Minimum Time Learning Model Predictive Control

Ugo Rosolia and Francesco Borrelli

Abstract—In this paper we present a Learning Model Predictive Control (LMPC) strategy for linear and nonlinear time optimal control problems. Our work builds on existing LMPC methodologies and it guarantees finite time convergence properties for the closed-loop system. We show how to construct a time varying safe set and terminal cost function using historical data. The resulting LMPC policy is time varying and it guarantees recursive constraint satisfaction and performance improvement. Computational efficiency is obtained by convexifying the safe set and terminal cost function. We demonstrate that, for a class of nonlinear system and convex constraints, the convex LMPC formulation guarantees recursive constraint satisfaction and performance improvement. Finally, we illustrate the effectiveness of the proposed strategies on minimum time obstacle avoidance and racing examples.

I. INTRODUCTION

In time optimal control problems, the goal of the controller is to steer the system from the staring point \(x_s\) to the terminal point \(x_F\) in minimum time, while satisfying state and input constraints. These problems have been studied since the 1950s [1]–[4] and it was shown that the optimal input strategy is a piece-wise function which saturates the input constraints [1]–[3]. Furthermore, while investigating the solution to time optimal control problems, researches formalized the maximum principle which describes the first order necessary optimality conditions [5], [6].

For linear systems, time optimal control problems can be solved applying the maximum principle. However, for nonlinear systems the optimality conditions are hard to solve, as those are described by a two boundary value problem for nonlinear systems the optimality conditions are hard to solve, while guaranteeing safety and performance improvement for a system of nonlinear differential equations [6]. For this reason, several approaches have been proposed to approximate the solution to time optimal control problems. These strategies can be divided in three different categories: i) hierarchical approaches, where in the first step a collision-free path is generated and afterwards it is computed the speed profile which minimizes the travel time along the path [7]–[12], ii) maximum principle-based strategies, which exploit the necessary optimality conditions [13]–[15] and iii) iterative optimization strategies, where the original control problem is approximated solving sequentially or in parallel simpler optimization problems [16]–[19]. A comprehensive literature review is out of the scope of this work. In the following, we focus on iterative optimization strategies, because the proposed approach falls into this category. In [16] the time optimal control problem is posed as a constrained nonlinear optimization problem and it is solved using a variable-order Legendre-Gauss-Radau method, where the initial guesses for the algorithm are obtained by solving a sequence of modified optimal control problems. A different approach was proposed in [17] where the path is parametrized using basis function which are amenable for optimization. The authors in [18] first proposed a smooth spatial system reformulation for the autonomous racing time optimal control problem. Afterwards, they used a nonlinear optimization solver based on a SQP algorithm to compute an optimal solution. In [19] the authors used at each time step a Model Predictive Controller (MPC) to compute a trajectory which drives the system from the current state to the end state.

We propose to iteratively approximate the solution to a time optimal control problem by repeatedly steering the system from the staring point \(x_s\) to the terminal point \(x_F\). At each \(j\)th iteration of the control task, the closed-loop trajectory is stored and used to update the control policy. Our work builds on existing Learning Model Predictive Control (LMPC) strategies for linear and nonlinear systems [20], [21]. In LMPC at each iteration \(j\), historical data are used to estimate i) a safe set of states from which the control task can be completed using a known policy \(\pi^j\) and ii) a value function which approximates the closed-loop cost associated with the policy \(\pi^j\).

The first contribution of this work is to design a LMPC schema for nonlinear systems where the safe set and the approximated value function are time varying. At each time \(i\) of iteration \(j\), the proposed time varying LMPC uses a subset of the stored data to compute the control action and it allows us to reduce the computational burden associated with time invariant LMPC methodologies [20]. We show that the proposed strategy guarantees safety, finite time convergence and performance improvement with respect to previous executions of the control task. The second contribution of this work is to propose a relaxed LMPC formulation which is based on a convexified time varying safe set and cost function. This strategy enables the reduction of the computational burden while guaranteeing safety and performance improvement for a specific class of nonlinear system and convex constraints. Furthermore, we show that the same properties hold for nonlinear systems, if a sufficient condition on the stored states and the system dynamics is satisfied. Finally, we test the proposed strategies on nonlinear time optimal control problems. We show that the proposed LMPC is able to match the performance of the strategy from [20], while being computationally faster. Furthermore, we demonstrate the effectiveness of the relaxed LMPC formulation both on a nonlinear double integrator and a dubins car minimum time problems.

U. Rosolia and F. Borrelli are with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA 94701, USA {ugo.rosolia, fborrelli}@berkeley.edu
II. PROBLEM FORMULATION

Consider the nonlinear system

\[ x_{j+1}^j = f(x_j^j, u_j^j), \]  

(1)

where \( x_j^j \in \mathbb{R}^n \) and \( u_j^j \in \mathbb{R}^m \) represent the system state and input at time \( t \) of the \( j \)th iteration, respectively. Furthermore, the system is subject to the following state and input constraints

\[ x_j^j \in X \text{ and } u_j^j \in U, \forall t \geq 0, \forall j \geq 0. \]  

(2)

The goal of the controller is to solve the following minimum time optimal control problem

\[
\begin{align*}
\min_{T, u_0^j, \ldots, u_{T-1}^j} & \sum_{t=0}^{T-1} 1 \\
\text{s.t.} & \quad x_{j+1}^j = f(x_j^j, u_j^j), \forall t = [0, \ldots, T - 1] \\
& \quad x_0^j \in X, u_0^j \in U, \forall t = [0, \ldots, T - 1] \\
& \quad x_T^j = x_F, \\
& \quad x_0^j \in S_S \end{align*}
\]  

(3)

where the goal state \( x_F \) is an unconstrained equilibrium point for system (1), i.e. \( f(x_F, 0) = x_F \).

In this paper we propose to solve Problem (3) iteratively. In particular, at each iteration we drive the system from the starting point \( x_S \) to the terminal state \( x_F \) and we store the closed-loop trajectories. After completion of the \( j \)th iteration, these trajectories are used to synthesize a control policy for the next iteration \( j + 1 \). We show that the proposed iterative design strategy guarantees recursive constraint satisfaction and iterative performance improvement. Next, we define the safe set and value function approximation which will be used in the controller design.

III. SAFE SET AND VALUE FUNCTION APPROXIMATION

At each \( j \)th iteration of the control task, we store the closed-loop trajectories and the associated input sequences. In particular, at the \( j \)th iteration we define the vectors

\[
\begin{align*}
\mathbf{u}^j &= [u_0^j, \ldots, u_{T^j}^j], \\
\mathbf{x}^j &= [x_0^j, \ldots, x_{T^j}^j],
\end{align*}
\]  

(4)

where \( x_j^i \) and \( u_j^i \) are the state and input of system (1). In (4), \( T^j \) denotes the time at which the closed-loop system reached the terminal state, i.e. \( x_{T^j} = x_F \).

A. Time Varying Safe Set

We use the stored data to build time varying safe sets, which will be used in the controller design to guarantee recursive constraint satisfaction. First, we define the time varying safe set at iteration \( j \) as

\[ SS_j^j = \bigcup_{i=0}^{T^j} x_i^j, \]  

(5)

where, for \( T^j = \min_{k \in \{0, \ldots, j\}} T^k \),

\[ \delta^j_i = \max(t + T^i - T^j, 0). \]  

(6)

Definition (6) implies that if at time \( t \) of the \( j \)th iteration \( x_i^j = x_{T^j}^j \), then system (1) can be steered along the \( i \)th trajectory to reach \( x_F \) in \( (T^j + t) \) time steps. Basically, at each time \( t \) the time varying safe set \( SS_j^j \) collects the stored states from which system (1) can reach the terminal state \( x_F \) in at most \( (T^j + t) \) time steps. A representation of the time varying safe set for a two-dimensional system is shown in Figure 1. We notice that, by definition, if a state \( x_i^j \) belongs to \( SS_j^j \), then there exists a feasible control action \( u_i^j \in U \) which keeps the evolution of the nonlinear system (1) into the time varying safe set at the next time step \( t + 1 \), i.e. \( f(x_i^j, u_i^j) \in SS_{j+1}^j \). This property will be used in the controller design to guarantee that state and input constraints (2) are recursively satisfied.

![Fig. 1. Representation of the time varying safe set SS2].(5)](image)

Finally, at each time \( t \) the we define the local convex safe set as the convex hull of \( SS_j^j \) from (5).

\[ CS_j^j = \text{Conv}(SS_j^j) \]  

\[ = \{ x \in \mathbb{R}^n : \exists [\lambda_0^j, \ldots, \lambda_{T^j}^j] \geq 0, \sum_{i=0}^{T^j} \lambda_k^j x_i^j = x \}. \]  

(7)

Later on we will show that for a class on nonlinear systems, if a state \( x_i^j \) belongs to \( CS_j^j \), then there exists a feasible control action \( u_i^j \in U \) which keeps the evolution of the nonlinear system (1) into the convex safe set at the next step \( t + 1 \). For such class of nonlinear systems, \( CS_j^j \) can be used to synthesize controllers which guarantee state and input constraint satisfaction at all time instants.

Remark 1: When the goal of the controller is to reach an invariant set \( X_F \) in minimum time, it is still possible to use the proposed iterative control strategy. In this case one should replace \( x_{T^j}^j = x_F \) with \( X_F \) in definition (5).

B. Time Varying Value Function Approximation

In this section, we show how to construct \( Q \)-functions which approximate the cost-to-go over the safe set and
convex safe set. These functions will be used in the controller design to guarantee iterative performance improvement.

We define the cost-to-go associated with the stored state \( x^j_t \) from (4).

\[
J^j_{t\rightarrow T_j}(x^j_t) = \sum_{k=t}^{T_j} \mathbb{1}_{x_F}(x^j_k),
\]

(8)

where the indicator function

\[
\mathbb{1}_{x_F}(x) = \begin{cases} 1 & \text{if } x_F \neq x \\ 0 & \text{Else} \end{cases}
\]

The above cost-to-go represents the time steps needed to steer system (1) from \( x^j_t \) to the terminal state \( x_F \) along the \( j \)-th trajectory, and it is used to construct the function \( Q^j_t(\cdot) \), defined over the safe set \( SS^j_t \),

\[
Q^j_t(x) = \min_{i \in \{0,\ldots, j\}} \left\{ \min_{k \in \{T^i_k,\ldots, T^j_k\}} J^j_{k\rightarrow T_k}(x^j_k) \right\}
\]

(9)

s.t. \( x = x^j_k \in SS^j_t \).

The function \( Q^j(\cdot) \) assigns to every point in the safe set \( SS^j_t \) from (5) the minimum cost-to-go along the stored trajectories from (4), i.e.

\[
\forall x \in SS^j_t, Q^j_t(x) = J^j_{k\rightarrow T_k}(x^j_k) = \sum_{k=k^*}^{T^j_k} \mathbb{1}_{x_F}(x^j_k)
\]

(10)

where \( i^* \) and \( k^* \) are the minimizers in (9):

\[
[i^*, k^*] = \argmin_{i \in \{0,\ldots, j\}} \left\{ \min_{k \in \{T^i_k,\ldots, T^j_k\}} J^j_{k\rightarrow T_k}(x^j_k) \right\}
\]

(11)

Finally, we define the convex Q-function over the convex safe set \( CS^j_t \) from [7].

\[
\bar{Q}^j_t(x) = \min_{|\lambda^j_{\delta^j_0},\ldots,\lambda^j_{T^j_k}| \geq 0} \sum_{i=0}^{j} \sum_{k=0}^{T^i_k} \lambda_i^j \cdot J^j_{1\rightarrow T^i_k}(x^j_k)
\]

s.t. \( \sum_{i=0}^{j} \sum_{k=0}^{T^i_k} \lambda_i^j \cdot x^j_k = x \)

(12)

\[\sum_{i=0}^{j} \sum_{k=0}^{T^i_k} \lambda_i^j = 1.\]

where \( \delta^j_k \) is defined in [9]. The convex Q-function \( \bar{Q}^j_t(\cdot) \) is simply a piecewise-affine interpolation of Q-function from [9] over the convex safe set, as shown in Figure 2. In Section [7] we will show that the convex Q-function can be used to guarantee iterative performance improvement for a particular class on nonlinear systems.

**IV. LEARNING MODEL PREDICTIVE CONTROL DESIGN**

In this section, we describe the controller design. We propose a Learning Model Predictive Control (LMPC) strategy for nonlinear systems which guarantees recursive constraint satisfaction and iterative performance improvement. Computing the control action from the LMPC policy is expensive. For this reason, we also present a relaxed LMPC policy, which allows us to reduce the computational cost and it guarantees recursive constraint satisfaction and performance improvement for a class of nonlinear systems.

**A. LMPC: Mixed Integer Formulation**

At each time \( t \) of the \( j \)-th iteration the controller solves the following finite time optimal control problem,

\[
\begin{align*}
\sum_{k=t}^{t+N-1} \mathbb{1}_{x_F}(x^j_{k|t}) + \bar{Q}^j_{t\rightarrow T_t}(x^j_{t+N}) \\
\text{s.t. } x^j_{k+1} = f(x^j_k, u^j_k), \forall k = t, \ldots, t + N - 1 \\
x^j_k \in \mathcal{X}, u^j_k \in \mathcal{U}, \forall k = t, \ldots, t + N - 1 \\
x^j_{t+N} \in \mathbb{S}^j_{t+N} \\
x^j_t = x^j_t
\end{align*}
\]

(13)

where \( \mathbb{S}^j_{t+N} = [u^j_{t}, \ldots, u^j_{t+N-1}] \in \mathbb{R}^{d \times N} \). The solution to the above finite time optimal control problem steers system (1) from \( x^j_t \) to the time varying safe set \( SS^j_{t+N} \) while satisfying state, input and dynamic constraints. Let

\[
\mathbb{U}^j_x = [u^j_x, \ldots, u^j_{t+N-1}]
\]

be the optimal solution to (12) at time \( t \) of the \( j \)-th iteration. Then, we apply to the system (1) the first element of the optimizer vector,

\[
u^j_t = \pi^j_{t\rightarrow t} \cdot u^j_x.
\]

The finite time optimal control problem (12) is repeated at time \( t \), based on the new state \( x^j_{t+1} = x^j_{t+1} \), until the iteration is terminated when \( x^j_{t+1} = x_F \).

Computing the control action from the LMPC policy (14) requires to solve a mixed-integer programming problem, as
Then, we apply to the system (1) the first element of the control problem. We show that the LMPC guarantees constraint satisfaction at all time instants, convergence in finite time to $x_F$ and iterative performance improvement. Furthermore, we demonstrate that the same properties are guaranteed when the relaxed LMPC is in closed-loop with a specific class on nonlinear systems or when a sufficient condition on the stored data and the system dynamics is satisfied.

A. Recursive Feasibility

We assume that a feasible trajectory which drives the system from the starting point $x_S$ to the terminal state $x_F$ is given, and we show that the controller recursively satisfies state and input constraints (2).

**Assumption 1:** We are given the closed-loop trajectory and associated input sequence

\[ x^0 = [x^0_0, \ldots, x^0_T] \] and \[ u^0 = [u^0_0, \ldots, u^0_T], \]

which satisfy state and input constraint sets in (2) are convex, then the relaxed LMPC (15) is feasible at time $t = 0$. Furthermore, we have that $x^F = x_S$ and $x^F_T = x_F$.

**Theorem 1:** Consider system (1) controlled by the LMPC (12) and (14). Let $SS_t$ be the time varying safe set at iteration $j$ as defined in [5]. Let Assumption 1 hold and assume that $x^0_j = x_S \land j \geq 0$, then at every iteration $j \geq 1$ the LMPC (12) and (14) is feasible. Furthermore, we have that $x^0_j = x_S$ and $x^F_T = x_F$. Therefore, at each iteration we solve the following finite time optimal control problem

\[ \min_{u^j_t, \lambda^j_t \geq 0} \sum_{k=t}^{t+N-1} L(x^j_{k+1}) + \sum_{i=0}^{t-1} \sum_{i=0}^{j-1} \lambda^j_i J^i_{k+1}(x^j_k) \]

s.t. \[ x^j_{k+1} = f(x^j_k, u^j_k), \forall k = t, \ldots, t + N - 1 \]

\[ x^j_k \in X, u^j_k \in U, \forall k = t, \ldots, t + N - 1 \]

\[ \sum_{i=0}^{t-1} \sum_{i=0}^{j-1} \lambda^j_i x^j_k = x^F_j \]

\[ \sum_{i=0}^{t-1} \sum_{i=0}^{j-1} \lambda^j_i = 1 \]

\[ x^j_k \in \bar{X}, \forall k = t, \ldots, t + N - 1 \]

where $U_t = [u^j_t, \ldots, u^j_{t+N-1}] \in \mathbb{R}^{d \times N}$ and the vector $\lambda^j_t = [\lambda^j_0, \ldots, \lambda^j_{t-1}] \in \mathbb{R}^{d \times N}$ describes the terminal constraint set $CS^{j-1}_t$ and terminal cost function $Q^j_{t+N}(\cdot)$. Let the optimal solution to (12) at time $t$ of the $j$th iteration be

\[ U^j_t \in U^j_t \approx [u^j_t, \ldots, u^j_{t+N-1}] \]

\[ \lambda^j_t \in [\lambda^j_0, \ldots, \lambda^j_{t-1}] \]

Then, we apply to the system (1) the first element of the optimal input sequence,

\[ u^j_t = \bar{u}^{j-1} (x^j_t) = u^j_t \approx \]

Notice that the terminal constraints in (15) is enforced using nonlinear equality constraint, and therefore the computation burden is reduced with respect to the LMPC from Section IV-A. In the next section we will show that for a class on nonlinear system the relaxed LMPC (15) and (17) has the same safety and performance improvement properties of the LMPC presented in Section IV-A.

V. PROPERTIES

This section describes the properties of the proposed control strategies. We show that the LMPC guarantees constraint satisfaction at all time instants, convergence in finite time to...
Assumption 3: The state and input constraint sets $\mathcal{X}$ and $\mathcal{U}$ in (2) are convex.

Theorem 2: Consider system (11) controlled by the relaxed LMPC (15) and (17). Let $\mathcal{S}^i_j$ be the convex safe set at iteration $j$ as defined in (7). Let Assumptions (12) hold and assume that $x_0 = x_S \forall j \geq 0$, then at every iteration $j \geq 1$ the relaxed LMPC (15) and (17) is feasible for all $t \geq 0$ when (17) is applied to system (1).

Proof: We notice that by Assumption 2 it follows that $\forall x \in \mathcal{S}^i_1$ there exists $u \in U$ such that $f(x, u) \in \mathcal{C}^i_{t+1}$.

Therefore, by (20) we have that at time $t^*$

\[ T^j - 1, i^* - N \geq 0 \]

we have that at time $t^* = T^j - 1, i^* = x_F$.

Therefore, by Proposition 1 the closed-loop system converges at most in $T^* = T^j - 1, i^*$ steps. Finally, we notice that $T^j = T^* + 1, i^* \leq T^*$, $\forall k \in \{0, \ldots, j - 1\}$.

Next, we show that if the relaxed LMPC (15) and (17) is in closed-loop with system (11) which satisfies Assumption 2 then $T^j$ is non-increasing with the iteration index. The proof follows as in Theorem 3 exploiting the recursive feasibility of the relaxed LMPC (15) and (17) from Theorem 2.

Proposition 2: Consider system (11) controlled by the LMPC (15) and (17). Assume that $\mathcal{S}^i_j = x_F$ and $Q^i_j = 0$ for all $t \geq 0$. If at time $t$ Problem (15) is feasible, then the closed-loop system (11) and (14) converges in at most $T^*$ time steps to $x_F$.

Proof: The proof follows as in Proposition 1 replacing the LMPC cost $J^\text{arc}, (\cdot)$ with the relaxed LMPC cost $J^\text{arc}, (\cdot)$.

Theorem 4: Consider system (11) controlled by the LMPC (15) and (17). Let $\mathcal{S}^i_j$ be the time varying safe set at iteration $j$ as defined in (7). Let Assumptions (13) hold and assume that $x_0 = x_S \forall j \geq 0$, then the time $T^j$ at which the closed-loop system (11) and (14) converges to $x_F$ is non-increasing with the iteration index,

\[ T^j \leq T^k, \forall k \in \{0, \ldots, j - 1\} \]

Proof: By Theorem 2 we have that Problem (15) is feasible at all time $t \geq 0$. Therefore, the proof follows as for Theorem 3 using Proposition 2.

C. Sufficient Condition for the Relaxed LMPC

In the previous sections we discussed the properties of the relaxed LMPC strategy in closed-loop with nonlinear systems which satisfy Assumption 2. Next, we show that the recursive constraint satisfaction and performance improvement properties still hold, if we replace Assumption 2 with the following assumption on the system dynamics and stored data.

Assumption 4: Consider $j$ stored feasible closed-loop trajectories $x^i$ and associated input sequences $u^i$. For all $x \in \mathbb{R}^n$ which can be expressed as convex combination of $n + 1$ stored states, i.e.

\[ x \in \text{Conv} (\bigcup_{t \in I(x)} x^i) \]

where the set $I(x) = \{t_0, i_0, \ldots, t_n, i_n\}$ collects $n + 1$ time and iteration indices associated with the stored states, we have that there exists $u \in U$ such that

\[ f(x, u) \in \text{Conv} (\bigcup_{t \in I(x)} f(x^i, u^i)) \]

We underlined that the above assumption is hard to verify in general. In practice, Assumption 4 may be approximately checked using sampling strategies, as shown in the result section.
Finally, we state the following theorem which summaries the necessary conditions that guarantee recursive constraint satisfaction, convergence in finite time and iterative performance improvement for the relaxed LMPC in closed-loop with the nonlinear system \(1\).

**Theorem 5:** Consider system \(1\) controlled by the relaxed LMPC \((15)\) and \((17)\). Let \(CS^j_t\) be the time varying convex safe set at iteration \(j\) as defined in \((7)\). Let Assumptions \(1\) \(3\) and \(4\) hold and assume that \(x_0 = x_s\) \(\forall j \geq 0\). Then, the relaxed LMPC \((15)\) and \((17)\) satisfies state and input constraints \((2)\) at all time. Furthermore, the time \(T^j\) at which the closed-loop system \((1)\) and \((17)\) converges to \(x_T\) is non-increasing with the iteration index,

\[
T^j \leq T^k, \quad \forall k \in \{0, \ldots, j-1\}.
\]

**Proof:** We assume that at time \(t\) the relaxed LMPC \((15)\) and \((17)\) is feasible, let \((13)\) be the optimal solution. As Assumption \(2\) holds, we have that there exists \(u \in U\) such that

\[
\begin{align*}
[x_{t+1}^1, \ldots, x_{t+Nf}^j, f(x_{t+Nf}^j, u) \in CS^j_{t+1}] \\
[u_{t+1}^j, \ldots, u_{t+Nf-1}^j, u \in U],
\end{align*}
\]

satisfy state and input constraints \((2)\), and therefore the relaxed LMPC \((15)\) and \((17)\) is feasible at time \(t + 1\). The rest of the proof follows as in Theorems \(2\) and \(4\). \(\blacksquare\)

**VI. DATA REDUCTION**

In this section, we show that the proposed LMPC can be implemented using a subset of the time varying safe set from \((5)\). In particular, we show the controller may be implemented using the last \(l\) iterations and \(P\) data points per iteration.

**A. Safe Subset**

We define the time varying safe subset from iteration \(l\) to iteration \(j\) and for \(P\) data points as

\[
SS^{j,l}_{t,P} = \bigcup_{i=l}^{j} \bigcup_{k=\delta^i_l}^{\delta^i+P} x^i_k,
\]

where \(\delta^i_l\) is defined in \((6)\). Furthermore, in the above definition we set \(x^i_k = x_F\) for all \(k > T^i\) and \(i < 0\). A representation of the time varying safe subset for a two-dimensional system is shown in Figure \(3\). Compare the safe subset \(SS^{j,l}_{t,P}\) with the safe set \(SS^j_t\) from \((5)\). We notice that, \(SS^{j,l}_{t,P}\) is contained into \(SS^j_t\). Therefore, at time \(t\) the safe subset collects the stored states from which the LMPC \((14)\) can reach the terminal state \(x_F\) in at most \((T^{j_1} - t)\) time steps. Finally, by definition, if a state \(x^i_k\) belongs to \(SS^{j,l}_{t,P}\), then there exists a feasible control action \(u^i_k \in U\) which keeps the evolution of the nonlinear system \((1)\) into the time varying safe set at the next time step \(t + 1\), i.e. \(f(x^i_k, u^i_k) \in SS^{j,l+1}_{t+1}\).

This property allows us to replace \(SS^{j,l}_{t,P}\) with \(SS^j_t\) in the design of the LMPC \((14)\) and \((12)\), without loosing the recursive constraint satisfaction property from Theorem \(1\).

Finally, at each time \(t\) we define the local convex safe subset as the convex hull of \(SS^{j,l}_{t,P}\) from \((5)\).

\[
CS^{j,l}_{t,P} = \text{Conv}(SS^{j,l}_{t,P}).
\]

We underline that relaxed LMPC from Section \(V-B\) may be implemented replacing the the convex safe set \((7)\) with the convex safe subset \((22)\).

**Fig. 3.** Representation of the time varying safe subset \(SS^{j,l}_{t,P}\). We notice that just a subset of the stored states are used to define \(SS^{j,l}_{t,P}\).

Finally, at each time \(t\) the we define the local convex safe subset as the convex hull of \(SS^{j,l}_{t,P}\) from \((5)\).

\[
CS^{j,l}_{t,P} = \text{Conv}(SS^{j,l}_{t,P}).
\]

We underline that relaxed LMPC from Section \(V-B\) may be implemented replacing the the convex safe set \((7)\) with the convex safe subset \((22)\).

\[
\begin{align*}
&\text{Closed-loop at iteration } 0 \text{ with } T^0 = 7 \text{ and } \delta^l = 4 \\
&\text{Closed-loop at iteration } 1 \text{ with } T^1 = 6 \text{ and } \delta^l = 1 \\
&\text{Closed-loop at iteration } 2 \text{ with } T^2 = 5 \text{ and } \delta^l = 0
\end{align*}
\]

**B. Q-function**

In the section, we construct the Q-function which assigns the cost-to-go to the states contained into the time varying safe subset from \((21)\). In particular, we introduce the function \(Q^{j,l}_{t,P}(\cdot)\), defined over the safe subset \(SS^{j,l}_{t,P}\), as

\[
\begin{align*}
Q^{j,l}_{t,P}(x) &= \min_{i \in \{l, \ldots, j\}} \sum_{t \in \{\delta^i_l, \ldots, \delta^i+P\}} J^i_{t \rightarrow T_f}(x^i_t) \\
&\text{s.t. } x = x^i_l \in SS^j_t
\end{align*}
\]

Compare the above function \(Q^{j,l}_{t,P}\) with \(Q^j_t\) from \((9)\). We notice that, the domain of \(Q^{j,l}_{t,P}\) is the safe subset \(SS^{j,l}_{t,P}\) and the domain of the \(Q^j_t\) is the safe set \(SS^j_t \supseteq SS^{j,l}_{t,P}\). Moreover, we have that

\[
\forall x \in SS^{j,l}_{t,P}, Q^{j,l}_{t,P}(x) = Q^j_t(x).
\]

Therefore, if we replace \(Q^j_t\) with \(Q^{j,l}_{t,P}\) in the design of the LMPC policy \((14)\), then the finite time convergence and performance improvements properties still hold.

Furthermore, we define the convex Q-function \(\bar{Q}^{j,l}_{t,P}\) from iteration \(l\) to iteration \(j\) and \(P\) data points as

\[
\begin{align*}
\bar{Q}^{j,l}_{t,P}(x) &= \min_{\lambda^i_1 \geq \ldots \lambda^i_{\delta^i_l+P} \geq 0} \sum_{i=0}^{j} \sum_{k=\delta^i_l}^{\delta^i+P} \lambda^i_k J^i_{k \rightarrow \delta^i+P}(x^i_k) \\
&\text{s.t. } \sum_{i=0}^{j} \sum_{k=\delta^i_l}^{\delta^i+P} \lambda^i_k x^i_k = x \\
&\quad \sum_{i=0}^{j} \sum_{k=\delta^i_l}^{\delta^i+P} \lambda^i_k = 1.
\end{align*}
\]
where $\delta^i_t$ is defined in (6). The above convex $Q$-function $\bar{Q}^{i,t}_t(\cdot)$ is simply a piecewise-affine interpolation of $Q$-function from (23) over the convex safe subset, as shown in Figure 4. We underline that $\bar{Q}^{i,t}_t(\cdot)$ can be used in the relaxed LMPC design instead of $Q^i_t(\cdot)$.

Figure 4. We underline that $Q^{i,t}_t(\cdot)$ can be used in the relaxed LMPC design instead of $\bar{Q}^{i,t}_t(\cdot)$ over the convex safe subset, as shown in Figure 4. We underline that $Q^{i,t}_t(\cdot)$ can be used in the relaxed LMPC design instead of $Q^i_t(\cdot)$.

$Q$-function $Q^{i,0}_{0,3}(\cdot)$ at time 0

convex $Q$-function $Q^{0,0}_{0,3}(\cdot)$ at time 0

$J^0_{1,T+\pi}(x_1^0)$

$J^0_{2,T+\pi}(x_2^0)$

stored states

$x_F = x_{f_T}, \forall j \geq 0$

$x_0 = x_0^0, \forall j \geq 0$

Fig. 4. Representation of the $Q$-function $Q^{i,0}_{0,3}(\cdot)$ and convex $Q$-function $Q^{0,0}_{0,3}(\cdot)$. We notice that the $Q$-function $Q^{i,0}_{0,3}(\cdot)$ is defined over a set of discrete data points, whereas the convex $Q$-function $Q^{0,0}_{0,3}(\cdot)$ is defined over the convex safe set.

VII. RESULTS

We test the proposed strategy on 3 time optimal control problems. In first example, the LMPC is used to drive a dubins car from the staring point $x_S$ to the terminal point $x_F$ while avoiding an obstacle. In the second example, we control a nonlinear double integrator system, which satisfies Assumption 2. Finally, the third example is a dubins car racing problem, which we solved using the relaxed LMPC after checking Assumption 4 via sampling. The code for these examples is available at https://github.com/urosolia/LMPC in the NonlinearLMPC folder.

A. Minimum time obstacle avoidance

We use the LMPC policy synthesized with the mixed integer approach from Section IV-A on the minimum time obstacle avoidance optimal control problem from [20].

$$\min_{T, \theta_0, \ldots, \theta_{T-1}} \sum_{t=0}^{T-1} 1$$

s.t. $x_{t+1}^{F} = \begin{bmatrix} x_t + v_t \cos(\theta_t) \\ y_t + v_t \sin(\theta_t) \\ v_{t+1} + a_t \\ (x_t - x_{obs})^2 + (y_t - y_{obs})^2 \end{bmatrix} \geq 1, \forall t \geq 0$

$$\begin{bmatrix} -\pi/2 \\ -1 \end{bmatrix} \leq \begin{bmatrix} \theta_t \\ a_t \end{bmatrix} \leq \begin{bmatrix} \pi/2 \\ 1 \end{bmatrix}, \forall t \geq 0$$

$x_T = x_F = [54, 0, 0]^T$

$x_0 = x_S = [0, 0, 0]^T$

The goal of the above minimum time optimal control problem is to steer the dubins car from the starting state $x_S$ to the terminal point $x_F$, while satisfying input saturation constraints and avoiding an obstacle. The obstacle is represented by an ellipse centered at $(x_{obs}, y_{obs}) = (27, -1)$ with semi-axis $(a_x, a_y) = (8, 6)$. At iteration 0, we compute a first feasible trajectory using a brute force algorithms and we use the closed-loop data to initialize the LMPC (12) and (14) with $N = 6$.

We compare the performance of the LMPC from [20] and of the LMPC policies (14) synthesized using different number of data points $P = \{8, 10, 40\}$ and iterations $i = \{1, 2, 3\}$, as described in Section VII (i.e. basically in definition (21) we set $l = j - 1 - i$). Figure 5 shows the time $T^j$ at which the closed-loop system converged to the terminal state $x_F$ at each iteration index. We notice that all LMPC policies converge to a steady state behavior which steers the system from $x_S$ to $x_F$ in 16 time steps. Furthermore, Figure 5 shows that the number of iterations needed to reach convergence is proportional to the amount of data used to synthesize the LMPC policy.

Fig. 5. Time steps $T^j$ to reach $x_F$ as a function of the iteration index. We notice that as more data points are used in the synthesis process, the number of iterations needed to reach a steady state behavior decreases.

Fig. 6. Computational cost associated with the LMPC policy at each time $t$ as function of the iteration index. We notice that as more data points are used in the synthesis process, the computational cost increases.
Figure 6 shows that the computational time increases as more data points $P$ are used in the control design. Therefore, there is a trade-off between the computational burden and the performance improvement at each iteration from Figure 5. It is interesting to notice that the computational cost associated with the proposed time varying LMPC strategy converges to a steady state value. On the other hand, the computation cost associate with the LMPC strategy from [20] increases at each iteration. Therefore, we confirm that the proposed time varying LMPC (12) and (14) allows us to reduce the computational cost while achieving the same closed-loop performance.

**B. Nonlinear Double Integrator**

In this section, we test the relaxed LMPC (15) and (17) on the following nonlinear double integrator problem

$$\begin{align*}
\min_{T,a_0,\ldots,a_{T-1}} & \sum_{t=0}^{T-1} 1 \\
\text{s.t. } & [x_{t+1}, v_{t+1}] = [x_t + v_t dt, v_t + (1 - \frac{v_t^2}{v_{\max}^2})a_t dt], \forall t \geq 0 \\
& 0 \leq v_t \leq v_{\max}, \forall t \geq 0 \\
& -1 \leq a_t \leq 1, \forall t \geq 0 \\
& x_T = x_F = [0,0]^T, \quad x_0 = x_S = [-10,0]^T, \\
\end{align*}$$

(25)

where the state of the system are the velocity $v_t$ and the position $x_t$. The control action is the acceleration $a_t$ which is scaled by the concave function $g(v_t) = (1 - \frac{v_t^2}{v_{\max}^2})$.

In Section X-A of the Appendix we show that the above nonlinear double integrator satisfies Assumption 2. We used an handcrafted policy to perform the first feasible trajectory used to initialize the relaxed LMPC policies synthesized with $N = 4$. Furthermore, we implemented the strategy from Section VI using $P = \{12, 25, 50, 200\}$ data points and $i = \{1, 3, 4, 10\}$ iterations.

Figures 9 shows the number of iterations needed to reach convergence. We notice that as more data points $P$ are used in the policy synthesis process, the closed-loop system convergence faster in the iteration domain to a trajectory which performs the task in 14 time steps.

Finally, we analyse the closed-loop trajectories associated with the LMPC policy (14) synthesized with $P = 8$ data points and $i = 1$ iteration. Figure 7 shows the first feasible trajectory, the stored data points and the closed-loop trajectory at convergence. We confirm that the LMPC is able to explore the state space while avoiding the obstacle and steering the system from the starting state $x_S$ to the terminal state $x_F$. Furthermore, we notice that the LMPC accelerates during the first part of the task, and then it decelerates to reach the terminal state with zero velocity, as shown in Figure 8.

**Fig. 7.** First feasible trajectory, stored data points and closed-loop trajectory at the 6th iteration. We notice that the LMPC is able to avoid the obstacle at each iteration.

**Fig. 8.** Acceleration and speed profile at convergence. We notice that the controller accelerates for the first 8 time steps and afterwards it decelerates to reach the terminal state goal state with zero velocity.

**Fig. 9.** Time steps $T^j$ to reach $x_F$ as a function of the iteration index. We notice that, also in this example, as more data points are used in the synthesis process, the number of iterations needed to reach a steady state behavior decreases.
Finally, Figures 10 and 11 show the steady-state closed-loop trajectories and the associated input sequences for all tested policies. We notice that after few iterations of the control task, the closed-loop systems converged to a similar behavior. In particular, the controller saturates the acceleration and deceleration constraints, as we would expect from the optimal solution to a time optimal control problem (Fig. 11). It is interesting to notice that accelerating the nonlinear double integrator to a peak speed requires more control effort than slowing down the system to zero speed. Therefore, the controller accelerates for the first 6 time steps and then it decelerates for the last 8 time steps to reach the terminal state with zero velocity.

As mentioned in Remark 1, in order to implement the LMPC to steer the system to terminal set instead of a terminal state with zero velocity. In particular, the controller saturates the control task, the closed-loop systems converged to a steady state behavior which steers the system to terminal set instead of a terminal state with zero velocity. Finally, Figures 10 and 11 show the steady-state closed-loop trajectories and associated input sequence at converged. In order to solve the following minimum time optimal control problem

\[
\min_{T,a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} s_t + \frac{v_t \cos(\theta_t - \gamma(s_t))}{1 - e_t/R} dt
\]

s.t.

\[
\begin{bmatrix}
  s_{t+1} \\
  e_{t+1} \\
  \theta_{t+1}
\end{bmatrix} = \begin{bmatrix}
  s_t + v_t \sin(\theta_t) dt \\
  e_t + v_t \sin(\gamma(s_t)) dt \\
  \theta_t + a_t dt
\end{bmatrix}, \forall t \geq 0
\]

\[
\begin{bmatrix}
  -2 \\
  -1
\end{bmatrix} \leq \begin{bmatrix}
  \theta_t \\
  a_t
\end{bmatrix} \leq \begin{bmatrix}
  2 \\
  1
\end{bmatrix}, \forall t \geq 0
\]

\[
e_{min} \leq e_t \leq e_{max}, \forall t \geq 0
\]

\[
x_T \in X_F,
\]

\[
x_0 = x_S = [0, 0, 0]^T,
\]

where \(\gamma(s_t)\) is the angle of the tangent vector to the centerline of the road at the curvilinear abscissa \(s_t\), the discretization time \(dt = 0.5\) and the lane half width \(e_{max} = -e_{min} = 2.0\). The control actions are the heading angle \(\theta_t\) and the acceleration command \(a_t\). The system is represented in the curvilinear abscissa reference frame where the state \(s_t, e_t\) and \(v_t\) are the distance travelled along the centerline, the lateral distance from the center of the lane and the velocity, respectively. Notice that in the curvilinear abscissa reference frame the lane boundaries are represented by convex constraints on the state \(e_t\), and therefore Assumption 3 is satisfied. The finish line is described by the following terminal set

\[
X_F = \left\{ x \in \mathbb{R}^3 \left| \begin{bmatrix} 18.19 \\ -e_{min} \\ 0 \end{bmatrix} \leq x \leq \begin{bmatrix} 18.69 \\ e_{min} \\ 0 \end{bmatrix} \right. \right\}.
\]

As mentioned in Remark 1, in order to implement the LMPC to steer the system to terminal set instead of a terminal point, we replaced \(x_T = x_F\) with the vertices of \(X_F\) in definitions (7) and (11).

In order to compute the first feasible trajectory needed to initialize the LMPC, we set \(\theta^0_t = \gamma(s^0_t)\) and we controlled the acceleration to steer the dubins car from \(x_S\) to the terminal set \(X_F\). Notice that for \(\theta^0_t = \gamma(s^0_t)\) the system is linear and consequently Assumption 4 is satisfied for iteration \(j = 0\). However, for \(j > 0\) it is hard to verify analytically if Assumption 4 holds, therefore we used a sampling strategy to approximately check this condition, as shown in the Appendix X-B.

We test different LMPC policies synthesised with \(N = 4\) and using the strategy described in Section VIII for \(P = \{15, 25, 50, 200\}\) data points and \(i = \{1, 3, 4, 10\}\) iterations. Figure 12 shows time steps \(T^j\) needed to reach the terminal set (27). We notice that after few iterations all LMPC policies converged to a steady state behavior which steers the system to the goal set in 16 time steps. Also in this example, convergence is reached faster as more data points are used in the LMPC synthesis process.

Finally, Figures 13 and 14 show that closed-loop trajectories and associated input sequence at converged. In order
to minimize the travel time, the LMPC cuts the curve and it drives the system to a state in the terminal set which is close to the road boundaries. Furthermore, we notice that the controller saturates the acceleration and deceleration constraints, as we expect from an optimal solution to a minimum time optimal control problem.

VIII. CONCLUSIONS

We presented a time varying Learning Model Predictive Controller (LMPC) for time optimal control problem. The proposed control framework uses historical data to construct time varying safe sets and approximations to the value function. Furthermore, we showed that these quantities can be convexified to synthesize a relaxed LMPC policy. We showed that the proposed control strategies guarantee safety, finite time convergence and performance improvement with respect to previous task execution. Finally, we tested the controllers on two dubins car minimum time optimal control problems.

IX. ACKNOWLEDGMENT

The authors would like to thank Nicola Scianca from Sapienza University of Roma for the interesting discussions on repetitive LMPC. Some of the research described in this review was funded by the Hyundai Center of Excellence at the University of California, Berkeley. This work was also sponsored by the Office of Naval Research. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of Naval Research or the US government.

X. APPENDIX

A. Nonlinear Double Integrator

In this section, we show that the following nonlinear double integrator

\[ z_{k+1} = \begin{bmatrix} x_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k dt \\ v_k + g(v_k) a_k dt \end{bmatrix} = f_n(z_k, a_k) \]

for \( g(v_k) = \frac{1}{2} \left( 1 - \frac{v_k^2}{v_{max}^2} \right) \) satisfies Assumption 2. First we notice that given \( P \) states \( x_i \in \mathcal{X} \) and inputs \( u_i \in \mathcal{U} \) for \( i \in \{1, \ldots, P\} \) and a set of multiplies \( \{\lambda_0, \ldots, \lambda_P\} \geq 0 \)

\[ x = \sum_{k=1}^{P} \lambda_k x_k \text{ and } \sum_{k=1}^{P} \lambda_k = 1, \]

we have that

\[ \sum_{k=0}^{P} \lambda_k f_n(z_k, a_k) = \sum_{k=0}^{P} f_n(\lambda_k z_k, a) \]

where

\[ a = \frac{\sum_{k=0}^{P} \lambda_k g(v_k) a_k}{g(\sum_{k=0}^{P} \lambda_k v_k)}. \]
Finally, by concavity of $g(v_k) \geq 0$ for all $z_k = [x_k, v_k]^T \in \mathcal{X}$ we have that
\[
\min_{k=1, \ldots, P} a_k \leq \frac{\sum_{k=0}^P \lambda_k g(v_k) a_k}{g\left(\sum_{k=0}^P \lambda_k v_k\right)} \leq \max_{k=1, \ldots, P} a_k
\]
and therefore Assumption 2 is satisfied.

**B. Dubins Car**

We used a sampling strategy to check if Assumption 4 is approximately satisfied. Before running the $(j+1)$th iteration of the relaxed LMPC, we randomly sample
\[
x^{(l)} \in \text{Conv} \left( \bigcup_{\{t, i\} \in I(x^{(l)})} x_{i}^t \right)
\]
for $l \in \{0, \ldots, 10^5\}$, where $I(x^{(l)})$ is defined in Assumption 4. Afterwards, we checked if $\exists u \in \mathcal{U}$ such that
\[
f(x^{(l)}, u) \in \text{Conv} \left( \bigcup_{\{t, i\} \in I(x^{(l)})} f(x_{i}^t, u_{i}^t) \right).
\]
For all tested data points and iterations Assumption 4 was satisfied. Notice that as we used a subset of the stored data to construct (7) and (9), we checked Assumption 4 for the stored closed-loop trajectory performed at iteration $j$. Finally, for $j = \{3, 5, 10\}$ Figures 15, 16 and 17 show the randomly generated states where we have verified that Assumption 4 holds.

![Fig. 15. Randomly sampled states used to check that Assumption 4 is approximately satisfied.](image)

![Fig. 16. Randomly sampled states used to check that Assumption 4 is approximately satisfied.](image)

![Fig. 17. Randomly sampled states used to check that Assumption 4 is approximately satisfied.](image)

**REFERENCES**

[1] I. Bogner and F. L. Kazda, “An investigation of the switching criteria for higher order contactor servomechanisms,” *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, vol. 73, no. 3, pp. 118–127, 1954.

[2] R. Bellman, I. Glicksberg, and O. Gross, “On the bang-bang control problem,” *Quarterly of Applied Mathematics*, vol. 14, no. 1, pp. 11–18, 1956.

[3] R. V. Gamkrelidze, “On the theory of optimal processes in linear systems,” Joint Publications Research Service Arlington VA, Tech. Rep., 1961.

[4] J. P. LaSalle, “The time optimal control problem;” *Contributions to the theory of nonlinear oscillations*, vol. 5, pp. 1–24, 1959.

[5] V. Boltyanskiy, R. V. Gamkrelidze, and L. Pontryagin, “Theory of optimal processes,” Joint Publications Research Service Arlington VA, Tech. Rep., 1961.

[6] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[7] J. E. Bobrow, S. Dubowsky, and J. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[8] N. R. Kapania, J. Subosits, and J. C. Gerdes, “A sequential two-step algorithm for fast generation of vehicle racing trajectories,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091005, 2016.

[9] Z. Shiller and H.-H. Lu, “Computation of path constrained time optimal motions with dynamic singularities,” *Journal of dynamic systems, measurement, and control*, vol. 114, no. 1, pp. 34–40, 1992.

[10] A. Nagy and I. Vajk, “Sequential time-optimal path-tracking algorithm for robots,” *IEEE Transactions on Robotics*, 2019.

[11] V. Rajan, “Minimum time trajectory planning,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 759–764.
[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” IEEE Transactions on Automatic Control, vol. 54, no. 10, pp. 2318–2327, 2009.

[13] E.-B. Meier and A. E. Ryson, “Efficient algorithm for time-optimal control of a two-link manipulator,” Journal of Guidance, Control, and Dynamics, vol. 13, no. 5, pp. 859–866, 1990.

[14] G. Manor, J. Z. Ben-Asher, and E. Rimon, “Time optimal trajectories for a mobile robot under explicit acceleration constraints,” IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 5, pp. 2220–2232, 2018.

[15] W. L. Scott and N. E. Leonard, “Time-optimal trajectories for steered agent with constraints on speed and turning rate,” in ASME 2016 Dynamic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection, 2017.

[16] K. F. Graham and A. V. Rao, “Minimum-time trajectory optimization of multiple revolution low-thrust earth-orbit transfers,” Journal of Spacecraft and Rockets, vol. 52, no. 3, pp. 711–727, 2015.

[17] J. E. Bobrow, “Optimal robot plant planning using the minimum-time criterion,” IEEE Journal on Robotics and Automation, vol. 4, no. 4, pp. 443–450, 1988.

[18] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, “Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm,” in 2016 European Control Conference (ECC). IEEE, 2016, pp. 141–147.

[19] S. Al Homsi, A. Sherikov, D. Dimitrov, and P.-B. Wieber, “A hierarchical approach to minimum-time control of industrial robots,” in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016, pp. 2368–2374.

[20] U. Rosolia and F. Borrelli, “Learning model predictive control for iterative tasks. a data-driven control framework,” IEEE Transactions on Automatic Control, vol. 63, no. 7, pp. 1883–1896, 2017.

[21] ——, “Learning model predictive control for iterative tasks: A computationally efficient approach for linear system,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 3142–3147, 2017.