2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

# Learning Convex Terminal Costs for Complexity Reduction in MPC

Shokhjakon Abdufattokhov*, Mario Zanon, Alberto Bemporad

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683069

*Abstract*— **Despite recent advances in computing hardware and optimization algorithms, solving model predictive control (MPC) problems in real time still poses some technical challenges when long prediction and control horizons are used, due to the presence of several optimization variables and constraints. In this paper, we propose to reduce the computational burden by shortening the prediction and control horizon to a single step while preserving good closed-loop performance. This is achieved by using machine learning techniques to construct a tailored quadratic and convex terminal cost that approximates the cost-to-go function of constrained linear (possibly parameter-dependent) MPC formulations. The potentials of the proposed MPC with Learned Terminal Cost (LTC-MPC) approach is demonstrated in two numerical examples.**

## I. Introduction

Model predictive control (MPC) is an optimization-based control technique that has been applied successfully in many real-life problems, thanks to its ability to achieve good closed-loop performance while enforcing constraints on both inputs and outputs of multi-variable systems [1]–[3]. Despite recent advances in computing hardware and optimization algorithms, the numerical operations required by MPC might be excessive in certain applications characterized by high sampling frequencies and/or cheap processing units.

One option to overcome the computational burden is to use explicit MPC, which precomputes the feedback control law as a piecewise affine function of the state vector by solving offline a multi-parametric programming problem [4]–[6]. The main drawback of this approach is that exact multiparametric solutions are limited to small-scale time-invariant MPC problems, since the high memory requirements should be satisfied to store the polyhedral partitions and the resulting control gains. On the other hand, to simplify the complexity and to overcome the issues of the exact explicit MPC scheme for large-scale problems, approximate explicit MPC techniques have been proposed in the literature, such as using neural networks [7]–[9], orthogonal trees [10], piecewise affine basis functions [11] to mention a few. However, approximate explicit solutions obtained by function approximation methods might be difficult to get due to the complexity of the control profile, and moreover constraints should be included to preserve feasibility of at least the commanded input.

When implicit MPC is the only viable option, the computational burden can be reduced by reducing the number of degrees of freedom in the optimization, e.g., by shortening the control horizon (i.e., the number of free control moves). This is done by using move blocking [12], [13], that describes the idea of letting control input be time-varying over the first few prediction steps and then forcing it to be constant over the remaining steps. A different method to reduce the number of degrees of freedom by parametrizing both the input and state sequences with basis functions is investigated in [14]. Alternatively, the authors in [15] propose to use a method based on the combination of a singular value decomposition and machine learning classification algorithms to choose basis functions that are used to parametrize the decision variables as an affine function of a small number of optimization variables. However, computations may remain significant in these methods if the prediction horizon and/or the number of output constraints are large. Additionally, these approaches do not adapt the objective function to (at least partially) compensate for the optimality loss due to the reduction of degrees of freedom.

In this paper, we propose an approach to significantly reduce the online computational burden of implicit MPC by considering a prediction and control horizon of length one. If naively implemented, this strategy results in a considerable loss of closed-loop performance, hence the need of introducing a suitable terminal cost. An attempt to construct an appropriate terminal cost using state decomposition has been investigated in [16], while [17], [18] proposes an algorithm that optimizes a time-varying terminal cost. In both cases, a prediction and control horizon larger than one is considered. Ideally, the terminal cost should be the optimal cost-to-go from the second step to the end of the prediction. Unfortunately, the cost-to-go is typically a complicated function. In case of linear time invariant systems with constraints, the cost-to-go is piecewise quadratic. Such a function is both difficult to compute, and would make the online optimization problem difficult to solve.

To keep online optimization simple, in this paper we propose to learn a cost-to-go function that is quadratic and positive semidefinite with respect to the command input (i.e., the optimization vector) and rather arbitrary with respect to the initial state, reference signals, and possibly other parameters entering the MPC problem. We employ machine learning algorithms to learn the terminal cost by fitting samples of the cost-to-go, that we assume informative enough. We therefore call our approach MPC with Learned Terminal Cost (LTC-MPC).

The remainder of this paper is organized as follows. After providing formulations to construct a MPC optimization problem with a single optimizer in Section II, we discuss how to learn a quadratic convex approximation of the cost-

*Corresponding author: s.abdufattokhov@imtlucca.it. This paper was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems" , contract no. 2017J89ARP.

to-go function in Section III. In section IV, we demonstrate the effectiveness of the proposed approach in simulations with two examples. Finally, concluding remarks are drawn in section V.

## II. Problem formulation

In this section, we first formulate a standard MPC controller based on a linear time-varying model with full-length prediction horizon. This will be reduced to a linear MPC controller with prediction horizon one by learning a suitable convex terminal cost, that approximates the optimal cost-to-go under input and output constraints from time 2 to $N$.

Consider the following MPC problem

$$\begin{aligned}
\min_{U,E}\ & \sum_{k=0}^{N-1} \ell_k(y_{k+1}, u_k, \Delta u_k, \epsilon_{k+1}, p_t) && (1)\\
\text{s.t. } & x_0 = M_x p_t \\
& x_{k+1} = A_k(p_t)x_k + B_k(p_t)u_k + b_k(p_t) && k \in \mathbb{I}_0^{N-1}\\
& u_k^{\min}(p_t) \le u_k \le u_k^{\max}(p_t) && k \in \mathbb{I}_0^{N-1}\\
& \Delta u_k^{\min}(p_t) \le \Delta u_k \le \Delta u_k^{\max}(p_t) && k \in \mathbb{I}_0^{N-1}\\
& \Delta u_k = 0 && k \in \mathbb{I}_{N_u}^{N-1}\\
& y_k = C_k(p_t)x_k + c_k(p_t) && k \in \mathbb{I}_1^{N}\\
& -\epsilon_k + y_k^{\min}(p_t) \le y_k \le y_k^{\max}(p_t) + \epsilon_k && k \in \mathbb{I}_1^{N}\\
& \epsilon_k \ge 0 && k \in \mathbb{I}_1^{N}
\end{aligned}$$

where $t$ is the current time instant, $N$ the prediction horizon, $N_u$ the control horizon, $\mathbb{I}_a^b$ is the set of all integers in the interval $[a, b]$ , $U = (u_0, \ldots, u_{N-1})$ is the sequence of manipulated variables $u_k \in \mathbb{R}^{n_u}$ to optimize, $x_k \in \mathbb{R}^{n_x}$ is the state predicted $k$-steps ahead, $y_k \in \mathbb{R}^{n_y}$ is the output vector, and we define $\Delta u_{k+1} := u_{k+1} - u_k$. Parameter $p \in \mathbb{R}^{n_p}$ is an exogenous vector of parameters that includes the initial state $x_0$, such that $x_0 = M_x p_t$, the previous input $u_{-1} = M_u p_t$, reference signals, and other known parameters affecting the affine prediction model described by $A_k(p_t)$, $B_k(p_t)$, $b_k(p_t)$, $C_k(p_t)$, $c_k(p_t)$, $\ell_k : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \in \mathbb{R}_{\ge 0}$ are convex stage cost functions, such as quadratic functions, and $E = (\epsilon_1, \ldots, \epsilon_N)$ is a vector of slack variables $\epsilon_k \in \mathbb{R}_{\ge 0}^{n_y}$ used to soften output constraints.

By Bellman's principle of optimality, Problem (1) can be reformulated as

$$\begin{aligned}
\min_{u_0,\epsilon_1}\ & \ell_0(y_1, u_0, \Delta u_0, \epsilon_1, p_t) + V(x_1, p_t) && (2a)\\
\text{s.t. } & x_0 = M_x p_t && (2b)\\
& x_1 = A_0(p_t)x_0 + B_0(p_t)u_0 + b_0(p_t) && (2c)\\
& u_0^{\min}(p_t) \le u_0 \le u_0^{\max}(p_t) && (2d)\\
& \Delta u_0^{\min}(p_t) \le \Delta u_0 \le \Delta u_0^{\max}(p_t) && (2e)\\
& y_1 = C_1(p_t)x_1 + c_1(p_t) && (2f)\\
& -\epsilon_1 + y_1^{\min}(p_t) \le y_1 \le y_1^{\max}(p_t) + \epsilon_1 && (2g)\\
& \epsilon_1 \ge 0 && (2h)
\end{aligned}$$

where $V : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the *cost-to-go* defined by the sum of stage costs starting from 2 to $N$, that is

$$\begin{aligned}
& V(x_1, p_t) = \\
& \min_{U_1,E_1}\ \sum_{k=1}^{N-1} \ell_k(y_{k+1}, u_k, \Delta u_k, \epsilon_{k+1}, p_t) && (3)\\
& \text{s.t. } x_{k+1} = A_k(p_t)x_k + B_k(p_t)u_k + b_k(p_t) && k \in \mathbb{I}_1^{N-1}\\
& \quad u_k^{\min}(p_t) \le u_k \le u_k^{\max}(p_t) && k \in \mathbb{I}_1^{N-1}\\
& \quad \Delta u_k^{\min}(p_t) \le \Delta u_k \le \Delta u_k^{\max}(p_t) && k \in \mathbb{I}_1^{N-1}\\
& \quad \Delta u_k = 0 && k \in \mathbb{I}_{N_u}^{N-1}\\
& \quad y_k = C_k(p_t)x_k + c_k(p_t) && k \in \mathbb{I}_2^{N}\\
& \quad -\epsilon_k + y_k^{\min}(p_t) \le y_k \le y_k^{\max}(p_t) + \epsilon_k && k \in \mathbb{I}_2^{N}\\
& \quad \epsilon_k \ge 0 && k \in \mathbb{I}_2^{N}
\end{aligned}$$

where $U_1 = (u_1, \ldots, u_{N-1})$, $E_1 = (e_2, \ldots, e_N)$. In case $\ell_k$ are convex quadratic functions, the optimal value function is convex and piecewise quadratic in $x_1$ [5], and in the absence of input and output constraints it is quadratic. More generally, $V$ is a convex function of $x_1$ [19], although it may not be easy to express analytically.

### A. Complexity reduction

While appealing due to the simplicity of implementation, MPC without terminal constraints such as problem (1) may require a long prediction horizon to ensure stability, good closed-loop performance, and constraint satisfaction [20]. Additionally, in many cases, the system dynamics are nonlinear and given in continuous-time, such that linearization and discretization are required. Consequently, increasing $N$ directly leads to a significant increase in the computations required to solve the problem in (1).

If the cost-to-go function $V(x_1, p_t)$ were known and simple enough, one could solve problem (1) more easily in the form (2), as the number of optimization variables and constraints would be drastically reduced. The main idea of this paper is to construct an approximation

$$\hat{V}(x_1, p_t) = (x_1 - \hat{x}(p_t))' \hat{P}(p_t)(x_1 - \hat{x}(p_t)) \qquad (4)$$

of $V(x_1, p_t)$, where $\hat{x}(p_t) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ and $\hat{P}(p_t) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_x \times n_x}$ are functions of $p_t$ to be learned, as we will describe next. Accordingly, problem (2) is approximated as

$$\begin{aligned}
\min_{u_0,\epsilon_1}\ & \ell_0(y_1, u_0, \Delta u_0, \epsilon_1, p_t) + \hat{V}(x_1, p_t) && (5a)\\
\text{s.t. } & \text{(2b)} - \text{(2h)} && (5b)
\end{aligned}$$

which we call MPC with Learned Terminal Cost (LTC-MPC). Parameterizing $\hat{V}$ as in (4) leads to two main advantages: $(i)$ when $\ell_0$ is also quadratic, the LTC-MPC problem (5) is a small quadratic program (QP), $(ii)$ the input $u_0$ and slack $\epsilon_1$ remain as optimization variables, so that input and output constraints can be enforced exactly at the initial step, and in particular the optimal input $u_0^\star(p_t)$ applied to the process is feasible.

The LTC-MPC law is evaluated on line as described in Algorithm 1.

**Algorithm 1:** LTC-MPC law

**Input:** $p_t$

**Output:** $u_0^\star$

evaluate $A_0(p_t)$, $B_0(p_t)$ and $C_1(p_t)$;

evaluate matrix $\hat{P}(p_t)$ and vector $\hat{x}(p_t)$;

solve the LTC-MPC problem (5) and get $u_0^\star$;

apply $u_t = u_0^\star(p_t)$.

## III. Learning the cost-to-go function

In this section, we focus learning an approximation $\hat{V}$ of the cost-to-go function $V$ as in (4). As we need to ensure convexity of $\hat{V}$ with respect to $x_1$, the learning algorithm must ensure that $\hat{P}(p_t)$ is positive semidefinite for all $p_t$. To this end, instead of directly learning $\hat{P}$, we learn a lower-triangular matrix $\hat{L} : \mathbb{R}^{n_p} \to \mathbb{R}^{n_x \times n_x}$, so that

$$\hat{P}(p_t) = \hat{L}(p_t)\hat{L}(p_t)' \tag{6}$$

is positive semidefinite by construction for all possible values of the parameter vector $p_t$. The formulation (4), (6) requires solving a regression problem parameterized by $\frac{n_x(n_x+1)}{2}+n_x$ functions and $n_p$ features (or inputs). Note that directly parameterizing the symmetric part of $\hat{P}$ is also possible, and still involves $\frac{n_x(n_x+1)}{2}$ predictors. The advantage of such a parameterization is the reduced nonlinearity of the learning problem, the disadvantage is that one must introduce constraints in the learning process to ensure that $\hat{P}(p_t) \succeq 0$ for all $p_t \in \mathbb{R}^{n_p}$. A thorough investigation comparing the alternative parameterizations of $\hat{P}$ is left for future research.

Given a dataset of $M$ samples $(p_i, x_{1,i}, V_{1,i})$, with $V_{1,i} = V(x_{1,i}, p_i)$ as in (3), we solve the following regression problem

$$\min_{\hat{L},\hat{x}} \ \frac{1}{M}\sum_{i=1}^{M} \phi(V_{1,i}, (x_{1,i}-\hat{x}(p_i))'\hat{L}(p_i)\hat{L}(p_i)'(x_{1,i}-\hat{x}(p_i))) \tag{7}$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}$ is a training loss function. In this paper, we consider the commonly used mean squared error (MSE) loss function, i.e., $\phi(a,b) = (a-b)^2$, although other functions could be used; for example if the values $V_{1,i}$ range between very small and very large values, one could use $\phi(a,b) = (\log(a) - \log(b))^2$.

The general problem (7) is infinite-dimensional. We adopt a parametric approach in this paper by modeling $\hat{L}$, $\hat{x}$ as feedforward neural networks (NNs) $\hat{L}_\theta$, $\hat{x}_\theta$. Let $\theta \in \mathbb{R}^{n_\theta}$ denote the overall set of weights/bias terms of NNs. Then problem (7) is approximated by

$$\min_{\theta} \ \gamma\|\theta\|_2^2 + \frac{1}{M}\sum_{i=1}^{M} \phi(V_{1,i}, V_\theta(x_{1,i})) \tag{8a}$$

where $\gamma$ is an $L_2$-regularization parameter and

$$V_\theta(x_{1,i}) := (x_{1,i} - \hat{x}_\theta(p_i))'\hat{L}_\theta(p_i)\hat{L}_\theta(p_i)'(x_{1,i} - \hat{x}_\theta(p_i)) \tag{8b}$$

We refer to [21] for a wide range of possible alternative loss functions, regularization terms, and optimization algorithms to solve (8), as well as to [22], [23] for well-established open-source frameworks for solving the training process.

The posed learning problem requires a training dataset that is rich enough to make LTC-MPC (5) a good approximation of the original MPC problem (1). A possibility is to generate random samples $(p_i, x_{1,i})$ that well cover a given operating range of states, set-points, and other parameters of interest. To avoid the curse-of-dimensionality issue when the number of components of $(p_i, x_{1,i})$ is high, a possibility is to run a closed-loop simulation in which the next state vector coincides with

$$x_{1,i} = A_0(p_i)M_x p_i + B_0(p_i)u_0^\star(p_i) + b_0(p_t) \tag{9}$$

and $u_0^\star(p_i)$ is the first sample of the optimal sequence returned by the full MPC problem (1), for $i = 1, \ldots, M$. An intermediate approach that ensures better exploration is to generate random perturbations of $x_{1,i}$ and $p_i$ around the obtained closed-loop signals. How to best generate suitable perturbations is application dependent. Note that in case $x_{1,i}$ is the optimal state given by (9) (unperturbed case) the cost-to-go $V_{1,i}$ is immediately available as a by-product of the solution of (1). Otherwise, for a generic (perturbed) value $x_{1,i}$ one necessarily needs to evaluate the cost-to-go as per (3).

## IV. Illustrative Example

In this section, we demonstrate the potentials of the proposed strategy on two simulation examples. First, we consider a linear quadratic regulation (LQR) problem without constraints, and show that the optimal cost-to-go solving the MPC problem (1) is recovered from data. Second, we consider a simplified lane-keeping problem for autonomous driving with full/partial road preview and constraints on inputs and outputs.

To run our numerical experiments, we collect $M$ samples in closed-loop without perturbation as follows: $(i)$ we generate a set of $N_{\text{init}}$ random initial conditions and reference signals; $(ii)$ starting from each initial condition, we run a closed-loop simulation over a horizon $N_{\text{sim}}$ using the full MPC controller and evaluate the optimal cost-to-go $V(x_1, p_t)$ on the optimal predicted states $x_1$ as in (9) obtained during the simulation. Out of the resulting $M = N_{\text{init}}N_{\text{sim}}$ samples of $(x_1, V)$, we use $M_{\text{train}} = 0.6M$ samples for learning the parameters of the model, $M_{\text{val}} = 0.2M$ samples for the validation that helps to specify the architecture of the model, for example to choose the number of hidden units, and $M_{\text{test}} = 0.2M$ test samples used to assess the performance of the specified model. We validate NNs model by measuring the prediction accuracy using the commonly used Normalized Root Mean Square Error($NRMSE$) metric and the $R^2_{\text{score}}$-coefficient of determination. We use Pytorch [22] to fit the NNs required to approximate the cost-to-go function. All simulations are executed in MATLAB 2018b on a machine equipped with an Intel Core i5-5200U (2.7GHz) processor.

### A. LQR problem

We consider the following discrete Linear Time Invariant (LTI) system

$$x_{i+1} = \begin{bmatrix} 0.9 & -0.2 \\ 0.1 & 1.0 \end{bmatrix} x_i + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} u_i$$

We formulate an LQR problem over a prediction horizon $N = 30$ using state weighting matrix $Q = I$, control weighting matrix $R = 0.1$ and terminal cost matrix $Q$, and get the closed-loop solution through running Riccati backward iterations. We run $N_{\text{init}} = 150$ simulations of length $N_{\text{sim}} = 40$ steps each, visualized in Figure 1, resulting in a dataset of $M = 6000$ samples. The parameter vector $p_i \in \mathbb{R}^{5\times 1}$ consists of the initial state $x_{0,i} \in \mathbb{R}^{2\times 1}$, reference states $x_i^{\text{r}} \in \mathbb{R}^{2\times 1}$ and a reference input $u_i^{\text{r}} \in \mathbb{R}$ (both constant over the prediction horizon), such that

$$p_i = (x_{0,i},\ x_i^{\text{r}},\ u_i^{\text{r}})$$

Since $\Delta u$ is not used in this example, we eliminated it from the formulation and we removed $u_{-1,i}$ from $p_i$, since it becomes unnecessary. We fix $\hat{x}(p_i) = x_i^{\text{r}}$ and learn matrix $\hat{L}(p_i)$ using a neural networks composed of fully connected layers, with an input layer with 5 neurons is followed by one hidden layer with 100 neurons and an output layer with 3 neurons. A sigmoidal function is chosen as an activation function for the hidden layer, while a linear activation function is applied to the output layer. We solve problem (8) using the ADAM algorithm [24] with a learning rate $\alpha = 10^{-2}$, coefficients used for computing running averages of gradient and its square exponential decay rates $\beta = (0.95, 0.995)$, $L_2$-regularization parameter $\gamma = 10^{-4}$ and $epochs = 1000$. The prediction accuracy analysis of chosen model is given in Table I.

TABLE I: LQR cost-to-go prediction accuracy analysis.

| | Training | Validation | Test |
|---|---|---|---|
| NRMSE | 0.005 | 0.004 | 0.004 |
| $R^2_{\text{score}}$ | 0.995 | 0.995 | 0.995 |

We simulate the system with $N_{\text{sim}} = 50$, constant references $x_r = \begin{bmatrix} 0 & 2 \end{bmatrix}'$ and $u_r = 4$. In order to check the consistency between full MPC and LTC-MPC, we investigate the maximum of the absolute difference between terminal weight matrices, respectively $P_{\text{full}}$ and $\hat{P}$, and linear state feedback gain matrices, respectively $G_{\text{full}}$ and $\hat{G}$. In the proposed simulation, we obtain a good approximation:

$$\max_{i=1,\dots,N_{\text{sim}}} \frac{\|\hat{P}(p_i) - P_{\text{full}}\|_{\max}}{\|P_{\text{full}}\|_{\max}} = 0.08$$

$$\max_{i=1,\dots,N_{\text{sim}}} \frac{\|\hat{G}_i - G_{\text{full}}\|_{\max}}{\|G_{\text{full}}\|_{\max}} = 0.03$$

The closed-loop performance of LTC-MPC is close to the one of full MPC, as displayed in Figure 2.

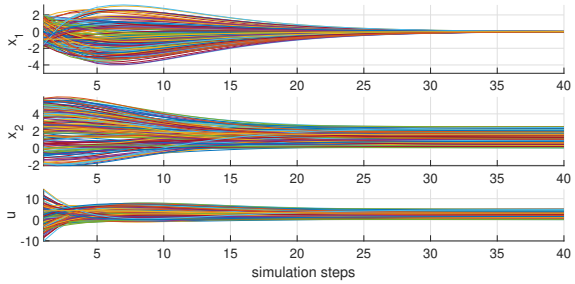



Fig. 1: Trajectory samples used in the LQR training.

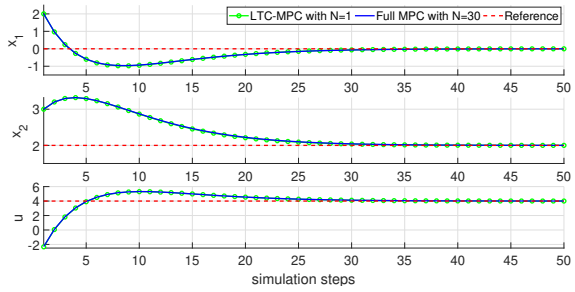



Fig. 2: Trajectory tracking results. The top subplot shows trajectories drawn for state one $x_1$ by full MPC with $N = 30$ (blue line) and LTC-MPC with $N = 1$ (green circle), while the middle and lower subplots illustrates lines for state two $x_2$ and control input $u$, respectively.

### B. Lane-keeping problem

To pose a control problem for lane-keeping in autonomous driving, consider the following continuous-time bicycle-like kinematic model of the vehicle dynamics

$$\begin{cases} \dot{s}_{\text{x}} &= v\cos(\psi + \delta) \\ \dot{s}_{\text{y}} &= v\sin(\psi + \delta) \\ \dot{\psi} &= \dfrac{v}{W_b}\sin(\delta) \end{cases} \tag{10}$$

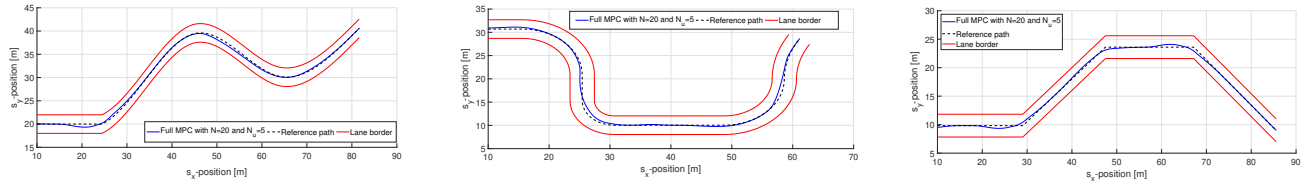
where $(s_{\text{x}}, s_{\text{y}})$ [m] is the Cartesian position of the front wheel on a fixed reference frame, $\psi$ [rad] is the orientation of the vehicle with respect to the $x$-axis, $\delta$ [rad] is the commanded steering angle, $v$ [m/s] is the commanded longitudinal velocity, and $W_b = 4.5$ m is the wheelbase. We then summarize the model as $\dot{x} = f(x, u)$, with $x = [s_{\text{x}}\ s_{\text{y}}\ \psi]'$ and $u = [v\ \delta]'$, and $f : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^3$ is given by (10). The model is linearized around the current state $x_t$ and the last computed optimal input $u^\star_{1,t-1}$, and discretized using the first-order forward Euler method with sampling time $T_{\text{s}} = 0.05$ s, obtaining the linear parameter varying (LPV) system as in (1) with

$$A_k(p_t) = I + T_s \frac{\partial f(x_t, u^\star_{1,t-1})}{\partial x}$$

$$B_k(p_t) = T_s \frac{\partial f(x_t, u^\star_{1,t-1})}{\partial u},$$

$$b_k(p_t) = x_t + T_s f(x_t, u^\star_{1,t-1}) - A_k(p_t)x_t - B_k(p_t)u^\star_{1,t-1}$$

$$C_k(p_t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad c_k(p_t) = 0$$




(a) (b) (c)

Fig. 3: Lane-change maneuver examples.

where the parameter $p_t \in \mathbb{R}^{5+2N_r}$ is given by

$$p_t = (x_t,\ u^{\star}_{1,t-1},\ y^{\mathrm{r}}_{1,t},\ \ldots,\ y^{\mathrm{r}}_{N_r,t}) \tag{11}$$

assuming that that at each time $t$ the future samples $y^{\mathrm{r}}_{1,t}, \ldots, y^{\mathrm{r}}_{N_r,t}$ of the reference are known, $N_r \leq N$.

Our goal is to make the vehicle follow the reference path while staying within the lane margins. For this purpose, we make the output $y$ track reference signals $y^{\mathrm{r}}$ by considering the following quadratic stage cost

$$\ell_k(y_{k+1}, \Delta u_k, \epsilon_{k+1}, p_t) = \\ \|y_{k+1} - y^{\mathrm{r}}_{k+1}\|_2^2 + \|\Delta u_k\|_R^2 + 100\|\epsilon_{k+1}\|_2^2$$

with $R = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}$. Moreover, the following upper and lower bounds are included in the control problem formulation:

$$u_k^{\min} = \begin{bmatrix} -5.5 \\ -\pi/4 \end{bmatrix} \qquad u_k^{\max} = \begin{bmatrix} 19.5 \\ \pi/4 \end{bmatrix}$$
$$\Delta u_k^{\min} = \begin{bmatrix} -1.0 \\ -\pi/18 \end{bmatrix} \qquad \Delta u_k^{\max} = \begin{bmatrix} 5.0 \\ \pi/18 \end{bmatrix}$$
$$y_k^{\min} = y_k^{\mathrm{r}} - 2 \qquad y_k^{\max} = y_k^{\mathrm{r}} + 2$$

Simulations are carried out in two scenarios. First, we design the vector $p_t$ with $N_r = 1$ and learn the cost-to-go function given in (8). Second, we include full preview information in the vector $p_t$ for a fair comparison with the Full MPC scheme. In both cases, we parameterize the 6 components of $\hat{L}_\theta(p_i)$ and the 3 components of $\hat{x}_\theta(p_i)$ as NNs consisting of an input layer with $5 + 2N_r$ neurons, one hidden layer with 200 neurons and sigmoidal activation functions, and a linear output layer with 9 neurons. We run $N_{\mathrm{init}} = 150$ simulations of length $N_{\mathrm{sim}} = 120$ using 3 different lane-change maneuvers as displayed in Figure 3, yielding a dataset of $M = 18000$ samples. We employ the ADAM algorithm with a learning rate $\alpha = 10^{-4}$, coefficients used for computing running averages of gradient and its square exponential decay rates $\beta = (0.99, 0.995)$, $L_2$-regularization parameter $\gamma = 10^{-5}$ and $epochs = 2000$ to train both model. The prediction accuracy analysis of chosen models is given in Table II. As expected, one can see that the predictions of the cost-to-go function trained with full preview are more precise than the function trained with a 1-step-ahead preview. The full MPC with $N = 20$, $N_u = 5$ and the LTC-MPC controllers are tested in simulation by running the vehicle from the initial state $x_0 = [10\ 29.5\ 0]'$, assuming $u_{-1} = [0\ 0]'$. The quadratic programs associated

TABLE II: Lane-keeping cost-to-go prediction accuracy analysis ($NRMSE$ / $R^2_{\mathrm{score}}$).

| $N_r$ | Training | Validation | Test |
|---|---|---|---|
| 1 | 0.03 / 0.90 | 0.05 / 0.88 | 0.05 / 0.87 |
| 20 | 0.01 / 0.98 | 0.02 / 0.96 | 0.03 / 0.94 |

with both MPC problems are solved using qpOASES [25] and simulation results are the obtained. The Figure 4 and Figure 5 show that LTC-MPC is able to keep the vehicle within the lane margins even in case the 1-step-ahead preview is used. While this results in a large performance loss, good performance is recovered by using the full preview information to compute the terminal cost. Moreover, the cost-to-go predictions and corresponding approximation errors with respect to Full MPC scheme are illustrated in Figure 6.

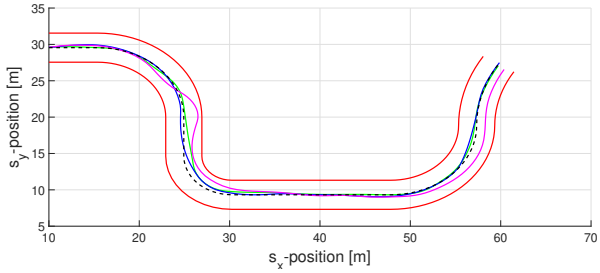



Fig. 4: Lane-keeping control results. The reference path (black-dashed line), Full MPC with $N = 20$ and $N_u = 5$ (blue line), LTC-MPC with $N = N_u = 1$ and $N_r = 20$ (green line), LTC-MPC with $N = N_u = N_r = 1$ (magenta line) and lane borders (red line).

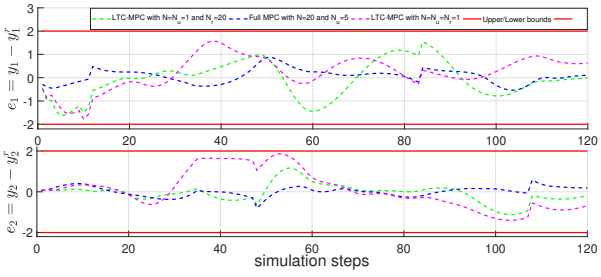



Fig. 5: Output tracking errors.

The computation time required by the full and LTC-MPC controllers shown in Figure 7 are measured on an Intel

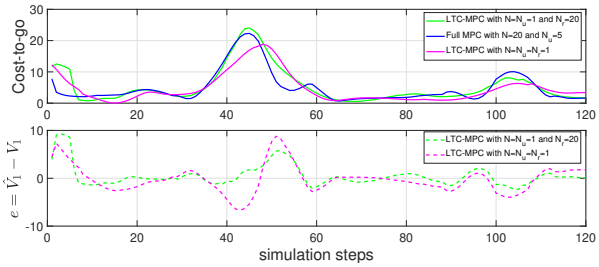



Fig. 6: Cost-to-go predictions(up) and corresponding approximation errors with respect to Full MPC scheme(down).

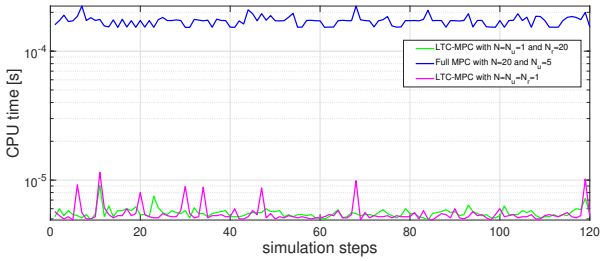



Fig. 7: Computational time for each MPC iteration of the closed loop simulation.

Core i5-5200U (2.7GHz) processor. Table III illustrates the comparison of the corresponding average and worst-case CPU time required to solve the QP associated with the MPC law, and the time to evaluate the NN yielding $\hat{L}(p_t)$ and $\hat{x}(p_t)$. Note that while the solution time of the QP varies from one sample step to the next due to the possible different number of active-set iterations, evaluating the NNs requires a fixed number of floating-point operations at each execution.

TABLE III: CPU times in [ms] for evaluating the full MPC (excluding condensing) and the LTC-MPC schemes at each sample step.

| | $N_r$ | $N$ | $N_u$ | NNs | QP worst-case | QP average |
|---|---|---|---|---|---|---|
| Full MPC | 20 | 20 | 5 | - | 0.220 | 0.171 |
| LTC-MPC | 20 | 1 | 1 | 0.094 | 0.010 | 0.005 |
| | 1 | 1 | 1 | 0.051 | 0.012 | 0.006 |

## V. Conclusion and Future Work

This paper has presented a novel approach to construct convex terminal costs for MPC problems that allow a drastic shortening of the prediction horizon, with consequent considerable reduction of computation time. In order to preserve good performance, the terminal cost is computed as an approximation of the convex MPC cost-to-go, and is supported by neural networks. The reported numerical example showed that closed-loop performance is preserved in spite of the introduced approximation.

Future work will address the investigation of theoretical guarantees of feasibility and stability of the LTC-MPC law, and the extension of the approach to nonlinear MPC formulations.

## References


[1] E. Camacho and C. Bordons, *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing, London: Springer, 1999.

[2] D. Mayne, J. Rawlings, and M. Diehl, *Model Predictive Control: Theory and Design*. Madison,WI: Nob Hill Publishing, LCC, 2 ed., 2018.

[3] F. Borrelli, A. Bemporad, and M.Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[4] M.Seron, J. DeDona, and G. Goodwin, "Global analytical model predictive control with input constraints," in *In Proc. 39th IEEE Conf. on Decision and Control*, (Sydney, Australia), pp. 154–159, 2000.

[5] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[6] A. Bemporad, "A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares," *IEEE Trans. Automatic Control*, vol. 60, no. 11, pp. 2892–2903, 2015.

[7] S. Chen, K. Saulnier, N. Atanasov, D. Lee, V. Kumar, and G. Pappas, "Approximating explicit model predictive control using constrained neural networks," in *American Control Conference*, pp. 1520–1527, June 2018.

[8] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020.

[9] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[10] T. Johansen and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 810–815, 2003.

[11] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," *IEEE Trans. Automatic Control*, vol. 56, no. 12, pp. 2883–2897, 2011.

[12] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[13] R. Shekhar and C. Manzie, "Optimal move blocking strategies for model predictive control," *Automatica*, vol. 61, pp. 27–34, 2015.

[14] M. Michael and D. Raffaello, "A method for reducing the complexity of model predictive control in robotics applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2516–2523, 2019.

[15] A. Bemporad and G. Cimini, "Reduction of the number of variables in parametric constrained least-squares problems," 2020. Submitted for publication. Also available on arXiv at `http://arxiv.org/abs/2012.10423`.

[16] B. Pluymers, L. Roobrouck, J. Buijs, J. Suykens, and B. De Moor, "Constrained linear mpc with time-varying terminal cost using convex combinations," *Automatica*, vol. 41, no. 5, pp. 831–837, 2005.

[17] H. H. Bloemen, T. J. van den Boom, and H. B. Verbruggen, "Optimizing the end-point state-weighting matrix in model-based predictive control," *Automatica*, vol. 38, no. 6, pp. 1061–1068, 2002.

[18] H. Zhang, J. Huang, and F. L. Lewis, "Updated terminal cost rhc for continuous-time systems," in *Proceedings of the 48h IEEE Conference on Decision*, pp. 4056–4061, 2009.

[19] O. Mangasarian and J. Rosen, "Inequalities for stochastic nonlinear programming problems," *Operations Research*, vol. 12, pp. 143–154, 1964.

[20] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. Communications and Control Engineering, Springer International Publishing, 2 ed., 2017.

[21] C. C. Aggarwal, *Neural networks and deep learning: a textbook*. Cham, Switzerland: Springer, 2018.

[22] Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.

[23] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," *Advances in Neural Information Processing Systems*, 2015.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] H. J. Ferreau, C. Kirches, A. Potschka, G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.