OCP/MPC Workshop 2024
Impact tutorial: part 2

Alejandro Astudillo Vigoya, Wilm Decré, Louis Callens, Alex Gonzalez García, Dries Dirckx

July 17, 2024

1 Point-to-point MPC for a robot manipulator

In this assignment, you will create a model predictive controller for a point-to-point motion of a robot manipulator (Franka Panda).

Given:

- $\mathbf{x}_{\text{current}} = \begin{bmatrix} \mathbf{q}_0 \\ \dot{\mathbf{q}}_0 \end{bmatrix}$: Initial state of the robot
- $\mathbf{p}_f$: Desired position to reach in Cartesian coordinates

We want to minimize the position error along and at the end of the horizon T and regularize the joint variables over the trajectory. The state vector is $\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}$ and the control input is $\ddot{\mathbf{q}}$. The optimization problem can be formulated as:

$$\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \quad \int_0^T w_1 \left(\|\mathbf{p}(\mathbf{q}(t)) - \mathbf{p}_f\|^2 + w_2\|\mathbf{q}(t)\|^2 + w_3\|\dot{\mathbf{q}}(t)\|^2 + w_4\|\ddot{\mathbf{q}}(t)\|^2 \right) dt + w_5\|\mathbf{p}(\mathbf{q}(T)) - \mathbf{p}_f\|^2 \tag{1a}$$

$$\begin{aligned}
\text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_{\text{current}}, && \text{(1b)}\\
& \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), && \text{(1c)}\\
& \mathbf{q}_{\min} \leq \mathbf{q}(t) \leq \mathbf{q}_{\max}, \quad \forall t \in [0, T], && \text{(1d)}\\
& \dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}}(t) \leq \dot{\mathbf{q}}_{\max}, \quad \forall t \in [0, T], && \text{(1e)}\\
& \ddot{\mathbf{q}}_{\min} \leq \ddot{\mathbf{q}}(t) \leq \ddot{\mathbf{q}}_{\max}, \quad \forall t \in [0, T], && \text{(1f)}\\
& \dot{\mathbf{q}}(T) = \mathbf{0} && \text{(1g)}
\end{aligned}$$

where:

- $\mathbf{p}(\mathbf{q})$ is the forward kinematics mapping from joint space to Cartesian space.
- w_i is a weighting factor for the control effort.
- $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are the lower and upper bounds on joint positions.
- $\dot{\mathbf{q}}_{\min}$ and $\dot{\mathbf{q}}_{\max}$ are the lower and upper bounds on joint velocities.
- $\ddot{\mathbf{q}}_{\min}$ and $\ddot{\mathbf{q}}_{\max}$ are the lower and upper bounds on joint accelerations.

2 Getting started

This exercise uses the robot manipulator of a Franka Panda robot provided in the Robotics Toolbox for Python. This library can be installed by running either

```
pip install roboticstoolbox-python
```
or
```
conda install -c conda-forge roboticstoolbox-python
```

To generate robot dynamics and kinematics, we use the interface between the state-of-the-art rigid-body dynamics library `Pinocchio`. You can install Pinocchio by running:

```
conda install pinocchio -c conda-forge
```

and the interface by executing:

```
pip install git+https://gitlab.kuleuven.be/meco-software/robot-models-meco.git@pin3-devel
```

All this libraries are already included in the provided `mecoverse-robotics` conda environment, which you can create by running

```
conda env create -f mecoverse_robotics_environment.yml
```

Once you have your environment ready, go to the directory `Tutorials/3_impact/part-2`.

You should first run the `generate_robot_model.py` script to generate a symbolic robot model based on the robot description file (URDF file) of the Franka Panda robot that is included in the same directory. This script will generate a `franka_panda.impz` file (which you can open with any zip viewer) which includes serialized CasADi functions (`.casadi` files) for the robot kinematics, dynamics and their Jacobians, in addition to a yaml file that includes the definition of the robot model (including its variables and some additional constants, such as lower and upper bounds on joint positions, velocities and torques).

Notice that such `.impz` file is already provided. If you want to generate a new one, you should delete the existing `.impz` file before.

Once you have the generated `.impz` file, you can execute the `MAIN.py` script. The `controller.py` script is used to specify the optimal control problem underpinning the model predictive controller for the task at hand.