6.3100: Dynamic System Modeling and Control Design

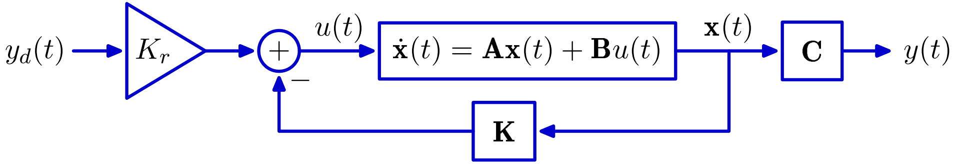
State-Space Design

November 06, 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 **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:

$$\dot{\mathbf{x}}(t) = \Big(\mathbf{A} - \mathbf{B}\mathbf{K}\Big)\mathbf{x}(t) + \mathbf{B}\mathbf{K_r} y_d(t) \equiv \mathbf{A_c}\mathbf{x}(t) + \mathbf{B_c} y_d(t)$$

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 $|Y(s)| \to \infty$.

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

$$\left|s\,\mathbf{I}-\mathbf{A_c}\right| = 0$$

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: $\dot{\mathbf{x}}_\mathbf{h}(t) = \mathbf{P}\mathbf{x}_\mathbf{h}(t)$

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

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

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

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

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:

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

To avoid computing infinite sums, we can diagonalize the matrix $\mathbf{P}$.
Start with the eigenvector/eigenvalue property:

$$\begin{tikzcd} \mathbf{v_i} \arrow[r] & \boxed{\mathbf{P}} \arrow[r] & \lambda_i \mathbf{v_i} \end{tikzcd}$$

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:

$$\mathbf{V} = \left[\mathbf{v_1} \middle| \mathbf{v_2} \middle| \mathbf{v_3} \middle| \cdots \middle| \mathbf{v_n}\right]$$

and the eigenvalues into an eigenvalue matrix:

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

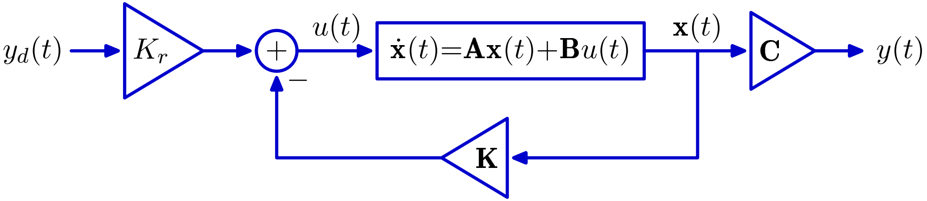
If $\mathbf{P}$ is full rank and if none of the eigenvalues are repeated

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

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

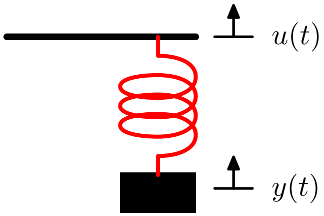
Controller Design

Optimizing the gains $\mathbf{K}$ and K_r of a state-space controller.



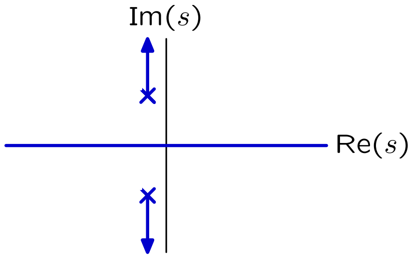
Last time we reviewed two examples of classical design.

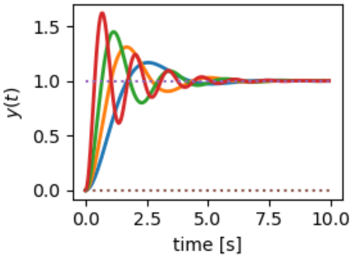
Root-Locus Analysis of Proportional Control



Consider the closed-loop poles $s_1, s_2 = \dfrac{-b \pm \sqrt{b^2 - 4mk(1+K_p)}}{2m}$

when $m = 1$, $b = 1.4$, $k = 2$, and K_p increases from 0 to ∞.





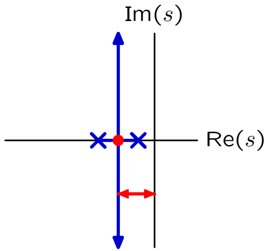
As $K_p \uparrow$, frequency of ringing $\uparrow$, but peaks decay with same time constant.

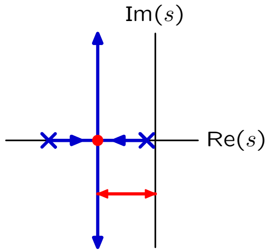
Root-Locus Analysis of PD Control

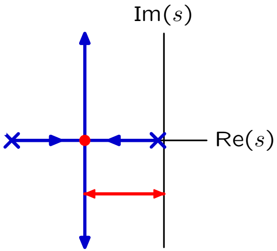
Increasing K_d enables faster closed-loop poles (red dots) without overshoot.

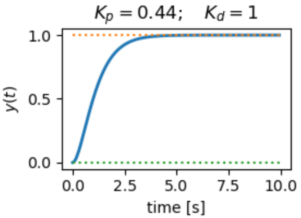
$$s = \frac{-b - kK_d \pm \sqrt{(b+kK_d)^2 - 4mk(1+K_p)}}{2m}$$

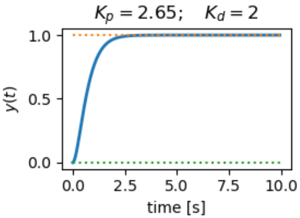
$m = 1;\ b = 1.4;\ k = 2;\ K_d = 1, 2, 3;\ K_p\colon 0 \to \infty.$

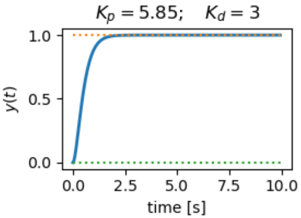












More Advanced Methods in Classical Control

Derivative feedback is just one way to optimize a classical controller.

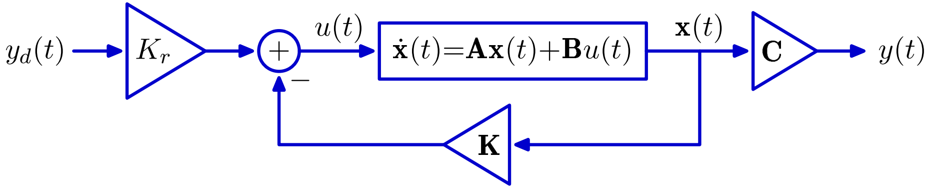
Other advanced classical methods include

- integral feedback for PID control,
- optimizing gain and phase margins,
- lead compensation, lag compensation, lead/lag compensation, and
- many other techniques.

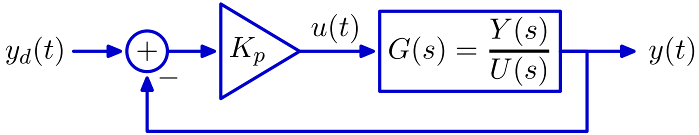
How well do these methods translate to state space?

Check Yourself: Proportional Control in State Space

Choose $\mathbf{K}$ and K_r so that the state-space controller:



is equivalent to a proportional controller:

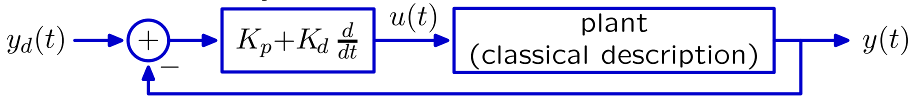


What $\mathbf{K}$ and K_r implement a proportional controller?

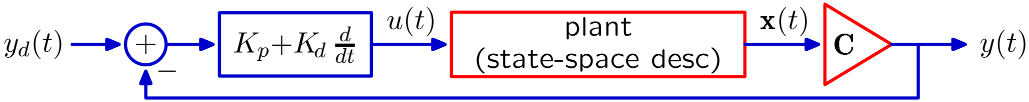
1. $\mathbf{K}=K_r\mathbf{C}$ and $K_r = K_p$
2. $\mathbf{K}=\mathbf{CB}$ and $K_r = 1$
3. $\mathbf{K}=\mathbf{C}$ and $K_r = K_p$
4. $\mathbf{K}=K_p\mathbf{C}$ and $K_r = K_p$
5. none of the above

Proportional Plus Derivative Control in State-Space

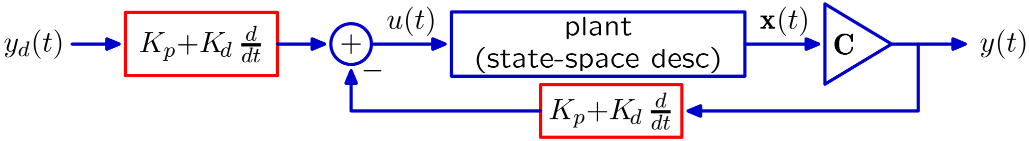
Start with a classical system with PD control:



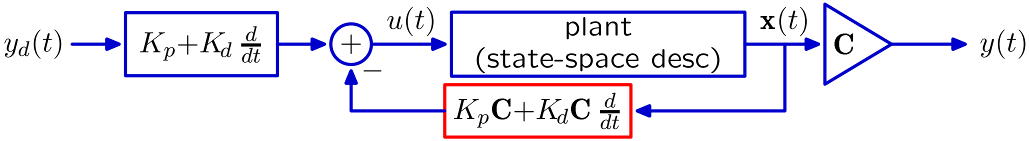
Replace classical description of plant with equiv. state-space description:



Distribute the PD controller over the inputs to the subtractor:



Feedback $\mathbf{x}(t)$ instead of $y(t)$:



Result is a state-space controller with $K_r=K_p+K_d\frac{d}{dt}$ and $\mathbf{K}=K_p\mathbf{C}+K_d\mathbf{C}\,\frac{d}{dt}$.

More Advanced Methods in Classical Control

Much of the design power of the more advanced methods results from their ability to **move poles and zeros** to locations that are more favorable for

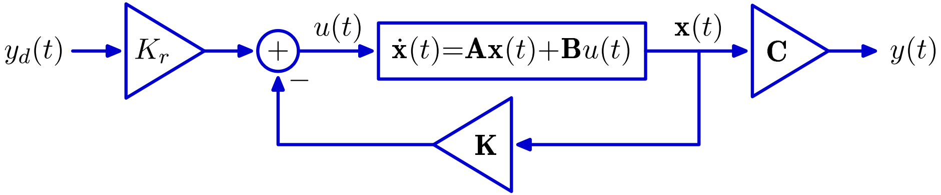
- stability,
- disturbance rejection,
- noise immunity, etc.

We can similarly optimize state-space controllers.

And the state-space formulation is much more powerful!

Pole Placement

With the correct choice of gains $\mathbf{K}$ and K_r, we can move 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:

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

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

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

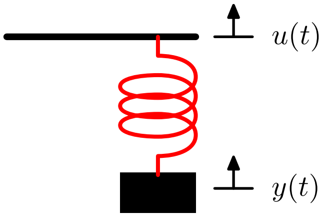
$\rightarrow$ characteristic polynomial can be written as a product of first-order terms:

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

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

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

Example: Pole Placement for Mass-Spring-Dashpot



Plant:

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

Rewrite this second-order differential equation as two **first-order** equations:

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

which can be expressed as a single matrix equation:

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

Example: Pole Placement for Mass-Spring-Dashpot

For $m = 1$, $b = 1.4$, and $k = 2$:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -1.4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

With $\mathbf{K} = [\, k_1 \quad k_2 \,]$:

$$|s\mathbf{I}-(\mathbf{A}-\mathbf{BK})| = \left| \begin{bmatrix} s & -1 \\ 2+2k_1 & s+1.4+2k_2 \end{bmatrix} \right| = s^2+(1.4+2k_2)s+(2+2k_1)$$

To place poles at **s = −0.5 and s = −1**, set

$$s^2+(1.4+2k_2)s+(2+2k_1) = (s+0.5)(s+1) = s^2+1.5s+0.5$$

$\rightarrow k_1 = -0.75$ and $k_2 = 0.05$.

Alternatively, to place poles at **s = −0.5 and s = −0.6**, set

$$s^2+(1.4+2k_2)s+(2+2k_1) = (s+0.5)(s+0.6) = s^2+1.1s+0.3$$

$\rightarrow k_1 = -0.85$ and $k_2 = -0.15$.

Check Yourself

Let $\mathbf{A}$ represent the system matrix and $\mathbf{B}$ represent the input matrix for a state-space control system, where these matrices are given by

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

Which $\mathbf{K}$ vector will produce closed-loop poles at 0 and -2?

1. $\mathbf{K} = [\,1 \quad 2\,]$
2. $\mathbf{K} = [\,3 \quad 4\,]$
3. $\mathbf{K} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
4. $\mathbf{K} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$
5. none of the above

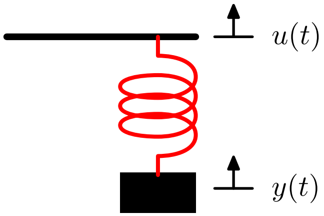
Pole Placement

With full-state feedback, the gains $\mathbf{K}$ can be adjusted to produce **ANY** set of n closed-loop poles! $\rightarrow$ **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).

Example: Optimizing Performance



Plant:

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

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

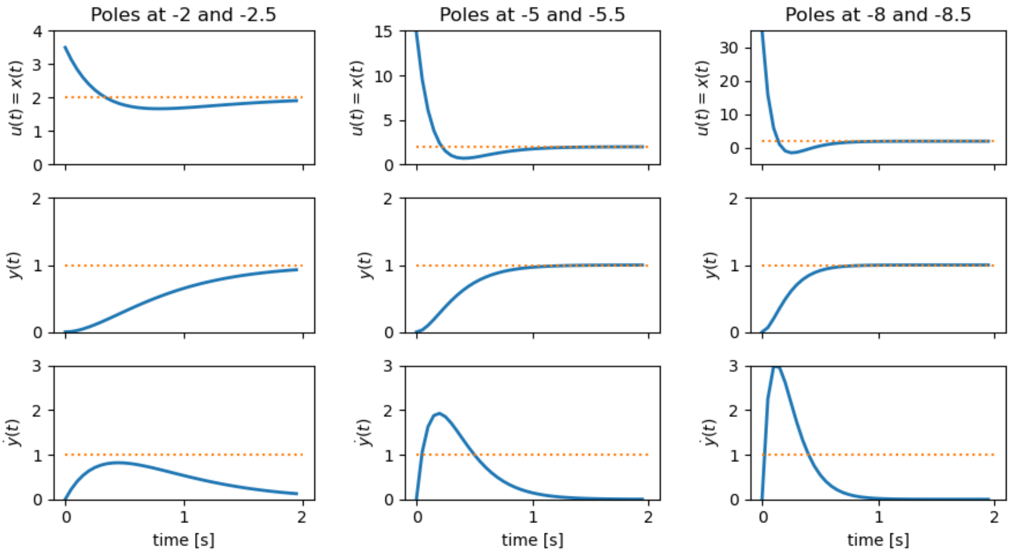
$$\underbrace{\frac{d}{dt}\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} 0 \\ k/m \end{bmatrix}}_{\mathbf{B}}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

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

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

Example: Optimizing Performance

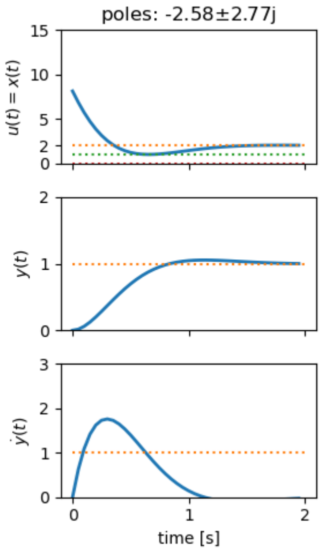
We can place the poles anywhere – which places are best?

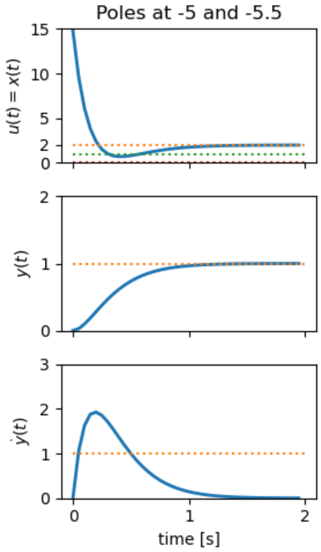


Which pole pair is better: left, center, or right column?

Example: Optimizing Performance

Complex poles can reduce initial displacement without sacrificing speed.





How can we systematically find good pole locations?

Example: Optimizing Performance

How do we find the "best" pole locations?

Which is better: a small input or a fast response?

– **comparing apples to oranges**!

Cost Functions

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

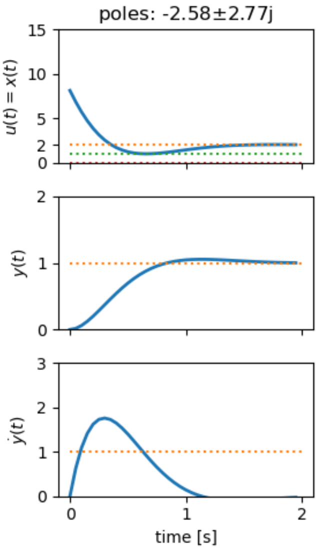
	mode	dollars	time	cost
	walk	$0.00	3h 50m	23
BLUEbikes	bike	$10.00	1h 4m	16.4
	subway/bus	$2.40	1h 2m	8.6
Uber	auto	$38.33	46m	42.93

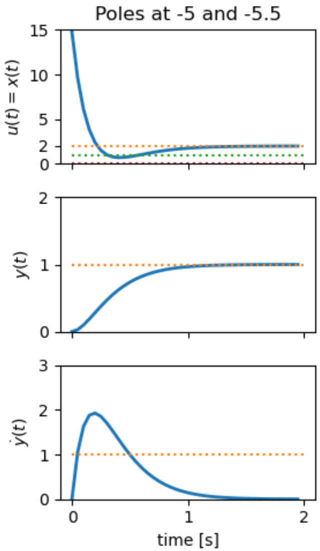
What would you optimize?

- dollars
- time
- something else?

Cost Functions for the Mass-Spring-Dashpot

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

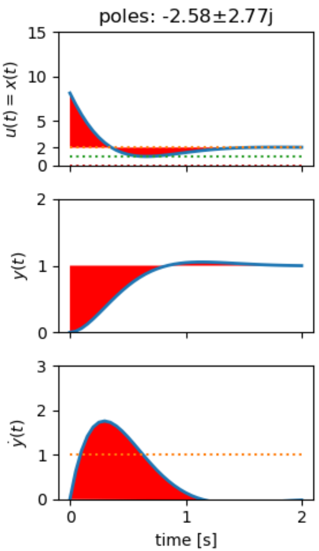


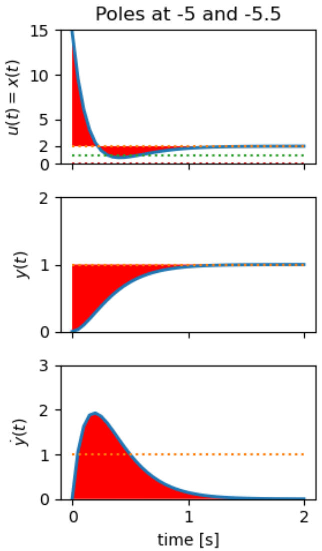


A better cost function might consider entire time functions ($x(t)$ and $y(t)$).

Cost Functions for the Mass-Spring-Dashpot

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





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

Quadratic Cost Functions

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

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

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.

Check Yourself

Consider the cost function

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

when $\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ and $\mathbf{u}(t)$ is a scalar. Find $\mathbf{Q}$ and $\mathbf{R}$ so that

- $\int_0^\infty x_1^2(t)\,dt$ is weighted **twice** as much as $\int_0^\infty u^2(t)\,dt$, and
- $\int_0^\infty x_2^2(t)\,dt$ is weighted **three times** as much as $\int_0^\infty u^2(t)\,dt$.

Which (if any) of the following satisfy these weights?

1. $\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\mathbf{R} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$
2. $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$, $\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
3. $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$, $\mathbf{R} = [1]$
4. $\mathbf{Q} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{3} \end{bmatrix}$, $\mathbf{R} = [1]$
5. none of the above

Linear Quadratic Regulator (LQR[1])

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

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

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

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

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

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

[1] quadratic regulation of a linear system

Algebraic Riccati Equation

Let $\mathbf{A}=\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}$, $\mathbf{B}=\begin{bmatrix}1\\ 1\end{bmatrix}$, $\mathbf{Q}=\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}$, and $\mathbf{R}=[\,1\,]$.

Find the matrix $\mathbf{S}$ that satisfies the **algebraic Riccati equation**:

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

$$\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}a & b\\ b & c\end{bmatrix}+\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}-\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}[\,1\,][\,1 \quad 1\,]\begin{bmatrix}a & b\\ b & c\end{bmatrix}+\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}a & b\\ 2b & 2c\end{bmatrix}+\begin{bmatrix}a & 2b\\ b & 2c\end{bmatrix}-\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}a & b\\ b & c\end{bmatrix}+\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}a & b\\ 2b & 2c\end{bmatrix}+\begin{bmatrix}a & 2b\\ b & 2c\end{bmatrix}-\begin{bmatrix}(a+b)^2 & (a+b)(b+c)\\ (a+b)(b+c) & (b+c)^2\end{bmatrix}+\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}(a+b)^2 & (a+b)(b+c)\\ (a+b)(b+c) & (b+c)^2\end{bmatrix}=\begin{bmatrix}2a+2 & 3b\\ 3b & 4c+3\end{bmatrix}$$

These equations are nonlinear, so there are multiple solutions (4 here):

$$\mathbf{S}=\begin{bmatrix}40.293 & -49.381\\ -49.381 & 65.682\end{bmatrix}\text{ or }\begin{bmatrix}8.569 & -4.194\\ -4.194 & 1.318\end{bmatrix}\text{ or }\begin{bmatrix}-0.059 & 1.431\\ 1.431 & 1.699\end{bmatrix}\text{ or }\begin{bmatrix}-0.782 & 0.122\\ 0.122 & -0.674\end{bmatrix}.$$

Algebraic Riccati Equation

Let $\mathbf{A}=\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}$, $\mathbf{B}=\begin{bmatrix}1\\ 1\end{bmatrix}$, $\mathbf{Q}=\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}$, and $\mathbf{R}=[\,1\,]$.

Then the algebraic Riccati equation:

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

has four possible solutions (row 1 below).

Each possible solution $\mathbf{S}$ corresponds to a different gain matrix $\mathbf{K}$ (row 2):

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

Each gain matrix $\mathbf{K}$ generates a different set of eigenvalues (row 3 below).

	solution 1	solution 2	solution 3	solution 4
$\mathbf{S}$:	$\begin{bmatrix}40.293 & -49.381\\ -49.381 & 65.682\end{bmatrix}$	$\begin{bmatrix}8.569 & -4.194\\ -4.194 & 1.318\end{bmatrix}$	$\begin{bmatrix}-0.059 & 1.431\\ 1.431 & 1.699\end{bmatrix}$	$\begin{bmatrix}-0.782 & 0.122\\ 0.122 & -0.674\end{bmatrix}$
gain $\mathbf{K}$:	$[\,-9.088 \quad 16.301\,]$	$[\,4.375 \quad -2.876\,]$	$[\,1.372 \quad 3.130\,]$	$[\,-0.660 \quad -0.552\,]$
poles s_1, s_2:	$(-1.357, -2.856)$	$(-1.356, 2.857)$	$(1.356, -2.857)$	$(1.356, 2.856)$

Which (if any) of these solutions should we select?

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);
```

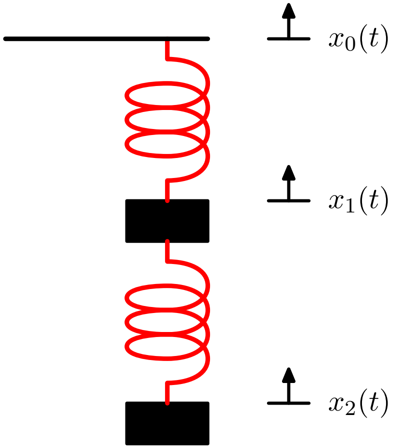
finds the optimal solutions and returns

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

Example: Two-Spring System

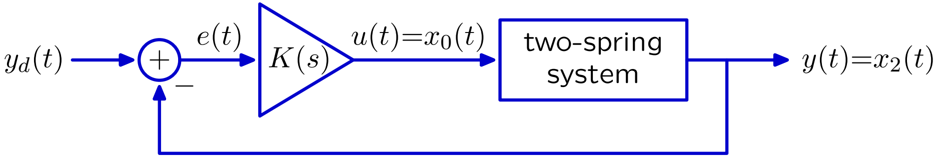
The **plant** consists of two springs and two masses.

The goal is to move the input $u(t) = x_0(t)$ so as to move the bottom mass to a desired location $x_2(t) = y_d(t)$.



Classical Control

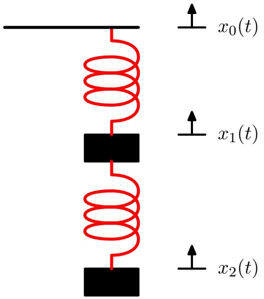
A classical controller for this problem has the following form.



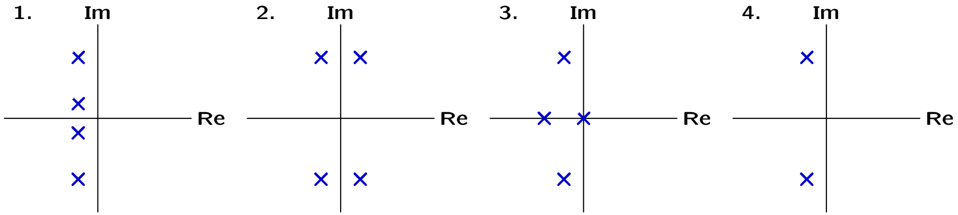
To solve this classical control problem, we must

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

Check Yourself

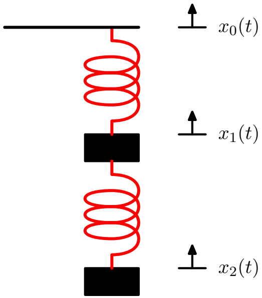


Which plot (if any) shows the poles of the two-spring system?



Two-Spring System

Equations of motion.



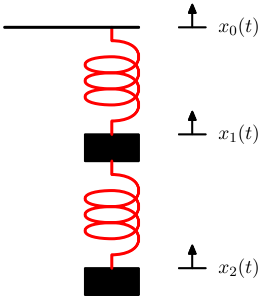
$$f_{m1} = m\ddot{x}_1(t) = k\Big(x_0(t){-}x_1(t)\Big) - k\Big(x_1(t){-}x_2(t)\Big) - b\dot{x}_1(t) - mg$$
$$f_{m2} = m\ddot{x}_2(t) = k\Big(x_1(t){-}x_2(t)\Big) - b\dot{x}_2(t) - mg$$

mass $m = 1$, stiffness $k = 2$, damping $b = 1.4$.

Positions $x_1(t)$ and $x_2(t)$ result from two separable inputs: gravity mg, which generates constant offsets, and $x_0(t)$, which determines the dynamics.

Two-Spring System

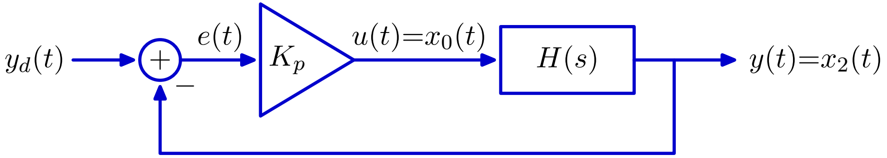
Transfer function.



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

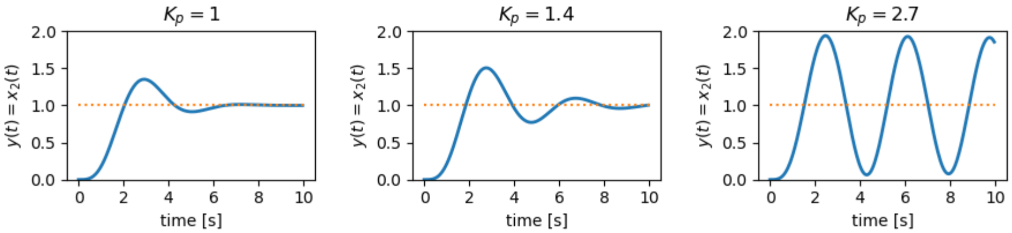
Classical Control

A proportional controller has the following form.



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

Step responses:

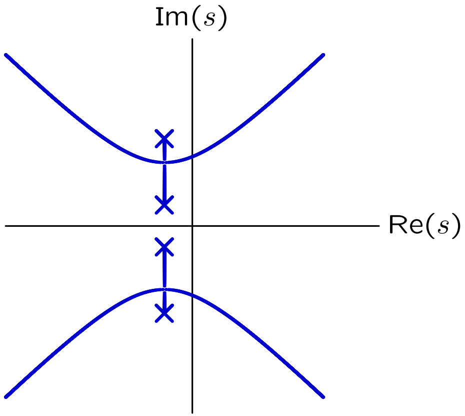


Slow convergence and large oscillatory overshoots.

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.

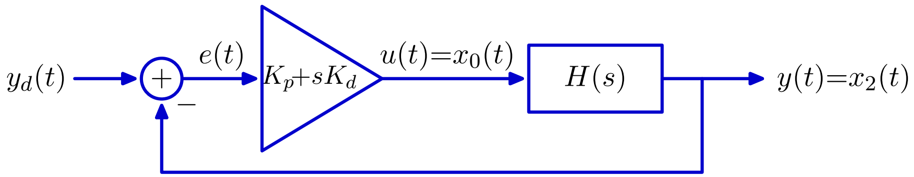


Good explanation of what happened.

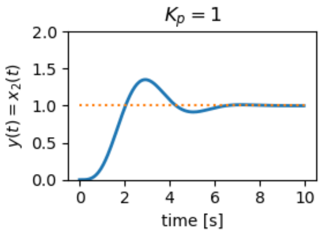
Try proportional plus derivative control.

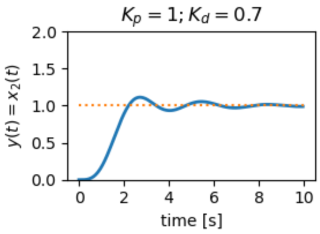
Classical Control

Proportional plus derivative performance is only slightly better.



Step responses:

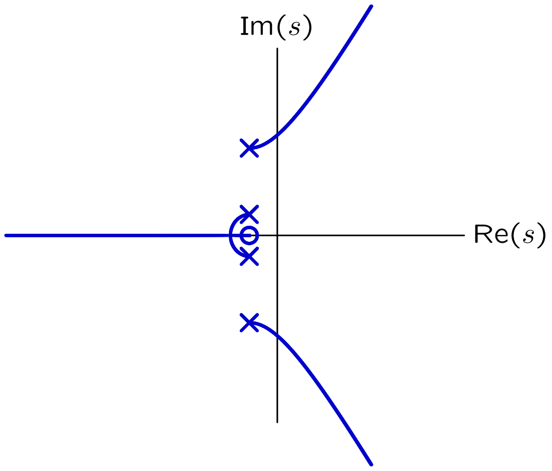




Somewhat smaller overshoot, but still slow convergence.

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.



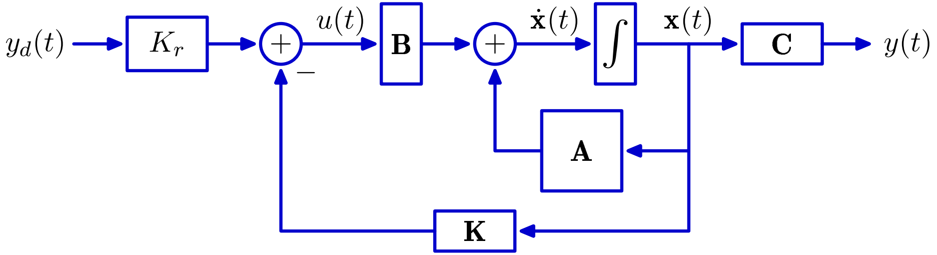
Good explanation of what happened.

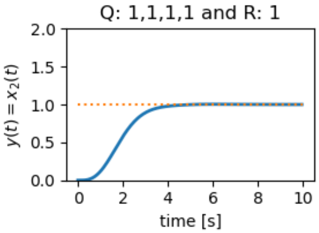
But how do we make it faster?

State-Space Control

Try state-space control.

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

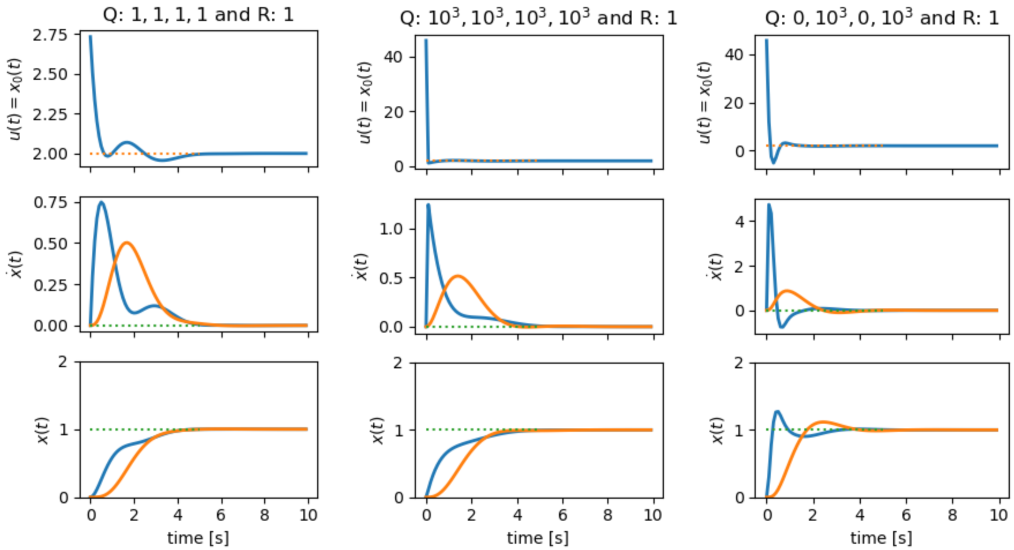




Convergence is slow but **monotonic**. Can we make it faster?

State-Space Control

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

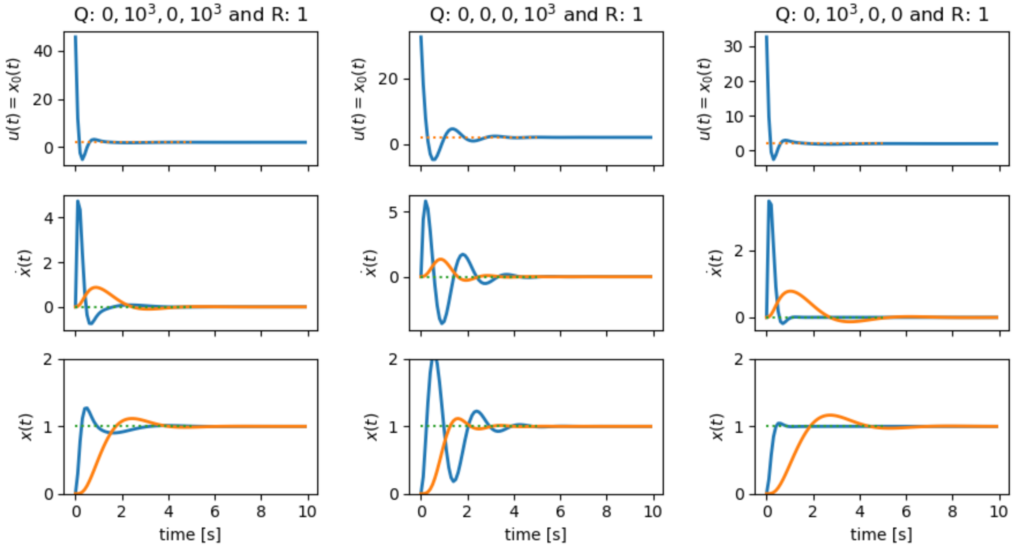


Increasing all values of $\mathbf{Q}$ doesn't have a big effect.

Setting $\mathbf{Q}$ to zero for velocity terms increases speed and overshoot.

State-Space Control

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



Reducing x_1 cost causes large change in x_1 and faster convergence of x_2.

Reducing x_2 cost speeds x_1 with little effect on x_2.

Summary

State-Space controllers offer design advantages over classical controllers.

With full-state feedback, the gains $\mathbf{K}$ can be adjusted to produce ANY set of n closed-loop poles! $\rightarrow$ **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).

The LQR algorithm provides intuitive control of the relative importance of errors in each state and the amount of effort exerted on the plant.