# Learning the MPC objective function from human preferences

Pablo Krupa, Hasna El Hasnaouy, Mario Zanon, Alberto Bemporad

*Abstract*— In Model Predictive Control (MPC), the objective function plays a central role in determining the closed-loop behavior of the system, and must therefore be designed to achieve the desired closed-loop performance. However, in real-world scenarios, its design is often challenging, as it requires balancing complex trade-offs and accurately capturing a performance criterion that may not be easily quantifiable in terms of an objective function. This paper explores preference-based learning as a data-driven approach to constructing an objective function from human preferences over trajectory pairs. We formulate the learning problem as a machine learning classification task to learn a surrogate model that estimates the likelihood of a trajectory being preferred over another. The approach provides a surrogate model that can directly be used as an MPC objective function. Numerical results show that we can learn objective functions that provide closed-loop trajectories that align with the expressed human preferences.

*Index Terms*— Model predictive control, Calibration, Preference-based learning, Machine learning, Classification.

## I. INTRODUCTION

The main advantage of Model Predictive Control (MPC) is its ability to handle constraints while optimizing a given objective function [1], [2]. The parameters defining the MPC controller and its objective function need to be tuned to achieve the desired system behavior and performance. However, tuning MPC parameters remains a challenging task in practice, as it requires expert knowledge and significant manual effort to achieve the desired closed-loop behavior [3]. Moreover, a limiting factor in many industrial applications is that obtaining an explicit measure quantifying the desired performance may be difficult or impossible. Instead, only qualitative assessments by a human expert may be available.

The challenge of calibrating the parameters of an MPC controller, otherwise known as *MPC tuning*, has received considerable attention from the research community. Many approaches leverage the availability of an explicit characterization of the desired system performance. For instance, Bayesian optimization [4] and other global optimization techniques [5] have been applied to MPC tuning [6], [7], or to adapt the learning procedure of the predictive model utilized in MPC to improve closed-loop performance [8]. These techniques, among others such as sensitivity-based approaches, have also been extended to tuning PID and LQR controllers [9], [10], [11]. Other approaches that have been used for MPC tuning include the use of reinforcement learning [12], [13] or inverse optimality [14].

Preference-based learning provides a promising alternative when there is no available explicit function characterizing system performance. Instead, human preferences are used, which can be expressed in different ways: pairwise comparisons, where a human selects the preferred option between two choices; ordered preferences, where multiple options are ranked in sequence; or binary yes/no choices. A common application that considers this paradigm is training large language models to align with user preferences [15], [16]. Different approaches have been employed to learn from human preferences, such as reinforcement learning [17], [18], or preferential Bayesian optimization along with Gaussian processes for classification [19]. The result is a surrogate model, typically in the form of a deep Neural Network (NN), that has been trained to capture the underlying preference function driving the human decision-making.

Several preference-based learning approaches have been explored for tuning controllers. In [20], preferential Bayesian optimization is applied to tune proportional-integral controllers from human-in-the-loop feedback providing pairwise preferences. In [21], [22], a semi-automated calibration approach was proposed for MPC tuning based on pairwise preferences between experiment outcomes. A surrogate model of the underlying preference function is fitted using the global optimization algorithm GLISp [23] by iteratively proposing new calibration parameters in an exploration-exploitation framework. Inverse reinforcement learning is another approach that is strongly related to preference-based learning and that has also been used for MPC tuning [24].

This paper proposes an MPC tuning approach where a dataset of pairwise comparisons of input and state trajectories are used to train a surrogate model that captures the underlying human preference function. In contrast with [21], [22], we pose the learning problem as a binary classification problem that is solved by standard machine learning tools. The problem is posed such that the fitted model can be directly used as the objective function of an MPC controller to achieve a closed-loop behavior that aligns with the collected human preference.

The rest of the paper is organized as follows. Section II presents the problem formulation, the proposed approach for training a surrogate model of the underlying preference function, and how it is then used as the objective function of an MPC controller. Section III shows numerical results for a system of oscillating masses, where we fit quadratic objective functions to achieve a closed-loop behavior that aligns with the human preferences. We conclude with Section IV.

**Notation:** Given $x, y \in \mathbb{R}^n$, $x \leq (\geq) y$ denotes componentwise inequalities. The exponential function is denoted by $\exp(\cdot)$. The set of natural numbers (including 0) is denoted by $\mathbb{N}$. For $i, j \in \mathbb{N}$ with $i \leq j$, $\mathbb{N}_{[i,j]} \doteq \{i, i+1, \ldots, j\}$.

## II. LEARNING MPC COST FROM HUMAN PREFERENCES

This section presents the proposed preference-based learning process to obtain an objective function for MPC that reflects human preferences. The idea is to use a dataset containing pairwise comparisons of system trajectories. The dataset can be used to learn the objective function of an MPC controller such that the closed-loop behavior of the system is likely to be preferred by the human over any other alternative.

### A. Problem formulation

Consider a system modeled by the discrete-time dynamics

$$x(t+1) = f(x(t), u(t)), \quad (1a)$$
$$y(t) = g(x(t)), \quad (1b)$$

where $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ and $y(t) \in \mathbb{R}^{n_y}$ are the system state, input and output at the discrete-time instant $t \in \mathbb{N}$, and $f \colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, $g \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ define the system state dynamics and output equation, respectively. Without loss of generality, the control objective is to steer the system state to the origin $x_r = 0$, $u_r = 0$, which is assumed to be an equilibrium of (1); that is, $x(t) \to x_r$ and $u(t) \to u_r$ as $t \to +\infty$. Moreover, we assume that the system must satisfy (possibly coupled) state-input constraints $(x(t), u(t)) \in \mathcal{Z}, \forall t$, for some given $\mathcal{Z} \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$.

Let $\mathcal{T} = (\mathcal{X}, \mathcal{U}, \mathcal{Y}) \in \mathbb{T}$ be a finite trajectory of (1), i.e., $\mathcal{X} = (x_0, x_1, \ldots, x_N)$, $\mathcal{U} = (u_0, u_1, \ldots, u_{N-1})$ and $\mathcal{Y} = (y_0, y_1, \ldots, y_{N-1})$, where $x_i \in \mathbb{R}^{n_x}$, $u_i \in \mathbb{R}^{n_u}$ and $y_i \in \mathbb{R}^{n_y}$ satisfy $x_{i+1} = f(x_i, u_i)$, $y_i = g(x_i)$, $\forall i$, and $N \in \mathbb{N}$.[1]

We assume that the human expresses their assessments according to a preference function $\Pi \colon \mathbb{T} \times \mathbb{T} \to \{0, 1\}$ that encodes a preference between two trajectories $\mathcal{T}_i$ and $\mathcal{T}_j$ as:

$$\Pi(\mathcal{T}_i, \mathcal{T}_j) = \begin{cases} 1 \text{ if } \mathcal{T}_i \text{ is preferred over } \mathcal{T}_j, \\ 0 \text{ otherwise.} \end{cases}$$

In our framework, $\Pi$ is unknown but accessible, in that we can evaluate $\Pi(\mathcal{T}_i, \mathcal{T}_j)$ for any pair of trajectories $\mathcal{T}_i, \mathcal{T}_j$. In a real scenario, $\Pi$ could be defined as the preference that an expert operator assigns by looking at two different trajectories. The exact reason for the expert operator to choose one trajectory over another may be difficult to formally represent, but we assume that there is some underlying $\Pi$ that the operator uses, consciously or unconsciously.

Let $\{\mathcal{T}_i\}_{i=1}^{n_t}$ be a collection of $n_t$ trajectories $\mathcal{T}_i \in \mathbb{T}$ of system (1). We assume the availability of a dataset

$$\mathcal{D} = \{(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell}, p_\ell)\}_{\ell=1}^{n_p}$$

that collects $n_p$ trajectory pairs $(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell})$, where $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$ are the indices of the trajectories from $\{\mathcal{T}_i\}_{i=1}^{n_t}$ considered in the $\ell$-th preference $p_\ell \doteq \Pi(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell})$.

### B. Training a surrogate model of the preference function

Our objective is to learn a surrogate model $\pi \colon \mathbb{T} \times \mathbb{T} \to \{0, 1\}$ of $\Pi$ that would ideally satisfy

$$\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) = \Pi(\mathcal{T}_i, \mathcal{T}_j), \; \forall \mathcal{T}_i, \mathcal{T}_j \in \mathbb{T}, \quad (2)$$

where $\theta \in \mathbb{R}^{n_\theta}$ is the vector of parameters defining $\pi$. Vector $\theta$ will be trained using the dataset $\mathcal{D}$ so as to maximize the satisfaction of (2). Furthermore, our objective is to be able to use $\pi$ to define the objective function of an MPC controller so as to obtain closed-loop trajectories that would generally be preferred over others according to $\Pi$. To this end, we take

$$\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) = \begin{cases} 1 \text{ if } \sigma(\mathcal{T}_i; \theta) \leq \sigma(\mathcal{T}_j; \theta), \\ 0 \text{ if } \sigma(\mathcal{T}_i; \theta) > \sigma(\mathcal{T}_j; \theta), \end{cases} \quad (3)$$

where $\sigma \colon \mathbb{T} \to \mathbb{R}$ is parameterized by $\theta$. We note that this choice of $\pi$ is taken to allow us to use $\sigma$ as the objective function of an MPC controller, as explained in Section II-C.

The objective of obtaining a function $\pi$ that always satisfies (2) is generally unattainable due to the lack of knowledge of the underlying (and unknown) preference function $\Pi$. However, we can formulate a learning problem of function $\pi$ to maximize the accuracy in satisfying (2) for the trajectory pairs in the dataset $\mathcal{D}$. In particular, we train $\theta$ by solving a binary classification problem using the dataset $\mathcal{D}$ [25]. As typically done in logistic regression for binary classification, we use the sigmoid function to model the probability of the preference between two trajectories $\mathcal{T}_i, \mathcal{T}_j \in \mathbb{T}$. We take this probability as

$$P_\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) = \frac{1}{1 + \exp(\sigma(\mathcal{T}_i; \theta) - \sigma(\mathcal{T}_j; \theta))}. \quad (4)$$

Taking this expression for $P_\pi$, the surrogate model (3) can then equivalently be expressed as

$$\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) = \begin{cases} 1 \text{ if } P_\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) \geq 0.5, \\ 0 \text{ if } P_\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) < 0.5. \end{cases} \quad (5)$$

The learning problem is given by

$$\min_\theta r(\theta) + \frac{1}{n_p} \sum_{\ell=1}^{n_p} \mathcal{L}(p_\ell, P_\pi(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell}; \theta)), \quad (6)$$

where $\mathcal{L} \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the cross-entropy loss

$$\mathcal{L}(p, \hat{p}) = -p \log(\hat{p}) - (1-p) \log(1-\hat{p})$$

measuring the likelihood of the prediction $\hat{p} \in \mathbb{R}$ matching the target $p \in \{0, 1\}$, and $r \colon \mathbb{R}^{n_\theta} \to \mathbb{R}$ is an $\ell_2$-regularization term $r(\theta) = \rho \|\theta\|^2$, for a given penalty parameter $\rho > 0$.[2] The optimal solution $\theta^*$ of (6) provides the parametrization of $\sigma$ that maximizes the likelihood of the surrogate model $\pi$ (3) satisfying (2) over the trajectory pairs in the dataset $\mathcal{D}$. Problem (6) can be solved using standard machine learning procedures for different popular choices of the loss function and regularization term, including popular optimization methods such as Adam [26] or L-BFGS-B [27]; see, e.g., the Python package `jax-sysid` [28], which uses Adam to warmstart L-BFGS-B.

---

[1] We omit $N$ from the notation of $\mathcal{T}$ for convenience. Throughout the article, all trajectories are of length determined by $N$ unless stated otherwise.

[2] We note that other regularization terms could be used instead, e.g., $\ell_1$-regularization, or a combination of $\ell_1$ and $\ell_2$ norms.

## C. Using the surrogate model for MPC

Our choice of taking the surrogate model $\pi$ as in (3) and the probability model $P_\pi$ as in (4) differs from the straightforward approach in machine learning for binary classification, cf. [25]. Indeed, the straightforward approach would be to consider a function $\tilde{\sigma}\colon \mathbb{T}\times\mathbb{T} \to \mathbb{R}$ parameterized by $\theta$ (typically a NN where $\theta$ collects the weights and biases), and then take (4) as

$$P_\pi(\mathcal{T}_i, \mathcal{T}_j; \theta) = \frac{1}{1 + \exp(-\tilde{\sigma}(\mathcal{T}_i, \mathcal{T}_j; \theta))}.$$

The surrogate model in this case would be given by (5).

Instead, the idea is to consider a function $\sigma\colon \mathbb{T} \to \mathbb{R}$ and then take its difference $\sigma(\mathcal{T}_i; \theta) - \sigma(\mathcal{T}_j; \theta)$ in the exponent term of (4). In doing so, we obtain a function $\sigma(\mathcal{T}; \theta)$ that can be used to predict the preference between two trajectories as the one given by the trajectory resulting in the smaller value of $\sigma$, cf. (3). Therefore, once $\theta^*$ has been obtained by solving the training problem (6), we obtain a function $\sigma(\mathcal{T}; \theta^*)$ such that (the possibly non-unique) $\mathcal{T}^* \doteq \arg\min_\mathcal{T} \sigma(\mathcal{T}; \theta^*)$ satisfies $\pi(\mathcal{T}^*, \mathcal{T}; \theta^*) = 1$, $\forall \mathcal{T} \in \mathbb{T}$. That is, $\mathcal{T}^*$ would be preferred (or deemed equal) to any other trajectory $\mathcal{T} \in \mathbb{T}$ according to our learned surrogate model $\pi$. The parameter vector $\theta^*$ has been learned so as to increase the likelihood of objective (2) being satisfied for the trajectory pairs in the dataset $\mathcal{D}$. Therefore, assuming that the dataset $\mathcal{D}$ is rich enough to capture the underlying preference function $\Pi$ and assuming that (6) is successfully solved, the hope is for $\mathcal{T}^*$ to be preferred by $\Pi$ over any other trajectory (not necessarily included in the dataset), that is, for $\Pi(\mathcal{T}^*, \mathcal{T}) = 1$, $\forall \mathcal{T} \in \mathbb{T}$.

We can exploit this feature to pose an MPC controller [1], [2] whose objective function is the learned function $\sigma(\mathcal{T}; \theta^*)$:

$$\min_\mathcal{T} \ \sigma(\mathcal{T}; \theta^*) \tag{7a}$$

$$\text{s.t. } x_0 = x(t), \tag{7b}$$

$$x_{k+1} = f(x_k, u_k), \ k \in \mathbb{N}_{[0, N-1]}, \tag{7c}$$

$$(x_k, u_k) \in \mathcal{Z}, \ k \in \mathbb{N}_{[0, N-1]}, \tag{7d}$$

$$x_N \in \mathcal{X}_N, \tag{7e}$$

where (7b) imposes the initial state constraint, (7c) imposes the system dynamics (1) on the predicted states and inputs $(x_k, u_k)$, (7d) impose the state-input constraints on the given set $\mathcal{Z} \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$, and (7e) imposes a terminal constraint on a given terminal set $\mathcal{X}_N \subseteq \mathbb{R}^{n_x}$. Let $\mathcal{T}^*(t)$ be the optimal solution of (7) at time $t$. The control law of the MPC controller is $u(t) = u_0^*(t)$, where $u_0^*(t)$ corresponds to the first control action in the optimal trajectory $\mathcal{T}^*(t)$.

By posing the minimization of $\sigma(\mathcal{T}; \theta^*)$ as an MPC problem, we have that $\mathcal{T}^*(t)$ minimizes the objective $\sigma(\mathcal{T}; \theta^*)$ for all trajectories satisfying the constraints (7b)-(7e), i.e., for all trajectories satisfying the system dynamics and constraints. Therefore, we have that

$$\sigma(\mathcal{T}^*(t); \theta^*) \leq \sigma(\mathcal{T}; \theta^*), \forall \mathcal{T} \in \mathbb{T}\colon \mathcal{T} \text{ satisfies } (7b) - (7e).$$

The results is that, according to the learned model $\pi(\mathcal{T}; \theta^*)$, $\mathcal{T}^*(t)$ is preferred by $\Pi$ over any $\mathcal{T} \in \mathbb{T}$ satisfying (7b)-(7e).

**Remark 1.** *In this paper, we allow $\sigma$ to be any function such that problem (6) can be solved using standard machine learning tools. This includes non-smooth functions $\sigma$ such as feedforward NN with ReLU activation functions. In this general setting, the MPC controller (7) does not guarantee stability of the closed-loop system. Additional consideration should be taken to ensure its stability for the choice of $\sigma$.*

**Remark 2.** *Problem (7) can be particularized to linear MPC, where we note that using $\sigma$ as the objective function is not restrictive, since it can be taken as a convex function. Indeed, $\sigma$ can be taken as a quadratic function, cf. Section III, or, more generally, as an input-convex neural network [29].*

**Remark 3.** *In Section II-A we define $\mathcal{T}$ as a collection of state, input, and output trajectories. However, $\sigma$ does not need to be a function of all three. That is, $\sigma$ could be a function of state and inputs $\sigma(\mathcal{X}, \mathcal{U}; \theta)$, inputs and outputs $\sigma(\mathcal{Y}, \mathcal{U}; \theta)$, or only consider the initial state $\sigma(x_0, \mathcal{U}; \theta)$. Furthermore, $\mathcal{T}$ could contain reference information $(x_r, u_r) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$, leading to functions such as $\sigma(\mathcal{X}, \mathcal{U}, x_r, u_r; \theta)$ and an MPC formulation (7) parameterized by $(x_r, u_r)$.*

## III. NUMERICAL RESULTS

This section presents numerical results highlighting the proposed preference-based learning approach for MPC objective functions. We present two case studies: one where the preference function is based on evaluating a quadratic function, and one where we consider a more complex preference function. Both case studies consider the system of oscillating masses described in [30, §IV.A], which is formed by three objects connected by springs. The state $x = (p_1, p_2, p_3, v_1, v_2, v_3)$ is given by the position $p_i$ and velocity $v_i$ of each object, and the input $u = (F_1, F_2)$ by the forces $F_i$ applied to the two outermost objects. We take the output as $y = (p_1, p_2, p_3)$. The dynamics of this system are given by a linear state-space model

$$x(t + 1) = Ax(t) + Bu(t), \tag{8a}$$

$$y(t) = Cx(t). \tag{8b}$$

The control objective is to steer the system to $(x, u) = (0, 0)$ while satisfying the input constraints $|u_i| \leq 1$, $i \in \mathbb{N}_{[1,2]}$.

To solve problem (6), we start by some initial guess of the parameters $\theta$. We first perform 200 iterations of the Adam solver from the `optax` Python package (version `0.2.4`), taking the learning rate as 0.01 and decay rates as $\beta_1 = 0.9$, $\beta_2 = 0.999$. We then run a maximum of 1000 iterations of the L-BFGS-B solver (with a history size of 10) from the `jaxopt` Python package (version `0.8.3`), starting from the solution returned by Adam. We take $\rho = 10^{-6}$ for the $\ell_2$-regularization term of (6). Tests are performed using Python `3.11.10` on a Macbook Air with an M3 microprocessor.

## A. Learning a quadratic-based preference function

We first evaluate the proposed approach by considering a scenario where the preference function is given by

$$\Pi_\phi(\mathcal{T}_j, \mathcal{T}_j) = \begin{cases} 1 \text{ if } \phi_N(\mathcal{T}_i; Q, R) \leq \phi_N(\mathcal{T}_j; Q, R), \\ 0 \text{ if } \phi_N(\mathcal{T}_i; Q, R) > \phi_N(\mathcal{T}_j; Q, R), \end{cases} \tag{9}$$

3

TABLE I: Training and performance results for $\Pi_\phi$.

| MPC cost | Train | Test | Time [s] | Avrg. $\phi_{30}$ | Max. $\phi_{30}$ | Min. $\phi_{30}$ |
|---|---|---|---|---|---|---|
| From $\Pi$ | – | – | – | 1.000 | 4.426 | 0.054 |
| Random | – | 82.6 | – | 1.833 | 10.532 | 0.106 |
| $\sigma_{20}$ | 100.0 | 92.0 | 1.13 | 1.355 | 5.491 | 0.086 |
| $\sigma_{60}$ | 100.0 | 94.8 | 1.13 | 1.108 | 4.598 | 0.059 |
| $\sigma_{100}$ | 98.0 | 95.0 | 1.16 | 1.230 | 5.192 | 0.059 |
| $\sigma_{400}$ | 100.0 | 99.4 | 1.53 | 1.043 | 4.536 | 0.057 |
| $\sigma_{1000}$ | 100.0 | 99.8 | 1.87 | 1.013 | 4.404 | 0.056 |

"Train", and "Test" are the accuracy on training and test datasets, in %. "Time" is the average computation time of solving problem (6). The average, maximum and minimum performance, with performance measured by $\phi_{30}$, have all been normalized by the average performance obtained with the MPC using the $Q$ and $R$ from $\Pi_\phi$.

with

$$\phi_N(\mathcal{T}; Q, R) = \sum_{k=0}^{N-1} \|x_k\|_Q^2 + \|u_k\|_R^2, \qquad (10)$$

for unknown positive definite matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$. That is, $\Pi_\phi$ prefers the trajectory with the smallest quadratic cost for matrices $Q$, $R$, and horizon $N$, which we take in our simulations as $Q = \texttt{diag}(40, 40, 40, 5, 5, 5)$, $R = \texttt{diag}(0.2, 0.2)$, and $N = 10$.

We build a dataset $\mathcal{D}$ by first generating $n_t = 50$ closed-loop trajectories of (8), each starting from a random $x(0)$ and controlled using the Linear Quadratic Regulator (LQR) for random matrices $\hat{Q} \in \mathbb{R}^{n_x \times n_x}$ and $\hat{R} \in \mathbb{R}^{n_u \times n_u}$ penalizing states and inputs, respectively. For $x(0)$, we take the position and velocity of each object from uniform distributions in the intervals $[-0.3, 0.3]$ and $[-0.05, 0.05]$, respectively. For $\hat{Q}$ and $\hat{R}$ we generate predominantly-diagonal positive definite matrices, with diagonal elements in the range $[0.1, 10]$. Datasets of paired trajectories are obtained by randomly selecting pairs from the $n_t$ trajectories and then evaluating the preference function on each pair.

We take $\sigma$ as a quadratic function (10):

$$\sigma(\mathcal{T}; \theta) = \phi_N(\mathcal{T}; Q_\theta, R_\theta), \qquad (11)$$

where $Q_\theta \in \mathbb{R}^{n_x \times n_x}$ and $R_\theta \in \mathbb{R}^{n_u \times n_u}$ are encoded by $\theta$ as follows. Let $n_Q = n_x + \frac{n_x(n_x-1)}{2}$ and $n_R = n_u + \frac{n_u(n_u-1)}{2}$. Then $\theta \in \mathbb{R}^{n_Q + n_R}$ contains the non-zero elements of the Cholesky decomposition of matrices $Q_\theta$ and $R_\theta$. This encoding allows us to impose positive definiteness on matrices $Q_\theta$ and $R_\theta$ by enforcing $\theta_{(i)} \geq \epsilon$, with $\epsilon > 0$, for all components $i$ of $\theta$ corresponding to an element in the diagonal of $Q_\theta$ or $R_\theta$. We take $\epsilon = 0.01$ This lower-bound constraint on $\theta$ is imposed when solving problem (6). Additionally, we impose the first element in the diagonal of $Q_\theta$ to be greater or equal to 1, which is formulated as a lower bound on the corresponding element of $\theta$ in (6).[3]

By taking $\sigma$ as in (11), we guarantee the existence of matrices $Q_\theta$ and $R_\theta$ such that our model perfectly captures the preference function (9). Indeed, (2) is always satisfied if $Q_\theta = Q$ and $R_\theta = R$. However, we assume that matrices

---

[3]This additional constraint is added to limit the degree of freedom in scaling matrices $Q_\theta$ and $R_\theta$, while also improving numerical conditioning by avoiding small values in $\theta^*$.

$Q$ and $R$ are unknown by the learning process to simulate a real scenario where the underlying preference function is unknown. Therefore, we randomly initialize $Q_\theta$ and $R_\theta$ using the same approach taken to generate matrices $\hat{Q}$ and $\hat{R}$ for the dataset $\mathcal{D}$. The $\theta$ corresponding to these random $Q_\theta$ and $R_\theta$ is used to initialize the Adam solver.

We consider training datasets of different sizes and a testing dataset with 500 paired trajectories, which we use to evaluate the accuracy of the trained surrogate models. Model accuracy is measured as

$$\frac{1}{n_p} \sum_{\ell=1}^{n_p} [\pi(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell}; \theta^*) = \Pi(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell})],$$

where $n_p$ is the number of trajectory pairings $(\mathcal{T}_{\mathcal{I}_\ell}, \mathcal{T}_{\mathcal{J}_\ell})$ in the test dataset. For each dataset size $n_p$, we train 20 functions $\sigma_{n_p}$, each one starting from a different random initialization of $\theta$. We keep the $\sigma_{n_p}$ with the highest accuracy on the test dataset. We then use the functions $\sigma_{n_p}$ as objective functions of the following particularization of the MPC controller (7):

$$\min_{\mathcal{T}} \phi_N(\mathcal{T}; \hat{Q}, \hat{R}) \qquad (12a)$$

$$\text{s.t. } x_0 = x(t), \qquad (12b)$$

$$x_{k+1} = Ax_k + Bu_k, \ k \in \mathbb{N}_{[0,N-1]}, \qquad (12c)$$

$$\underline{u} \leq u_k \leq \overline{u} \in, \ k \in \mathbb{N}_{[0,N-1]}. \qquad (12d)$$

We perform 200 closed-loop simulations with system (8), each one starting from a random $x(0)$ computed as for the generation of the datasets. Simulations have a length of 30 sample times. Additionally, for the purpose of comparison, we take an MPC (12) with $\hat{Q} = Q$ and $\hat{R} = R$, and, for each simulation, an MPC (12) with random matrices $\hat{Q}$ and $\hat{R}$.

Table I shows the training and test results. We report the accuracy of the models on training and test datasets, as well as the average computation time of solving (6).[4] We also report the average, maximum and minimum performance of the closed-loop trajectories for each MPC controller, with performance measured as $\phi_{30}(\mathcal{T}; Q, R)$. That is, performance is measured using matrices $Q$ and $R$ from the preference function (9). Therefore, a smaller value of the performance index corresponds to a closed-loop trajectory that would be preferred by $\Pi_\phi$ (9). We normalize all performance values by the average performance obtained using the MPC controller with $Q$ and $R$ from the preference function $\Pi_\phi$ (9). Fig. 1 shows performance results of the MPC controllers as well as an example comparing the closed-loop trajectories obtained by different MPC controller for a random initial state $x(0)$ with position of the central object set to $-0.3$.

The results show that as the size $n_p$ of the training dataset increases, the learned functions $\sigma_{n_p}$ tend to provide a higher accuracy on the 500 trajectory pairs within the test dataset. That is, we obtain matrices $Q_\theta$ and $R_\theta$ that capture the preference function $\Pi_\phi$. We note that matrices $Q_\theta$ and $R_\theta$ obtained in these tests do not necessarily resemble matrices

---

[4]This includes the execution of Adam, L-BFGS-B and overhead.

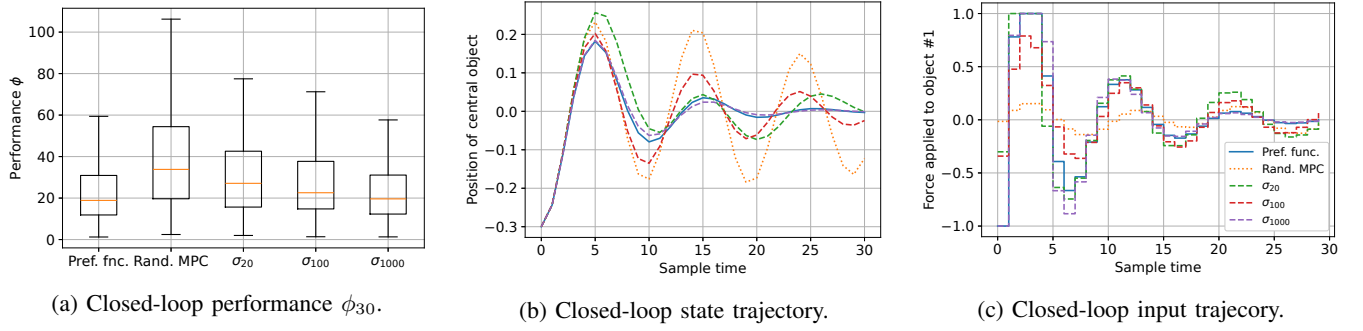(a) Closed-loop performance $\phi_{30}$.     (b) Closed-loop state trajectory.     (c) Closed-loop input trajecory.

Fig. 1: Results for quadratic-based preference function $\Pi_\phi$ and quadratic $\sigma$.

$Q$ and $R$. In particular, $Q$ and $R$ are diagonal matrices, whereas $Q_\theta$ and $R_\theta$ typically have significant non-diagonal values. Nevertheless, the MPC controllers using $\sigma_{n_p}$ provide good closed-loop performance. Indeed, results indicate that the use of functions $\sigma_{n_p}$ as objective function of the MPC (12) provide a closed-loop behavior and performance that resemble the ones obtained by using the $Q$ and $R$ from $\Pi_\phi$ (9), when using a sufficiently large training dataset. Note that for $\sigma_{1000}$, the average performance only increases by 1.3%. We also observe that the use of functions $\sigma_{n_p}$ obtained from very small datasets provide better results than using MPC controllers with random matrices $\hat{Q}$ and $\hat{R}$ (generated with the same procedure used to populate the dataset).

### B. Learning a complex preference function

We now consider a preference function that selects the trajectory with the smallest 2-norm output settling time:

$$\kappa_\varepsilon(\mathcal{T}) \doteq \min_{k \geq 0} k : k \in \mathbb{N}, \|y_k\| \leq \varepsilon, \|y_\ell\| \not\succ \varepsilon \; \forall \ell > k,$$

where we take $\varepsilon = 0.1$. That is, we consider

$$\Pi_{\kappa_\varepsilon}(\mathcal{T}_j, \mathcal{T}_j) = \begin{cases} 1 \text{ if } \kappa_\varepsilon(\mathcal{T}_i) < \kappa_\varepsilon(\mathcal{T}_j), \\ 0 \text{ if } \kappa_\varepsilon(\mathcal{T}_i) > \kappa_\varepsilon(\mathcal{T}_j), \\ \Pi_u(\mathcal{T}_j, \mathcal{T}_i) \text{ otherwise,} \end{cases}$$

with

$$\Pi_u(\mathcal{T}_j, \mathcal{T}_j) = \begin{cases} 1 \text{ if } \max \|u_i(t)\|_\infty \leq \max \|u_j(t)\|_\infty, \\ 0 \text{ otherwise,} \end{cases}$$

where $u_i(t)$ and $u_j(t)$ are the input trajectories associated with $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. The combination of $\Pi_{\kappa_\varepsilon}$ with $\Pi_u$ is designed to predominantly choose trajectories with small output settling times $\kappa_\varepsilon$ while encouraging the use of small maximum values of the control action.

As in Section III-A, we obtain a dataset of paired trajectories from closed-loop simulations with random LQR controllers. In this case, we take $N = 15$, and $\hat{Q}$ and $\hat{R}$ as diagonal matrices with non-zero elements taken from uniform distributions on the following intervals: $[5, 20]$ for elements of $Q$ corresponding to object positions, and $[0.1, 1.0]$ for the rest. This change provides datasets with a rich variety of values of $\kappa_\varepsilon$ that typically satisfy $\kappa_\varepsilon(\mathcal{T}) \leq N$ for all $\mathcal{T}$ in the dataset. The median value of $\kappa_\varepsilon$ for the generated

trajectories is 6. As in Section III-A, we take $\sigma$ as a quadratic function (11) and perform 200 closed-loop simulations with the MPC controllers (12) taking random initial states $x(0)$. In this case, we remove the input constraints (12d) so as to provide the controller with freedom to achieve smaller output settling times $\kappa_\varepsilon$.

Table II shows the training results, the median and maximum values of $\kappa_\varepsilon$ for the closed-loop tests, and the average value of $\max \|u(t)\|_\infty$. We also report the results on $\kappa_\varepsilon$ and $\max \|u(t)\|_\infty$ for the same tests but removing $\Pi_u$ from $\Pi_{\kappa_\varepsilon}$.[5] Fig. 2 shows the results for the output settling time and the average values of $\|y(t)\|$ for each sample time $t$ of the simulations. It also shows the average values of $\|y(t)\|$ obtained when removing $\Pi_u$ from $\Pi_{\kappa_\varepsilon}$. The results show that we learn objective functions that tend to provide smaller output settling times $\kappa_\varepsilon$ as the size of the dataset increases. Additionally, we observe that removing $\Pi_u$ from $\Pi_{\kappa_\varepsilon}$ results in smaller values of $\kappa_\varepsilon$ at the expense of an increase in the control actions, as expected.

### IV. CONCLUSIONS

We have presented a preference-based learning approach for learning the objective function of an MPC controller from human preferences expressed over trajectory pairs. We learn a surrogate function of the underlying human preference function by posing a binary classification problem, which can be solved using standard machine learning tools. The problem is posed so that the fitted model can be directly used as the objective function of an MPC controller so as to provide a closed-loop behavior that aligns with the human preferences. Case studies show good results when using 50 closed-loop trajectories and a few hundred evaluations of the preference function, indicating that the MPC can be tuned using a relatively small amount of machine and human time.
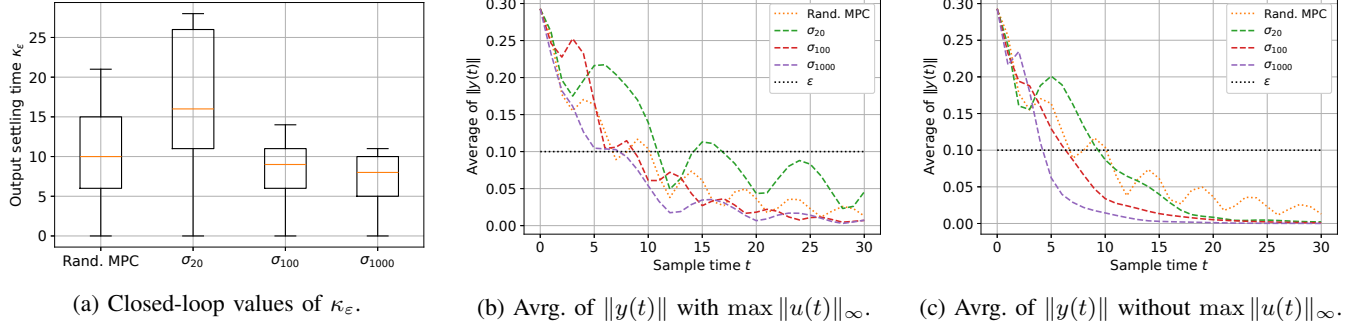
### REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing Madison, Wisconsin, 2017.
[2] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. London, UK: Springer-Verlag, 2007.
[3] J. L. Garriga and M. Soroush, "Model predictive control tuning methods: A review," *Industrial and Engineering Chemistry Research*, vol. 49, no. 8, pp. 3505–3515, 2010.

---

[5]We remove from the dataset all trajectory pairs with the same $\kappa_\varepsilon$.

TABLE II: Training and performance results for complex preference function $\Pi_{\kappa_\varepsilon}$.

| MPC cost | Training and test results after solving (6) | | | Results for $\Pi_{\kappa_\varepsilon}$ with $\Pi_u$ | | Results for $\Pi_{\kappa_\varepsilon}$ without $\Pi_u$ | |
|---|---|---|---|---|---|---|---|
| | Acc. train (%) | Acc. test (%) | Avrg. comp. time [s] | Med./Max. $\kappa_\varepsilon$ | Avrg. $\max \|u(t)\|_\infty$ | Med./Max. $\kappa_\varepsilon$ | Avrg. $\max \|u(t)\|_\infty$ |
| Random | – | 83.17 | – | 10 / 21 | 0.52 | 10 / 21 | 0.52 |
| $\sigma_{20}$ | 100.0 | 90.60 | 1.35 | 16 / >30 | 0.51 | 10 / 16 | 1.53 |
| $\sigma_{100}$ | 100.0 | 93.60 | 1.43 | 9 / 14 | 3.21 | 7 / 9 | 3.06 |
| $\sigma_{400}$ | 95.25 | 94.80 | 1.91 | 9 / 17 | 2.38 | 5 / 9 | 5.48 |
| $\sigma_{1000}$ | 95.20 | 94.20 | 2.75 | 8 / 11 | 2.38 | 5 / 6 | 6.38 |



(a) Closed-loop values of $\kappa_\varepsilon$.    (b) Avrg. of $\|y(t)\|$ with $\max \|u(t)\|_\infty$.    (c) Avrg. of $\|y(t)\|$ without $\max \|u(t)\|_\infty$.

Fig. 2: Results for preference function $\Pi_{\kappa_\varepsilon}$ and quadratic $\sigma$.

[4] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv:1012.2599*, 2010.

[5] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, no. 2, pp. 571–595, 2020.

[6] M. Forgione, D. Piga, and A. Bemporad, "Efficient calibration of embedded MPC," in *IFAC-PapersOnLine*, vol. 53, no. 2, 2020, pp. 5189–5194.

[7] A. Lucchini, S. Formentin, M. Corno, D. Piga, and S. M. Savaresi, "Torque vectoring for high-performance electric vehicles: An efficient MPC calibration," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 725–730, 2020.

[8] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[9] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning," in *IJCAI International Joint Conference on Artificial Intelligence*, 2019, pp. 5850–5856.

[10] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on gaussian process global optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 270–277.

[11] D. Masti, M. Zanon, and A. Bemporad, "Tuning LQR controllers: A sensitivity-based approach," *IEEE Control Systems Letters*, vol. 6, pp. 932–937, 2022.

[12] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2020.

[13] M. Mehndiratta, E. Camci, and E. Kayacan, "Automated tuning of nonlinear model predictive controller by reinforcement learning," in *IEEE International Conference on Intelligent Robots and Systems*, 2018, pp. 3016–3021.

[14] M. Zanon and A. Bemporad, "Constrained controller and observer design by inverse optimality," *IEEE Transactions on Automatic Control*, vol. 67, no. 10, pp. 5432–5439, 2022.

[15] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv:1909.08593*, 2019.

[16] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano, "Learning to summarize with human feedback," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3008–3021.

[17] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, 2017.

[18] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, 2017, pp. 4300–4308.

[19] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, "Preferential Bayesian optimization," in *34th International Conference on Machine Learning (ICML)*, vol. 3, 2017, pp. 2080–2089.

[20] J. P. Coutinho, I. Castillo, and M. S. Reis, "Human-in-the-loop controller tuning using preferential Bayesian optimization," in *IFAC-PapersOnLine*, vol. 58, no. 14, 2024, pp. 13–18.

[21] M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference (ECC)*. IEEE, 2021, pp. 638–645.

[22] M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

[23] A. Bemporad and D. Piga, "Global optimization based on active preference learning with radial basis functions," *Machine Learning*, vol. 110, pp. 417–448, 2021.

[24] R. Tao, S. Cheng, X. Wang, S. Wang, and N. Hovakimyan, "DiffTune-MPC: Closed-loop learning for model predictive control," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7294–7301, 2024.

[25] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of Biomedical Informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

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

[27] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[28] A. Bemporad, "An L-BFGS-B approach for linear and nonlinear system identification under $\ell_1$ and group-lasso regularization," *IEEE Transactions on Automatic Control*, 2025, in press. Also available on arXiv at http://arxiv.org/abs/2403.03827.

[29] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. PMLR, 2017, pp. 146–155.

[30] P. Krupa, R. Jaouani, D. Limon, and T. Alamo, "A sparse ADMM-based solver for linear MPC subject to terminal quadratic constraint," *IEEE Transactions on Control Systems Technology*, vol. 32, no. 6, pp. 2376–2384, 2024.