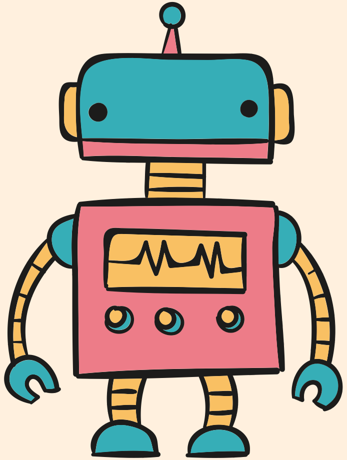
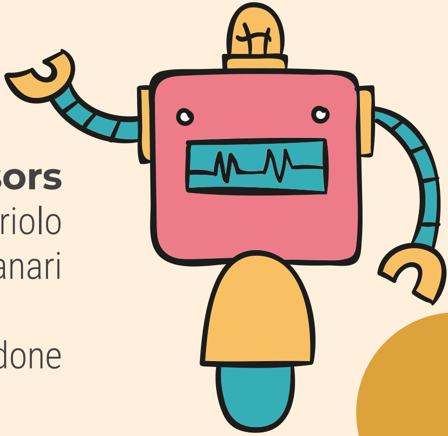
Elective in Robotics – Underactuated Robots

A.Y. 2020/2021



Modeling and Control of a Hybrid Wheeled Jumping Robot

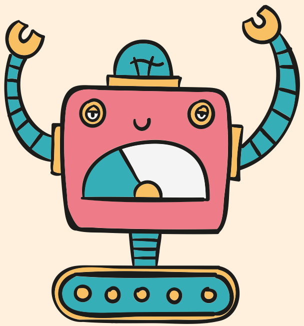


Candidates
Rossella Bonuomo, 1923211
Marco Pennese, 1749223
Veronica Vulcano, 1760405

Professors
Giuseppe Oriolo
Leonardo Lanari

Supervisor Filippo Smaldone

Table of Contents



Introduction

- We apply Model Predictive Control to a wheeled robot with a prismatic extension joint, that can be considered a wheeled-legged system.
- These systems can combine the benefits of both **mobile robots** and **legged systems**.

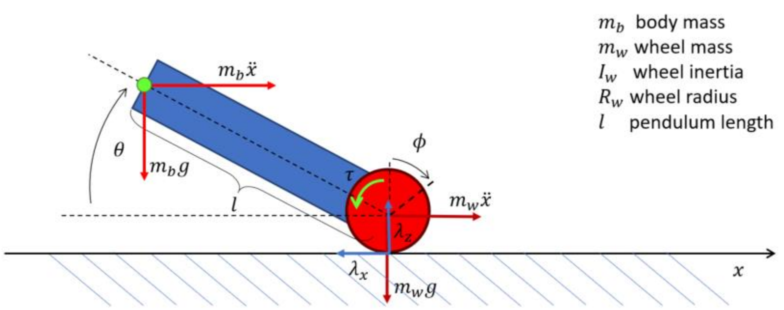
- After showing the **system dynamics**, we propose a scenario in which the robot should swing-up and balance, drive upright and jump over a gap under the control of the MPC.
- We also consider a PD controller in order to make a comparison between the two approaches. To test **robustness** of both controller, we have also included noise in sensor's reading.

Phase 0 Model

- In the first phase of the simulation, the robot has to **slow down and swing-up** by applying torque to the wheel. We constraints l (length of the body) to be constant, so to simplify the model.

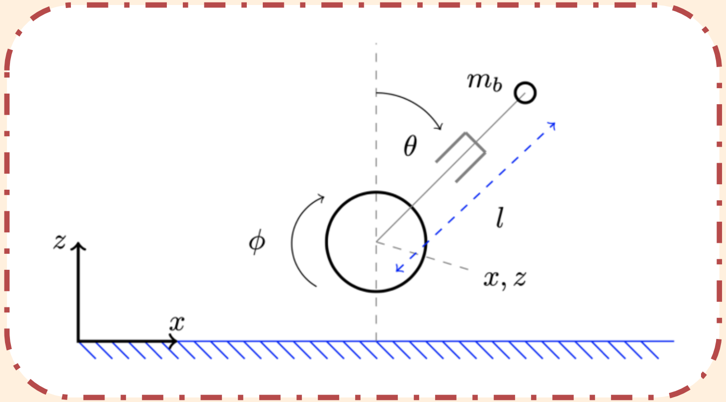
- By considering the forces and torque applied to the car we obtain the **dynamic model**.

$$\begin{cases} \ddot{\phi} = \dfrac{\tau}{(I_w + m_t R_w^2)} \\ \ddot{\theta} = \dfrac{1}{m_b l^2}\left(-m_b l \sin\theta \, R_w \ddot{\phi} + m_b g l \cos\theta - \tau\right) \end{cases}$$



With:
- ϕ: angle of rotation of the wheel;
- θ: angle of the car with respect to the ground;
- m_t: total mass (sum of the body mass m_b and wheel mass m_w);
- τ: control input consisting in the torque applied to the wheel.

Variable-Length Wheeled Inverted Pendulum



- It consists of a **wheel** and a **pole** modeled as a point mass located at a certain distance from the wheel.
- The **prismatic joint** has to mimic the capability of a leg. The distance of the point mass to the wheel is not constant.

The **generalized coordinates** are $q = [x \quad z \quad \phi \quad l \quad \theta]^T$, the **control inputs** are $u = [\tau \quad f]^T$

The **dynamic model**, derived through the Lagrangian method, is:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)u = S^T u + J_c^T \lambda$$

Contact forces $\lambda = [\lambda_x, \lambda_z]$

Inertia matrix

Coriolis matrix

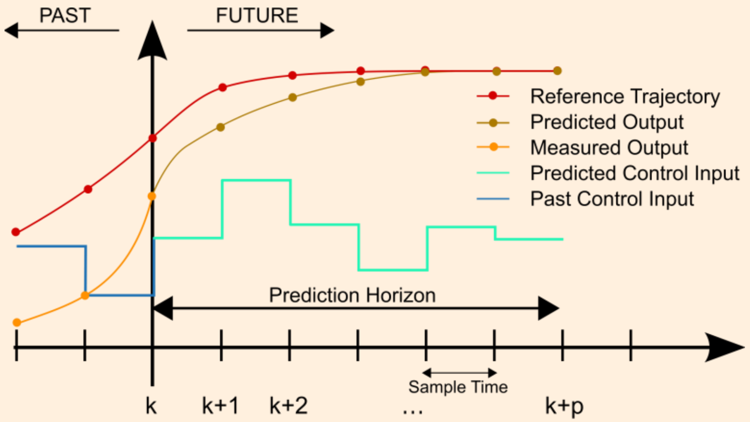
Gravity vector

Selection matrix

Contact Jacobian

Model Predictive Control

- MPC includes a family of control methods for achieving **optimal performance** while satisfying a *set of constraints*.
- The optimal control action is found by solving an **optimization control problem** over a finite *prediction horizon*.
- The *cost function* is minimized considering the **process dynamics** along the horizon.



We want to control the system during different phases:

- Swing-up and balance
- Driving upright
- Jumping over a gap

MPC – Problem formulation

Cost function $$\min_{X,U,\Lambda} \sum_i \left((x_i^* - x_i)^T Q (x_i^* - x_i) + u_i^T U u_i \right) + (x_N^* - x_N)^T P (x_N^* - x_N)$$

$$s.t.$$

Start state $$x_0 = x_0^*$$

Goal state $$x_N = x_N^*$$

Dynamics $$x_{t+1} = f(x_t, u_t, \lambda_t), \qquad t \in [0, N]$$

State bounds $$x^- \leq x_t \leq x^+, \qquad t \in [0, N]$$

Control bounds $$u^- \leq u_t \leq u^+, \qquad t \in [0, N-1]$$

Flight phase
(no ground force)

If $t \in [T_{tf}, T_{td}]$

$$\lambda_t = 0$$

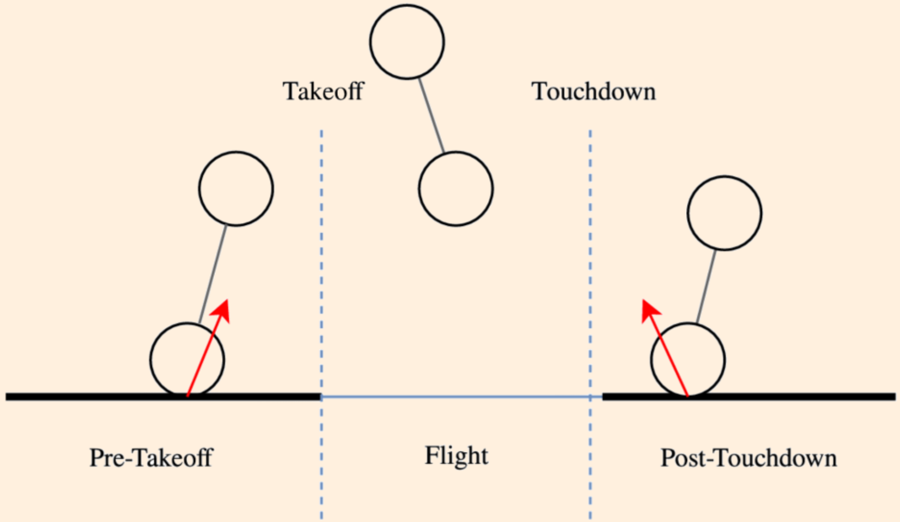
Ground phases
(friction cone)
(unilateral force)

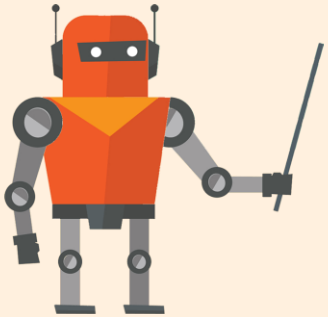
If $t \notin [T_{tf}, T_{td}]$

$$|\lambda_t^x| \leq \mu \lambda_t^z$$

$$\lambda_t^z \geq 0$$

No slip on ground $$\dot{x}_t = R_w \dot{\phi}_t$$





PD Controller

We want to control the robot during the **balancing phase** with a **Proportional Derivative controller**

$$\tau = -K_p^{\theta} e(\theta) - K_D^{\theta} e(\dot{\theta}) - K_P^{\phi} e(\phi) - K_D^{\phi} e(\dot{\phi})$$
$$f = f_{des} + K_P^{l} e(l) + K_D^{l} e(\dot{l})$$

The interpretation of the terms is:

- **Proportional action:** it immediately reacts to the variations of the error that decreases as the proportional constant K_P increases. Of course, there is a limit for the increase of the proportional constant.

- **Derivative action**: it is proportional to the derivative of the error and it is used to improve the transient of the response since it can decrease the overshooting.

Experiments

The simulations have been implemented in Python using the acados solver.

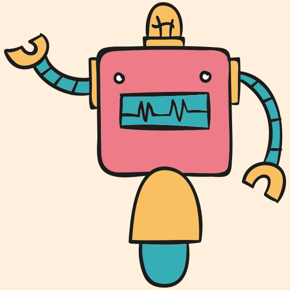
Jumping over a gap (MPC)

In the simulation, we set the following values for the robot **parameters**:

$$m_b = 4\;kg$$
$$m_w = 2\;kg$$
$$R_w = 0.17\;m$$
$$I_w = m_w R_w^2 = 0.0578\;m \cdot kg^2$$

The integrations have been performed using **Runge-Kutta of 4th order** and the **sampling time** of the controller is $0.05\;s$.

- In the phase 0, we set a **prediction horizon** of $0.5\;s$.
- In the second phase, the **prediction horizon** is equal to $2\;s$.



Jumping over a gap (MPC)

We define the problems to be solved with the MPC controller:

1. The first uses the phase 0 model to drive the four-wheels robot from a velocity of $\dot{x} = 7\ m/s$ up to a velocity of $\dot{x} = 0\ m/s$; this phase lasts less than one second.

State bounds	Control bounds
$X_{lim} = \begin{bmatrix} -\infty & 0 & -\infty & -\infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$	$U_{lim} = \begin{bmatrix} -10 \\ 10 \end{bmatrix}$

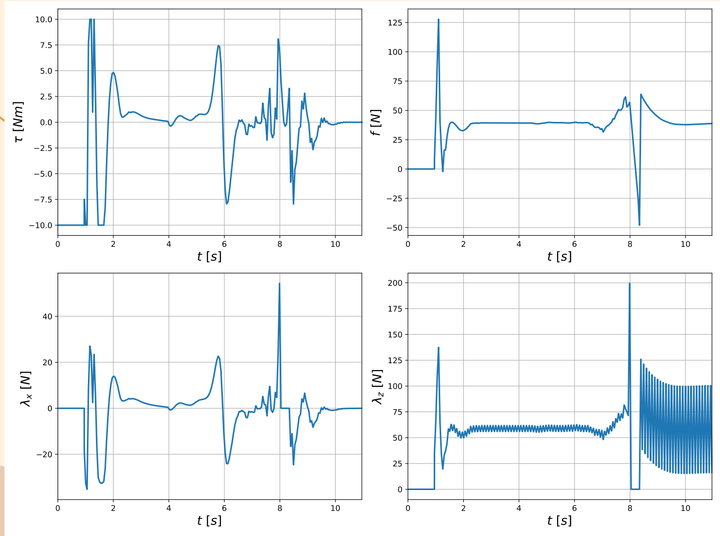
2. Once we reach the target velocity, we switch to the VL-WIP model.

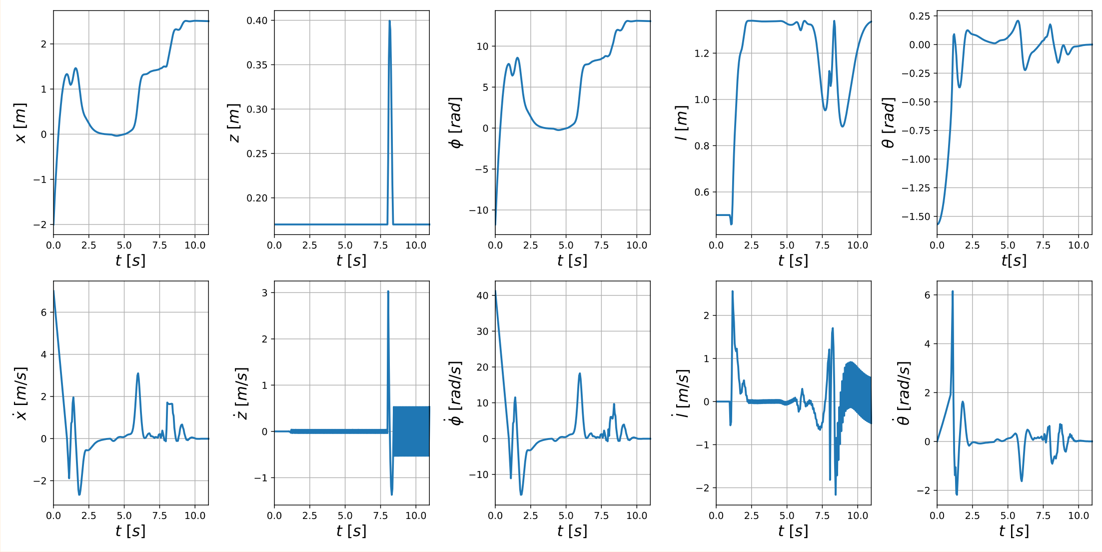
State bounds	Control bounds
$X_{lim}^{pre-tf} = \begin{bmatrix} -\infty & R_w & -\infty & 2R_w & -\pi/2 & -\infty & -\infty & -\infty & -\infty & -\infty \\ 1.5 & R_w & \infty & 1+2R_w & \pi/2 & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$	$U_{lim}^{\text{ground}} = \begin{bmatrix} -10 & -200 & -\infty & 0 \\ 10 & 200 & \infty & \infty \end{bmatrix}$
$X_{lim}^{flight} = \begin{bmatrix} -\infty & R_w & -\infty & 2R_w & -\pi/2 & -\infty & -\infty & -\infty & -\infty & -\infty \\ \infty & \infty & \infty & 1+2R_w & \pi/2 & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$	$U_{lim}^{\text{flight}} = \begin{bmatrix} -10 & -200 & 0 & 0 \\ 10 & 200 & 0 & 0 \end{bmatrix}$
$X_{lim}^{post-td} = \begin{bmatrix} 2 & R_w & -\infty & 2R_w & -\pi/2 & -\infty & -\infty & -\infty & -\infty & -\infty \\ \infty & R_w & \infty & 1+2R_w & \pi/2 & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$	

Jumping over a gap (MPC)

The robot has first to drive horizontally on four wheels, then it should get up and balance on two wheels, drive forward and finally jump over a gap. A two stage controller is used to switch from one model to the other.

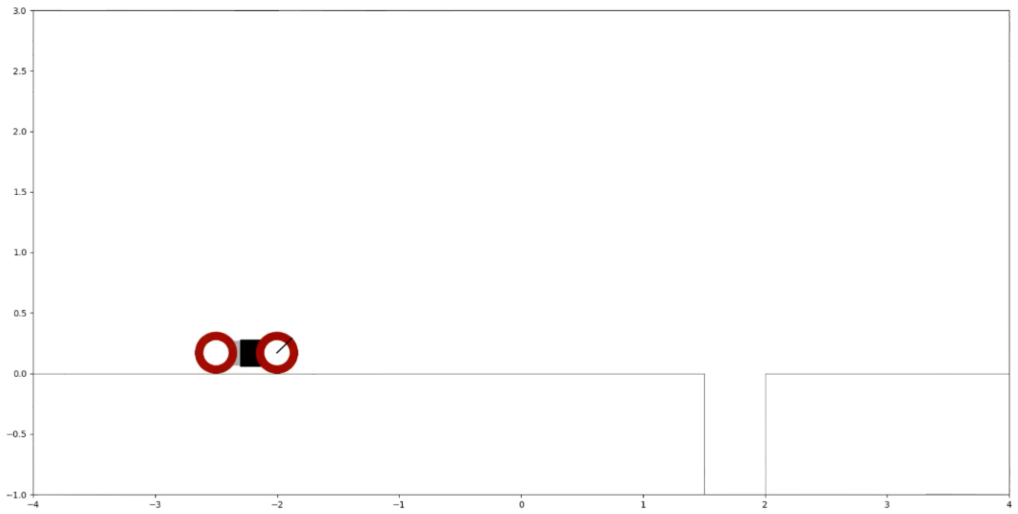
Control variables





State variables

Jumping over a gap (MPC)



The robot contracts its body by restricting the prismatic joint before the take-off and then extends it before approaching the ground again.

PD vs MPC – simplified model

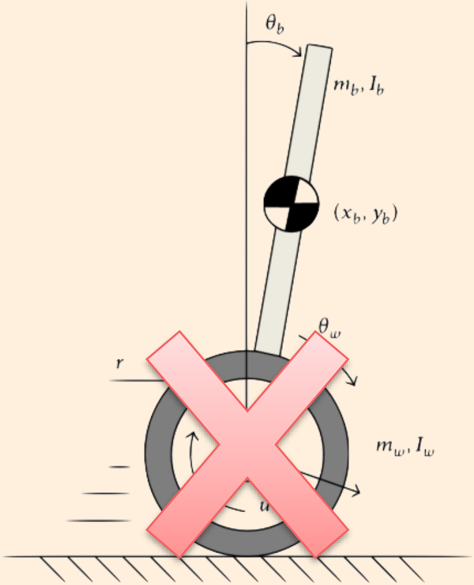
- We are going to control the robot during the *balancing phase* only, so we do not care about the floating base; we always want the wheel in contact with the ground.

- We can use a **simplified model**:

  - x is not an independent variable since $x = R_w\varphi$;
  - z is kept constant and equal to R_w;
  - The vector of generalized coordinates becomes $q = [\varphi \quad l \quad \theta]^T$.

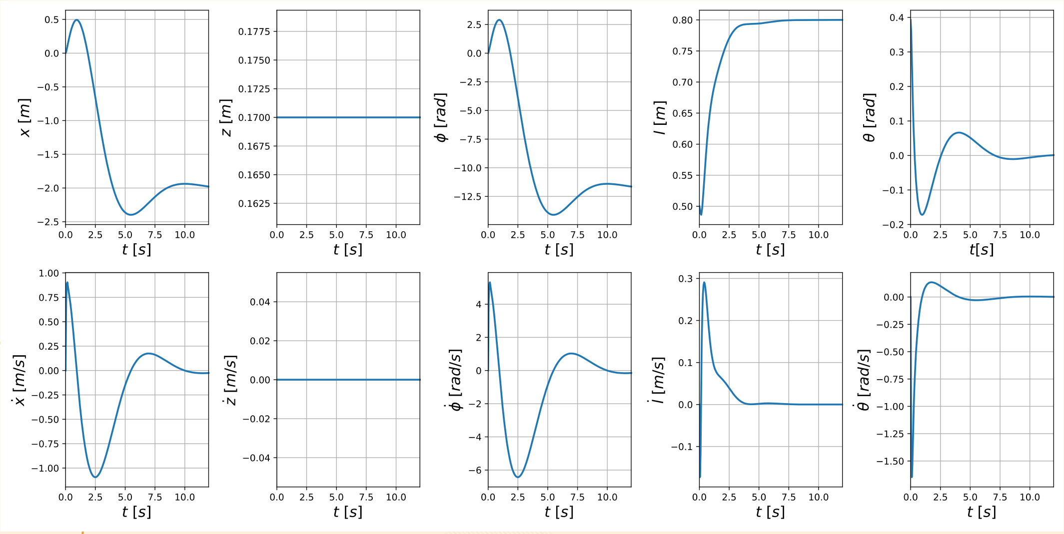
- The new dynamics are derived using the *Lagrangian formulation*.

- The control inputs are $\mathbf{u} = \begin{bmatrix} \tau \\ f \end{bmatrix}$.

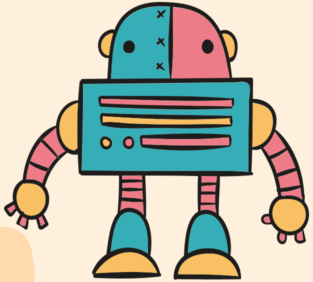


PD vs MPC – results

- We compare the results for the balancing phase, from $q_i = \begin{bmatrix} 0 & 0.5 & \frac{\pi}{8} \end{bmatrix}$ to $q_f = \begin{bmatrix} -\frac{2}{R_w} & 0.8 & 0 \end{bmatrix}$.
- We plot the whole state variables knowing that $x = R_w\phi$ and $z = R_w$



PD results



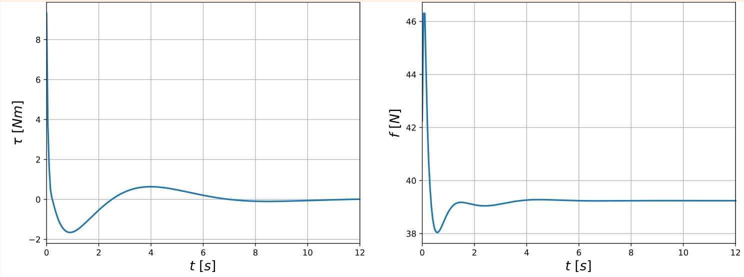
$K_P^\theta = 20, \quad K_D^\theta = 4$

$K_P^\phi = 0.125, \quad K_D^\phi = 0.2$

$K_P^l = 20, \quad K_D^l = 20$

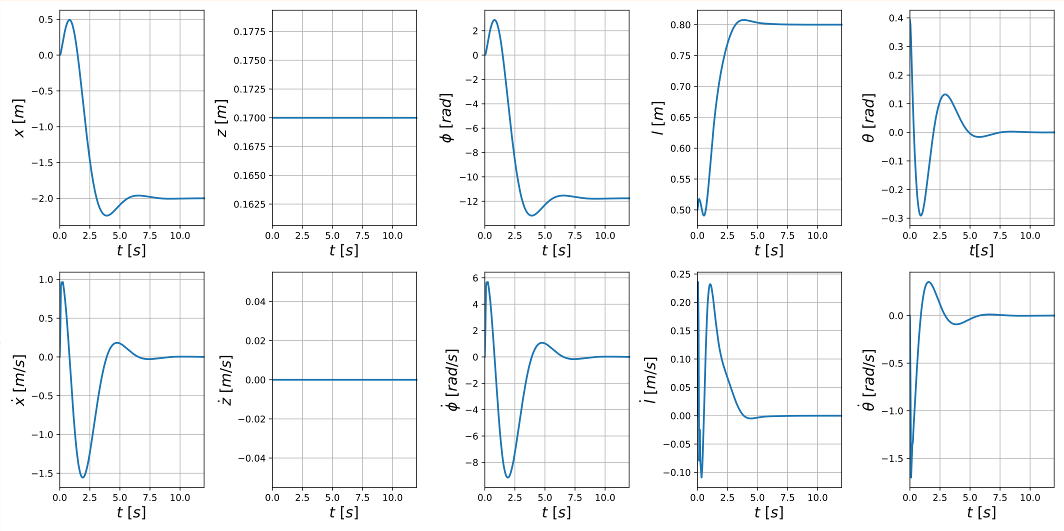
State variables

Control variables

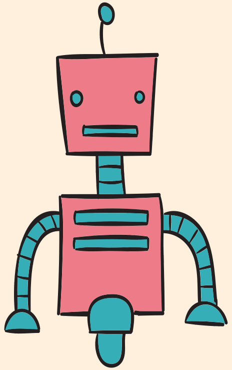


PD vs MPC – results

- We compare the results for the balancing phase, from $q_i = \begin{bmatrix} 0 & 0.5 & \frac{\pi}{8} \end{bmatrix}$ to $q_f = \begin{bmatrix} -\frac{2}{R_w} & 0.8 & 0 \end{bmatrix}$.
- We plot the whole state variables knowing that $x = R_w \phi$ and $z = R_w$

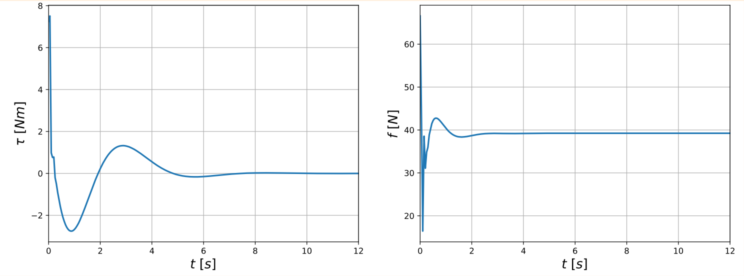


MPC results



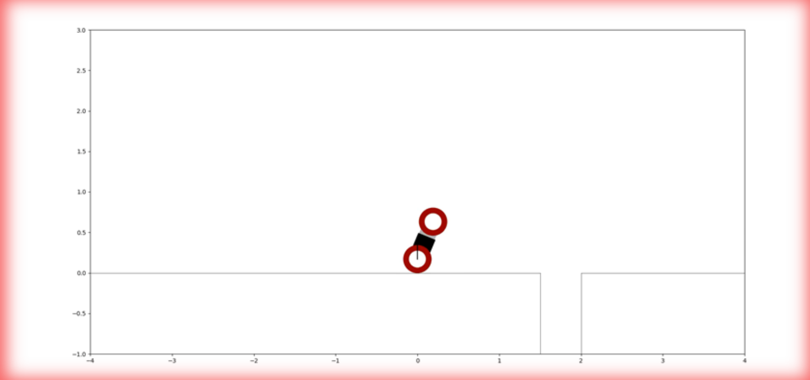
State variables

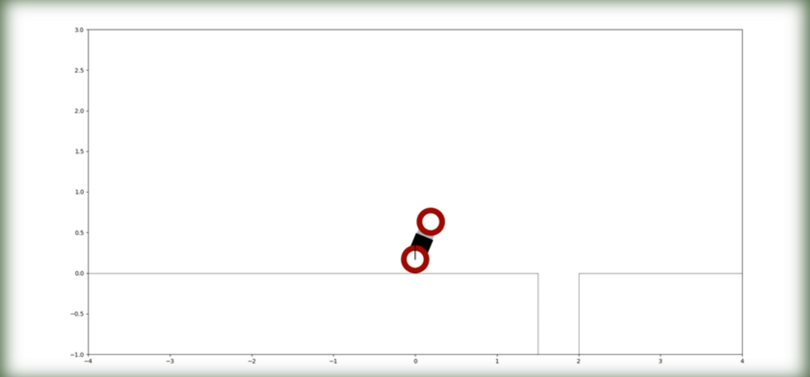
Control variables



PD vs MPC – results

PD





MPC

PD vs MPC – computational time

We want to compare the computational times for the two controllers considering:

- **Mean time**: mean time value among all the iterations
- **Std dev**: standard deviation among all the iterations
- **Max time**: time that the slowest iteration needs to finish
- **Total time**: time needed to execute 12 s of simulation

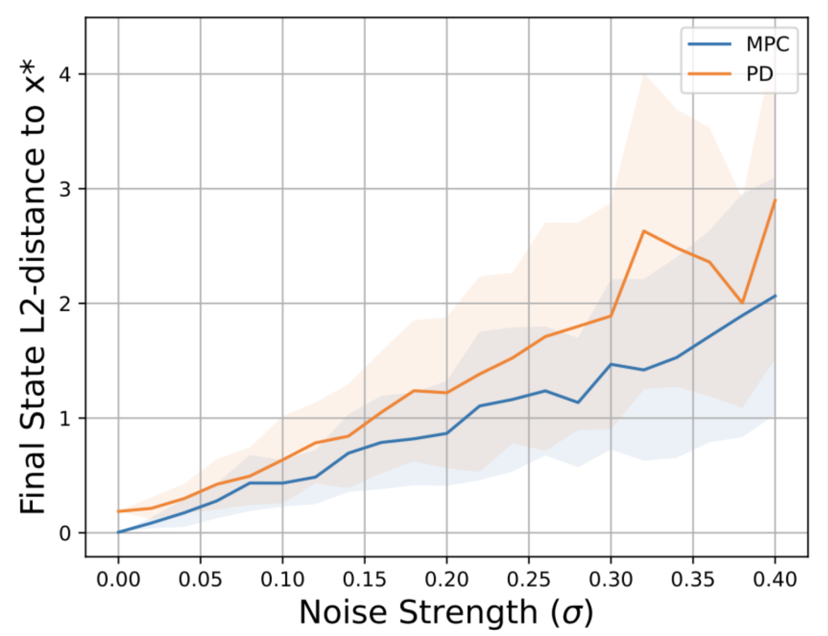
	Mean time	Std dev	Max time	Total time
PD	$8.131 \cdot 10^{-5}$	$2.189 \cdot 10^{-5}$	$2.546 \cdot 10^{-4}$	$1.951 \cdot 10^{-2}$
MPC	$2.534 \cdot 10^{-3}$	$6.131 \cdot 10^{-4}$	$5.418 \cdot 10^{-3}$	$6.082 \cdot 10^{-1}$

MPC is ***slower*** than PD controller!

Extension: noise on sensor's reading

We added noise on sensor's reading both on MPC and PD for the balancing phase.

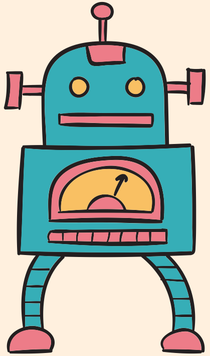
- Sensor reading at each step is:
$$x_{read} = x + \mathcal{N}(0, \sigma\mathbb{I})$$
- Noise varies in range 0-0.4 with a step size 0.02.
- For each value of σ we run 40 experiments.

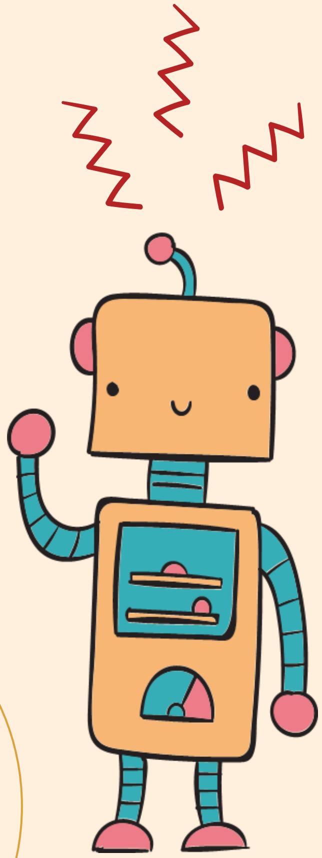


Conclusions

Despite its very simple dynamics, the VL-WIP is able to generate **complex motions**.

- **MPC** allows to control it in a precise way, obtaining high accuracy motions throughout all the phases.
- We can also use a **PD controller**, but we have to tune gains in an empirical way.
- With the MPC we can obtain even more difficult tasks (as **jumping**) in an easier way; however, the computational times are higher.
- MPC is more robust than PD controller since it is less sensible to **noise**.





Thank you for listening!