**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**Motion Planner**](https://github.com/epfl-lasa/mpc_motion_planner/tree/python_binding) **– Python library overview   
for any questions, please contact** [**stephen-monnet@hotmail.com**](mailto:stephen-monnet@hotmail.com)

1. **Main goals**

This library is based on the [mpc\_motion\_planner](https://github.com/AlbericDeLajarte/mpc_motion_planner/tree/0046c76840a71e86c6a057cb6caf798c75756847) project (C++), which aims to generate a trajectory between two arbitrary points and for the Franka Panda robot with 7 degrees of freedom. This trajectory can be generated either (a) using [ruckig](https://github.com/pantor/ruckig) or (b) using [polympc](https://github.com/PREDICT-EPFL/polympc) with ruckig as the starting solution. It should be noted that (a) only allows for considering box constraints on velocity, acceleration, and jerk, while (b) allows for adding non-linear constraints such as the minimum height of the tool or the maximum permissible torque.

In addition to making the functionalities of mpc\_motion\_planner accessible in Python, this library extends its capabilities to the Kuka iiwa7 and Kuka iiwa14 robots, offering numerous possibilities for manipulating and analyzing the generated data.

1. **Model**
   1. **Dynamic**

For polympc, the robot's dynamics are simply modelled as a double integrator between the input and the joint positions (thus, ):

* 1. **Objective Function (Polympc)**

The objective of the NOCP (Nonlinear Optimal Control Problem) is to minimize the total duration of the trajectory that connects the initial state to the final state . For more information on the formulation, refer to [Jennings problem](https://apmonitor.com/do/index.php/Main/MinimizeFinalTime).

* 1. **Ruckig constraints :**

**: Joint Velocity**

**: Joint Acceleration**

**: Joint Jerk**

* 1. **Polympc constraints :**

**: Joint Position**

**: Joint Velocity**

**: Joint Acceleration**

**: Joint Torque**

**: End Effector Height**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Main Architecture of the Framework**

The main task handled by the mpc\_motion\_planner block is to set up the Nonlinear Optimal Control Problem (NOCP) in a form that polympc can solve. It is responsible for evaluating the state of constraints and defining the dynamics of the system .

polympc (C++)

ruckig (C++)

mpc\_motion\_planner (C++)

Setup NOCP

Solve NOCP starting from initial guess , provided by ruckig

Compute time-optimal jerk-limited trajectory between and

Constraints  
Objective (minimize trajectory duration)  
Initial guess ,

motion\_planner (Python)

pybind11

Figure 1 : Framework Architecture

1. **Constraints Margin**

For practical reasons, it is important to incorporate a safety margin on certain constraints. For example, for a state constraint , new bounds will be calculated as  :

Note that if , then :

In the case of symmetrical constraints such as velocity, acceleration, or torque constraints. Please mind the notation, which might be a bit confusing, where is called “margin” while it is usually defined in the literature as .

1. **Classes**

The Python library is organized with two main classes:

* 1. **Trajectory**

A trajectory object stores mainly 5 vector values (time , joint position , joint velocity , joint acceleration , joint torque ) and provides convenient methods to assess constraint satisfaction.

* 1. **MotionPlanner**

A MotionPlanner object is used to interface with the C++ class that corresponds to the desired robot (Panda, Kuka-iiwa7, Kuka-iiwa14). It provides methods to set the constraint limits and boundary conditions ( and ), solve a trajectory using ruckig and/or polympc and compute inverse / forward kinematics.

For more information about how to use these objects, please refer to the GitHub repository : mpc\_motion\_planner/pyMPC/howToUse.ipynb