Table of Contents I

notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

Full state feedback control

notes

Contents map

developed content units	taxonomy levels
poles placement	u1, e1

prerequisite content units	taxonomy levels
feedback control	u1, e1
state space LTI systems	u1, e1

notes

Main ILO of sub-module "Full state feedback control"

Formulate a state feedback control law $u = -K\boldsymbol{x}$ to modify the closed-loop dynamics of a linear time-invariant system, given matrices A and B in state-space form

Compute the matrix K to place the poles of the closed-loop system at specified locations, using characteristic polynomial matching

Apply the pole placement algorithm to determine the feedback matrix K for a system with A, B in control canonical form, using time-domain specifications

notes

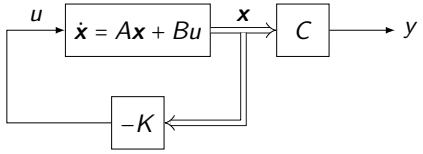
- by the end of this module you shall be able to do this

note: the considerations below are the same for both discrete time and continuous time LTIs

notes

- I want to emphasize this important point: the control design approach we're about to discuss applies equally well to both discrete time and continuous time linear time-invariant systems. This is quite powerful because it means once you understand the fundamental concepts, you can apply them in either domain. Many of you will encounter both types of systems in your careers, so this unified approach will save you a lot of time and effort.

Control-law design for full-state feedback – assumed structure



$$u = -Kx = -\begin{bmatrix} K_1 & \dots & K_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

(estimating x from the measurements = later on)

notes

- Here we're looking at the fundamental structure of state feedback control. The control input u is computed as a linear combination of all state variables, with the coefficients stored in the gain vector K. This is a very powerful control structure because it allows us to influence the system dynamics in a precise way.
- Notice the negative sign in front of K - this is conventional because we typically want negative feedback for stability. Think of it as "pushing back" against the system's natural behavior.
- For now, we're assuming we have direct access to all state variables. In real-world applications, this is often not the case, which is why we'll cover state estimation later in the course. But first we need to understand what to do with the states once we have them.

Finding the control law

$$\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} \end{cases} \qquad \text{“+”} \qquad u = -K\boldsymbol{x}$$

$$\Downarrow$$

$$\begin{aligned} \dot{\boldsymbol{x}} &= (A - BK)\,\boldsymbol{x} \\ y &= C\boldsymbol{x} \end{aligned}$$

Important:

$$BK = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \begin{bmatrix} K_1 & \dots & K_n \end{bmatrix} = \begin{bmatrix} b_1K_1 & \cdots & b_1K_n \\ \vdots & & \vdots \\ b_nK_1 & \cdots & b_nK_n \end{bmatrix}$$

notes

- This is where the magic happens! When we substitute our control law into the original system dynamics, we get a new closed-loop system with the matrix $(A - BK)$ instead of just A. This is why state feedback is so powerful - we're essentially redesigning the system dynamics.
- Pay close attention to the matrix multiplication BK. The dimensions must match correctly: B is an $n \times m$ matrix and K is $m \times n$, so BK is $n \times n$, which can be subtracted from A.
- Notice that the output equation doesn't change. This means that what we observe from outside the system depends on what states we choose to measure (matrix C), and isn't directly affected by our controller.
- The key insight here is that by choosing K appropriately, we can change the behavior of the closed-loop system to achieve our design goals.

Finding the control law – what are the poles now?

$$\begin{aligned} \dot{\boldsymbol{x}} &= (A - BK)\,\boldsymbol{x} \\ y &= C\boldsymbol{x} \end{aligned} \qquad \Rightarrow \qquad \det\Big(sI - (A - BK)\Big) = 0$$

choose K so that the closed-loop poles are where we like

Poles allocation algorithm

- from time-domain specifications, numerically determine the n desired poles $p_1, \dots, p_n$
- numerically compute the desired denominator of the closed loop TF as $\prod_{i=1}^{n} (s - p_i)$
- compute $\det\Big(sI - (A - BK)\Big)$ as a function of $K_1, \dots, K_n$
- find $K_1, \dots, K_n$ by equating the two polynomials

notes

- This is the core idea of pole placement: we can choose the poles of our closed-loop system by selecting the right feedback gains in K. Remember that poles determine the system's dynamic behavior - things like stability, settling time, and oscillation characteristics.
- The poles of our closed-loop system are the roots of the characteristic equation shown here. By changing K, we can change where these poles are located in the complex plane.
- The algorithm I've outlined gives us a systematic way to find the right values of K. We start with the desired performance in the time domain (like settling time or damping ratio), translate these into desired pole locations, and then solve for the K values that will give us those poles.
- Step 3 can be challenging for large systems because computing this determinant and extracting the coefficients as functions of the K_i values gets messy. We'll see how to address this shortly.
- This is an incredibly powerful feature of state feedback - in fact, if the system is fully controllable, we can place the closed-loop poles anywhere we want!

Example

Close the loop around the open loop system $\left\{ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u \right.$ so that the closed loop is stable and with raise time not longer than 5 seconds

$$\left(\text{recall:} \quad G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad \rightarrow \quad \text{rise time } t_r = \frac{1.8}{\omega_n} \right)$$

notes

- Let's work through this example step by step. First, notice that the open-loop system has one unstable pole at $s = 1$ and one stable pole at $s = -0.5$. We need to stabilize the system and ensure a rise time less than 5 seconds.
- We'll use the relationship between rise time and natural frequency: $t_r = 1.8/\omega_n$. Since we want $t_r < 5$, we need $\omega_n > 1.8/5 = 0.36$.
- For a good balance between speed and overshoot, let's choose a damping ratio of $\zeta = 0.7$ (typically a good compromise) and $\omega_n = 0.5$ to meet our rise time requirement.
- This gives us desired poles at $s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -0.35 \pm j0.357$.
- Now we need to find $K = [K_1, K_2]$ such that the characteristic polynomial of $(A - BK)$ has these roots. Let's go through the calculations...
- The closed-loop characteristic polynomial is $\det(sI - (A - BK)) = (s - 1 + K_1)(s + 0.5 + K_2) - (0 \cdot K_2) = s^2 + (K_1 + K_2 - 0.5)s + (K_1K_2 + 0.5K_1 - 0.5)$.
- Our desired polynomial is $(s - (-0.35 + j0.357))(s - (-0.35 - j0.357)) = s^2 + 0.7s + 0.25$.
- Comparing coefficients: $K_1 + K_2 - 0.5 = 0.7$ and $K_1K_2 + 0.5K_1 - 0.5 = 0.25$. Solving these equations gives us our gain vector K.

Finding the control law – drawback when A has no special structure

Example:

$$A = \begin{bmatrix} 5 & 1 & 3 & 8 & 3 \\ 7 & 3 & 9 & 6 & 9 \\ 9 & 4 & 4 & 1 & 7 \\ 2 & 2 & 5 & 3 & 6 \\ 1 & 1 & 0 & 1 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 3 \\ 4 \\ 1 \\ 0 \\ 9 \end{bmatrix}$$

drawback: doing as before is cumbersome

$\downarrow$

is there any alternative way of finding K?

notes

- Look at this 5x5 matrix A - it has no obvious structure, unlike the diagonal matrix from our previous example. Imagine trying to compute the characteristic polynomial of $(A - BK)$ by hand! It would be extremely tedious and error-prone.
- For larger systems like this, the direct approach we used earlier becomes impractical. Computing the determinant of $sI - (A - BK)$ and extracting the coefficients as functions of K_i would be a nightmare, even with computer assistance.
- This is a common situation in control engineering - the mathematics becomes unwieldy as system dimensions increase. Fortunately, there are better approaches.
- In the following slides, we'll see how transforming the system to a canonical form makes this problem much more manageable. And eventually, we'll discover Ackermann's formula, which gives us a direct way to compute K without all the tedious algebra.
- This is a great example of why understanding mathematical structures and system transformations is so important in control theory - they often lead to enormous simplifications in practical problems.

Determinant of a matrix in control canonical form

Let

$$A = \begin{bmatrix} -a_1 & -a_2 & \dots & \dots & -a_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix}$$

then

$$\det(sI - A) = s^n + a_1 s^{n-1} + \dots + a_n$$

notes

- This is an extremely useful property of the control canonical form. The structure of this matrix is special - notice the 1's along the subdiagonal and the coefficients $-a_i$ in the first row.
- What's remarkable is that when we compute the characteristic polynomial of this matrix, the coefficients appear directly in the result. There's a beautiful one-to-one correspondence between the matrix elements and the polynomial coefficients.
- You might wonder why this happens. The control canonical form is designed specifically to have this property. It's closely related to the companion matrix in linear algebra.
- This property will become very helpful for pole placement, as we'll see in the next slides. Instead of messy determinant calculations, we can work directly with the polynomial coefficients.
- I encourage you to verify this property for a 2x2 or 3x3 case to convince yourself that it works. It's not immediately obvious, but once you work through the calculations, you'll see how elegantly it all fits together.

Incidentally. . .

$$Y(s) = \frac{b_1 s^{n-1} + \dots + b_n}{s^n + a_1 s^{n-1} + \dots + a_n} U(s)$$

$$\mapsto \quad A = \begin{bmatrix} -a_1 & -a_2 & \dots & \dots & -a_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix}$$

$$\mapsto \quad \det(sI - A) = s^n + a_1 s^{n-1} + \dots + a_n$$

notes

- Here's another powerful connection - there's a direct mapping between a transfer function representation and the control canonical form. The coefficients in the denominator of the transfer function become the elements of the first row of the A matrix.
- This gives us a systematic way to go back and forth between transfer function representations and state-space models in control canonical form. This is incredibly useful in practice.
- Remember that we often obtain system models as transfer functions from frequency response data or from basic physical principles. This mapping allows us to easily convert such models to state-space form for control design.
- Think of the control canonical form as a "standard format" for representing dynamics. Just like you might convert units to a standard system for calculations, we often convert system representations to canonical forms to simplify analysis and design.
- And as we'll see next, this particular canonical form makes the pole placement problem much more tractable.

Finding the control law with (A, B) in control canonical form

$$B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Rightarrow \quad BK = \begin{bmatrix} K_1 & K_2 & \cdots & K_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

so that

$$A - BK = \begin{bmatrix} -a_1 - K_1 & -a_2 - K_2 & \dots & \dots & -a_n - K_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix}$$

and thus the poles of the closed loop system are the roots of

$$\det\Big(sI - (A - BK)\Big) = s^n + (a_1 + K_1)\, s^{n-1} + \ldots + (a_n + K_n)$$

notes

- Now we're seeing the real power of the control canonical form. When the system is in this form, the control matrix B has a very simple structure - just a 1 in the first position and zeros elsewhere.
- This means that BK only affects the first row of the A matrix when we compute $A - BK$. All the other rows remain unchanged. This is a huge simplification!
- Look at what happens to the characteristic polynomial: the coefficient of s^{n-i} is simply $a_i + K_i$. This is remarkably clean and direct - each gain K_i affects exactly one coefficient in the characteristic polynomial.
- This is what makes pole placement so straightforward in the control canonical form. We can directly "dial in" the coefficients of our desired characteristic polynomial by choosing the appropriate gains.
- For instance, if we want the closed-loop polynomial to be $s^n + \alpha_1 s^{n-1} + \ldots + \alpha_n$, we simply set $K_i = \alpha_i - a_i$ for each i. It's that simple!

Summary (valid also for discrete-time systems!)

(A, B) in control canonical form + K generic

$$\Downarrow$$

closed loop is $\dot{\boldsymbol{x}} = (A - BK)\boldsymbol{x} = \begin{bmatrix} -a_1 - K_1 & -a_2 - K_2 & \dots & \dots & -a_n - K_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}$

$$\Downarrow$$

poles of the closed loop system = roots of

$$\det\Big(sI - (A - BK)\Big) = s^n + (a_1 + K_1)\, s^{n-1} + \ldots + (a_n + K_n)$$

notes

- Let me emphasize an important point here: everything we've discussed applies equally well to discrete-time systems. For a discrete-time system in control canonical form, the pole placement procedure works exactly the same way - the gain K_i directly affects the coefficient of z^{n-i} in the characteristic polynomial.
- This is another example of the elegant unification in control theory - many concepts apply similarly across continuous and discrete domains, especially when working with state-space representations.
- The closed-loop system maintains the special structure of the control canonical form, just with modified coefficients in the first row. This is what makes the analysis so clean.
- Looking at these equations, you can see exactly how each feedback gain affects the closed-loop dynamics. There's a direct, one-to-one mapping between the gains and the coefficients of the characteristic polynomial.
- This transparency is why control engineers often convert systems to canonical forms before design - it makes the relationship between design parameters and system behavior much more explicit.

Summary of the algorithm for (A, B) in control canonical form

- from time domain specifications, find the desired poles $p_1, \ldots, p_n$
- form the desired characteristic polynomial

$$\alpha(s) = \prod_{i=1}^{n} (s - p_i) = s^n + \alpha_1 s^{n-1} + \ldots + \alpha_n$$

- find K s.t. $\det(sI - A + BK) = \alpha(s)$ by solving

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} a_1 + K_1 \\ \vdots \\ a_n + K_n \end{bmatrix}$$

notes

- Here's the complete algorithm for pole placement when working with a system in control canonical form. The beauty of this approach is its simplicity and directness.
- In step 1, we translate our performance requirements (like settling time, damping ratio, or bandwidth) into specific pole locations. This is where your understanding of second-order system dynamics comes into play.
- Step 2 involves multiplying out the factors $(s - p_i)$ to get the coefficients α_i of the desired characteristic polynomial. This is just algebra.
- And finally, in step 3, we simply set $K_i = \alpha_i - a_i$ for each i. This couldn't be more straightforward!
- Compare this to the general case we saw earlier - no messy determinant calculations, no system of equations to solve. Just a direct computation of the gain vector K.
- Of course, the catch is that our system needs to be in control canonical form. If it's not (which is usually the case), we need to transform it first, or use a different approach like Ackermann's formula, which we'll see shortly.

Example

Close the loop around the discrete time open loop system $A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

with sampling period 0.2 seconds so that the closed loop raise time is not longer than 10 seconds

$$\left(\text{recall:} \quad G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad \rightarrow \quad \text{rise time } t_r = \frac{1.8}{\omega_n} \quad \text{but we need to discretize!}\right)$$

notes

- Let's work through this discrete-time example. Notice that the system is already in control canonical form, which is convenient! We can directly apply our algorithm.
- First, we need to translate our continuous-time specifications to the discrete domain. We want a rise time of no more than 10 seconds with a sampling period of 0.2 seconds.
- From the relationship $t_r = 1.8/\omega_n$, we need $\omega_n \geq 1.8/10 = 0.18$ rad/s. Let's choose $\omega_n = 0.2$ for simplicity, and a damping ratio of $\zeta = 0.7$ for good behavior.
- These give us continuous-time poles at $s = -0.14 \pm j0.143$. Now we need to map these to the z-plane using $z = e^{sT}$ with $T = 0.2$ seconds.
- The discrete poles will be $z = e^{(-0.14 \pm j0.143)\cdot 0.2} = e^{-0.028} \cdot e^{\pm j0.0286} \approx 0.972 \angle \pm 0.0286$ rad.
- Our desired characteristic polynomial is therefore $\alpha(z) = (z - 0.972e^{j0.0286})(z - 0.972e^{-j0.0286})(z - 0)$ (I've added a third pole at z=0 for stability).
- Multiplying this out: $\alpha(z) = z^3 - 1.944z^2 + 0.945z$.
- Looking at our A matrix, the open-loop characteristic polynomial is $\det(zI - A) = z^3 - z^2 - 2z - 3$.
- Therefore, we need $K = [K_1, K_2, K_3]$ such that $a_1 + K_1 = 1.944$, $a_2 + K_2 = -0.945$, and $a_3 + K_3 = 0$.
- Since $a_1 = 1$, $a_2 = 2$, and $a_3 = 3$, we get $K = [0.944, -2.945, -3]$.
- Always double-check by computing the eigenvalues of $(A - BK)$ to verify they match our desired poles!

Test this out: write a K that makes the discrete time open loop system

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

with sampling period 0.5 seconds have a raise time is not longer than 15 seconds.

notes

- This is a good exercise to test your understanding. Let me walk you through how I would approach it:
- First, I need to specify desired pole locations based on the rise time requirement. For rise time 15 seconds, we need $\omega_n \geq 1.8/15 = 0.12$ rad/s.
- I'll choose $\omega_n = 0.15$ rad/s and $\zeta = 0.7$ for good damping. The continuous-time poles are $s = -0.105 \pm j0.107$.
- Converting to discrete poles with sampling period $T = 0.5$ seconds: $z = e^{sT} = e^{(-0.105 \pm j0.107)\cdot 0.5} \approx 0.949 \angle \pm 0.0535$ rad.
- The desired characteristic polynomial is $\alpha(z) = (z - 0.949e^{j0.0535})(z - 0.949e^{-j0.0535}) = z^2 - 1.898z + 0.9$
- The open-loop characteristic polynomial is $\det(zI - A) = z^2 - z - 2$.
- So we need $K = [K_1, K_2]$ such that $a_1 + K_1 = 1.898$ and $a_2 + K_2 = -0.9$.
- Since $a_1 = 1$ and $a_2 = -2$, we get $K = [0.898, 1.1]$.
- I encourage you to verify this works by calculating the eigenvalues of $(A - BK)$ and checking if they match our desired poles!
- This example shows how straightforward pole placement becomes when using the control canonical form. You simply identify the desired polynomial coefficients and adjust the gains accordingly.

Fundamental difference with PIDs

$$\det\Big(sI - (A - BK)\Big) = s^n + (a_1 + K_1)\, s^{n-1} + \ldots + (a_n + K_n)$$

state-feedback in fully controllable systems allows allocating all the closed loop poles wherever one wants

notes

- This is a crucial advantage of state feedback over PID control. With PID, you're limited to adjusting just three parameters (K_p, K_i, K_d) regardless of the system order.
- State feedback gives you as many "knobs to turn" as you have states, allowing you to shape the entire closed-loop dynamics precisely.
- Think about it: for an nth-order system, PID gives you 3 parameters to adjust, while state feedback gives you n parameters. This extra flexibility is why state feedback can achieve better performance for complex systems.
- However, with great power comes great responsibility! You need to choose all these gains carefully. Poor pole placement can lead to excessive control effort or sensitivity to noise.
- Also remember that state feedback requires access to all state variables, which often isn't practical. That's why we'll later combine it with state estimators.
  item PID controllers are still incredibly useful for many applications - they're simple, robust, and often "good enough." But for systems where you need precise control over all dynamics, state feedback is the way to go.

Caveats

- weak controllability
- the more you move poles, the more you use actuators (risk of saturations!)
- effect of zeros

notes

- While pole placement is powerful, there are several important practical limitations to keep in mind:
- 1) Weak controllability: If the system is barely controllable (i.e., the controllability matrix is nearly singular), small errors in your model or implementation can make the actual pole locations very different from what you designed.
- 2) Control effort: Aggressive pole placement (fast poles) typically requires large control signals. Your actuators have physical limits - if you demand too much, they'll saturate and your system may become unstable.
- 3) Zeros: Remember that pole placement only affects the poles of the closed-loop system. The zeros remain unchanged and can significantly affect the response, especially for non-minimum phase systems.
- Other practical considerations include:
- - Sensitivity to noise (high gains amplify measurement noise)
- - Robustness to modeling errors
- - Computational limitations in implementation
- Always perform thorough simulations testing different scenarios before implementing your controller in the real world!

Do we actually need to compute the control canonical form?

no, there exists the so-called Ackermann's formula

$$K = [0 \ \dots \ 0 \ 1]\mathcal{C}^{-1}\alpha(A)$$

we will though do not cover it - will be in follow up courses!

notes

- Here's a powerful alternative to transformation to control canonical form: Ackermann's formula gives us the gain matrix K directly.
- The formula might look intimidating at first, but it is actually something that has a precise intuitive physical meaning (if one has been studying systems theory :))
- it won't be asked at the exam

But how do we select the locations of the poles?

Strategies:

- *in this course,* dominant second-order poles approximations
- *in follow up courses,* very many other ones!

notes

- this is not a trivial topic: one shall do courses just to learn this!
- for the exam we will go through just dominant poles approximations

Summarizing

Formulate a state feedback control law $u = -K\boldsymbol{x}$ to modify the closed-loop dynamics of a linear time-invariant system, given matrices A and B in state-space form

Compute the matrix K to place the poles of the closed-loop system at specified locations, using characteristic polynomial matching

Apply the pole placement algorithm to determine the feedback matrix K for a system with A, B in control canonical form, using time-domain specifications

notes

- we saw a bit of formulas and concepts about how to do this

Most important python code for this sub-module

notes

control (Python Control Systems Library)

main functions:

- `acker` (Ackermann's method)
- `place` (robust pole placement)

notes

- read the documentation :)

Self-assessment material

notes

Question 1

What is the primary advantage of state feedback control with pole placement compared to PID control?

Potential answers:

I: **(wrong)** PID control is always more stable than state feedback.

II: **(correct)** State feedback allows arbitrary placement of all closed-loop poles when the system is fully controllable.

III: **(wrong)** State feedback does not require knowledge of the system's state variables.

IV: **(wrong)** PID control can achieve faster response times than state feedback.

V: **(wrong)** I do not know

Solution 1:

The correct answer is that state feedback allows arbitrary placement of all closed-loop poles when the system is fully controllable. This is a fundamental advantage

notes

- see the associated solution(s), if compiled with that ones :)

Question 2

Why is the control canonical form particularly useful for pole placement problems?

Potential answers:

I: **(wrong)** It makes the system matrix A diagonal.
II: **(wrong)** It eliminates all zeros from the transfer function.
III: **(correct)** The coefficients of the characteristic polynomial appear directly in the first row of A.
IV: **(wrong)** It guarantees that the system will be observable.
V: **(wrong)** I do not know

Solution 1:

The correct answer is that in control canonical form, the coefficients of the characteristic polynomial appear directly in the first row of A. This makes pole placement straightforward since modifying these coefficients through state feedback directly affects the closed-loop poles. The other options are incorrect: control canonical form doesn't diagonalize A, doesn't affect zeros, and observability isn't guaranteed by this form.

notes

- see the associated solution(s), if compiled with that ones :)

Question 3

What is a major practical limitation of aggressive pole placement through state feedback?

Potential answers:

I: **(wrong)** It makes the system uncontrollable.
II: **(correct)** It may require large control inputs that could lead to actuator saturation.
III: **(wrong)** It always makes the system unstable.
IV: **(wrong)** It prevents the use of output feedback.
V: **(wrong)** I do not know

Solution 1:

The correct answer is that aggressive pole placement may require large control inputs that could lead to actuator saturation. While pole placement theoretically allows arbitrary pole locations, practical limitations include actuator limits. The other options are incorrect: pole placement doesn't make the system uncontrol-

notes

- see the associated solution(s), if compiled with that ones :)

Question 4

When designing state feedback control, why might we choose poles with dominant second-order characteristics?

Potential answers:

I: **(wrong)** Because higher-order systems cannot be controlled effectively.
II: **(wrong)** Because it eliminates all zeros from the system.
III: **(correct)** Because it allows us to approximate the response using familiar second-order performance measures.
IV: **(wrong)** Because it guarantees minimum-phase behavior.
V: **(wrong)** I do not know

Solution 1:

The correct answer is that choosing poles with dominant second-order characteristics allows us to approximate the response using familiar second-order performance measures (like rise time, overshoot, etc.). The other options are incorrect: higher-order systems can be controlled, zeros aren't affected by pole placement, and minimum-phase behavior isn't guaranteed by this approach.

notes

- see the associated solution(s), if compiled with that ones :)

Recap of sub-module "Full state feedback control"

- full state feedback enables placing the poles wherever one wants
- with respect to PID it has more flexibility
- this comes with the cost of having a sufficiently accurate model (and that the model can be written in control canonical form, something that is not always guaranteed!)

notes

- the most important remarks from this sub-module are these ones