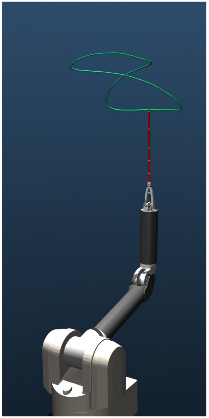
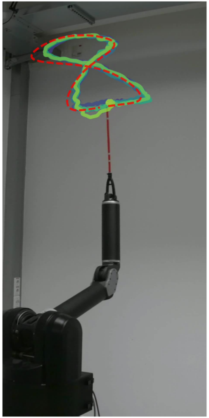
Integrated Project 2: Pendulum Acrobatics

Florian Wolf, supervised by: Pascal Klink and Kai Ploeger



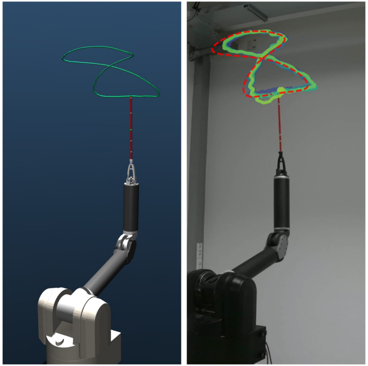


Overview

1. Experimental Setup and Aim of the Project
2. Formulation as an Optimal Control Problem
3. Results in Simulation (IP 1)
4. MPC Controller: Close Sim2Real Gap (Simulation, no transfer to the real system)
   - 4.1 Time Delays
   - 4.2 Emulate Optitrack Observations
   - 4.3 Noise
   - 4.4 Friction, Damping and Armature
5. LQR Controller (transfer to real system)
6. Conclusion

Experimental Setup and Aim of the Project

- Three-dimensional spherical cart pole system at end-effector of Barrett WAM robotic arm
- Complex system:
  - Highly nonlinear, unstable and underactuated
  - Requires fast-reactive controller
  - Can fall off in lateral direction too



Experimental Setup and Aim of the Project

- Three-dimensional spherical cart pole system at end-effector of Barrett WAM robotic arm
- Complex system:
  - Highly nonlinear, unstable and underactuated
  - Requires fast-reactive controller
  - Can fall off in lateral direction too
- Generic Goal (**Tracking Control**):

Follow a reference trajectory in 3D space with the pendulum's tip while keeping the 3D spherical inverted pendulum in a reasonably balanced state.

Formulation as an Optimal Control Problem

Given:

- MPC horizon of T_{MPC} seconds and a time step $t_0 \in [0, T - T_{\mathrm{MPC}}]$
- Target reference points $(\mathbf{x}_t^{\mathrm{PP,tip}})_{t\in[0,T]} \subset \mathbb{R}^3$ (task space) and target angles $(\theta_t^{\mathrm{PP}})_{t\in[0,T]} \subset \mathbb{R}$

Goal: Solve

$$\begin{aligned}
\min_{\substack{(\mathbf{q}_t)_{t\in[t_0,T_{\mathrm{MPC}}]},\\ (\mathbf{u}_t)_{t\in[t_0,T_{\mathrm{MPC}}]}}} \quad & \mu_{\mathbf{x}} \int_{t_0}^{T_{\mathrm{MPC}}} \left\| \mathbf{x}_t^{\mathrm{tip}} - \mathbf{x}_t^{\mathrm{PP,tip}} \right\| \mathrm{d}t + \mu_\theta \int_{t_0}^{T_{\mathrm{MPC}}} 1_{|\theta_t| \geq |\theta_t^{\mathrm{PP}}|} \cdot \|\theta_t\| \, \mathrm{d}t + \ldots \text{ (regularization)} \\
\text{s.t.} \quad & \forall t \in [t_0, T_{\mathrm{MPC}}] : \; \mathbf{u}_t \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}], \\
& \mathbf{u}_t = M(\mathbf{q}_t)\ddot{\mathbf{q}}_t + c(\mathbf{q}_t, \dot{\mathbf{q}}_t) + g(\mathbf{q}_t), \\
& \mathbf{x}_t = f_{\mathrm{F}}(\mathbf{q}_t),
\end{aligned} \tag{1}$$

$\Rightarrow$ OC Library Crocoddyl (Mastalli et al. 2020) with Feasibility driven DDP (FDDP) solver in MPC setting

Results in Simulation (IP 1)

Setpoint Reach and Tracking Control

Tracking Control: **Video 1**

Transfer Results to the Real System: Closing the Sim2Real Gap

Closing the Sim2Real Gap

Forward Integrator to compensate for Time Delays

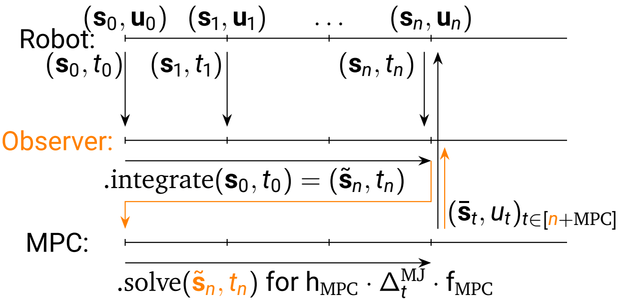


Figure: **All torques can be executed:** Solution states $(\bar{\mathbf{s}}_t)_{t\in[n+\mathrm{MPC}]} := \{\mathbf{s}_n, \bar{\mathbf{s}}_{n+1}, \dots, \bar{\mathbf{s}}_{n+\mathrm{h}_{\mathrm{MPC}}\cdot\mathrm{f}_{\mathrm{MPC}}}\}$ and corresponding torques. (Assume n steps (in 500 Hz) of computation time.)

Closing the Sim2Real Gap

Emulate Optitrack observations with MuJoCo-sites

Problem: No Joint Encoders for the pendulum joints, have to use OptiTrack

- Reconstruct joint positions:
  - Linear regression on *yz*-coordinates of the four markers on the pendulum pole
  - Transform unit vector back to local endeffector frame
  - Trigonometric identities to recover joint positions

$\Rightarrow$ Only joint positions, NO velocities

- Naive idea: use finite difference to compute joint velocities
  - Strong noise amplifications due to imperfect observations
  - Not applicable (even in simulation)

Solution: Use a nonlinear state observer

Closing the Sim2Real Gap

Using a nonlinear State Observer to not rely on joint velocity inputs

Solution:

- Nonlinear Luenberg-Observer
  - Uses internal model of the system
  - PD gains only on positional differences

  $\Rightarrow$ only relies on joint position observations, no velocities needed
- Fine-tuning of observer configurations is crucial:
  - Hyperparameter grid search for feedback gains
  - Relative integration timestep of $\frac{t_{\text{curr}} - t_{\text{last}}}{5}$ s

Closing the Sim2Real Gap

Custom Dynamics Model to incorporate Friction, Damping and Armature

- Damping and armature via a custom dynamics model in Crocoddyl
- Use `tanh`-model for friction
  - Unstable derivatives and causes huge computational times

  $\Rightarrow$ Use friction only in the forward predictor and the observer, but not in the solver

Show **Video 2**

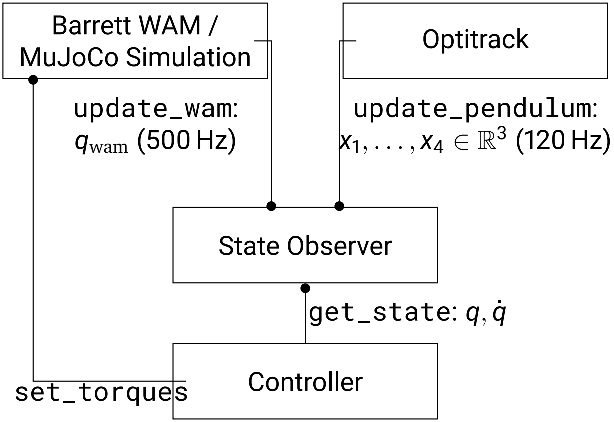


Figure: Controller schema

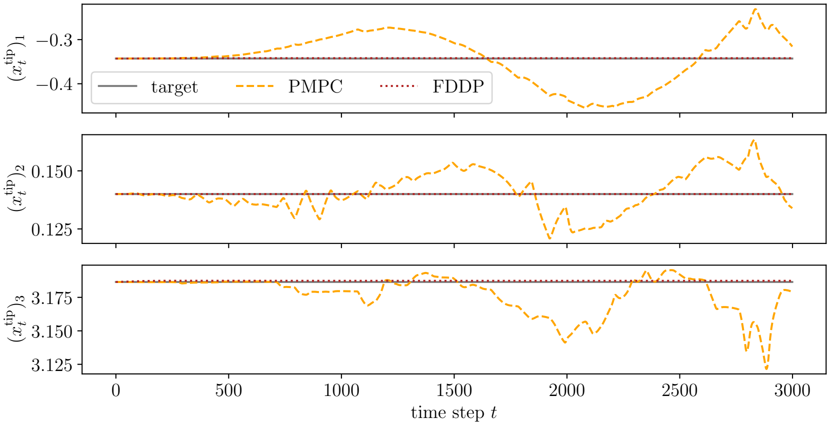


Figure: Task space results for the stabilization task with the final MPC controller scheme. We have 10^{-3} noise on the pendulum observations and $5 \cdot 10^{-4}$ noise on the joint observations. The average ℓ^2-tracking error between $\mathbf{x}^{\mathrm{tip}}$ and $\mathbf{x}^{\mathrm{PP,tip}}$ is $4.933 \cdot 10^{-2}$ m.

$\Rightarrow$ Not applicable on the real system, try LQR controller instead

LQR Controller
Linear Feedforward Controller with PD-feedback can be transferred to the Real System

Biggest problems with the MPC controller:

- Cannot react to noise while the solver is computing
- Solver torques are really aggressive and thus not robust against noise

$\Rightarrow$ Torques cannot be stably executed on the real system

Solution: Use a simplified Feedforward Controller with PD-feedback gains

- LQR-Controller scheme: $\mathbf{u}_t = \mathbf{u}_t^{\mathrm{FF}} + K_k \cdot \left(\mathbf{s}_k^{\mathrm{d}} - \mathbf{s}_k\right) + k_k$
- Each computation is just a matrix multiplication $\Rightarrow$ no online computation time needed
- Disadvantage: Have to ensure that the system stays close to the internally planned nominal trajectory

To our surprise, the LQR controller works really well in simulation and could be transferred to the real system (Implementation in C++)

- Real System **Video 3** (Stabilization), **Video 4** (Tracking Control)

Conclusion

LQR Emerges as the Preferred Controller Over MPC Due to Its Simplicity and Responsiveness

1. **LQR Preferred for Simplicity:** Streamlined gain tuning.
2. **No Time Delays with LQR:** No need for complex state estimation methods.
3. **Quick Response to Disturbances:** Every operation is just a matrix multiplication.
4. **Real-time Performance at 125Hz:** Ensuring real-time performance.
5. **Tuning Observer Crucial:** Meticulous gain tuning and careful observer selection is required.

Conclusion

LQR Emerges as the Preferred Controller Over MPC Due to Its Simplicity and Responsiveness

1. **LQR Preferred for Simplicity:** Streamlined gain tuning.
2. **No Time Delays with LQR:** No need for complex state estimation methods.
3. **Quick Response to Disturbances:** Every operation is just a matrix multiplication.
4. **Real-time Performance at 125Hz:** Ensuring real-time performance.
5. **Tuning Observer Crucial:** Meticulous gain tuning and careful observer selection is required.

Thank you for your Attention!