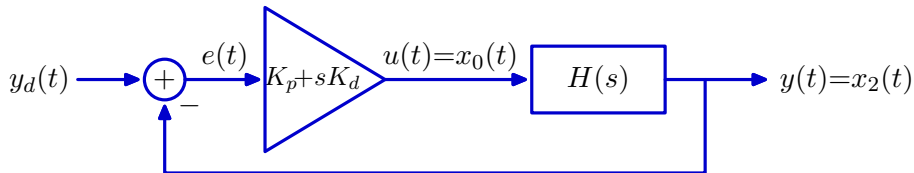


Good explanation of what happened.

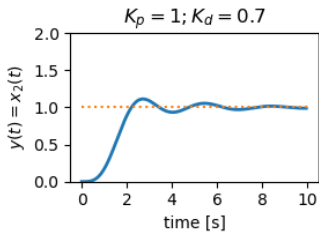
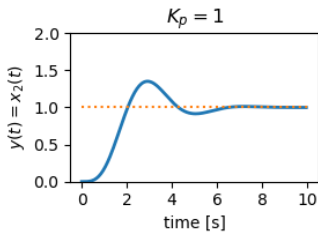
Try proportional plus derivative control.

Classical Control

Proportional plus derivative performance is only slightly better.



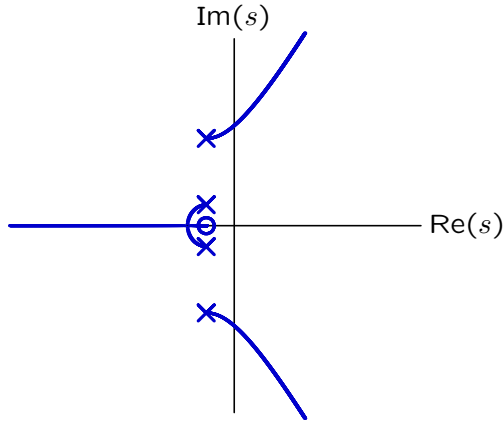
Step responses:



Somewhat smaller overshoot, but still slow convergence.

Classical Control

Root Locus: Increase K_p while holding $K_d = K_p/0.7$. Derivative term adds a zero and changes the asymptotic behavior, but closed-loop system still goes unstable.

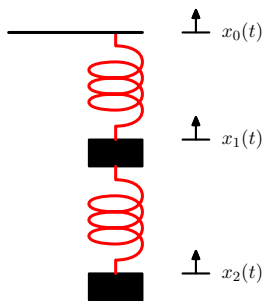


Good explanation of what happened – but how do we make it faster?
Try state-space approach.

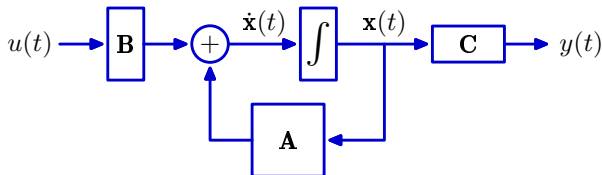
Check Yourself

Find \mathbf{A} , \mathbf{B} , and \mathbf{C} so that $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$ and $y = \mathbf{C}\mathbf{x}$.

How many non-zero entries are in \mathbf{A} ?



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



State-Space Description

Equations of motion:

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

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

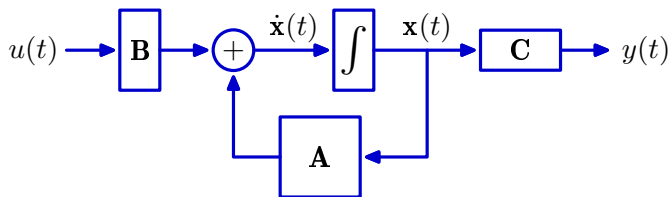
Four state variables (two displacements and their velocities):

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

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

State-Space Model

How many non-zero entries are in \mathbf{A} ? 8



$$\mathbf{A} = \begin{bmatrix} -\frac{b}{m} & -\frac{2k}{m} & 0 & \frac{k}{m} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{k}{m} & -\frac{b}{m} & -\frac{k}{m} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \frac{k}{m} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{C} = [0 \ 0 \ 0 \ 1]$$

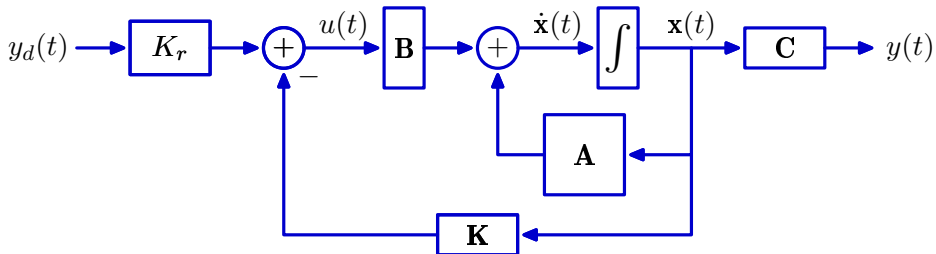
$$\mathbf{x}(t) = \begin{bmatrix} v_1(t) \\ x_1(t) \\ v_2(t) \\ x_2(t) \end{bmatrix}$$

$$u(t) = x_0(t)$$

$$y(t) = x_2(t)$$

State-Space Controller

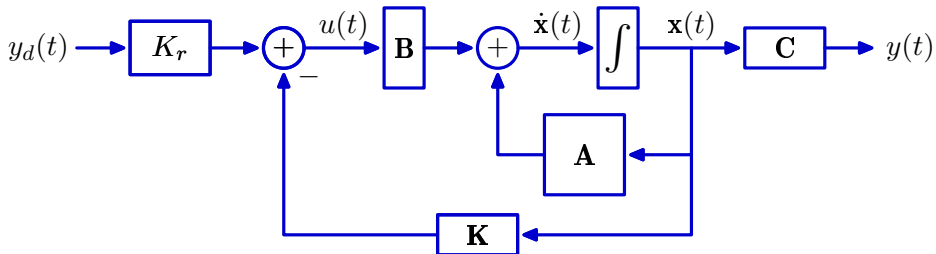
A state-space **controller** can then be expressed as follows.



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

State-Space Controller

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



Find **K** with **pole placement**:

$$K = \text{place}(A, B, [\text{poles}])$$

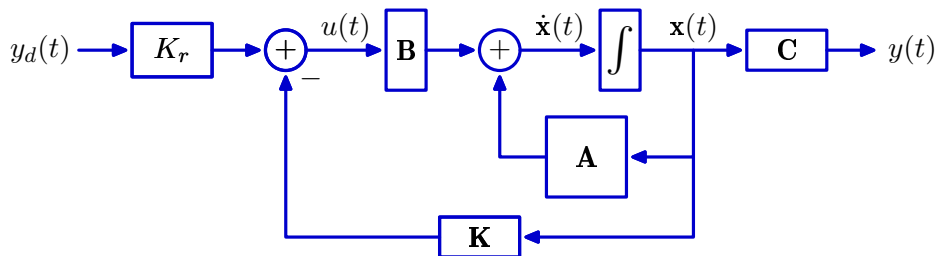
or **LQR**:

$$K = \text{lqr}(A, B, Q, R) \text{ where } Q = \text{diag}([1, 1, 1, 1]) \text{ and } R = 1$$

How to find K_r ?

State-Space Controller

K_r does not affect stability. Choose K_r to minimize steady-state error.



Find the steady-state values of \mathbf{x} :

$$\dot{\mathbf{x}} = \mathbf{0} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{BK}_r y_d$$

$$\mathbf{x} = -(\mathbf{A} - \mathbf{BK})^{-1} \mathbf{BK}_r y_d$$

We want $y = y_d$:

$$y = \mathbf{C}\mathbf{x} = -\mathbf{C}(\mathbf{A} - \mathbf{BK})^{-1} \mathbf{BK}_r y_d$$

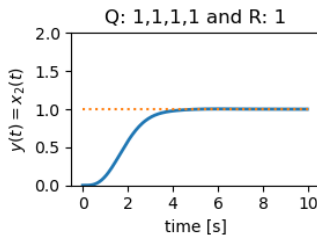
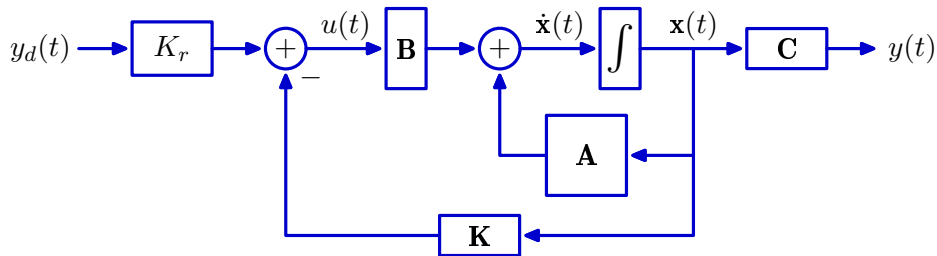
Divide out y_d (under the assumption that $y = y_d \neq 0$):

$$K_r = \frac{-1}{\mathbf{C}(\mathbf{A} - \mathbf{BK})^{-1} \mathbf{B}}$$

State-Space Control

Try LQR.

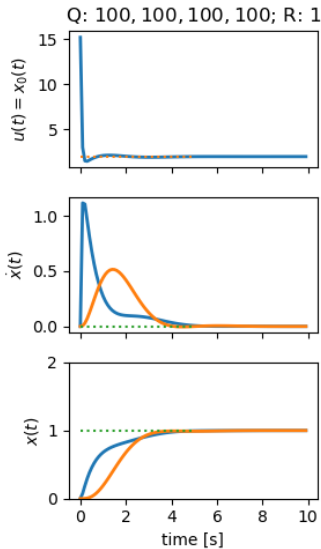
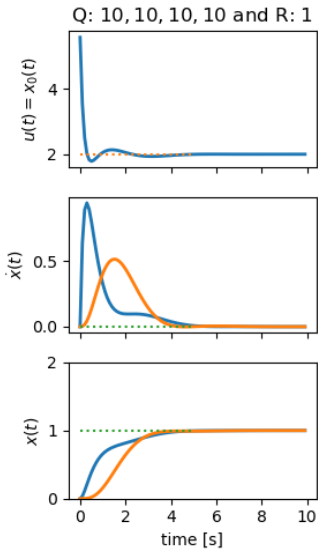
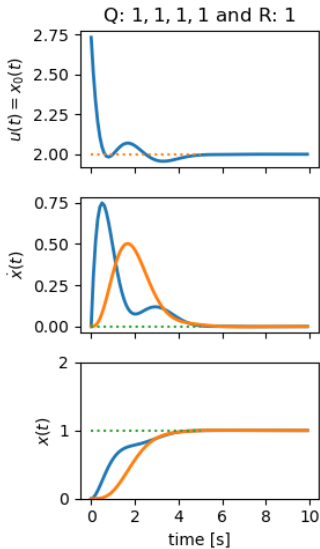
Start with flat parameters $\mathbf{Q} = \text{diag}([1, 1, 1, 1])$ and $\mathbf{R} = [[1]]$.



Convergence is slow but **monotonic**. Can we make it faster?

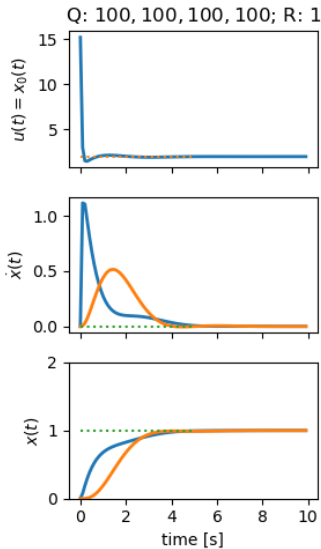
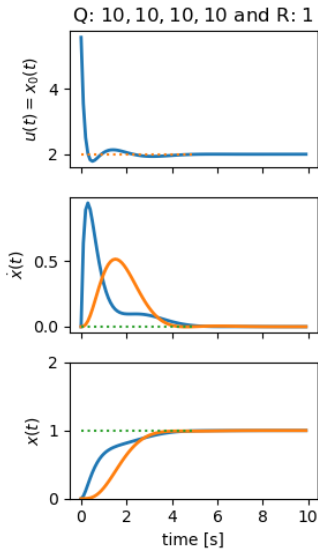
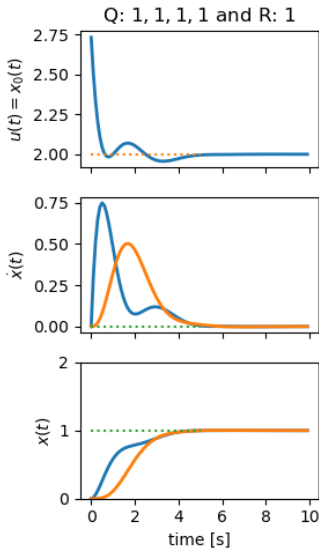
State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .



State-Space Control

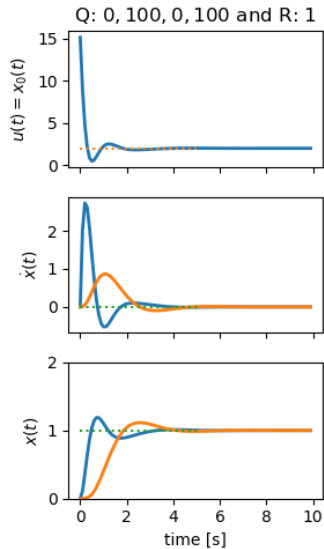
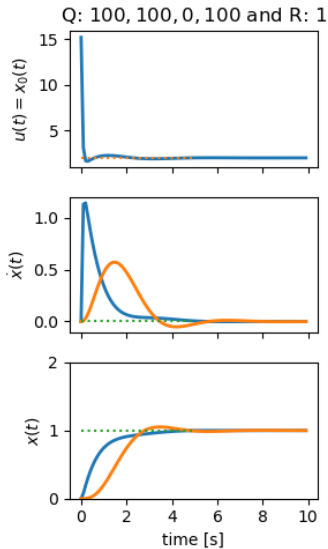
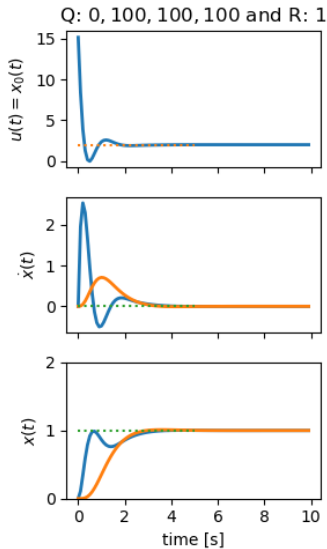
Try different values of \mathbf{Q} and \mathbf{R} .



Increasing all values of \mathbf{Q} doesn't have a big effect on positions x_1 or x_2 .
Has big effect on effort $u(t)$.

State-Space Control

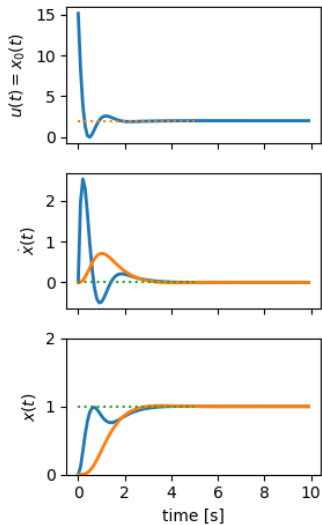
Try different values of \mathbf{Q} and \mathbf{R} .



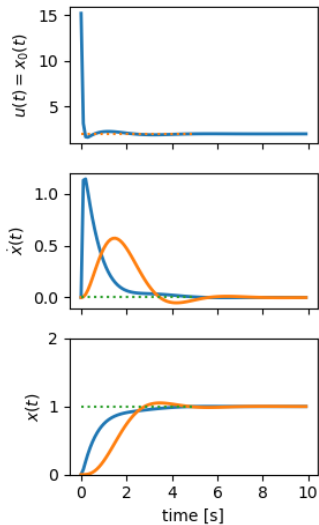
State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .

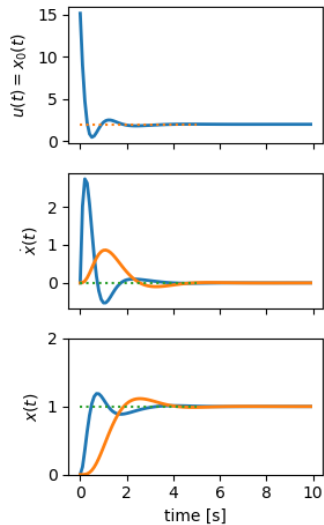
Q: 0, 100, 100, 100 and R: 1



Q: 100, 100, 0, 100 and R: 1



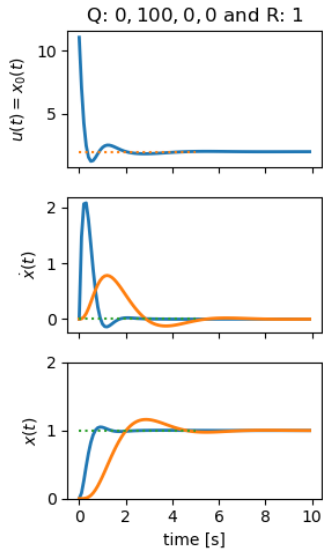
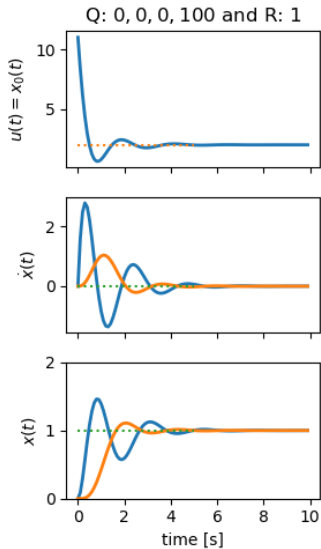
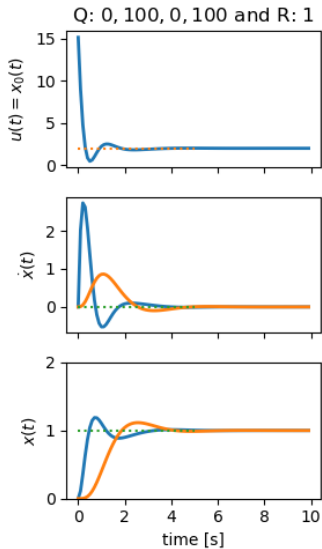
Q: 0, 100, 0, 100 and R: 1



Reducing weight on $v_1(t)$ has bigger effect than reducing weight on $v_2(t)$.
Reducing weight on both velocities speeds the displacements most.

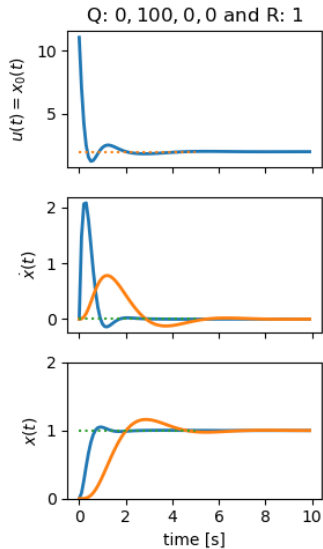
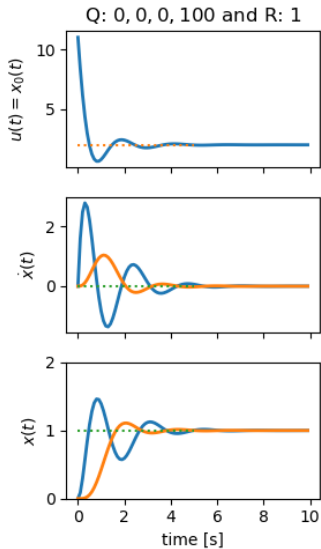
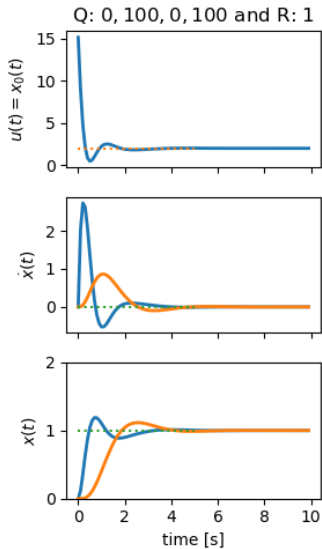
State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .



State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .

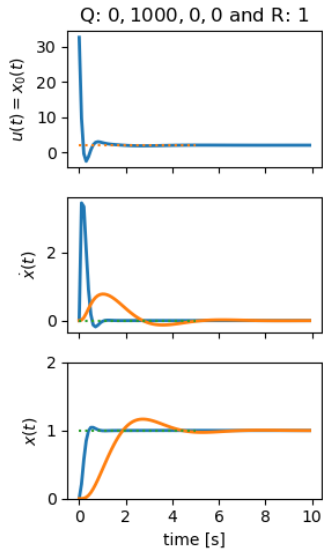
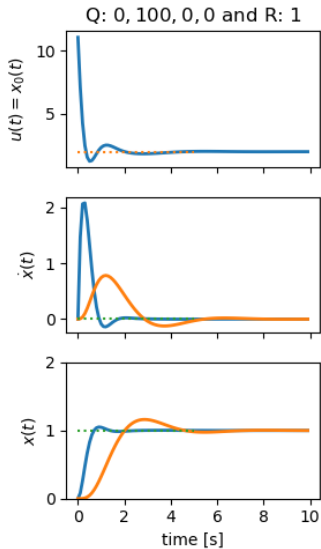
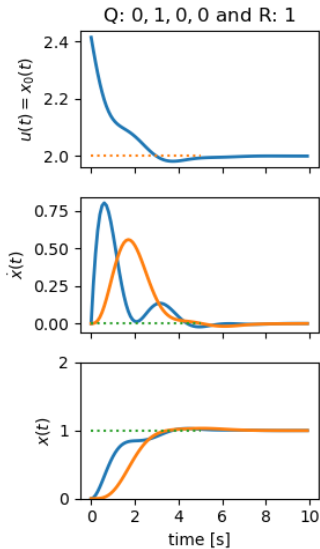


Weighting just the output x_2 lets $x_1(t)$ get very large.

Weighting just displacement x_1 works surprisingly well.

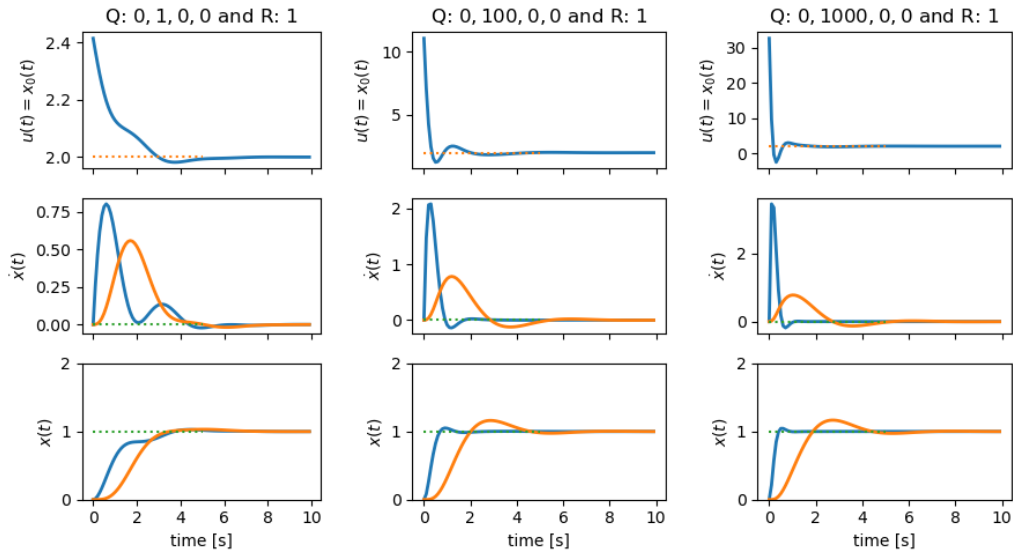
State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .



State-Space Control

Try different values of \mathbf{Q} and \mathbf{R} .



Increasing weight on x_1 improves convergence of $x_2(t)$ but increases effort.

Summary

State-space control with full-state feedback offers technical and intuitive advantages over the most common types of classical control.

The pole placement algorithm allows one to specify the locations of all of the closed-loop poles.

LQR provides intuitive refinement of feasible solutions to a control problem.