A Modular Software Ecosystem for Research in Learning-Based Receding Horizon Control - formulation notes

Andrea Patrizi[*†], Carlo Rizzardo[*], Arturo Laurenzi [*] and Nikos G. Tsagarakis[*]

Abstract— **Robotics research in manipulation and locomotion is undergoing a transformative shift towards the use AI-based tools, where purely learning-based control policies and pipelines are starting to take over the field. Despite being shown to be capable of remarkable robustness and performance even when applied to real-world environments, some key inherent limitations such as safety guarantees, interpretability, sample efficiency, etc.. persist. For this reason, it is the authors' belief that more classical control approaches should not be considered outdated yet. We thus advocate for a hybrid approach, combining offline data-based policy design, specifically through Reinforcement Learning (RL), with classical online Motion Planning, i.e. Receding Horizon Control (RHC). Even though this kind of hybrid approaches are not entirely new, to the authors' knowledge, there is no specific tool currently available for search in this domain. To this purpose, we developed a modular software ecosystem, hereby synthetically presented in its main components and features. We care to stress that the framework is currently under active development, and features might not be stable or could be lacking. To facilitate usability and diffusion, we made all the core components open source under the GPLv2 license. To showcase the potential of the framework, we furthermore briefly present a proof-of-concept example combining a high-level RL agent coupled with a lower-level MPC controller for the execution of a simple locomotion task on a hybrid wheeled-legged quadruped robot in simulation.**

I. Markov Decision Processes

Markov Decision Processes (MDPs) are a mathematical framework used to model decision-making in situations where outcomes are influenced by potentially probabilistic transitions and the actions taken by a decision-maker. MDPs are widely used in fields such as artificial intelligence, operations research, economics, and reinforcement learning to model and solve sequential decision-making problems. MDPs consist of a set of states, representing the possible configurations or conditions of the system being modeled. At any given time, the system is in one of these states. For each state in the MDP, there is a set of possible actions that the decision-maker can choose from. Actions represent the decisions or control inputs available to the decision-maker. Upon taking an action in a particular state, the system transitions to a new state according to a probability distribution. These transition probabilities capture the stochastic nature of the environment. Associated with each state-action pair is an immediate reward, representing the immediate benefit or cost incurred by the decision-maker upon taking that action in that state. A policy is a mapping from states to actions, specifying the decision-maker's behavior or strategy. It defines how the decision-maker selects actions in different states.
Key concepts:

- States: S
- Actions: A
- State Transition Probabilities (dynamics): $P(s' \mid s,a)$
- Immediate Rewards: $R(s,a,s')$

The objective in MDPs is to find an optimal policy that maximizes the cumulative reward obtained by the decision-maker over time. This involves balancing the trade-off between immediate rewards and long-term benefits, taking into account the probabilistic nature of the environment and the dynamics of the system.
Value function:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid s_0 = s\right] \quad (1)$$

Quality function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a\right] \quad (2)$$

II. Dynamic Programming and Bellman's optimality principle

Dynamic Programming (DP) is a powerful mathematical technique used for solving optimization problems by breaking them down into simpler subproblems. It's widely applicable in various fields, including computer science, operations research, economics, and engineering.

DP relies on the principle of optimal substructure, which states that an optimal solution to a problem can be constructed from optimal solutions to its subproblems. This property allows DP algorithms to efficiently solve complex problems by recursively solving smaller subproblems. DP algorithms store the solutions to subproblems in a table or memoization array to avoid redundant calculations. This enables efficient computation by reusing previously computed solutions when solving larger problems.

Bellman's Optimality Principle is a fundamental concept in DP, which states that an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. In other words, an optimal policy exhibits the property of optimality not only with respect to the current decision but also with respect to all subsequent decisions.

Let π^* be an optimal policy for a Markov Decision Process (MDP) with state space S, action space A, state transition

[†] Department of Informatics, Bioengineering, Robotics and Systems Engineering, Università di Genova, Via All'Opera Pia 13, 16145 Genova.

[*] Humanoids and Human-Centred Mechatronics (HHCM), Istituto Italiano di Tecnologia (IIT), Via San Quirico 19d, 16163 Genova.

probabilities $P(s' \mid s,a)$, and immediate rewards $R(s,a,s')$. Then, for any state s in the state space S, the following equation holds:

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^*(s') \right]$$

where:

- *$V^*(s)$ is the optimal value function for state s,*
- *$R(s,a)$ is the immediate reward received upon taking action a in state s,*
- *$P(s' \mid s,a)$ is the probability of transitioning to state s' after taking action a in state s,*
- *γ is the discount factor representing the importance of future rewards, and*
- *$\max_{a\in A}[\cdot]$ denotes the maximum over all possible actions in state s.*

Alternative, the same equation can be stated for the optimal action-value function (also named "quality function")

III. Reinforcement Learning

IV. Optimal control problem

Optimal Control is a field of study concerned with determining control inputs that optimize the behavior of a dynamical system over time. It aims to find the control strategy that minimizes or maximizes a certain performance criterion while satisfying system constraints. Optimal control has applications in a wide range of fields, including aerospace, robotics, manufacturing, finance, and economics. Optimal control deals with systems whose behavior evolves over time according to certain dynamics. These systems can be described by differential or difference equations that govern their evolution. In optimal control, the decision-maker (controller) selects control inputs or actions to influence the evolution of the system. These control inputs can be continuous (e.g., forces, torques) or discrete (e.g., on/off switches). Optimal control seeks to optimize a certain performance criterion that quantifies the quality of the system's behavior. This criterion could be to minimize energy consumption, maximize profit, track a desired trajectory, or regulate system variables within specified bounds. Optimal control problems often involve constraints on the system's state variables, control inputs, or other parameters. These constraints must be satisfied while optimizing the performance criterion. There are several types of optimal control:

1) **Open-loop Control:** In open-loop control, the control inputs are predetermined based on a model of the system and are not adjusted based on feedback from the system's output. Open-loop control is suitable for systems with predictable dynamics and no disturbances.
2) **Feedback Control:** Feedback control adjusts the control inputs based on feedback from the system's output, enabling adaptation to changes in the system or environment. Feedback control is essential for handling uncertainties, disturbances, and variations in the system's behavior.
3) **Stochastic Control:** Stochastic control deals with systems influenced by random disturbances or uncertainties. It seeks to find control strategies that optimize performance in the presence of uncertainty, using probabilistic models and decision-making criteria.

The optimal control problem is defined by the following constrained optimization program:

$$\begin{aligned} \underset{u_0,\cdots,u_{N-1}}{\text{minimize}} \quad & L(u) \\ \text{subject to} \quad & \dot{x}_k = f(x_k,u_k), \quad \forall k \\ & x_k \in \mathscr{X}, \quad \forall k \\ & u_k \in \mathscr{U}, \quad \forall k \\ & x_0 = x_{\text{init}} \end{aligned} \tag{3}$$

where

- System Dynamics: $\dot{x} = f(x,u)$
- Control Inputs: u
- State Constraints: $x \in \mathscr{X}$
- Control Constraints: $u \in \mathscr{U}$
- Objective Function: $L(u) = \sum_{k=0}^{N-1} l(x_k,u_k) + L_t(x_N)$

Solving this problem in real-time and in a receding-horizon fashion is usually referred to as Receding Horizon Control (RHC) or Model Predictive Control (MPC).