Iterative Model Predictive Control for Piecewise Systems

Ugo Rosolia[ID] and Aaron D. Ames[ID], *Fellow, IEEE*

Abstract—In this letter, we present an iterative Model Predictive Control (MPC) design for piecewise nonlinear systems. We consider finite time control tasks where the goal of the controller is to steer the system from a starting configuration to a goal state while minimizing a cost function. First, we present an algorithm that leverages a feasible trajectory that completes the task to construct a control policy which guarantees that state and input constraints are recursively satisfied and that the closed-loop system reaches the goal state in finite time. Utilizing this construction, we present a policy iteration scheme that iteratively generates safe trajectories which have non-decreasing performance. Finally, we test the proposed strategy on a discretized Spring Loaded Inverted Pendulum (SLIP) model with massless legs. We show that our methodology is robust to changes in initial conditions and disturbances acting on the system. Furthermore, we demonstrate the effectiveness of our policy iteration algorithm in a minimum time control task.

Index Terms—Iterative learning control, predictive control for nonlinear systems, hybrid systems.

I. INTRODUCTION

ROBOTS performing complex tasks can be described as hybrid systems, which are characterized by continuous dynamics and discrete events. Therefore, controllers designed for such systems can take control actions based on continuous and discrete decision variables. Yet the presence of discrete variables make planning and control problems challenging, as it is required to reason about all possible combinations of discrete events. This challenge can be mitigated by designing hierarchical strategies, where a high-level planner computes the discrete variables and a low-level controller optimizes the system trajectory described by continuous variables [1]–[4].

A popular methodology to synthesize policies, which can jointly plan over discrete and continuous states is Model Predictive Control (MPC) [5]–[8]. MPC is a control strategy which systematically uses forecast to compute control actions. At each time step, an MPC plans a trajectory over a short time window, then the first control action is applied to the system and the process is repeated at the next time step based on new measurements. When the system dynamics are hybrid, the MPC planning problem is a Mixed Integer Program (MIP) that is hard to solve online with limited computational resources. For this reason, significant work has focused on explicit MPC strategies where the solution to the MIP is solved offline as a parametric optimization problem [7]–[9]. For hybrid systems described by piecewise affine dynamics the parametric optimization problem can be solved exactly. Once the solution is computed offline, the MPC policy is given by a look-up table of feedback gains that can be efficiently implemented online in real-time [10], [11]. However, computing the explicit solution to hybrid MPC problems is computationally demanding.

Another strategy to speed-up the computation of the MPC policy is to leverage warm-starting strategies, where the optimization algorithm is initialized using a candidate solution. Several strategies have been proposed for warm-starting hybrid MPC problems [12]–[15]. These approaches leverage the trajectory computed at the previous time step to warm-start both the continuous and discrete variables. As the complexity of solving MIPs is given by the computation of the optimal integer variables, recent works have investigated the possibility of leveraging learning algorithms to predict the set of active discrete variables used to warm-start the MPC [16], [17].

In this letter, we focus on control tasks where the goal is to steer the system from a starting configuration to a goal state in finite time, while satisfying state and input constraints. We assume that a feasible trajectory that is able to perform the task is available. Then, we synthesize a control policy, which plans the system trajectory over a finite horizon that is shorter than the control task duration and may cause the controller to take unsafe shortsighted control actions. Thus, building upon [18], [19], we present a methodology to construct the MPC terminal components in order to guarantee satisfaction of the safety constraints and convergence in finite time of the closed-loop system to the goal set.

Compared to previous works [18], [19], we show how to handle piecewise systems by warm-starting the integer variables and we present a shrinking horizon strategy tailored to finite time control tasks. We present an algorithm which solves at most M smooth optimization problems and is guaranteed to find a feasible solution to the original MIP planning problem. Our approach is based on a suboptimal trajectory that can complete the task and affects the closed-loop performance of the controller. Therefore, we present a policy iteration algorithm, where simulations are used to iteratively update the controller. We prove that our algorithm returns safe policies that have non-decreasing performance. Finally, we demonstrate the effectiveness of our

approach on the discretized Spring Loaded Inverted Pendulum (SLIP) [20].

II. PROBLEM FORMULATION

In this section, we describe the system model and the control synthesis objectives. We consider discrete time piecewise nonlinear systems defined over R disjoint regions $\mathcal{D}_i \subseteq \mathbb{R}^n$ for $i \in \{1, \ldots, R\}$:

$$x_{t+1} = f_i(x_t, u_t), \quad \text{if } x_t \in \mathcal{D}_i, \tag{1}$$

where the state $x_t \in \mathbb{R}^n$ and the input $u_t \in \mathbb{R}^d$. In the above equation $f_i : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ represents the system dynamics, which describe the evolution of the discrete time system when the state x_t belongs to the region $\mathcal{D}_i \subseteq \mathbb{R}^n$. Furthermore, the system is subject to the following state and input constraints:

$$u_t \in \mathcal{U} \subseteq \mathbb{R}^d \text{ and } x_t \in \mathcal{X} \subseteq \mathbb{R}^n, \ \forall t \in \{0, \ldots, T-1\}, \tag{2}$$

where $T \in \{0, 1, \ldots\}$ is the duration of the control task. Notice that several robotic systems can be described by piecewise constrained nonlinear models defined over disjoint regions, such as the SLIP model presented in Section V-A.

Objective: Our goal is to design a control policy $\pi : \mathbb{R}^n \to \mathbb{R}^d$ which maps states to actions, i.e.,

$$u_t = \pi(x_t). \tag{3}$$

The above control policy should guarantee that the state and input constraints from (2) are satisfied and that the closed-loop system (1) and (3) converges in finite time to a goal set $\mathcal{G} \subset \mathcal{X}$. More formally, the control policy (3) should guarantee that for an initial condition x_I in a neighborhood of a starting state $x_S \in \mathcal{X}$, the trajectory of the closed-loop system (1) and (3) is a feasible solution to the following *Finite Time Optimal Control Problem (FTOCP)*:

$$\min_{\substack{u_0, \ldots, u_{T-1} \\ i_0, \ldots, i_{T-1}}} \sum_{t=0}^{T-1} l(x_t, u_t)$$

$$\begin{aligned}
\text{s.t.} \quad & x_{t+1} = f_{i_t}(x_t, u_t), \\
& x_t \in \mathcal{D}_{i_t} \cap \mathcal{X}, \ u_t \in \mathcal{U}, \ i_t \in \{1, \ldots, R\}, \\
& x_0 = x_I, x_T \in \mathcal{G}, \\
& \forall t \in \{0, \ldots, T-1\},
\end{aligned} \tag{4}$$

where the stage cost $l : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$. Notice that in the above problem the system dynamics are a function of the integer variables $i_t \in \{1, \ldots, R\}$. Therefore, for a feasible set of continuous inputs $[u_0, \ldots, u_{T-1}]$ and integer variables $[i_0, \ldots, i_{T-1}]$, we have that the resulting vector of states $[x_0, \ldots, x_T]$ must satisfy $x_t \in \mathcal{D}_{i_t} \cap \mathcal{X}, \ \forall t \in \{0, \ldots, T-1\}$. Throughout this letter we make the following assumptions.

Assumption 1: We are given the state-input trajectories which are feasible for the FTOCP (4) with $x_I = x_S$:

$$\mathbf{x}^0 = [x_0^0, \ldots, x_T^0] \text{ and } \mathbf{u}^0 = [u_0^0, \ldots, u_{T-1}^0],$$

where for all $t \in \{0, \ldots, T-1\}$ the state $x_t^0 \in \mathcal{X}$, the input $u_t^0 \in \mathcal{U}$ and $x_T^0 \in \mathcal{G}$.

Assumption 2: For any $x \in \mathcal{G}$, the stage cost $l(x, u) = 0$, for all $u \in \mathcal{U}$. Moreover, the set $\mathcal{G}$ is control invariant, i.e., for all $x \in \mathcal{G}$, there exists a control $u \in \mathcal{U}$ and index $i \in \{1, \ldots, R\}$ such that $f_i(x, u) \in \mathcal{G}$ and $x \in \mathcal{D}_i$.

Remark 1: The proposed methodology requires only feasibility of a trajectory $\mathbf{x}^0$. However, the optimality of the

trajectory $\mathbf{x}^0$ affects the performance of the proposed control synthesis strategy. For this reason, in Section III-C we present a policy iteration scheme that may be used to iteratively improve the closed-loop performance of the policy.

III. CONTROL DESIGN

In this section, we first introduce an FTOCP which can be recast as a non-linear program (NLP). Afterwards, we present the proposed strategy which leverages this NLP and the feasible trajectory from Assumption 1. Finally, we present a policy iteration scheme which can be used to improve the performance of the control policy.

A. Model Predictive Control

The FTOCP (4) is challenging to solve as at each time t the system dynamics change as a function of the state x_t, i.e., $x_{t+1} = f_i(x_t, u_t)$ if $x_t \in \mathcal{D}_i$. However, the computational complexity may be reduced when searching for a feasible sub-optimal solution.[1] In particular, a feasible solution to the FTOCP (4) may be computed fixing a priori a sequence of regions $\{\mathcal{D}_{i_0}, \ldots, \mathcal{D}_{i_{T-1}}\}$, where the system should be constrained at each time t. Clearly, a trajectory which steers the system from the starting state x_I to the goal set $\mathcal{G}$, while visiting the sequence of regions $\{\mathcal{D}_{i_0}, \ldots, \mathcal{D}_{i_{T-1}}\}$ is a feasible trajectory for the original FTOCP (4).

In order to reduce the computational complexity, we introduce an FTOCP defined over a horizon N shorter than T and for a set of indices $\mathcal{I}_t = \{i_t, \ldots, i_{t+N-1}\}$ associated with a sequence of N regions $\{\mathcal{D}_{i_t}, \ldots, \mathcal{D}_{i_{t+N-1}}\}$. In particular, given a set of indices $\mathcal{I}_t$, the terminal state x_F and an associated terminal cost $q_F \in \mathbb{R}$ we define the FTOCP:

$$J(x_t, x_F, q_F, \mathcal{I}_t, N) = \min_{\mathbf{u}_t} \sum_{k=t}^{t+N-1} l(x_{k|t}, u_{k|t}) + q_F$$

$$\begin{aligned}
\text{s.t.} \quad & x_{k+1|t} = f_{i_k}(x_{k|t}, u_{k|t}), \\
& x_{k|t} \in \mathcal{D}_{i_k} \cap \mathcal{X}, \ u_{k|t} \in \mathcal{U}, \\
& x_{t|t} = x_t, x_{t+N|t} = x_F, \\
& \forall k \in \{t, \ldots, t+N-1\},
\end{aligned} \tag{5}$$

where $\mathbf{u}_t = [u_{t|t}, \ldots, u_{t+N-1|t}]$ The optimal state-input sequences to the above FTOCP

$$[x_{t|t}^*, \ldots, x_{t+N|t}^*] \text{ and } [u_{t|t}^*, \ldots, u_{t+N-1|t}^*], \tag{6}$$

steer the system from the starting state x_t to the terminal state x_F while satisfying state and input constraints.

In the FTOCP (5), at each predicted time k the system state $x_{k|t} \in \mathcal{D}_{i_k} \cap \mathcal{X}$, and therefore problem (5) can be recast as an NLP, which is easier to solve than problem (4) where the optimization is carried out over continuous and integer variables. Next, we present the proposed algorithm which chooses the set $\mathcal{I}_t = \{i_t, \ldots, i_{t+N-1}\}$ associated with the sequence of regions $\{\mathcal{D}_{i_t}, \ldots, \mathcal{D}_{i_{t+N-1}}\}$, the terminal state x_F, and terminal cost q_F that are used in the FTOCP (5).

B. Policy Synthesis

This section describes the proposed strategy. For each state x_t^0 of the feasible trajectory $\mathbf{x}^0$ from Assumption 1, we define

---

[1]Notice that the computational cost may be reduced also using heuristics as discussed in [17], [18].

Algorithm 1 Control Policy π

1: **Init Parameters:** $\mathbf{q}^0$, $\mathbf{x}^0$, $\mathbf{i}^0$, M, $k_0 = N$, $N_0 = N$, T
2: **Input:** x_t
3: $\quad i_t = \texttt{getRegion}(x_t)$
4: **for** $m = [0, \ldots, M-1]$ **do** $\qquad \triangleright$ Solve M FTOCPs
5: $\quad$ set $t_F = \min(k_t + m, T)$
6: $\quad$ set $x_F = x_{t_F}^0$
7: $\quad$ set $q_F = q_{t_F}^0$
8: $\quad$ set $\mathcal{I}_t = \{i_t, i_{t_F-N_t+1}^0, \ldots, i_{t_F-1}^0\}$
9: $\quad$ solve the FTOCP $J(x_t, x_F, q_F, \mathcal{I}_t, N_t)$ from (5)
10: $\quad$ store $c_m = J(x_t, x_F, q_F, \mathcal{I}_t, N_t)$
11: $\quad$ store $\bar{u}_m = u_{t|t}^*$
12: $\quad$ store $\bar{x}_{F,m} = x_{t+N_t|t}^*$
13: $\quad$ **if** $m > 0$ and $c_{m-1} < c_m$ **then**
14: $\qquad$ $c_t^* = c_{m-1}$ $\qquad\qquad\qquad \triangleright$ Pick best cost
15: $\qquad$ $m_t^* = m - 1$ $\qquad\qquad \triangleright$ Pick best cost index
16: $\qquad$ **break**
17: $\quad$ **end if**
18: **end for**
19: **if** $\bar{x}_{F,m_t^*} = x_T^0$ **then** $\triangleright$ Horizon and Parameter Update
20: $\quad$ store $N_{t+1} = \max(1, N_t - 1)$
21: $\quad$ store $k_{t+1} = T$
22: **else**
23: $\quad$ store $N_{t+1} = N_t$
24: $\quad$ store $k_{t+1} = k_t + m_t^* + 1$
25: **end if**
26: $u_t = \bar{u}_{m_t^*}$
27: **Outputs** c_t^*, u_t

Algorithm 2 Iterative Policy Update

1: **Init Parameters:** $\mathbf{q}^0$, $\mathbf{x}^0$, $\mathbf{i}^0$, M, N, x_S, T
2: **Input:** j
3: define π^1 via Algorithm 1 initialized with $\mathbf{q}^0$, $\mathbf{x}^0$, $\mathbf{i}^0$,
$\qquad M$, $k_0 = N$, $N_0 = N$, T
4: **for** $i \in \{1, \ldots, j\}$ **do** $\qquad \triangleright$ Policy iteration loop
5: $\quad$ simulate the closed-loop system (8) from $x_0^i = x_S$
6: $\quad$ set $\mathbf{x}^i = [x_0^i, \ldots, x_T^i]$
7: $\quad$ compute the region indices $\mathbf{i}^i$ from $\mathbf{x}^i$
8: $\quad$ compute $\mathbf{q}^i$ from (7) with $q_T^i = 0$
9: $\quad$ define π^{i+1} via Algorithm 1 initialized with $\mathbf{q}^i$, $\mathbf{x}^i$, $\mathbf{i}^i$,
$\qquad M$, $k_0 = N$, $N_0 = N$
10: **end for**
11: **Outputs** $(\pi^0, \mathbf{x}^0, \mathbf{q}^0), \cdots, (\pi^{j+1}, \mathbf{x}^{j+1}, \mathbf{q}^{j+1})$

the vector

$$\mathbf{i}^0 = [i_0^0, \ldots, i_T^0],$$

where $i_t^0 \in \{1, \ldots, R\}$ identifies the region containing the system's state at time t, i.e., $x_t^0 \in \mathcal{D}_{i_t^0}$. Moreover, for each state x_t^0 we introduce the cost-to-go q_t^0 given by the recursion:

$$q_t^0 = l(x_k^0, u_k^0) + q_{t+1}^0, \tag{7}$$

for $q_T^0 = 0$ and the stage cost $l: \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ from (4). Finally, we define the cost vector

$$\mathbf{q}^0 = [q_0^0, \ldots, q_T^0].$$

The cost vector $\mathbf{q}^0$, the feasible trajectory $\mathbf{x}^0$ and the vector of indices $\mathbf{i}^0$, together with the parameters $M \in \{0, 1, 2, \ldots\}$ and $N \in \{0, 1, \ldots\}$, are used to initialize the proposed control policy, which is described by Algorithm 1. At each time t, Algorithm 1 takes as input the state of the system x_t and it returns the best cost and the control action u_t, which is applied to system (1). First, given the state x_t we identify the region's index i_t such that $x_t \in \mathcal{D}_{i_t}$ (line 1). Afterwards, we solve M times the FTOCP (5) with a different terminal state $x_F = x_{t_F}^0$, terminal cost $q_F = q_{t_F}^0$ and set of indices $\mathcal{I}_t$ (lines 4-18). Note that the set of indices $\mathcal{I}_t$ (line 8) is computed appending to the current region's index i_t a sequence of $N_t - 1$ indices selected backward from time t_F, which is chosen independently of N_t for $t > 0$. As a result, at each time t the controller solves the FTOCP (4) using a sequence of indices which is different from the one associated with the first feasible trajectory.[2] As we will

discuss in Section IV at time t and for $m = 0$, the FTOCP (5) is feasible when the terminal components are defined as in lines 5-7. Therefore if for $m > 0$ we have that $c_{m-1} < c_m$, we stop solving the set of M NLPs and we save the best cost and the associated index m_t^*. Then, we update the parameter k_t that is used to define the terminal MPC components and we shrink the prediction horizon, if the terminal predicted state $\bar{x}_{F,m_t^*}$ equals the terminal state x_T^0. Finally, we return the optimal control action $u_t = \bar{u}_{m_t^*}$.

C. Policy Iteration

The control policy from Algorithm 1 leverages the feasible trajectory $\mathbf{x}^0$ to compute the terminal components used in the MPC problem (5) and, as a result, the performance of the proposed methodology is affected by the optimality of $\mathbf{x}^0$. In this section, we discuss a policy iteration strategy to improve the performance of the policy from Algorithm 1.

At iteration $j \geq 1$, we define the control policy $\pi^j: \mathbb{R}^n \to \mathbb{R}^d$ which is given by Algorithm 1 initialized using the feasible trajectory $\mathbf{x}^{j-1}$, the vector of indices $\mathbf{i}^{j-1}$, and the cost vector $\mathbf{q}^{j-1}$. For an initial condition $x_0^j = x_S$, the policy π^j may be used to simulate the trajectory of the closed-loop system:

$$x_{t+1}^j = f_i(x_t^j, \pi^j(x_t^j)), \quad \text{if } x_t^j \in \mathcal{D}_i, \tag{8}$$

which is then used to update the policy π^{j+1} at the next iteration $j + 1$, as shown in Algorithm 2.

IV. PROPERTIES

A. Recursive Feasibility and Finite-Time Convergence

We show that at each time t Algorithm 1 in closed-loop with system (1) guarantees constraint satisfaction and that the closed-loop system converges in finite time[3] to the set $\mathcal{G}$.

Theorem 1: Consider the closed-loop system (1) and (3), where the policy π is given by Algorithm 1. Let Assumptions 1-2 hold. If at time $t = 0$ the initial condition $x_0 = x_S$, then the closed-loop system (1) and (3) satisfies state and input constraints from (2) and it reaches the terminal set $\mathcal{G}$ at time T, i.e., $x_t \in \mathcal{X}$, $u_t \in \mathcal{U}$, $\forall t \in \{0, \ldots, T-1\}$ and $x_T \in \mathcal{G}$.

Proof: We show that the time-varying terminal set from Algorithm 1 is a control invariant, thus feasibility follows

---

[2]For example, at time $t = 0$, for $m = 2$, $k_0 = N = 4$ and $T = 100$, we have that $t_F = k_0 + m = 6$; therefore $\mathcal{I}_0 = \{i_0, i_3^0, i_4^0, i_5^0\}$ and $x_F = x_6^0$.

[3]Note that finite-time convergence was shown also for the strategy presented in [21]. However, compared to our approach a stronger assumption on the cost function is required to guarantee finite-time convergence.

from standard MPC arguments [8], [22]. Assume that at time t Algorithm 1 returns a feasible control action u_t and let

$$[x^*_{t|t}, \ldots, x^*_{t+N_t|t} = x_F] \text{ and } [u^*_{t|t}, \ldots, u^*_{t+N_t-1|t}], \quad (9)$$

be the optimal solution associated with the m^*_tth FTOCP (5) solved at time t. Next, we consider three cases to show that the time-varying components defining the FTOCPs solved at line 9 of Algorithm 1 guarantee that Algorithm 1 returns a feasible control action u_{t+1} at the next time step $t+1$:

Case 1: If $\bar{x}_{F,m^*_t} = x^0_T$ and the horizon $N_t = 1$, then $k_{t+1} = T$, $N_{t+1} = 1$ and $x_{t+1} = x^0_T$, therefore by the invariance of $\mathcal{G}$ for $m = 0$ the FTOCP $J(x_{t+1}, x_F, q_F, \mathcal{I}_{t+1}, N_{t+1})$ with $x_F = x_{k_{t+1}} = x^0_T$ is feasible.

Case 2: If $\bar{x}_{F,m^*_t} = x^0_T$ and the horizon $N_t > 1$, then $k_{t+1} = T$ and for $m = 0$ the following state-input sequences $[x^*_{t+1|t}, \ldots, x^*_{t+N_t|t}]$ and $[u^*_{t+1|t}, \ldots, u^*_{t+N_t-1|t}]$ are feasible for the FTOCP $J(x_{t+1}, x_F, q_F, \mathcal{I}_{t+1}, N_{t+1})$ with $x_F = x_{k_{t+1}} = x^0_T$ as $x^*_{t+N_t|t} = x^0_T$ and $N_{t+1} = N_t - 1$.

Case 3: If $\bar{x}_{F,m^*_t} \neq x^0_T$, then $k_{t+1} = k_t + m^*_t + 1$ and for $m = 0$ the state-input sequences $[x^*_{t+1|t}, \ldots, x^*_{t+N_t|t} = x^0_{k_t+m^*_t}, x^0_{k_t+m^*_t+1}]$ and $[u^*_{t+1|t}, \ldots, u^*_{t+N_t-1|t}, u^0_{k_t+m^*_t}]$ are feasible for the FTOCP $J(x_{t+1}, x_F, q_F, \mathcal{I}_{t+1}, N_{k+1})$ with $x_F = x_{k_{t+1}} = x_{k_t+m^*_t+1}$.

From *Cases* 1–3, we have that if at time t Algorithm 1 returns a feasible action u_t, then at time $t+1$ Algorithm 1 returns a feasible control action u_{t+1}. Now, we notice that at time $t = 0$ the sequence of actions $[u^0_0, \ldots, u^0_{N-1}]$ is feasible for the FTOCP $J(x_0, x_F, q_F, \mathcal{I}_0, N_0)$ with $x_F = x^0_{N_0}$, which in turns implies that Algorithm 1 returns a feasible action u_t at all times and that state and input constraints are satisfied.

Finally, we show finite-time convergence. From Algorithm 1, we have that k_t increases at each time step until $k_t = T$ after at most $T - N$ time steps and, afterwards, that the horizon shrinks. Therefore, at time $T - 1$ we have that $N_{T-1} = 1$, $t_F = k_{T-1} = T$ and $x_F = x^0_T$, thus the predicted state at time $T - 1$ of the optimal trajectory from (9) satisfies $x^*_{T|T-1} = x^0_T$, which in turns implies that $x_T = x^*_{T|T-1} = x^0_T \in \mathcal{G}$. ∎

Corollary 1: Consider the closed-loop system (1) and (3), where the policy π is given by Algorithm 1. Let Assumptions 1-2 hold. If at time t Algorithm 1 returns a feasible control action $u_t \in \mathcal{U}$, then the closed-loop system (1) and (3) satisfies constraints (2) and it converges to the goal set $\mathcal{G}$.

The above corollary highlights the advantage of computing a policy that maps states to actions and it can be used to deal with perturbed initial conditions and uncertainties. In the result section we perform an empirical study where we test the robustness of the proposed methodology by changing initial conditions and simulating disturbances acting on the system.

B. Iterative Improvement

This section discusses the properties of the iterative Algorithm 2. In particular, we show that at each policy update the cumulative cost associated with the closed-loop trajectories from (8) is non-increasing. Notice that our strategy guarantees non-increasing cost at each update, but no guarantees are given about the optimality of the trajectory at convergence.

Theorem 2: For $i \in \{0, \ldots, j\}$ consider the closed-loop trajectories $\mathbf{x}^i$ from Algorithm 2. If Assumptions 1-2 hold, then we have that at each policy update the cost associated with the

closed-loop trajectories is non-increasing, i.e., $q^{i-1}_0 \geq q^i_0$, $\forall i \in \{1, \ldots, j\}$, where $q^i_0 = \sum_{t=0}^{T-1} l(x^i_t, u^i_t)$.

Proof: From Theorem 1 it follows that at each jth update the policy π^j from Algorithm 1 returns a feasible action u^j_t and the feasible state-input trajectories $\mathbf{x}^j$ and $\mathbf{u}^j$. At time t, let $c^{*,j}_t$ be the optimal cost of the m^*_tth FTOCP and let $[x^{*,j}_{t|t}, \ldots, x^{*,j}_{t+N_t|t}]$ and $[u^{*,j}_{t|t}, \ldots, u^{*,j}_{t+N_t-1|t}]$ be the optimal solution. Then we write the optimal cost as

$$c^{*,j}_t = \sum_{k=t}^{t+N_t-1} l(x^{*,j}_{k|t}, u^{*,j}_{k|t}) + q^{j-1}_{k_t+m^*_t}$$

$$= l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + \sum_{k=t+1}^{t+N_t-1} l(x^{*,j}_{k|t}, u^{*,j}_{k|t}) + q^{j-1}_{k_t+m^*_t}. \quad (10)$$

Next, we consider three cases to analyze the time evolution of the optimal cost $c^{*,j}_t$ associated with the m^*_tth FTOCP solved at line 9 of Algorithm 1:

Case 1: If $\bar{x}_{F,m^*_t} = x^{j-1}_T$ and the horizon $N_t = 1$, then $k_{t+1} = T$, $N_{t+1} = 1$, $x_{t+1} = x^{j-1}_T$ and the FTOCP for $m = 0$ is feasible (*Case 1* of Theorem 1) which together with Assumption 2 imply that $c^{*,j}_{t+1} = 0$. Thus, we have that

$$c^{*,j}_t = l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + q^{j-1}_{m^*_t} = l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + c^{*,j}_{t+1}, \quad (11)$$

as the terminal cost $q^{j-1}_{m^*_t} = 0 = c^{*,j}_{t+1}$ for $\bar{x}_{F,m^*_t} = x^{j-1}_T$.

Case 2: If $\bar{x}_{F,m^*_t} = x^{j-1}_T$ and the horizon $N_t > 1$, then we have that the last two terms in (10) represent the cost $\bar{c}^j_{t+1}$ associated with the state-input sequences $[x^{*,j}_{t+1|t}, \ldots, x^{*,j}_{t+N_t|t}]$ and $[u^{*,j}_{t+1|t}, \ldots, u^{*,j}_{t+N_t-1|t}]$, which are feasible at time $t+1$ (*Case 2* of Theorem 1) and therefore we have that

$$c^{*,j}_t = l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + \bar{c}^j_{t+1} \geq l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + c^{*,j}_{t+1}. \quad (12)$$

Case 3: If $\bar{x}_{F,m^*_t} \neq x^{j-1}_T$, then by definition (7) we have that the optimal cost can be written as

$$c^{*,j}_t = l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + \sum_{k=t+1}^{t+N_t-1} l(x^{*,j}_{k|t}, u^{*,j}_{k|t}) + q^{j-1}_{k_t+m^*_t}$$

$$= l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + \sum_{k=t+1}^{t+N_t-1} l(x^{*,j}_{k|t}, u^{*,j}_{k|t})$$

$$+ l(x^{j-1}_{k_t+m^*_t}, u^{j-1}_{k_t+m^*_t}) + q^{j-1}_{k_t+m^*_t+1}.$$

Notice that the last three terms in the above equation represent the open-loop cost $\bar{c}^j_{t+1}$ associated with the state-input sequences $[x^{*,j}_{t+1|t}, \ldots, x^{j-1}_{k_t+m^*_t|t}, x^{j-1}_{k_t+m^*_t+1|t}]$ and $[u^{*,j}_{t+1|t}, \ldots, u^{*,j}_{t+N_t-1|t}, u^{j-1}_{k_t+m^*_t|t}]$, which are feasible at time $t+1$ (*Case 3* of Theorem 1) and therefore we have that

$$c^{*,j}_t = l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + \bar{c}^j_{t+1} \geq l(x^{*,j}_{t|t}, u^{*,j}_{t|t}) + c^{*,j}_{t+1}. \quad (13)$$

Finally, we notice that $c^{j,*}_T = 0$ as from Theorem 1 $x^j_T \in \mathcal{G}$ and therefore from equations (11)–(13) we have that

$$c^{*,j}_0 \geq l(x^j_1, u^j_1) + c^*_2 \geq \ldots \geq \sum_{t=0}^{T-1} l(x^j_t, u^j_t) + c^{*,j}_T = q^j_0,$$

as $x^{*,j}_{t|t} = x^j_t$ and $u^{*,j}_{t|t} = u^j_t$ for all $t \in \{0, \ldots, T\}$. Furthermore, at time $t = 0$ and for $m = 0$ the sequence of open-loop actions
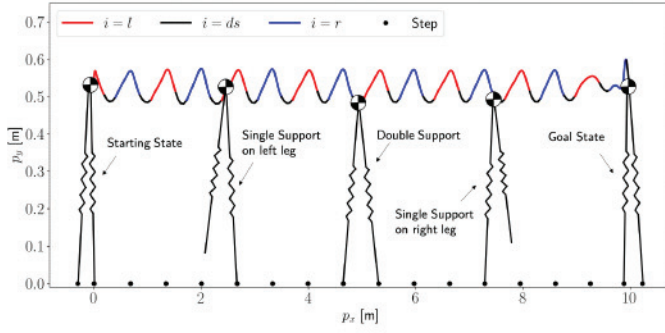
Fig. 1. Trajectory of the CoM and foot step locations associated with the feasible trajectory $\mathbf{x}^0$ used to initialize the control policy from Algorithm 1.

Fig. 2. Closed-loop trajectories of the CoM and foot steps for different initial conditions.

$[u^j_0, \ldots, u^j_{N-1}]$ is feasible for $J(x^j_0, x_F, q_F, \mathcal{I}_0, N_0)$ with $x_F = x^j_N$ and the associated cost is q^{j-1}_0, which in turns implies $q^{j-1}_0 \geq c^j_0 \geq q^j_0$. ∎

V. APPLICATION TO SLIP WALKING

A. Spring Loaded Inverted Pendulum Model

This section describes the Spring Loaded Inverted Pendulum (SLIP) model with massless legs [20]. The system state is $x = [p_x, p_y, v_x, v_y, z^l_x, z^r_x, z^l_y, z^r_y]$, where (p_x, p_y) is the position of the Center of Mass (CoM), (v_x, v_y) the velocity of the CoM, and (z^k_x, z^k_y) the position of the kth foot for $k \in \{l, r\}$. The input vector $u = [\delta, v_z, v^l_y, v^r_y]$, where δ represents the change of leg stiffness, v_z the velocity along the x-axis of the foot not in contact with the ground, and (v^l_y, v^r_y) the velocities of the two feet on the y-axis. Given the state of the system, we can compute the angle $\theta^k = \mathrm{tg}^{-1}((p_x - z^k_x)/p_y), \forall k \in \{l, r\}$ and the length $l^k = l_0 - \sqrt{(p_x - z^k_x)^2 + p_y^2}, \forall k \in \{l, r\}$, where $l_0 = 0.55$ is the rest leg length, for more details please refer to [20]. These quantities are used to define the dynamics:

$$f(x, u) = x$$
$$+ \begin{bmatrix} v_x \\ v_y \\ (\delta + \delta_0)[\gamma^l l^l \sin(\theta^l) + \gamma^r l^r \cos(\theta^r)] \\ (\delta + \delta_0)[\gamma^l l^l \cos(\theta^l) + \gamma^r l^r \cos(\theta^r)] - mg \\ (1 - \gamma^l)v_z \\ (1 - \gamma^r)v_z \\ v^l_y \\ v^r_y \end{bmatrix} dt$$

where the integer variable γ^k equals one if the kth foot is in contact with the ground. In particular, we have that $(\gamma^l, \gamma^r) = (1, 1)$ if $x \in \mathcal{D}_{ds}$, $(\gamma^l, \gamma^r) = (1, 0)$ if $x \in \mathcal{D}_l$, and $(\gamma^l, \gamma^r) = (0, 1)$ if $x \in \mathcal{D}_r$, where the regions are defined as follows:

$$\mathcal{D}_{ds} = \{x \in \mathbb{R}^n | l^r \leq l_{max}, l^l \leq l_{max}, z^l_y = 0 \text{ and } z^r_y = 0\},$$
$$\mathcal{D}_l = \{x \in \mathbb{R}^n | l^r \leq l_{max}, z^l_y = 0 \text{ and } z^r_y > 0\},$$
$$\mathcal{D}_r = \{x \in \mathbb{R}^n | l^l \leq l_{max}, z^l_y > 0 \text{ and } z^r_y = 0\}.$$

Finally, we notice that when a sequence of regions is fixed, the vertical motion of the feet can be computed independently from the other states.
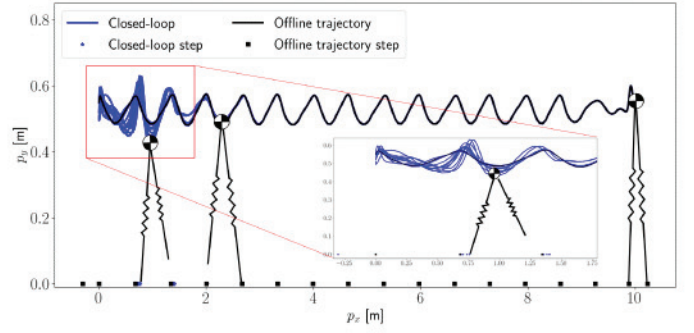
B. Simulations

First, we compute a feasible trajectory that steers the system from standing still at $(p_x, p_y) = (0, 0.85)$ to a goal state $x^g = (10, 0.85, 0, 0, 9.9, 10.2, 0, 0)$. In order to compute a feasible trajectory $\mathbf{x}^0$, we fixed a sequence of regions $\{\mathcal{D}_t\}^T_{t=0}$ where the system should be at each time t and we solved the resulting NLP with IPOPT [23] using CASADI [24]. Furthermore, we added a slack variable to the terminal constraint, we used

$$l(x, u) = \|p_x - p^g_x\|^2_2 + 10\|p_y - p^g_y\|^2_2 + \|v_x\|^2_2 + \|v_y\|^2_2$$
$$+ \|\delta\|^2_2 + 0.1\|v_z\|^2_2, \quad (14)$$

and the input constraints $\delta \in \{\bar{\delta} : \|\bar{\delta}\| \leq 10\}$ and $v_z \in \{\bar{v}_z : \|\bar{v}_z\| \leq 10\}$. Code available at https://github.com/urosolia/SLIP. Figure 1 shows the feasible trajectory which steers the system from the starting configuration to the goal position while transitioning through phases of double support (black) and single support with the left (red) and right (blue) legs. The figure also shows the locations where the feet are in contact with the ground. This feasible trajectory is used to initialize the control policy from Algorithm 1. We tested the proposed strategy with $N = 30$ on 10 different initial conditions in the neighborhood of the starting state x_S. We set $M = 1$ to test the robustness to disturbances and changes in initial conditions of Algorithm 1, when only one FTOCP is solve at each time t. Figure 2 shows that for all initial conditions the controller is able to steer the system to the goal state. Notice that in order to stabilize the system the proposed strategy is able to plan a sequence of foot steps, which are different from the one associated with the feasible trajectory. Finally, it takes on average less than $0.05s$ to run Algorithm 1 with a maximum computation time of $0.114s$ across all simulations.

Furthermore, we compared the proposed strategy with a tracking controller which is defined removing the terminal constraint from (4) and using a tracking cost instead of (14). Both our method and the tracking controller are able complete the task. However, as shown in Figure 3, the tracking controller fails to reach the goal when a disturbance hits the system. In Figure 3, it is interesting to notice that the proposed approach initially deviates more from the offline trajectory compared to the tracking controller. This deviation allows the controller to compensate for the disturbance and to stabilize the system back to a periodic gait.

Finally, we tested Algorithm 2 to iteratively update the control policy. We set $j = 40$ and we changed the stage cost to encode the objective of steering the system from the starting state to the goal state in minimum time. In particular, we defined the stage cost $\bar{l}(x, u) = \mathbb{1}_\mathcal{G}(x) + 0.0001 l(x, u)$, where
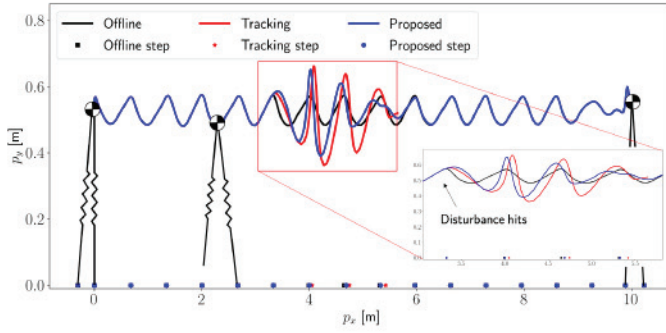
Fig. 3. Comparison between an MPC tracking controller and the proposed strategy, when a disturbance hits the system.
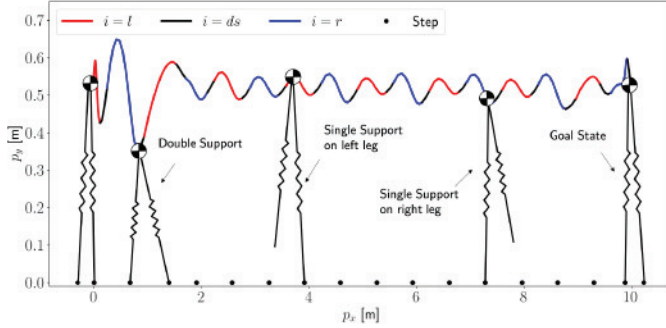

Fig. 4. Closed-loop trajectory computed from Algorithm 2 after 40 updates of the control policy.

the function $\mathbb{1}_{\mathcal{G}}(x) = 0$ if $x \in x_T^0$ and $\mathbb{1}_{\mathcal{G}}(x) = 1$ otherwise. Moreover, we set $M = 40$ to allow Algorithm 1 to solve multiple instances of the FTOCP (4). Note that as M gets larger the controller can better explore the state space, but as a trade off the computational cost increases. In this example it takes on average less than $0.4s$ and at most $0.9s$ to run Algorithm 1. We initialized the proposed policy iteration strategy with the feasible trajectory $\mathbf{x}^0$ from Figure 1 that steers the system from the starting point to the goal state in 396 time steps, and our algorithm returned a policy which completes the task in 193 time steps. The closed-loop trajectory is shown in Figure 4. First the controller accelerates and as a result the CoM oscillates more compared to the first feasible trajectory from Figure 1. Finally, the controller slows down and reduces the oscillation to reach the goal position with zero speed and two feet on the ground.

VI. CONCLUSION

We presented an algorithm to synthesize control policies for hybrid systems by leveraging a feasible trajectory for the control task. We showed that the proposed methodology guarantees constraints satisfaction and convergence in finite time. Building upon the proposed synthesis strategy, we presented a policy iteration algorithm which guarantees that at each policy update the closed-loop performance is non-decreasing. Finally, we tested the proposed strategy on a discretize SLIP model.

ACKNOWLEDGMENT

The authors would like to thank Andrew J. Taylor, Noel Csomay-Shanklin, and Wenlong Ma for suggestions and proofreading this letter.

REFERENCES

[1] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain—Online foothold optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5370–5376, Oct. 2020.
[2] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Paris, France, 2020, pp. 2436–2442.
[3] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," 2020. [Online]. Available: arXiv:2011.00032.
[4] R. Grandia, A. J. Taylor, A. Singletary, M. Hutter, and A. D. Ames, "Nonlinear model predictive control of robotic systems with control Lyapunov functions," 2020. [Online]. Available: arXiv:2006.01229.
[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.
[6] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *Proc. Amer. Control Conf. (ACC)*, vol. 2. Chicago, IL, USA, 2000, pp. 1190–1194.
[7] R. Oberdieck and E. N. Pistikopoulos, "Explicit hybrid model-predictive control: The exact solution," *Automatica*, vol. 58, pp. 152–159, Aug. 2015.
[8] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[9] V. Dua, N. A. Bozinis, and E. N. Pistikopoulos, "A multiparametric programming approach for mixed-integer quadratic engineering problems," *Comput. Chem. Eng.*, vol. 26, nos. 4–5, pp. 715–733, 2002.
[10] I. Naşcu, R. Oberdieck, and E. N. Pistikopoulos, "Explicit hybrid model predictive control strategies for intravenous anaesthesia," *Comput. Chem. Eng.*, vol. 106, pp. 814–825, Nov. 2017.
[11] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, "Hybrid predictive control for aerial robotic physical interaction towards inspection operations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, 2014, pp. 53–58.
[12] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot. (Humanoids)*, Birmingham, U.K., 2017, pp. 31–38.
[13] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2433–2448, Jun. 2021.
[14] D. Frick, A. Domahidi, and M. Morari, "Embedded optimization for mixed logical dynamical systems," *Comput. Chem. Eng.*, vol. 72, pp. 21–33, Jan. 2015.
[15] P. Hespanhol, R. Quirynen, and S. Di Cairano, "A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control," in *Proc. 18th Eur. Control Conf. (ECC)*, Naples, Italy, 2019, pp. 2763–2768.
[16] D. Bertsimas and B. Stellato, "The voice of optimization," *Mach. Learn.*, vol. 110, no. 2, pp. 249–277, 2021.
[17] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato, "Learning convex optimization control policies," in *Proc. 2nd Conf. Learn. Dyn. Control*, 2020, pp. 361–373.
[18] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. A data-driven control framework," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 1883–1896, Jul. 2017.
[19] U. Rosolia and F. Borrelli, "Minimum time learning model predictive control," *Int. J. Robust Nonlinear Control*, to be published.
[20] M. Shahbazi, R. Babuška, and G. A. D. Lopes, "Unified modeling and control of walking and running on the spring-loaded inverted pendulum," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1178–1195, Oct. 2016.
[21] A. Richards and J. How, "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility," in *Proc. Amer. Control Conf.*, vol. 5, 2003, pp. 4034–4040.
[22] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI, USA: Nob Hill, 2009. [Online]. Available: https://books.google.com/books?id=3_rfQQAACAAJ
[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.