POLYTECH NANCY

UNIVERSITÉ DE LORRAINE

CONTROL OF MOBILE ROBOTS

Linear Control Refresher

From a Physical Model to PID and State-Feedback Design

Refresher session before two guided labs on the Quanser QUBE-Servo 2
(PD position control, then pole-placement & LQR balance state feedback control)

Hugues GARNIER

hugues.garnier@univ-lorraine.fr

September 2026

COURSE OVERVIEW

Where Today Fits

34 hours total, including a 24-hour paired mini-project. Today's 2 hours are the bridge between last semester's automatic control course and the two guided labs.

TODAY'S GOAL

- Revisit the methodology used in both labs: physical model → linear model → controller synthesis → validation
- Apply it once with PID (Lab 1) and once with state feedback (Lab 2) - same platform, same workflow, two toolkits for model-based control
- Stay high level: the labs are already carefully guided step by step

A NOTE ON WHERE YOU LEFT OFF

Nothing here is new !

We are reactivating vocabulary and reflexes (overshoot, settling time, pole placement, PID, LQR…) you will need from the first minutes of the Labs.

Classic control vs. modern control

“Classic” Control

- Uses transfer functions to model systems
- Typically for Single-Input Single-Output (SISO) systems
- Uses control techniques like PID

Transfer function of motor speed

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1}$$

“Modern” Control

- Uses state-space variable to model system
- Good for Multiple-Input Multiple Output (MIMO) systems
- Uses control techniques like state-feedback

State-space equation of motor speed

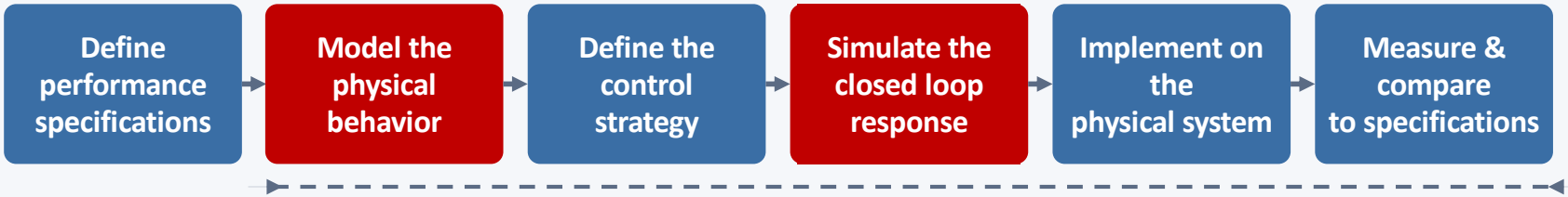
$$\dot{x} = -\frac{k_m^2}{J_{eq} R_m} x + \frac{k_m}{J_{eq} R_m} u$$

$$y = x$$

METHODOLOGY

The Control Design Methodology

The same workflow applies for both lab handouts. Come back to it whenever a design does not behave as expected.



TWO STEPS DO MOST OF THE WORK

- Modelling the physical behavior. Without a trustworthy model, nothing downstream can be trusted
- Simulating before implementing. Cheap iteration in Simulink before touching real hardware

TODAY, WE TRAVEL THIS WORKFLOW TWICE

Once for PID control (Lab 1) and once for state feedback (Lab 2) - same physical platform, same methodology, two different “Define control strategy” boxes.

Why Start From a Physical Model?

GREY-BOX

Model-based controller design

- Physical parameters carry meaning (inertia, damping, gain…)
- Controller gains are computed to meet some specifications, not guessed
- Same model unlocks state feedback, not just PID
- Predicts behaviour beyond the tested conditions

PURE TRIAL & ERROR

Empirical controller tuning (*e.g.* trial & error)

- Useful when no model is available at all (remember the line following for the 3pi+ robot)
- Gains tuned directly on the real system, by trial
- Little insight into why a gain value works
- Does not extend naturally to state feedback

In both labs, you have or can get a model, so we use it.
It is what makes state feedback possible in Lab 2

METHODOLOGY

Two Ways to Obtain a Physical Model

Both routes appear in your labs. Pick whichever the situation makes available.

ROUTE A

Equations of motion + datasheet

- Newton / Euler–Lagrange equations from the physics
- Numerical parameters from the manufacturer (mass, length, inertia, damping…)
  - Used in Lab 2 for the pendulum-up linearized state-space model

ROUTE B

Data-driven model identification

- Excite the real system (e.g. a voltage step) and record the response
- Fit a low-order transfer function to the data (e.g. with the CONTSID toolbox)
  - Used in Lab 1 for the DC-motor velocity loop model

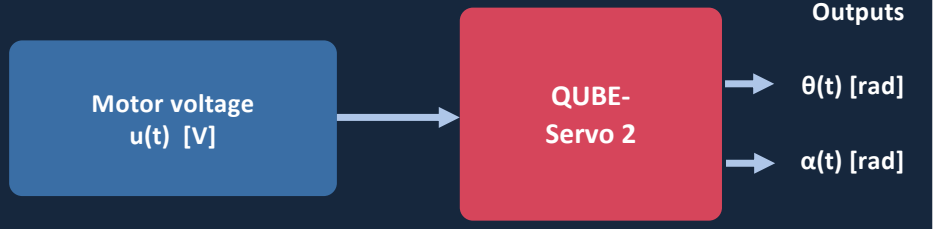
THE HARDWARE YOU WILL USE

The QUBE-Servo 2 Platform

ONE ACTUATOR, INTERCHANGEABLE MODULES

- Direct-drive DC motor, driven by a voltage u(t) [V]
- Inertia disc module - Lab 1 (P/PD position)
- Rotary pendulum module - Lab 2 (state-feedback balance control)
- Encoders measure angular position (rad)
- Tachometers or high-pass filters give angular velocity (rad/s)

INPUT / OUTPUT PICTURE

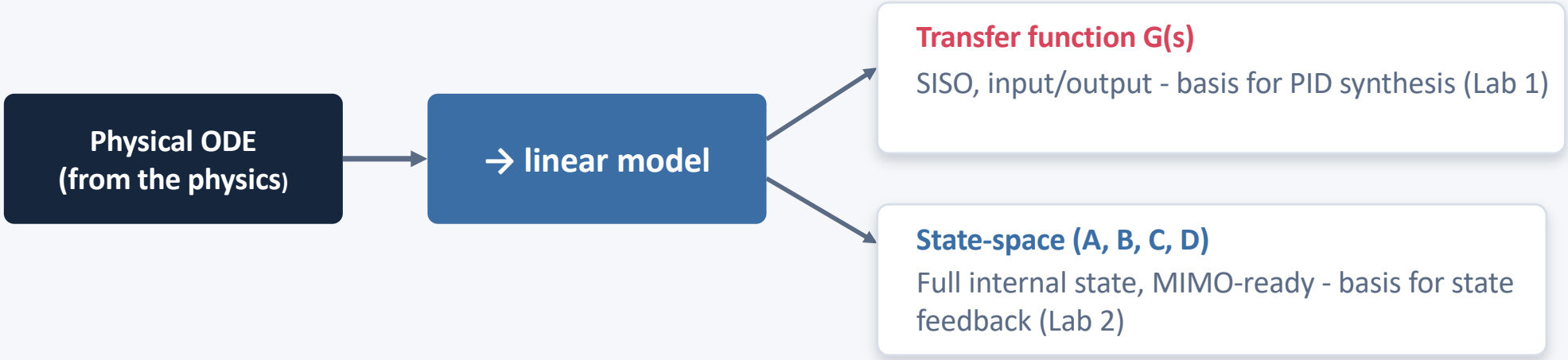


Same causal picture as your Lab 1 wiring: HIL Write Analog → plant → HIL Read Encoders.

Sensor calibration: encoder counts → rad, at 2048 counts / revolution.

METHODOLOGY

From an Equation of Motion to a Usable Model



SAME PHYSICAL SYSTEM, TWO REPRESENTATIONS

A single differential equation can always be written either way. Which one you pick depends on the controller you intend to design - not on the physics, which does not change.

Position loop, Lab 1: $\Theta(s)/U(s) = K / [s(1+Ts)]$ $\leftrightarrow$ $\dot{x} = Ax + Bu,\quad y = Cx + Du$

Pendulum, Lab 2: the same idea, linearized around the vertical equilibrium (next slide).

METHODOLOGY

Linearization Around an Operating Point

TAYLOR EXPANSION, FIRST ORDER

$$f(x) \approx f(x_0) + \partial f/\partial x \,|_{x_0} \cdot (x - x_0)$$

Valid for small deviations $\delta x = x - x_0$ around the chosen equilibrium x_0.

WHY IT MATTERS HERE

- Most control design tools (pole placement, LQR, classical PID tuning) are built for linear models
- A well-chosen operating point keeps the approximation valid over the whole manoeuvre

Lab 1 - DC MOTOR

Voltage-to-velocity behaviour is close to linear already: no explicit linearization step needed before identification.

Lab 2 - PENDULUM

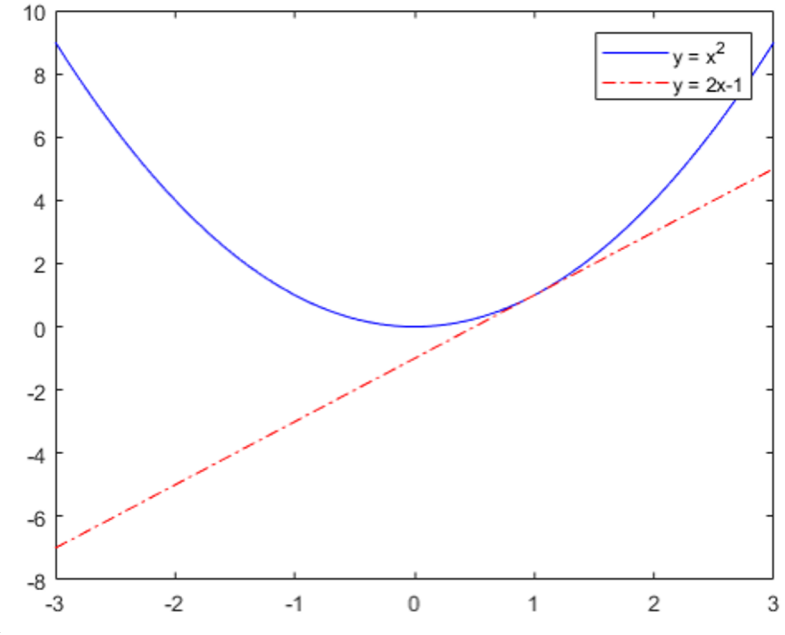
Nonlinear dynamics ($\sin \alpha$, coupling) are linearized around the vertical equilibrium $\alpha = 0$ (pendulum pointing straight up) - exactly the position you will balance it at.

Example: Function with square term

Linearize the nonlinear function $y = x^2$ about $x_0 = 1$

$$f_{lin}(x) = f(x_0) + \left.\frac{\partial f(x)}{\partial x}\right]_{x=x_0} (x - x_0)$$

$$f_{lin}(x) = 1 + 2(x - 1) = 2x - 1$$



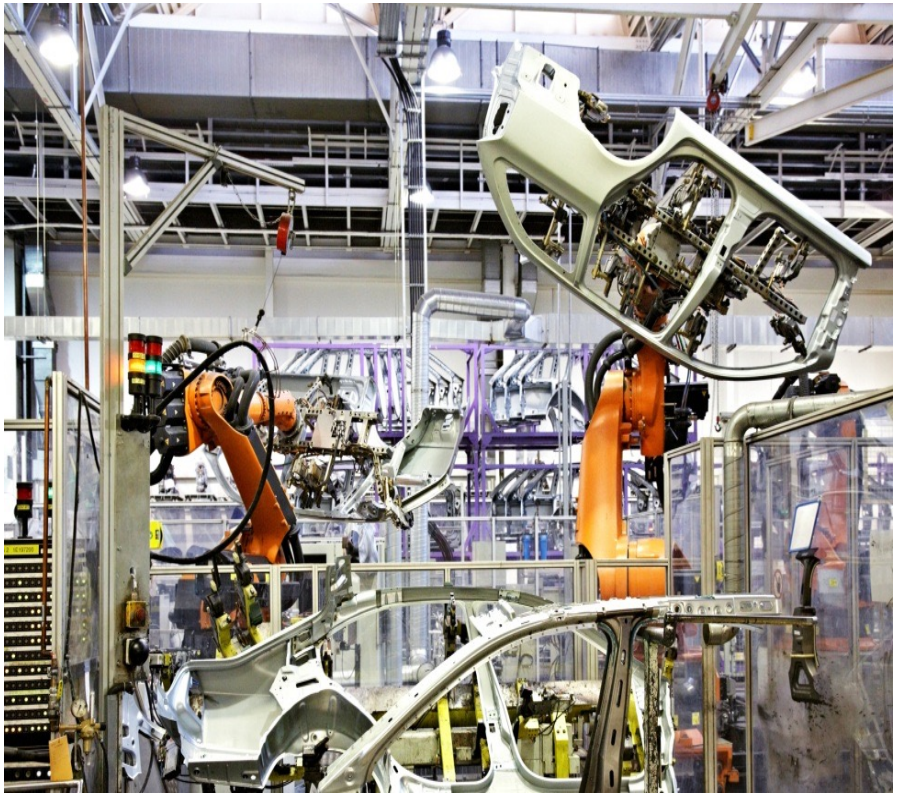
PART 1 OF 2

PID Control based on an Identified Physical Model

Lab 1 — the QUBE-Servo 2 with the inertia disc

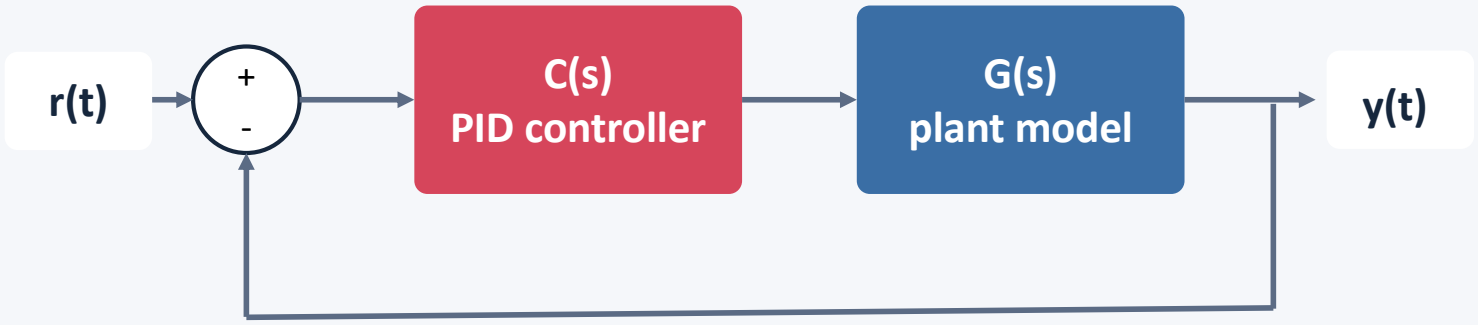
Why PID?

- **Solves 90% of the control problems**
- Easy to design and implement
- Industry standard



PID often used for controlling joint of robot arms

Closed-Loop Control and the Role of Each PID Term



P **Proportional**

Sets the raw reaction speed to the error.
Too high → overshoot / instability.

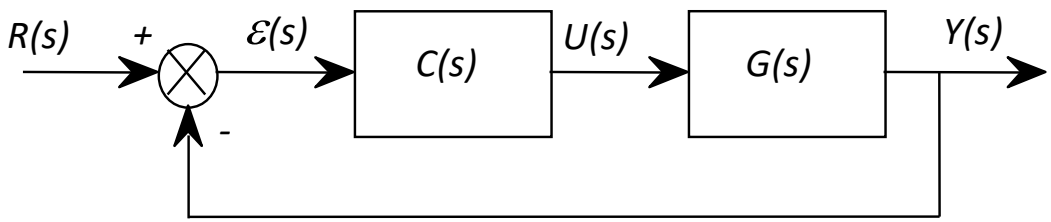
I **Integral**

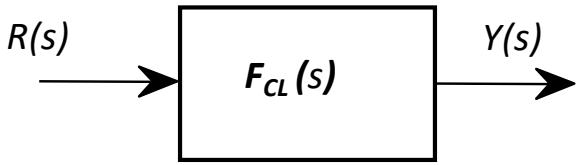
Removes steady-state error (a constant disturbance, dry friction…).
Adds a pole at the origin.

D **Derivative**

Anticipates the trend, adds damping, tames overshoot.
In Lab 1 it acts on the output, not the error.

Closed-loop transfer function





We want $F_{CL}(s)$ to behave like a reference model $\boldsymbol{F_{ref}(s)}$

which takes, usually, the form of a standard second-order transfer function model

$$F_{ref}(s) = \frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2}$$

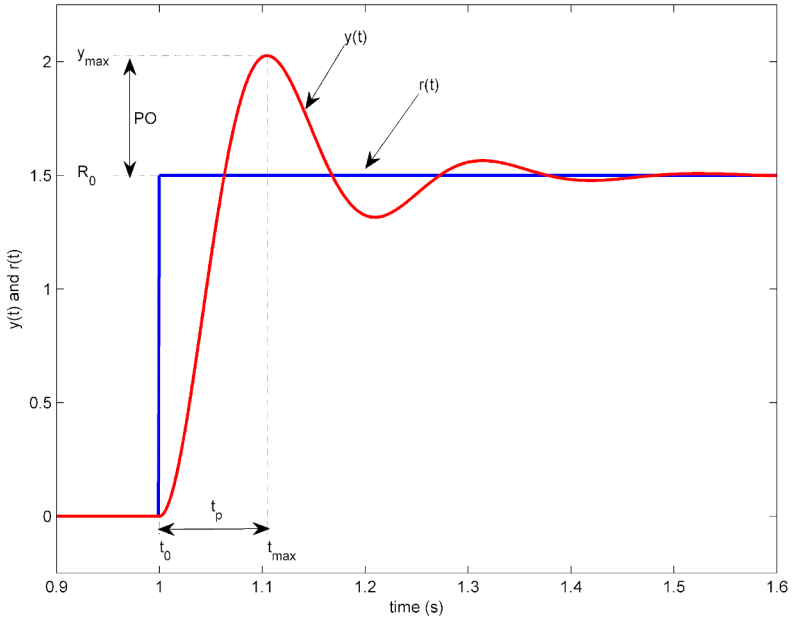
Standard 2nd order model

- Recall the standard second-order transfer function:

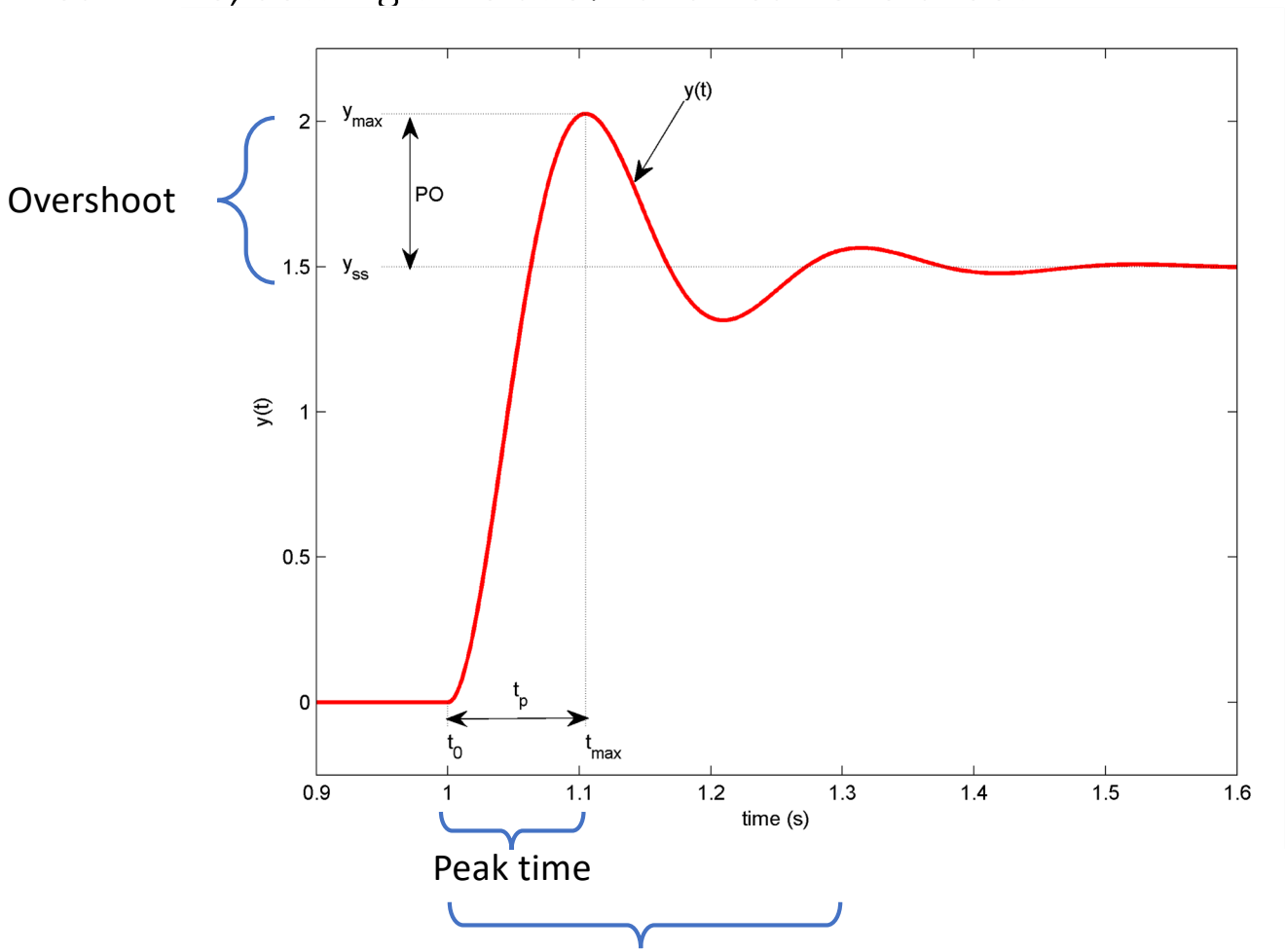
$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2}$$

- ω_n is natural frequency
- ζ is the damping ratio

Step response of 2nd order system



Peak Time, Settling time at 5% and Peak Overshoot



Settling time at 5% = The time required for the system response to enter and remain within ±5% of its final steady-state value.

PID — LAB 1

Step 1: Turn the Brief Into Numbers

Example — angular position control requirements, exactly as stated in your Lab 1 handout.

Requirement	Assessment criterion	Level
Control the position	Position setpoint tracking	**No steady-state error**
Actuator limit	Motor input voltage	**$-10\ V \leq u \leq +10\ V$**
Transient response	Peak overshoot D_1	**4.3 %**
Transient response	Settling time at 5 %	**$T_{5\%} = 0.05\ s$**
Robustness	Rejection of load effects	—

TRANSLATE, DON'T GUESS

Overshoot % and settling time are not gains. They must first be converted into (ζ, ω_n) targets for the closed-loop transfer function. That conversion is the bridge from specs to controller design (see next slides).

PID Control based on a Physical Model

PID — LAB 1

From a Spec to PID Gains: Model-Based Tuning

STANDARD SECOND-ORDER CLOSED LOOP

$$T(s) = \omega_n^2 / (s^2 + 2\zeta\omega_n s + \omega_n^2)$$

Overshoot and settling time depend only on (ζ, ω_n):

- D_1 (%) ⟷ damping ratio ζ
- $T_5\% \approx 3 / (\zeta\omega_n)$ ⟷ natural frequency ω_n

Matching the closed-loop transfer function's coefficients to the desired (ζ, ω_n) gives Kp (and Kd, Ki…) directly in terms of the physical model parameters (K, T).

Specifying the damping ratio ζ and the natural frequency ω_n implicitly defines the desired closed-loop pole locations.
The approach can therefore be viewed as a **pole-placement method** driven by transient-response specifications.

WHY BOTHER, IF ZIEGLER–NICHOLS EXISTS?

- Empirical rules (Ziegler–Nichols, etc.) are useful when no model is available
- Here you do have a model - so you can solve directly for the controller gains that hit the exact spec, not an approximate one
- This is precisely what you do analytically in Lab 1 before touching Simulink

Modelling Step 1: the motor-voltage-to-velocity transfer function

FIRST-ORDER MODEL

Ω(s) / U(s) = K / (1 + Ts)

Ω(t): motor angular velocity [rad/s] — U(t): motor voltage [V]

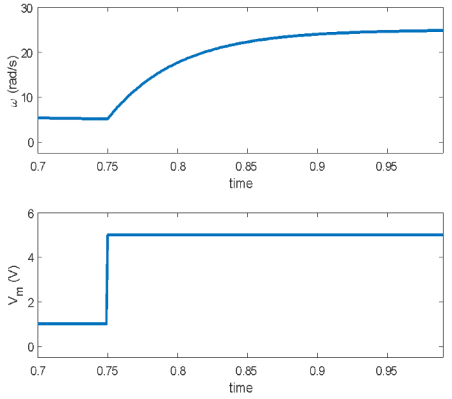
- K [rad/(V·s)]: steady-state gain
- T [s]: time constant

Why velocity, not position? Position adds a pure integrator

Θ(s)/U(s) = K / [s(1+Ts)] harder to identify directly from a step test.

IDENTIFICATION FROM A STEP TEST

Apply a voltage step to the motor and record the velocity response.



IN LAB 1 YOU WILL USE CONTSID

A continuous-time identification toolbox (developed at CRAN, Polytech Nancy) that fits K and T directly from the recorded input/output data - more accurate than reading them off the curve by hand.

Modelling Step 2: Angular Position Closed Loop + PD Design

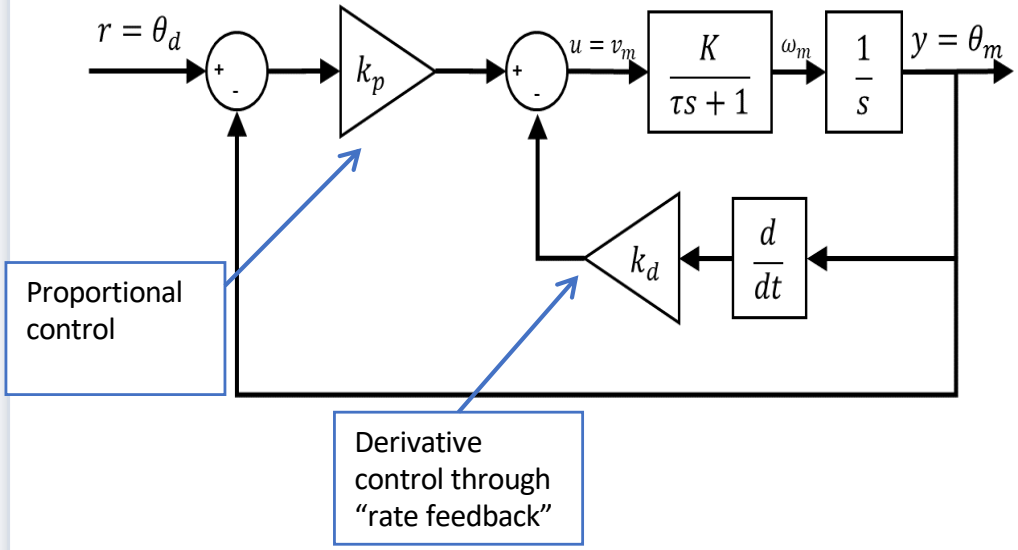
POSITION MODEL

Θ(s) / U(s) = K / [s(1+Ts)]

The same K, T as the velocity model - position adds one pure integration.

PD LAW USED IN LAB 1

Derivative action on the OUTPUT (not the error) — avoids a derivative kick on the setpoint step:



PD CONTROLER GAIN CALCULATION

- Convert D_1 = 4.3 %, T_5% = 0.05 s into (ζ, ω_n)
- Solve the closed-loop characteristic equation for Kp and Kd in terms of K, T, ζ, ω_n
- Check the resulting u(t) stays within ± 10 V

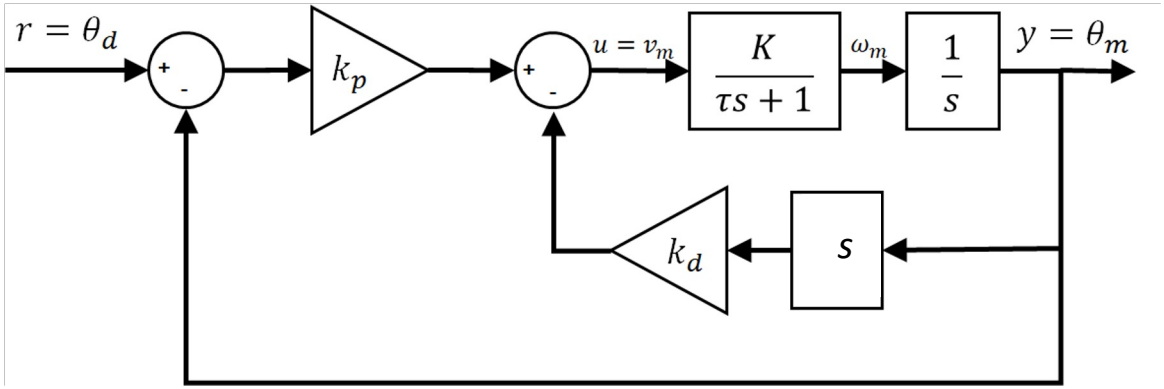
THEN, IN LAB 1

- Simulate the step response in Simulink, refine Kp / Kd against the spec
- Implement on the real QUBE-Servo 2, compare to simulation
- Test robustness: add a load disturbance, then swap in the pendulum module

Closed-Loop Transfer Function

- Plant model

$$\frac{Y(s)}{U(s)} = \frac{K}{s(Ts+1)}$$



- Control input (i.e. voltage)

$$V_m(s) = k_p(\theta_d - \theta_m) - k_d s\theta_m$$

- Find closed-loop transfer function

$$\frac{Y(s)}{R(s)} = \frac{Kk_p/\tau}{s^2 + \frac{(1+Kk_d)s}{\tau} + \frac{Kk_p}{\tau}}$$

We want $F_{CL}(s)$ to behave like a reference model $F_{ref}(s)$

which takes, usually, the form of a standard second-order transfer function model

Closed-loop transfer function

$$F_{CL}(s)=\frac{Y(s)}{R(s)}=\frac{Kk_p/\tau}{s^2+\frac{(1+Kk_d)s}{\tau}+\frac{Kk_p}{\tau}}$$

Standard second-order transfer function

$$F_{ref}(s)=\frac{Y(s)}{R(s)}=\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$$

$$\omega_n^2=\frac{Kk_p}{\tau}$$

$$2\zeta\omega_n=\frac{1+Kk_d}{\tau}$$

$$k_p=\frac{\omega_n^2\tau}{K}$$

$$k_d=\frac{2\zeta\omega_n\tau-1}{K}$$

PID Control Design based on an Identified Physical Model (K, τ here)

PD controller gain determination for a specified (ω_n, ζ)

1. Based on required settling time and overshoot, get ω_n and ζ

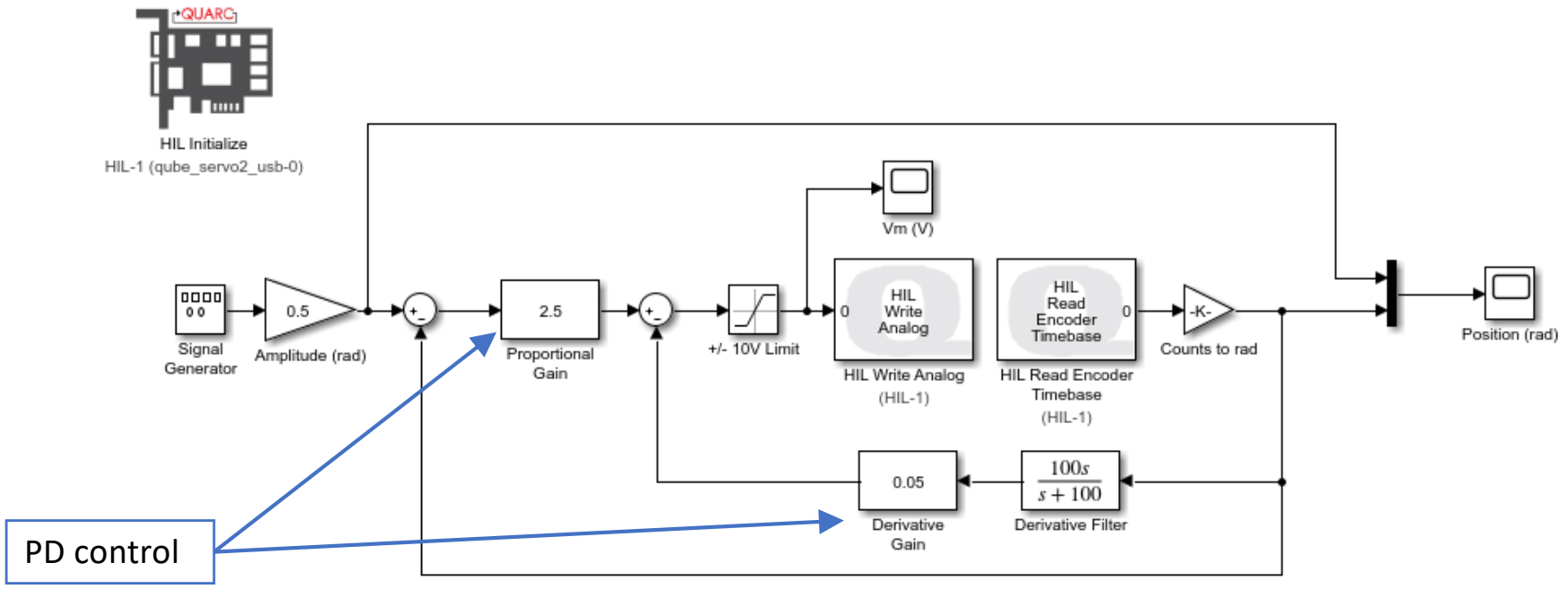
2. Given ω_n and ζ, what PD gains would I need?

$$k_p = \frac{\omega_n^2 \tau}{K}$$

$$k_d = \frac{2\zeta\omega_n\tau - 1}{K}$$

In-Lab Experiments

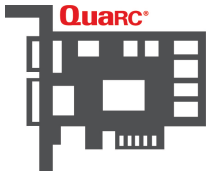
PD control implementation

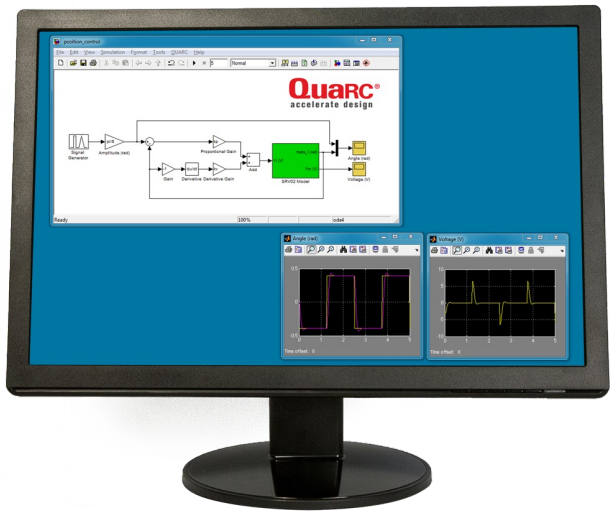


QUARC

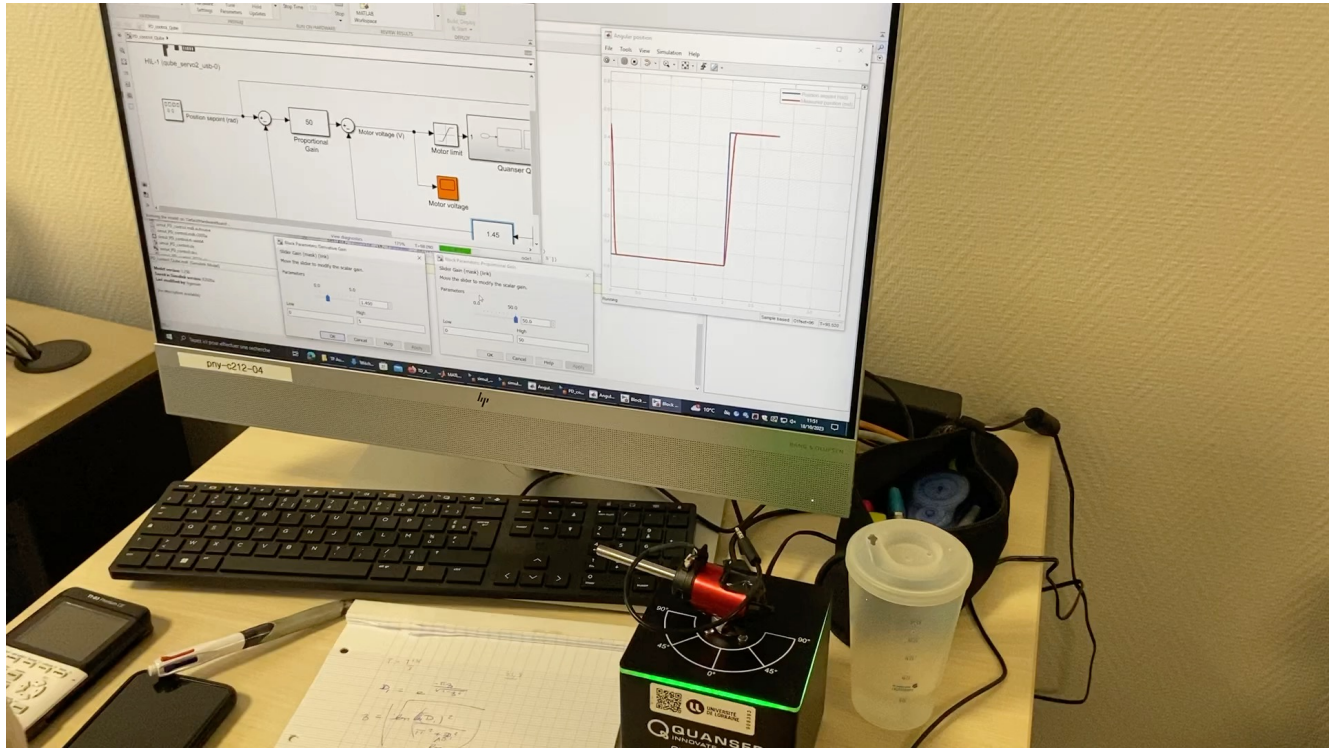
- QUARC is a Rapid Control Prototyping Software made by Quanser for Matlab/Simulink

- Generates real-time code from a Simulink diagram





PD control of motor angular position



PART 2 OF 2

State Feedback via the Linearized Physical Model

Lab 2 - the QUBE-Servo 2 with the rotary pendulum: balancing it upright

14

STATE FEEDBACK — LAB 2

Why Go Beyond PID Here?

PID performs well on many problems but the balancing task in Lab 2 breaks two of its usual assumptions.

1 System unstable in Open-loop

The pendulum balanced upright has an open-loop pole in the right half-plane.
A PID tuned near equilibrium is not guaranteed to stabilize it.

2 Two coupled outputs, one input

Both the arm angle θ and the pendulum angle α must be controlled from a single motor voltage.
Classic SISO PID has no natural way to use both signals together.

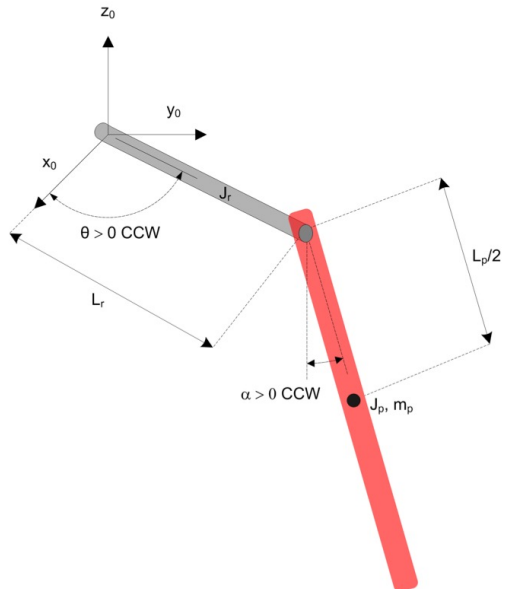
STATE FEEDBACK USES ALL AVAILABLE INFORMATION

Instead of feeding back a single output, state feedback feeds back the entire state vector x(t) - here, both angles and both angular velocities, which is exactly what makes it possible to stabilize an unstable, multi-output system.

STATE FEEDBACK - LAB 2

State-Space Model of the Rotary Pendulum

SCHEMATIC



α = 0 is the vertical, upright position - our chosen equilibrium.

STATE VECTOR

$$x(t) = [\ \theta(t) \quad \alpha(t) \quad \dot{\theta}(t) \quad \dot{\alpha}(t)\ ]^T$$

Two angular positions (from the encoders) and two angular velocities (estimated from high-pass filters).

GENERIC LINEAR STATE-SPACE FORM

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$
$$y(t) = C\,x(t) + D\,u(t)$$

u(t) = motor voltage;

y(t) = [θ, α] measured by the two encoders.

STATE FEEDBACK — LAB 2

Where does the state-space model actually come from

Route A from Slide 5: Euler–Lagrange equations + Quanser physical parameters - linearized around the pendulum-up equilibrium.

If the state vector x of the rotary pendulum system is chosen as

$$x(t) = \left[\theta(t); \alpha(t); \dot{\theta}(t); \dot{\alpha}(t)\right]$$

the linearized state-space model of the rotary pendulum-down system is:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

with

$$A = \frac{1}{J_T}\begin{bmatrix} 0 & 0 & J_T & 0 \\ 0 & 0 & 0 & J_T \\ 0 & \frac{1}{4}m_p^2 L_p^2 L_r g & -\left(J_p + \frac{1}{4}m_p L_p^2\right) D_r & \frac{1}{2}m_p L_p L_r D_p \\ 0 & -\frac{1}{2}m_p L_p g\left(J_r + m_p L_r^2\right) & \frac{1}{2}m_p L_p L_r D_r & -\left(J_r + m_p L_r^2\right) D_p \end{bmatrix}$$

$$B = \frac{1}{J_T}\begin{bmatrix} 0 \\ 0 \\ J_p + \frac{1}{4}m_p L_p^2 \\ -\frac{1}{2}m_p L_p L_r \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

STATE FEEDBACK — LAB 2

Where The Numbers Come From

PENDULUM-UP LINEARIZED MODEL

$x = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^T$ - numerical values from Lab 2:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 149.3 & -14.93 & 4.92 \\ 0 & 261.6 & -14.76 & -8.61 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 49.73 \\ -49.15 \end{bmatrix}$$

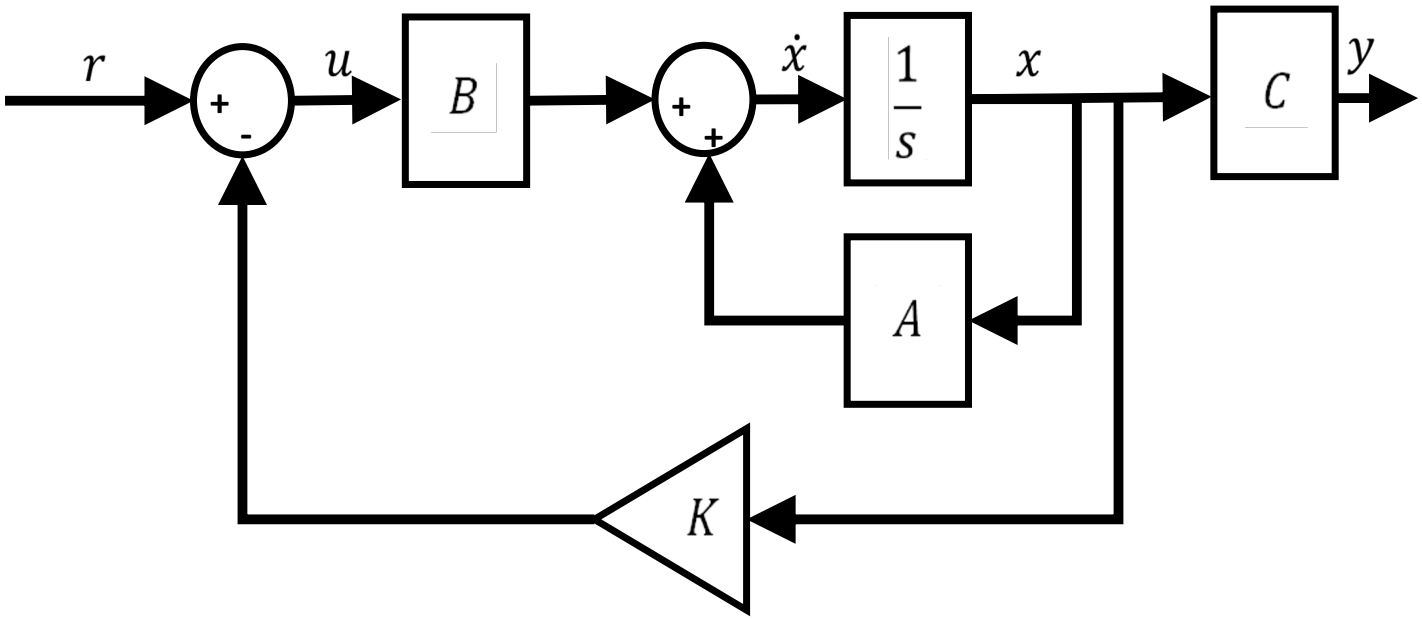
OPEN-LOOP POLES

{ 0, −28.64, +10.67, −5.57 }

One pole at +10.67 → the linearized system is unstable in open loop: exactly the situation state feedback is built for.

The numbers come from physical parameters (arm/pendulum length, mass, inertia, viscous damping) taken from the Quanser datasheet.

State-Feedback Control



State-feedback control shown above can only be used if a system is controllable.

STATE FEEDBACK — LAB 2

Controllability: Can We Actually Place the Poles?

DEFINITION

The pair (A, B) is controllable if any initial state can be steered to any desired state in finite time using u(t).

TEST

$$C = [\, B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B \,]$$

(A, B) is controllable $\Leftrightarrow$ rank(C) = n (here n = 4)

```
In Matlab:  Co = ctrb(A,B);  rank(Co)
```

WHY IT MATTERS BEFORE DESIGNING K

- An unstable system can only be stabilized by state feedback if it is controllable
- Confirming rank(C) = 4 for the QUBE-Servo 2 pendulum is the first check in Lab 2, before any pole-placement or LQR computation
- Practical caution: avoid placing poles at unrealistic locations weakly controllable modes need very high gain and become sensitive to noise

STATE FEEDBACK — LAB 2

Pole Placement Design

CONTROL LAW

u(t) = −K x(t) + r(t)

Closed loop: $\dot{x} = (A - BK)x + Br$

K is chosen so A − BK has the desired eigenvalues.

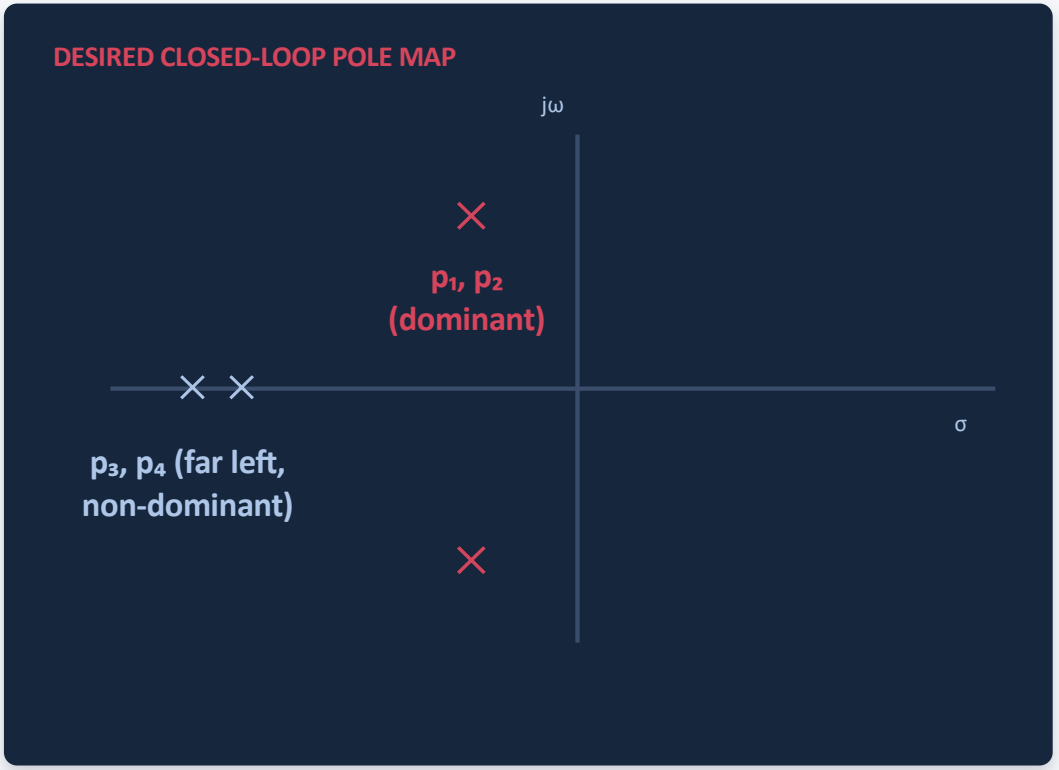
DOMINANT-POLE SPECIFICATION

Overshoot 6.8 %, $T_{5\%}$ = 1.5 s → $\zeta = 0.65$, $\omega_n = 4$ rad/s (dominant pair)

Remaining poles placed ≥ 4× further left (e.g. −40, −45) so they don't affect the transient shape.

```
Matlab:  K = place(A,B,P)
```

DESIRED CLOSED-LOOP POLE MAP



STATE FEEDBACK — LAB 2

An Alternative: The Optimal LQR

COST FUNCTION

$$J = \int (x_r - x)^T Q (x_r - x) + u^T R u \; dt$$

(integral from 0 to ∞; x_r = reference state)

- Q penalizes deviation of each state from its setpoint
- R penalizes control effort (voltage usage)
- Larger Q_{ii} → that state is corrected more aggressively
- Larger R → gentler, lower-effort control

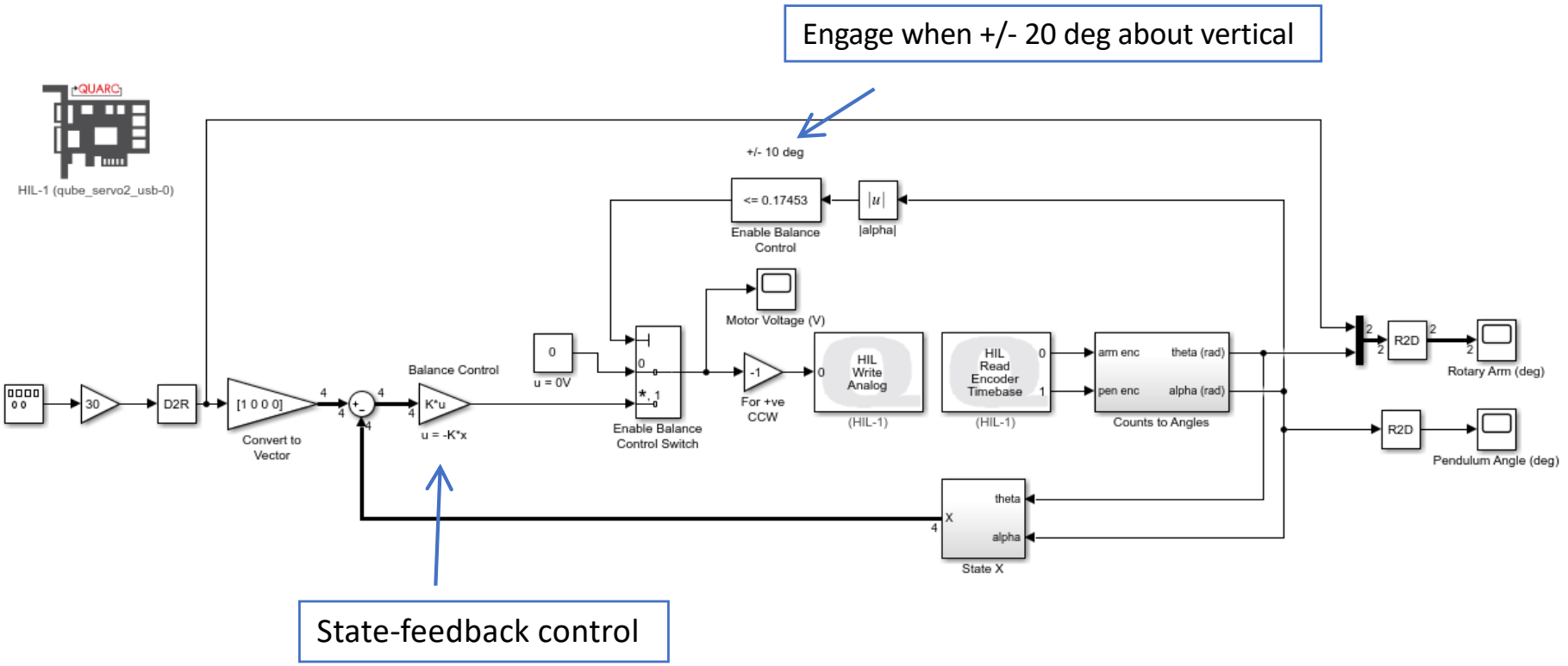
```
Matlab:  K = lqr(A,B,Q,R)
```

SAME GOAL, DIFFERENT ROUTE TO K

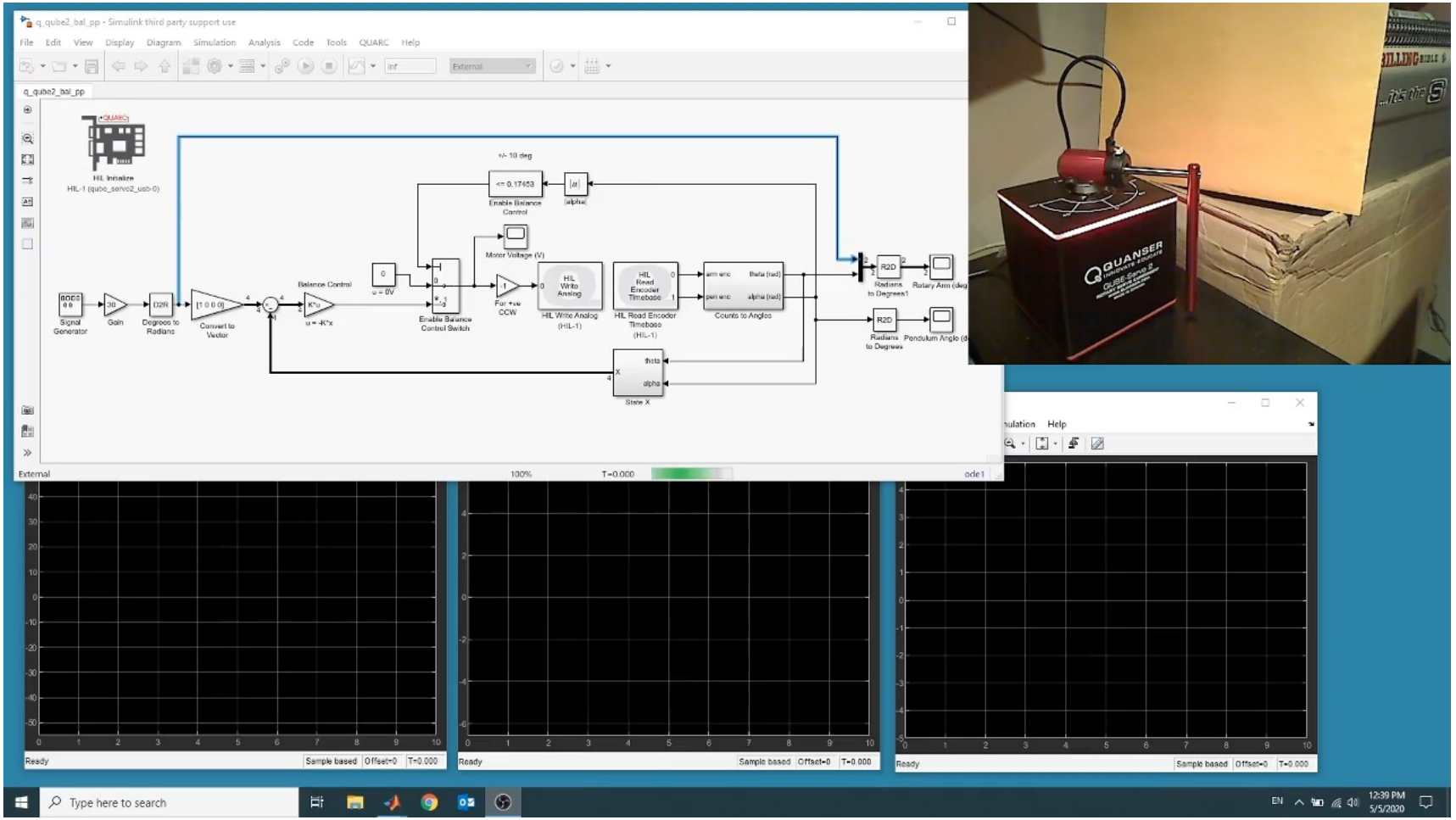
- Pole placement: you choose the closed-loop poles directly
- LQR: you choose a cost trade-off (Q, R), and the optimal K and its resulting poles falls out automatically
- In Lab 2 you will try both on the same linearized model and compare the balancing behaviour

In-Lab Experiments

LQR control implementation



Optimal LQR-based balance control

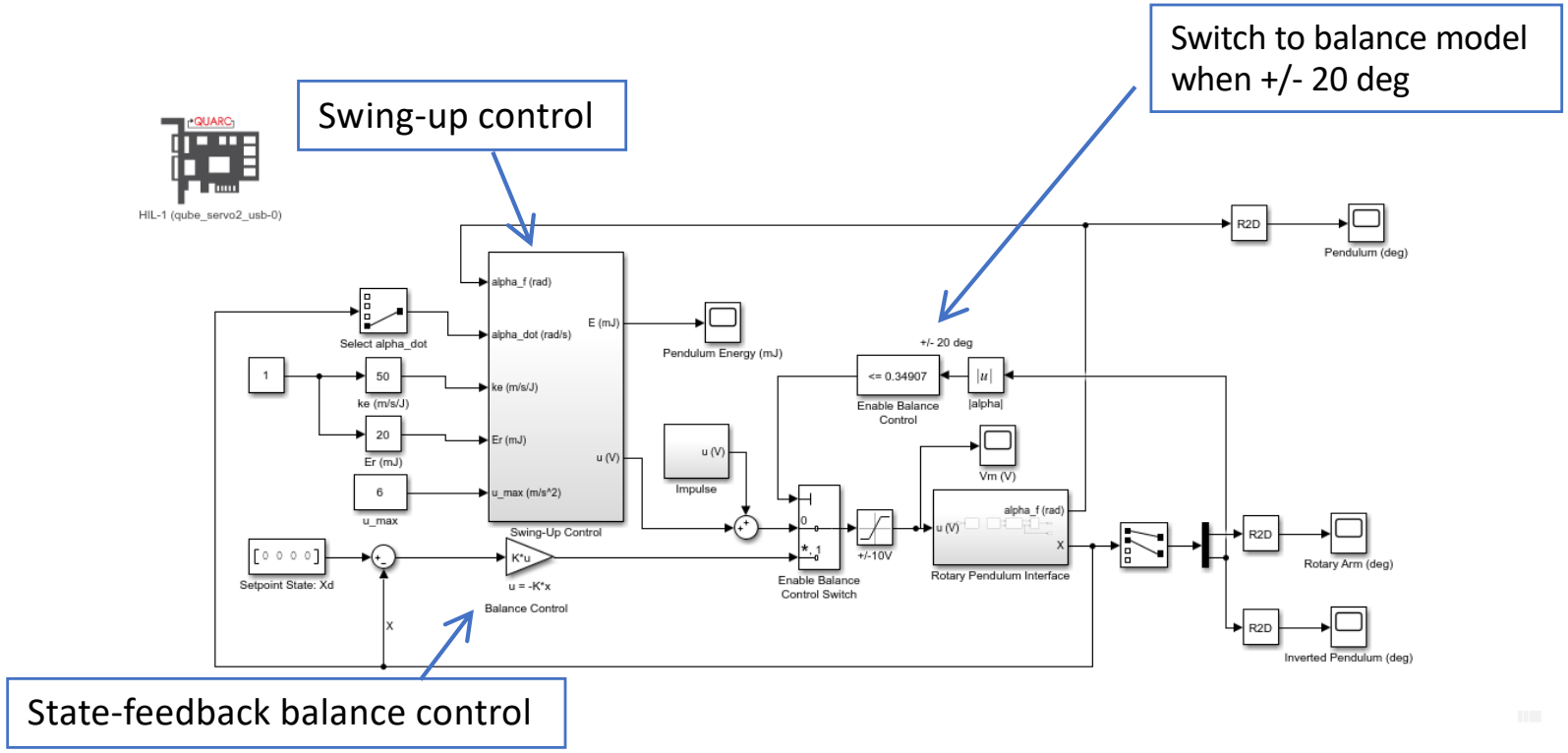


Pendulum swing-up by nonlinear energy control

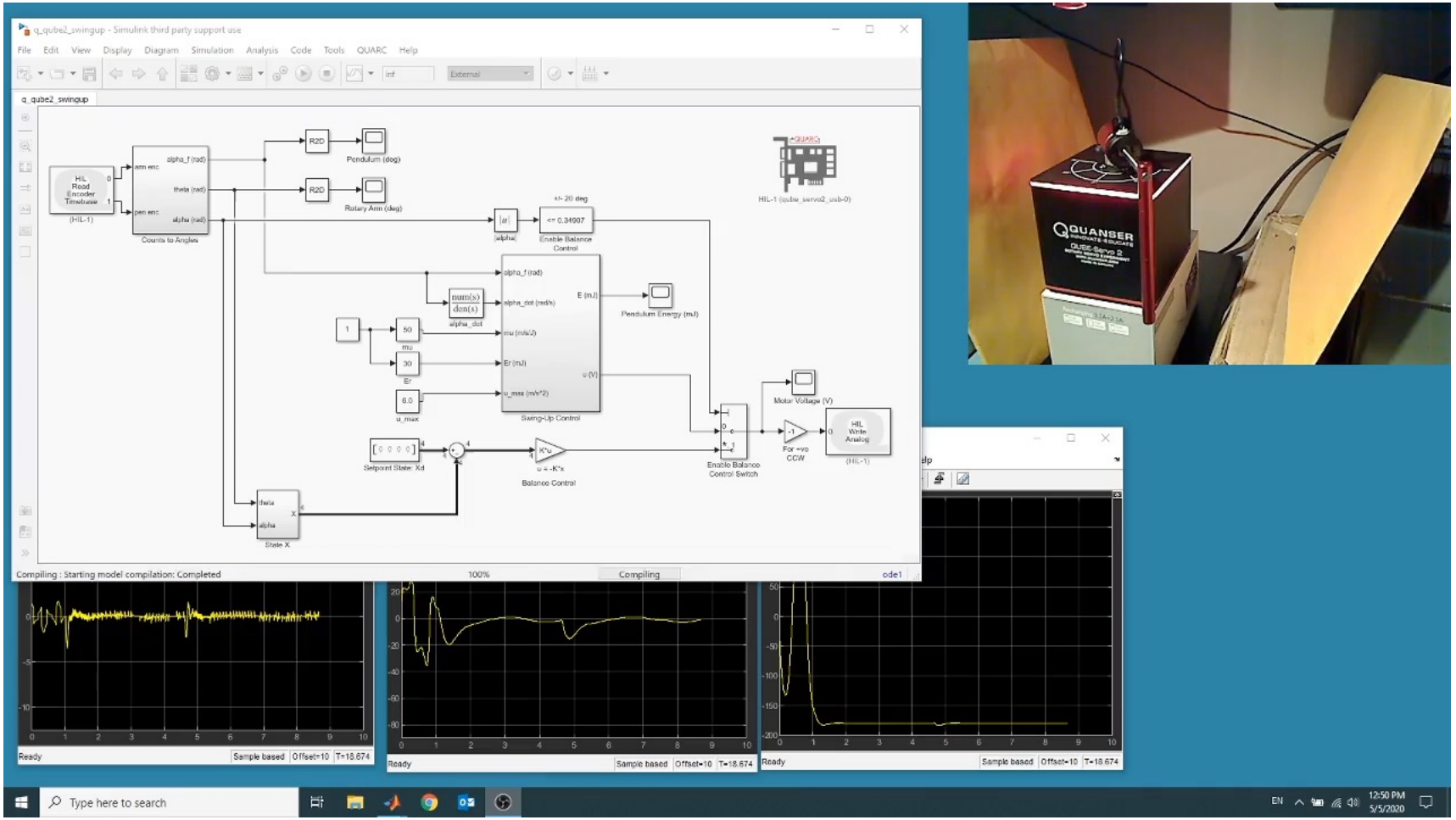
- Swinging up a pendulum can be done analytically using nonlinear energy control

Use dynamics of pendulum to swing it up automatically!

Pendulum swing-up + balance implementation



Swing-up and balance control of the QUBE Servo-2 inverted pendulum



WRAP-UP

One Methodology, Two Toolkits. Then Into the Labs

	PID (Lab 1)	State feedback (Lab 2)
Model form	Transfer function G(s)	State-space (A, B, C, D)
Handles instability?	Not naturally	Yes, if controllable
Handles multiple outputs?	Not naturally (SISO)	Yes - uses full state x
Design tool used today	Pole placement on 2nd-order Closed Loop TF	Pole placement or LQR

NEXT: LAB 1 (4H)

PD position control on the QUBE-Servo 2 with the inertia disc, then a robustness test with a disturbance load.

THEN: LAB 2 (4H)

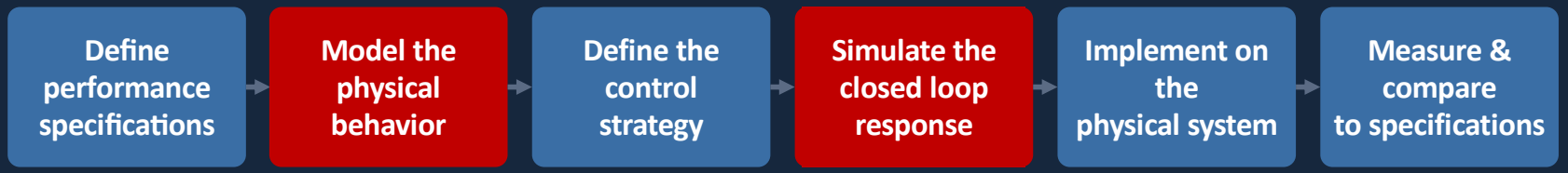
Pole-placement then LQR balance control on the QUBE-Servo 2 with the rotary pendulum.

If a design does not behave as expected in either lab, return to the methodology loop from Slide 3: revisit the model before touching the gains.

QUESTIONS?

Let's go build and balance !

The workflow is now yours to run, twice, on real hardware



22