University of Trento

Industrial Engineering Department

Laboratory Experience

ID numbers: 249485, 248735

Project B: Learning a Terminal Constraint Set

Final Assignment

Daniele Turrini, Gianluca Lona

Period of development: 18/11/2024 - 28/11/2024

1 Introduction

This report presents the development and implementation of the final assignment's project "B", aimed at learning an approximate control-invariant set to serve as a terminal constraint in a Model Predictive Control (MPC) framework.
A critical challenge in MPC is ensuring the feasibility of the control problem, which requires computing control inputs that guide the system to the target configuration whithout violating the constraints. Terminal constraints play a key role in maintaining feasibility, particularly for systems with a finite prediction horizon. While simple terminal constraints like zero velocity can ensure feasibility, they may result in slow convergence, especially with small prediction intervals.

This project explores the use of the N-step backward reachable set, that is the set of states from which the system can come to rest within N time steps, as a terminal constraint to ensure recursive feasibility and improve convergence time. Since analytical computation of the BRS is very challenging even for a relatively simple system like a double pendulum, this project employs a data-driven approach. Random state samples are used to solve the optimal control problem for the BRS, generating a labeled dataset that indicates whether each state is in the set. A neural network is then trained on this dataset to approximate the BRS. Finally, the trained network serves as a classifier to impose terminal constraints within the MPC framework.

2 Problem Description

To compare and analyze the results, we generated multiple datasets, each corresponding to an optimization problem with different constraints. These datasets were used to train distinct neural networks, which were subsequently integrated as terminal constraints within the MPC framework. The main performance metrics considered include:

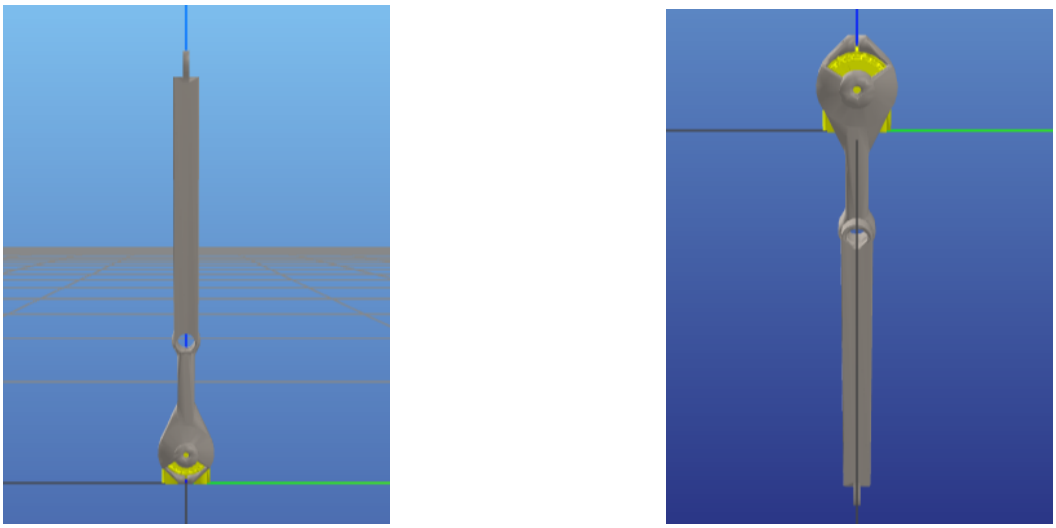
- **Convergence time**: The time required for the system to reach a steady state configuration.
- **Computation time**: The time required to solve the optimization problem at each step of the MPC. This value must remain low to enable real-time application of the controller.
- **Feasibility**: The ability of the solver to find a feasible solution that satisfies all constraints within the predefined limit of `SOLVER_MAX_ITER`, the maximum number of Newton's steps. This limit is set low to minimize computation time.
- **Offset**: The reached configuration could be different from the desired one so this value show the difference between them.

Four distinct models of the approximated backward reachable set were developed, each differing only in the number of time steps, N, within which the robot is required to come to a

stop ($N = 25, 50, 100, 200$). For smaller values of N, the terminal constraint closely resembles a zero-velocity terminal constraint. In contrast, larger values of N are expected to enable faster convergence to the target configuration while maintaining recursive feasibility.

2.1 Desired Configuration

The robot under consideration is a double pendulum with two joints, where the state is defined by four variables: two joint coordinates and two joint velocities. Starting from the initial position pointing upwards (with both joint coordinates set to zero), the robot is required to reach the opposite configuration, with the end effector pointing downwards, as shown below.



(a) Initial configuration (b) Final desired configuration

Figure (1) Configurations of the system

2.2 Joint Limits

2.2.1 Coordinate Limits

To increase the problem's complexity in terms of feasibility, coordinate bounds are imposed. These bounds are applied directly to the joints rather than the end effector, thereby avoiding the use of inverse kinematics. The desired configuration is $x_{\text{des}} = [-\pi, 0]$ (relative coordinates). The limits are defined as:

$$x_{\text{lim}} = \Big[-(\pi + \Delta), \quad 0 - \Delta\Big], \tag{1}$$

such that:

$$x_{\text{des}} \geq x_{\text{lim}}. \tag{2}$$

Here, Δ, set to 0.05, represents the limit factor that defines the maximum allowable joint rotation angle. This constraint, coupled with the reduced torque available at the joints, presents a significant challenge in terms of feasibility: the robot must decelerate its motion sufficiently

early to avoid exceeding these bounds while simultaneously compensating for the effects of gravity.

2.2.2 Torque Limits

We intentionally chose not to use the standard torque limits for the double pendulum model, opting instead for smaller maximum torque values at the joints. This was done to make the problem more challenging in terms of constraint satisfaction. The specific values used are provided below.

$$T_{\text{Joint 1}} = \Big[+0.6, \quad -0.6\Big], \quad T_{\text{Joint 2}} = \Big[+0.6, \quad -0.6\Big]. \tag{3}$$

2.2.3 Velocity Limits

In real-world applications, velocity constraints are typically applied to prevent excessive inertial forces that could potentially damage the robot. However, since our primary focus is on the system's behavior as it approaches the target configuration, we set the velocity bounds sufficiently large to ensure they don't impose a significant constraint on the problem. The values, in rad/s, are provided below.

$$v_{\text{Joint 1}} = \Big[-10, \quad +10\Big], \quad v_{\text{Joint 2}} = \Big[-10, \quad +10\Big]. \tag{4}$$

3 Dataset Generation

To create a dataset where each state is labeled (either 0 or 1) based on whether it lies within the backward reachable set, the process involves two main steps:

1. *Randomly sample a certain number of states from the state space.*
2. *Determine if the sampled states fall inside the backward reachable set.*

To maintain consistency with the MPC formulation, the state and input bounds used for the computation of the backward reachable set remain unchanged from those previously described. The only parameters that can be tuned are:

- *Number of samples generated*: Each sample consisting of the pair state and BRS label
- N: Number of steps for the backward reachable set formulation.

The control input invariant set used for this application is defined as:

$$\mathcal{S} = \{(q, \dot{q}) \mid q_{\min} \leq q \leq q_{\max},\, \dot{q} = 0\}, \tag{5}$$

This set consists of the zero-velocity states that are compatible with the given constraints. The states used for the generation of the dataset are randomly selected with a uniform probability within the following intervals:

$$\begin{aligned} -1.5 \leq q_1 \leq 0.2, \quad -0.5 \leq q_2 \leq 0.5, \\ -7.0 \leq \dot{q}_1 \leq 7.0, \quad -2.0 \leq \dot{q}_2 \leq 2.0, \end{aligned} \tag{6}$$

These intervals are centered around the trajectory that the system has to follow, helping to optimize the computations. Once all datasets are generated according to the specified *Number of samples*, determining whether a state lies within the backward reachable set involves solving an optimization problem. Given the discrete-time dynamics of the double pendulum, defined as:

$$x_{k+1} = f(x_k, u_k), \quad x_0 = x_{\text{init}}$$

the optimization problem is formulated as:

$$\begin{aligned} \underset{X,U}{\text{minimize}} \quad & 1 \\ \text{subject to:} \quad & x_{i+1} = f(x_i, u_i), \quad i = 0, \ldots, N-1, \\ & x_{i+1} \in \mathcal{X}, \quad u_i \in \mathcal{U}, \quad i = 0, \ldots, N-1, \\ & x_0 = x_{\text{init}}, \\ & x_N \in \mathcal{S}. \end{aligned}$$

Since we are only concerned with the feasibility of the optimization problem, the cost function is set to a constant value of 1. If the solver successfully finds a solution to the optimal control problem for a certain initial state, we can conclude that the state is within the N-step backward reachable set of the system.

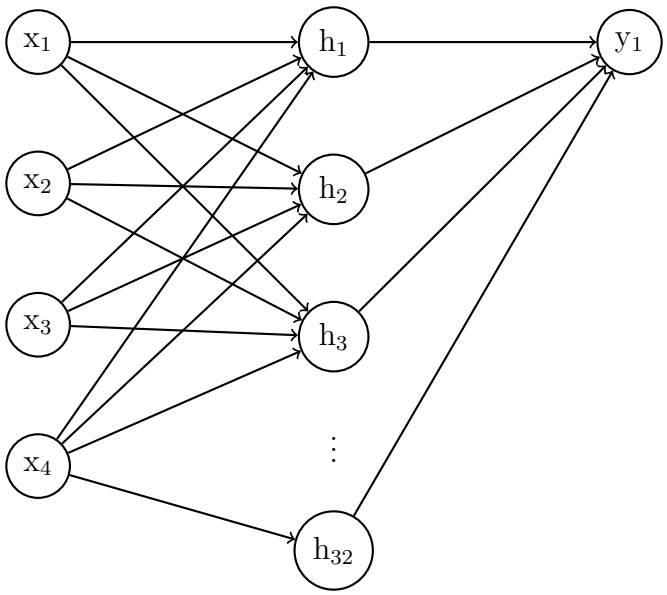
For all datasets created, the *Number of samples* was set to 5000.

4 Neural Network Training

After generating the dataset, we proceed to train a *Neural Network* to classify whether a given state lies within the backward reachable set. For this task, we utilized the PyTorch framework.

4.1 Architecture

The architecture of the model is as follows.



The model has the following dimensions:

- **Input size:** 4. The input state consists of four components (two positions and two velocities).
- **Hidden size:** 32. The problem's complexity is relatively low, and thus a larger hidden layer is unnecessary.
- **Output size:** 1. The output is a single scalar value representing whether the state is within the backward reachable set.

The model was trained over 50 epochs using 5000 samples. At the end of training, the test accuracy was consistently above 0.9, ensuring reliable performance.

4.2 Loss Function

The loss function evaluates the performance of the neural network by quantifying the difference between the predicted outputs and the true labels. For our model, we employ the *Binary Cross-Entropy with Logits Loss* (`BCEWithLogitsLoss`). This loss function is widely used in binary classification tasks, where the goal is to classify inputs into one of two categories. By combining the sigmoid activation function and the binary cross-entropy loss in a single operation, `BCEWithLogitsLoss` provides numerical stability and simplifies the training process.

4.3 Output

The neural network produces raw logits as output, which are subsequently processed through a sigmoid activation function. This transformation maps the logits to a probability value within the range $[0, 1]$. A value closer to 1 indicates a higher likelihood that the state lies within the backward reachable set, while a value closer to 0 indicates otherwise.

5 MPC formulation

The objective, at this point, consists in using the approximated BRS function obtained by training the neural network to constrain the final state computed in a single optimal control problem. As suggested, we used the libraries *CasADi* for the implementation of the MPC framework and *L4casadi* to embed the neural network inside the optimization problem.

The Model Predictive Control (MPC) problem is formulated as:

$$\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & J = \sum_{k=0}^{N-1} \ell(x_k, u_k) \\
\text{subject to:} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N-1, \\
& x_k \in \mathcal{X}, \quad k = 0, 1, \ldots, N, \\
& u_k \in \mathcal{U}, \quad k = 0, 1, \ldots, N-1, \\
& x_0 = x_{\text{init}}, \\
& g(x_N) \geq 0.5
\end{aligned}$$

where:

- N is the time horizon and it was set to 30 for all the simulations.
- $\ell(x_k, u_k)$ is the current cost at time k, which has the position error between the joints and the target configuration as main contribution, while only penalizing very little the joint velocities and torques. The weight used are: $w_{position} = 10^2$, $w_{velocity} = 10^{-8}$, $w_{acceleration} = 10^{-4}$.
- $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are, respectively, the state and control constraint sets, as described in the previous sections. These constraints must be consistent with those used to compute the BRS to ensure meaningful results.
- x_{init} is the real state of the robot at the beginning of each prediction horizon.
- $g(x)$ is the approximated BRS function used here as a final state constraint on x_N. It is important to note that the neural network's output lies between 0 and 1, providing the estimated probability that the state given as input belongs to the backward reachable set. The threshold for this probability is set at 0.5 to account for uncertainties in the model's training process, though this parameter, called `PROB_TRESHOLD` in the code, can be adjusted for different cases.

The MPC formulation described above doesn't contain terminal cost or constraints, except for the approximated BRS function; however, both of them were used during the simulations in order to compare the behavior of the double pendulum when subjected to different types of

constraints, in particular in terms of feasibility, computation time and number of convergence steps.
To reduce the computation time of the program, a *WARM START* is employed in order to use the current *k-th* solution as guess for the subsequent *(k+1)-th* optimal control problem.

6 Discussion of the Results

To evaluate the performance of the classifier, we implemented four different models, each corresponding to an N-step backward reachable set with respectively $N = 25, 50, 100, 200$. We analyzed and compared the behavior of the MPC problem in three different scenarios:

- *Naive implementation*: without any terminal cost or constraint
- *Terminal zero velocity constraint*
- *Backward reachable set terminal constraint*

The state space, control space, and desired configurations outlined in Chapter 2 were utilized for all simulations. These parameters were chosen appositely to ensure the problem was challenging with a time horizon of 30 steps. In fact, the naive implementation turns out to be unfeasible with this prediction interval.

The following table summarizes the key parameters for each simulation. For all the cases in which the BRS terminal constraint was used, the *probability threshold value* was set to 0.5. While modifying this threshold impacts the simulation's behavior, we retained the standard value of 0.5 for these comparisons.

Parameters	**Naive**	**Vel. T.C.**	**B. N=25**	**B. N=50**	**B. N=100**	**B. N=200**
Conv. Time	0.2	0.4	0.23	~ 0.21	0.17	0.16
Comp. Time	0.0303	0.0299	0.0318	0.0320	0.0323	0.0310
Feasibility	X	X	✓	✓	$\sim$	X
Offset	X	X	0.137	~ 0.145	0.099	0.060

Table (1) Data of all the relevant parameters of the different simulations in different conditions

The computation time depends on the machine's performance on which the software is running, of course all the simulations has been done from the same computer to have a truthful comparison.

6.1 Naive

The problem, in its naive configuration, is unfeasible as the pendulum exceeds both joint and torque bounds. Even after increasing the maximum number of solver iterations (`SOLVER_MAX_ITER`), the problem remains unresolved, with persistent constraint violations.

This unfeasibility arises primarily due to two factors:

1. **Insufficient Time Horizon:** When the horizon is too short the solver is not able to adjust the control input to meet the constraints because it realizes too late that an unfeasible region is approaching.

2. **Absence of Terminal Cost or Constraints:** The cost function heavily penalizes position deviations, with significantly lower weights assigned to velocity and acceleration. As a result, the double pendulum aggressively attempts to reach the desired configuration, regardless the velocity and acceleration limits. This behavior leads to violations of joint and torque bounds.

To address these issues, the time horizon of the MPC should be extended, or an appropriate terminal constraint should be introduced. This would compensate for the cost weights and ensure the feasibility of the solution within the given physical limits.

6.2 Terminal Constraint on Velocity

Introducing a terminal constraint on velocity enforces the condition that, at the end of the time horizon, the double pendulum must reach a zero-velocity configuration. This approach has several implications:

1. **Impact on Robot Dynamics:** The terminal constraint slows down the robot's dynamic, as it prioritizes reaching a zero-velocity state at the horizon's end. Consequently, the convergence time increases significantly, from 0.2 seconds in the naive configuration to 0.4 seconds with the terminal constraint.

2. **Recursive Feasibility:** By targeting the zero-velocity configuration, which represents a control-invariant set, recursive feasibility of the MPC is theoretically ensured.

Despite these adjustments, the problem remains unfeasible. This is probably due to the limited torque available at the joints, which is insufficient to counteract the gravitational forces acting on the pendulum during its motion; in fact, the issue remains even after increasing the number of maximum iterations of the solver.

6.3 Terminal Constraint with the Backward Reachable Set N = 25

By incorporating the terminal constraint derived from the classifier (with $N_{BRS} = 25$), the problem becomes feasible. In this configuration, no constraints are violated. The computational time increases slightly compared to the naive configuration, from 0.0303 seconds to 0.0318 seconds. However, a more relevant comparison would be with the simulation using a time horizon of 55 (i.e. $30 + 25$) and a terminal velocity constraint. This setup has a mean computational time of 0.0487 seconds.

One limitation of this approach is the presence of a final offset. Specifically, the simulation does not perfectly reach the desired configuration but instead converges to a nearby state. For instance, the first joint reaches a final configuration that deviates from the desired value by 0.137, a relatively large error. This issue appears to stem from the solver getting trapped in a local minimum introduced by the terminal constraint.

Increasing the maximum number of solver iterations (`SOLVER_MAX_ITER`) to 6 enables the solver to find a better optimum, thereby reaching the desired configuration. However, this adjustment results in the violation of other constraints, highlighting a trade-off between convergence accuracy and constraint satisfaction.

6.4 Terminal Constraint with the Backward Reachable Set N = 50

As with the previous configuration, adding the terminal constraint derived from the backward reachable set with $N_{BRS} = 50$ makes the problem feasible. The convergence time is slightly reduced, decreasing from 0.23 seconds to approximately 0.21 seconds. Here, the symbol $\sim$ indicates that the system does not reach a steady-state solution. Instead, the double pendulum continues oscillating without converging to the desired configuration. The offset error is approximately 0.145, but it oscillates over time rather than remaining constant. This issue can be solved by increasing to 5 the number of maximum Newton's steps. Adjusting this parameter enables the system to converge towards the desired configuration without violating the constraints, despite requiring a higher computation time.

Also in this case to compare the computation time the simulation to consider has a time horizon of 80 $(30 + 50)$ and its mean computation time is about 0.0698, much higher than 0.0320.

6.5 Terminal Constraint with the Backward Reachable Set N = 100

In this configuration, incorporating the terminal constraint derived from the backward reachable set with $N_{BRS} = 100$ results in a significant reduction in convergence time. It decreases from 0.23 and 0.21 seconds (observed in the previous two simulations) to 0.17 seconds.

Regarding feasibility, the problem encounters a minor torque boundary violation at a single step, with the magnitude of the violation being negligible. Since this does not involve a joint bound violation and the torque violation is minimal, the solution can be considered approximately feasible.

To provide a meaningful comparison of computation times, the relevant simulation to consider is one with a time horizon of 130 (i.e., $30 + 100$). The mean computation time for such a simulation is approximately 0.0950 seconds, which is significantly higher than the 0.0323 seconds observed in the current configuration.

6.6 Terminal Constraint with the Backward Reachable Set N = 200

In this simulation, the problem becomes unfeasible, with torque, velocity, and joint limits being exceeded. This indicates that the terminal constraint is no longer effective and is not sufficient to ensure feasibility.

To address this issue, the probability threshold parameter (`PROB_TRESHOLD`) must be increased from 0.5 to 0.9. This adjustment makes the problem almost feasible, with only a minor torque violation occurring at a single step. Moreover, the system's behavior becomes more acceptable under these conditions.

The convergence time is further reduced to 0.16 seconds, representing an improvement over the previous simulations.

For a meaningful comparison of computation times, the relevant simulation to consider is one with a time horizon of 230 (i.e., 30+200). The mean computation time for such a simulation is approximately 0.139 seconds, which is significantly higher than the 0.0310 seconds observed in the current configuration.

6.7 Final Considerations

Analyzing all simulations reveals several key insights into the behavior and feasibility of the MPC under different types of constraints:

1. **Effectiveness of Terminal Constraints:** The introduction of the terminal constraint based on the backward reachable set, generally improves the feasibility of the MPC problem. Theoretically its effectiveness, shouldn't depend on the size of the BRS prediction interval; however, as noted during the simulation, it appears that, without any additional adjustment, the behavior of the robot get worse as N_{BRS} increases, resulting in undesired trajectories. For practical applications N_{BRS} must be properly tuned in order to get a correct behavior of the system.

2. **Trade-offs Between Accuracy and Feasibility:** Increasing the maximum iteration limit enables the solver to find better local minima and get closer to the desired final configuration. However, this often results in other constraint violations, such as exceeding torque or velocity limit. Usually the `SOLVER_MAX_ITER` value is kept as low as possible to make the problem compatible with real time applications. In this case the system is quite simple so with a proper hardware we could increase the number of iterations keeping the computation time low, but generally this not always can be done.

3. **Computational Efficiency:** Simulations utilizing shorter horizons with terminal backward reachable set (BRS) constraints demonstrate significantly faster computation times compared to those with extended horizons and explicit terminal velocity constraints. A notable advantage of this approach is that the computation time remains relatively constant across simulations, regardless of the length of the BRS prediction interval.

4. **Convergence Time:** Unlike the terminal zero-velocity constraint, the BRS constraint permits non-stationary terminal states. As N_{BRS} increases, the constraint is considered "farther" in the optimal control problem, enabling faster motion. This behavior was observed in the simulation, resulting in a significant reduction in convergence time when N_{BRS} increased from 50 to 100. However, the reduction in convergence time was less pronounced when N_{BRS} increased further from 100 to 200, likely due to the solution being now limited by the physical velocity constraints of the system.

5. **Offset Issues:** The presence of final configuration offsets in several simulations suggests that the BRS constraint introduces some undesired local minima. While adjusting solver parameters can mitigate this, achieving consistent convergence to the desired configuration without constraint violations requires further refinement of the MPC formulation, such as increasing the maximum allowable torque.

In conclusion, while terminal constraints derived from backward reachable sets demonstrate promising improvements in efficiency and feasibility, several challenges remain, particularly in tuning solver parameters and selecting appropriate horizon lengths.

6.8 Future Improvements

Future work could focus on adaptive BRS tuning techniques, where the size of the BRS prediction interval is dynamically adjusted based on system feedback, ensuring better trade-offs between computational efficiency and convergence accuracy. Additionally, implementing advanced cost functions that better balance position, velocity, and control effort constraints could further improve performance.

7 Images

7.1 Naive Simulation

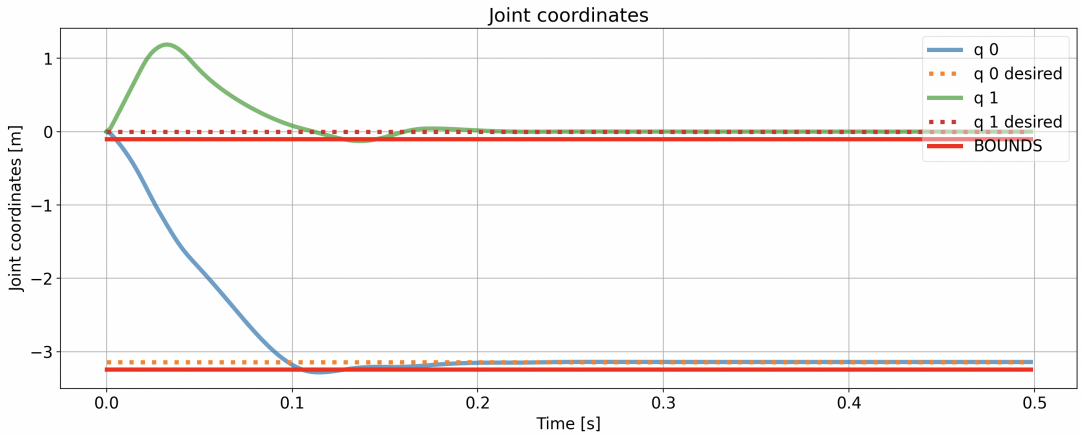


Figure (2) Joint coordinate trajectories

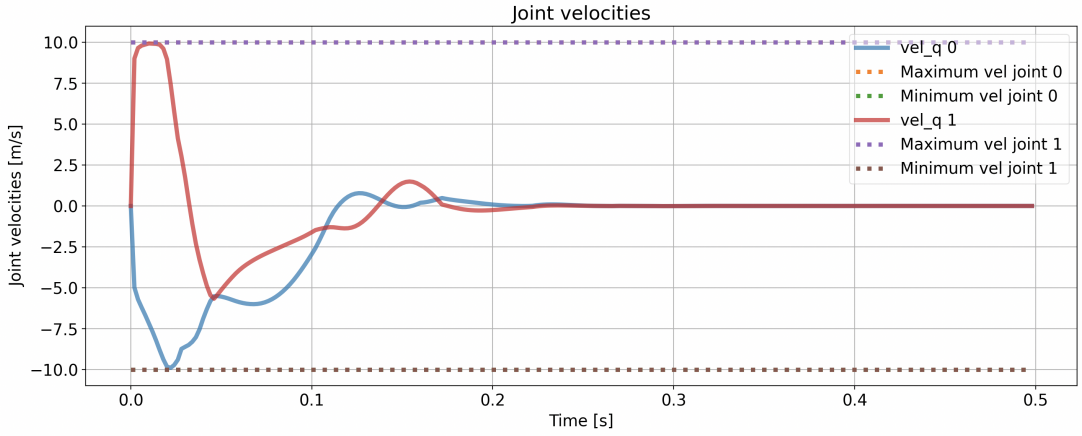


Figure (3) Joint velocities

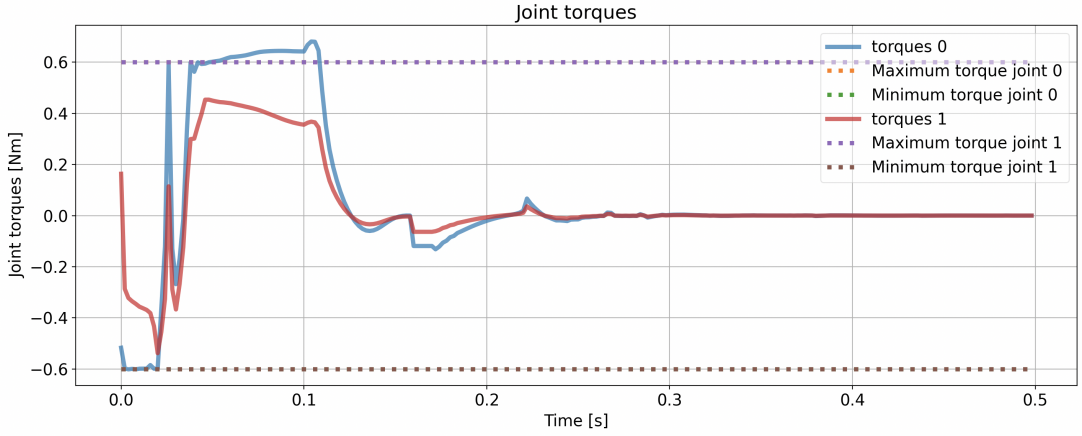


Figure (4) Joint torques

7.2 Terminal constraint on the velocity

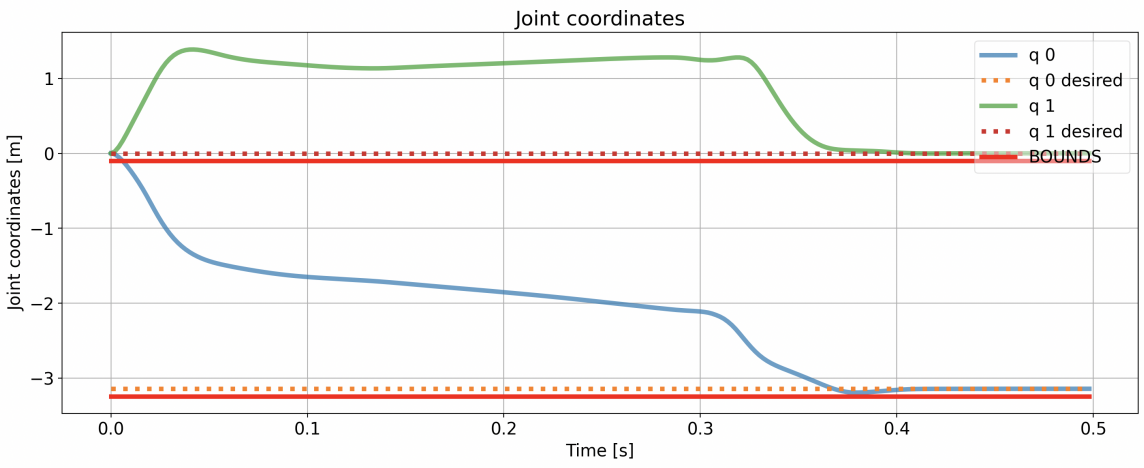


Figure (5) Joint coordinate trajectories

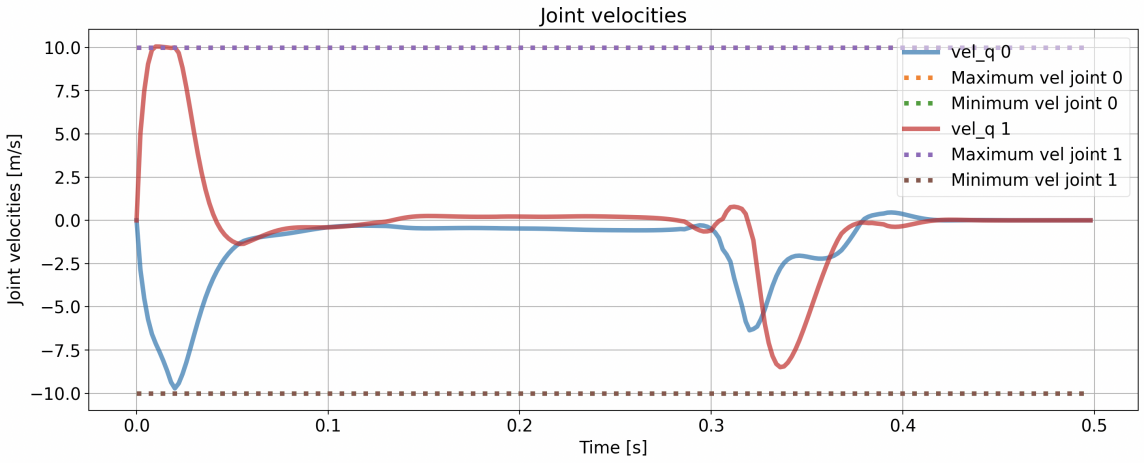


Figure (6) Joint velocities

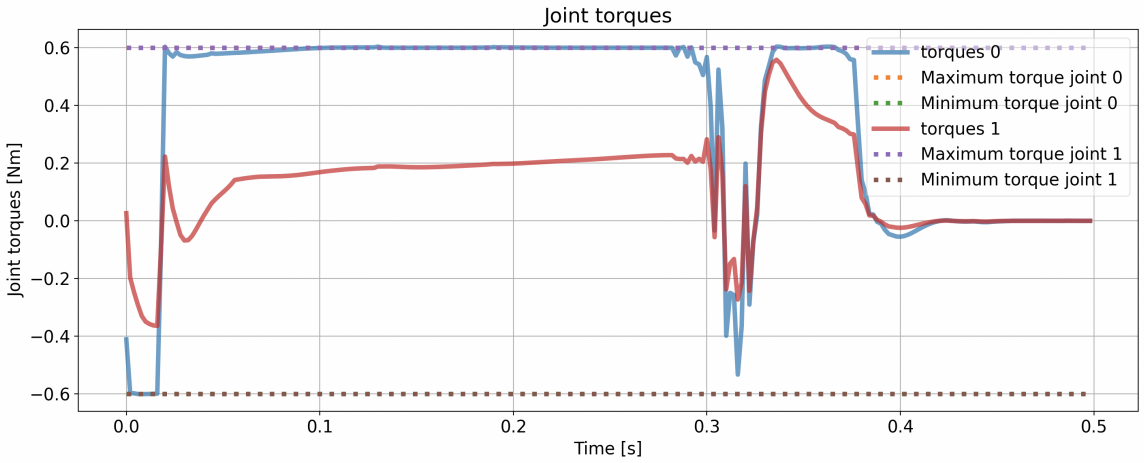


Figure (7) Joint torques

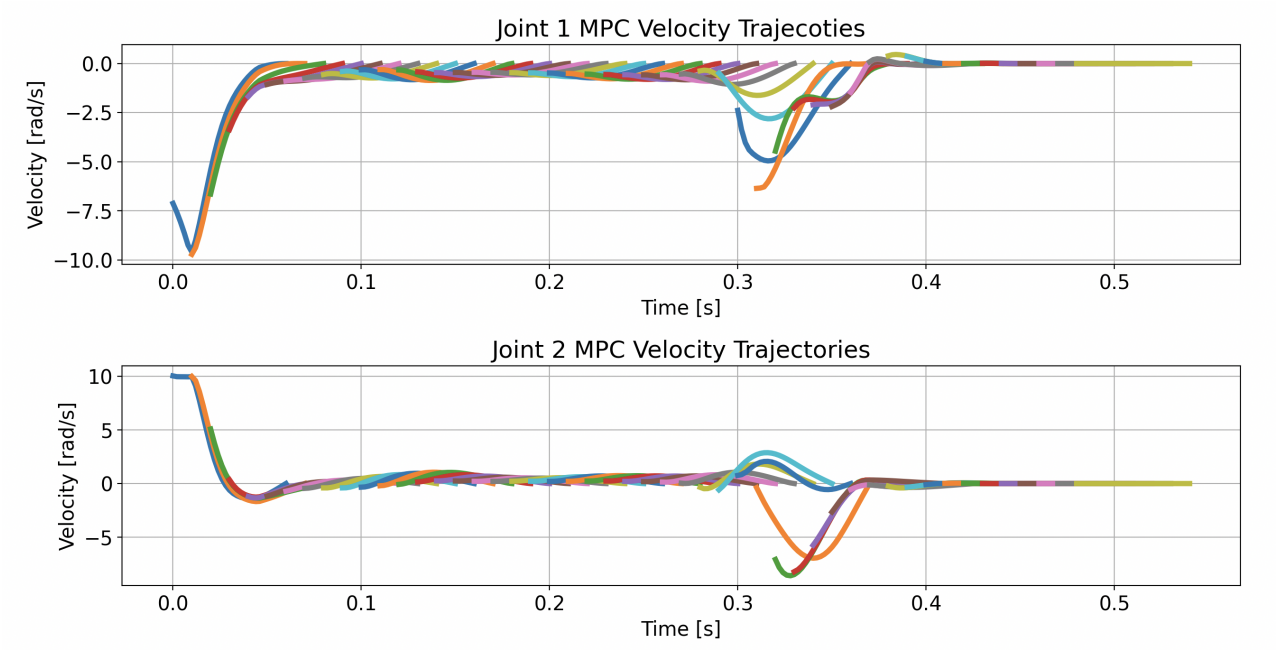


Figure (8) Velocity trajectories of the MPC every 5 step

7.3 Terminal Constraint with the Backward Reachable Set N = 25

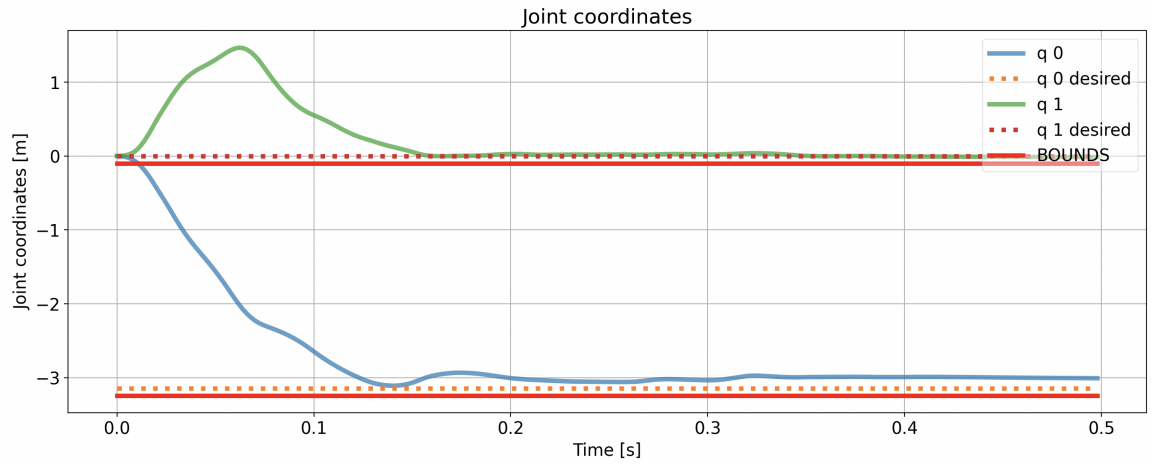


Figure (9) Joint coordinate trajectories

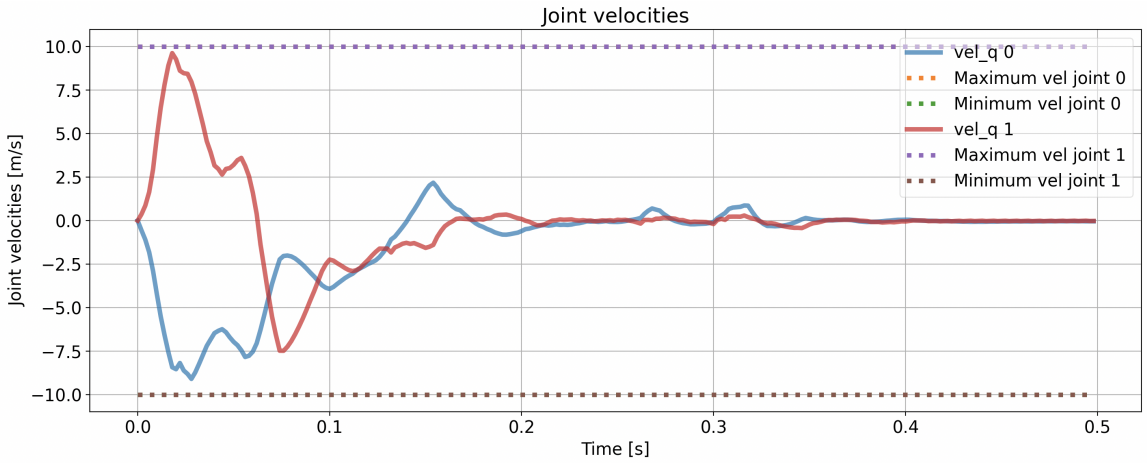


Figure (10) Joint velocities

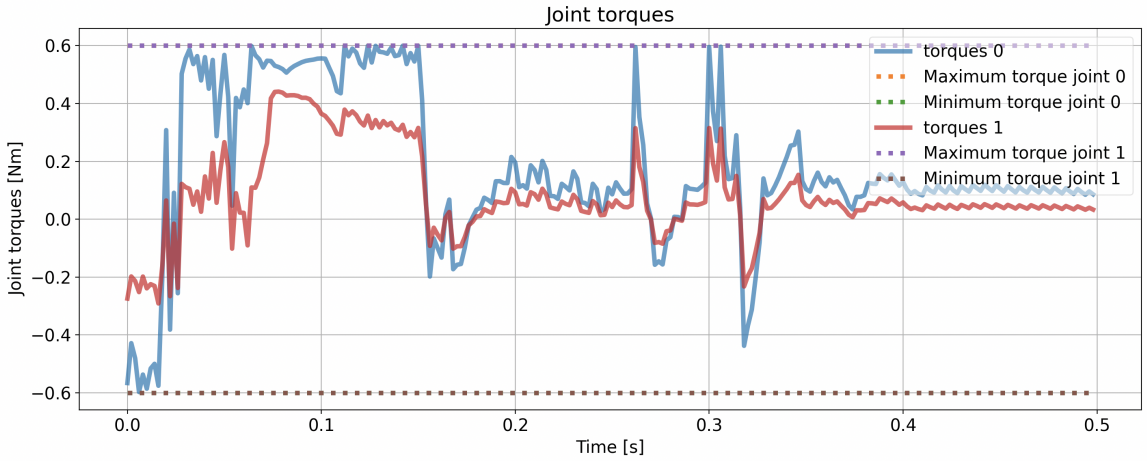


Figure (11) Joint torques

7.4 Terminal Constraint with the Backward Reachable Set N = 50

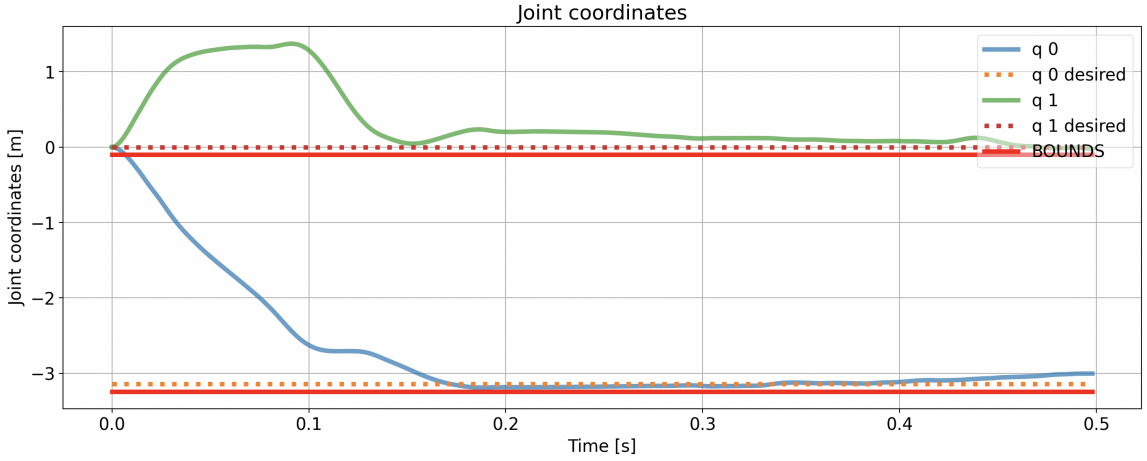


Figure (12) Joint coordinate trajectories

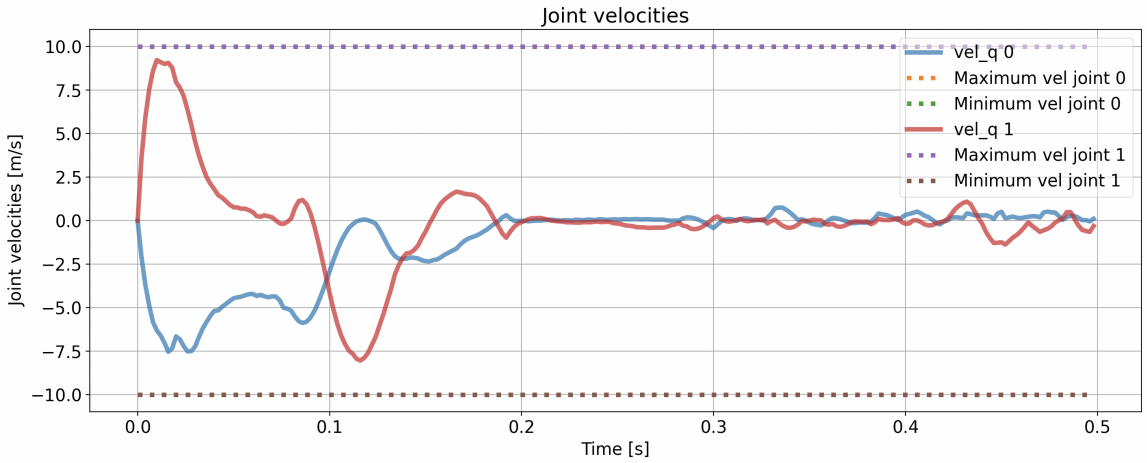


Figure (13) Joint velocities

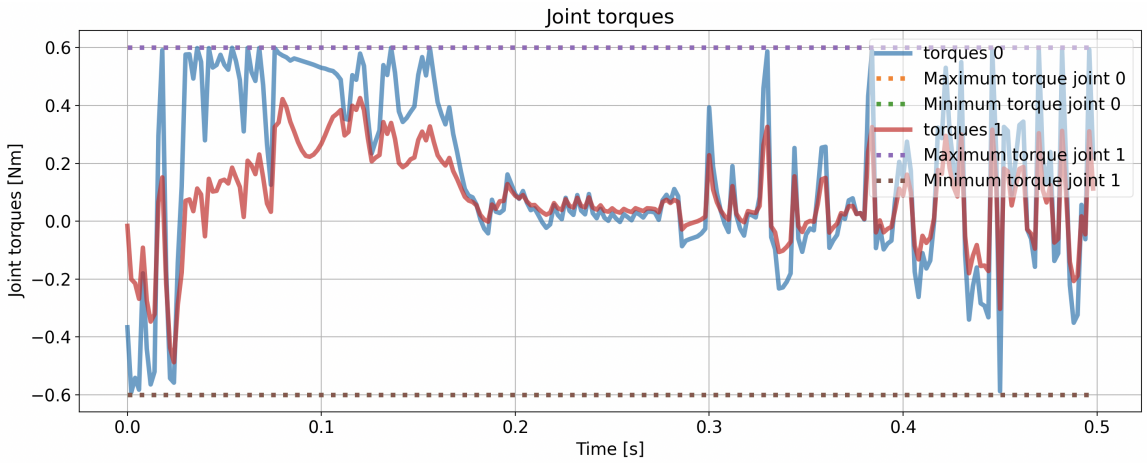


Figure (14) Joint torques

7.5 Terminal Constraint with the Backward Reachable Set N = 100

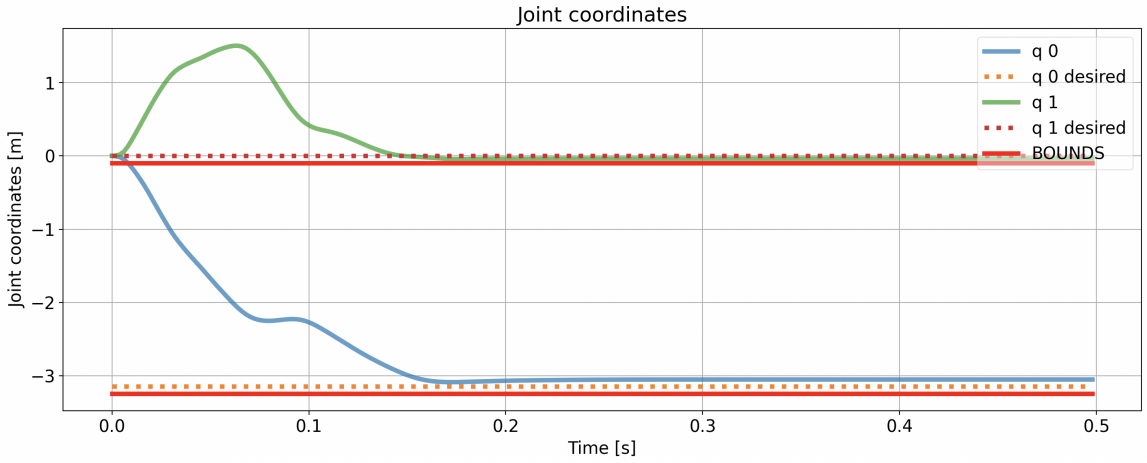


Figure (15) Joint coordinate trajectories

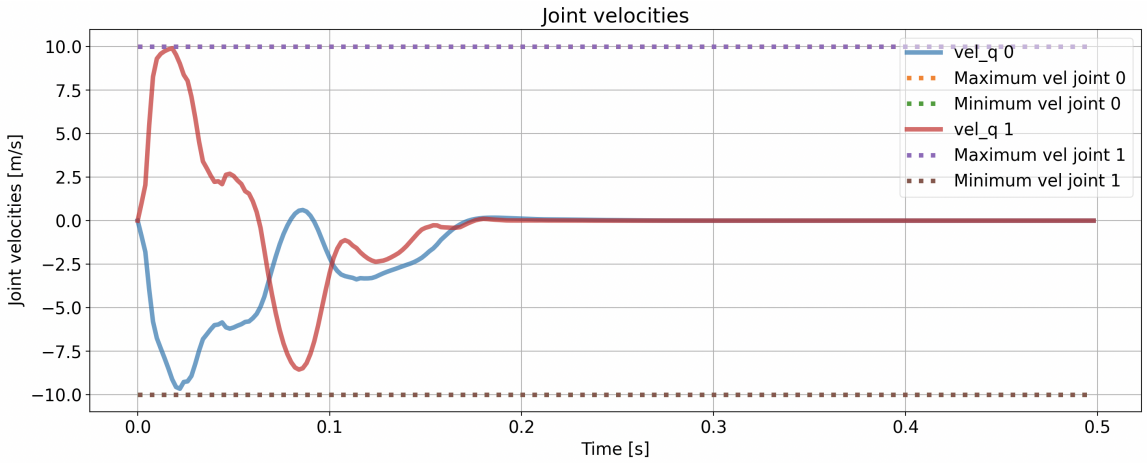


Figure (16) Joint velocities

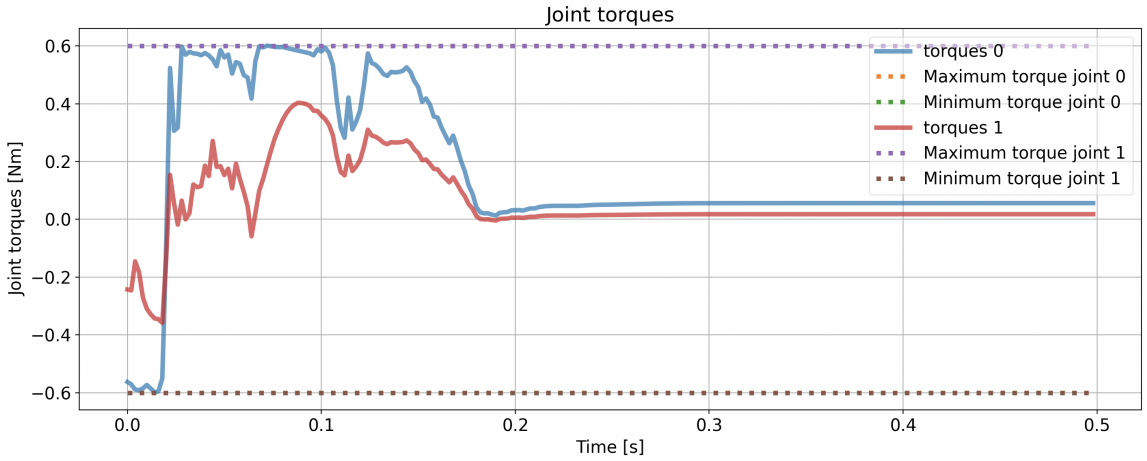


Figure (17) Joint torques

7.6 Terminal Constraint with the Backward Reachable Set N = 200

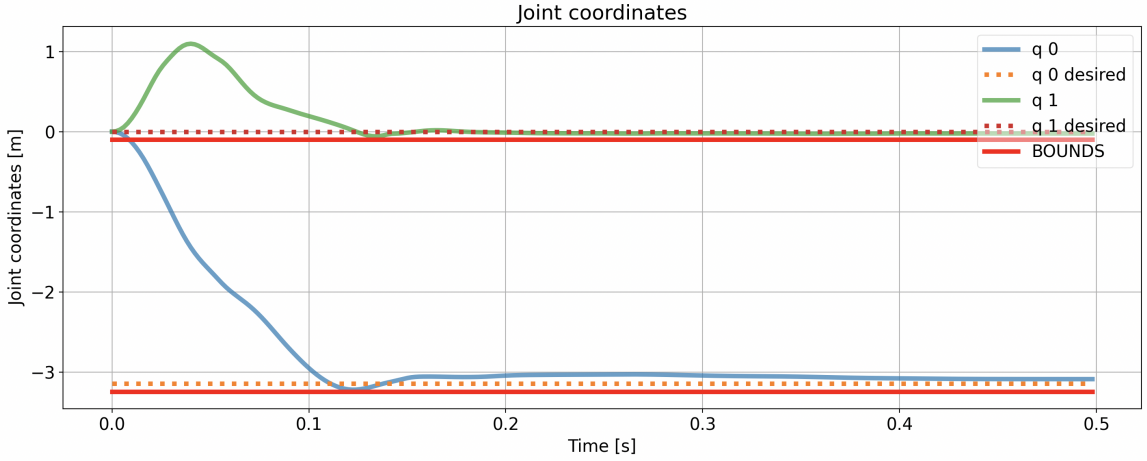


Figure (18) Joint coordinate trajectories

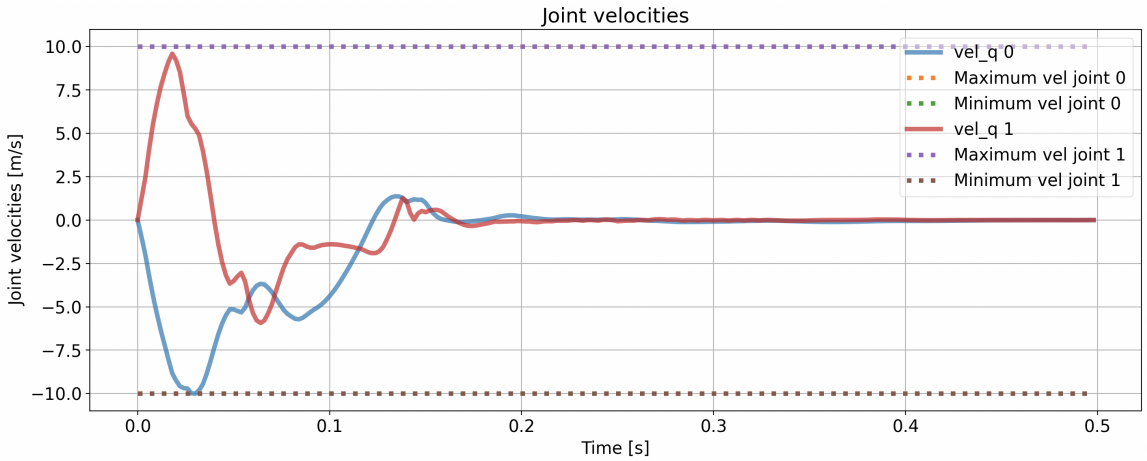


Figure (19) Joint velocities

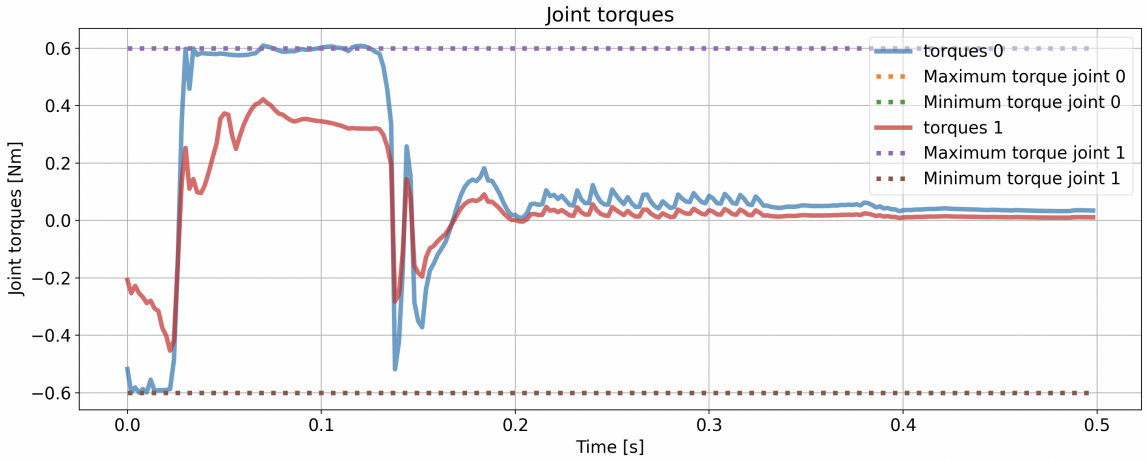


Figure (20) Joint torques