Priority-Driven Constraints Softening in Safe MPC for Perturbed Systems

Ying Shuai Quan, Mohammad Jeddi, Francesco Prignoli, *Student Member, IEEE*, and Paolo Falcone, *Member, IEEE*

***Abstract*—This letter presents a *safe* model predictive control framework designed to guarantee the satisfaction of *hard* safety constraints, for perturbed dynamical systems. Safety is guaranteed by softening the constraints selected on a priority basis from a subset of constraints defined by the designer. Since such an online selection is the result of an auxiliary optimization problem, its computational overhead is alleviated by off-line learning its approximated solution, rather than solving it exactly online. Simulation results, obtained from an automated driving application, show that the proposed approach provides guarantees of collision-avoidance *hard* constraints despite the *unpredicted* behaviors of the surrounding environment.**

***Index Terms*—Model predictive control, constrained control, uncertain systems, safety critical.**

I. Introduction

MODEL predictive control (MPC) is widely used in safety-critical applications of mobile autonomous systems thanks to its capability of embedding safety (collision-avoidance) constraints. However, the presence of external disturbances, arising from pop-ups or moving obstacles, may lead to infeasibility, thus to unsafe behaviors. To address such an issue, safe model predictive control (SMPC) frameworks have been developed to ensure the satisfaction of safety constraints at all times, provided that a set of assumptions holds [1], [2]. However, deploying these frameworks in real-world scenarios remains challenging, as unforeseen environmental factors can still violate the required assumptions, thus leading to infeasibility. MPC frameworks have been proposed [3] that incorporate time-varying constraints. However, these approaches rely on user-defined disturbances bounds, which can lead to overly conservative control strategies. Moreover, handling time-varying constraints increases the complexity of controller design, particularly due to the necessity of online computation of the time-varying terminal set [4]. An alternative approach introduces slack variables to soft constraints, improving adaptability to environmental changes [5]. This method also eliminates the need for explicit computation of a time-varying terminal set by incorporating terminal dynamics for the slack variable. In addition, the design of an exact penalty function ensures that, whenever a feasible solution to the original hard-constrained problem exists, the optimal solution of the slacked problem coincides with that of the original problem [6], [7]. However, determining which soft constraints are ultimately relaxed depends on the penalty weights assigned to each slack variable, a choice that becomes increasingly challenging as the number of slack variables grows. In [8], the authors address this issue by proposing an MPC design with prioritized constraints, formulated as a multi-objective optimization problem and solved through a sequence of single-objective MPCs. The approach in [9] recovers from infeasibility by solving an *auxiliary* linear program that finds the optimal relaxation of a set of constraints, selected by the user on a priority basis.

In this letter, we propose a priority-driven mechanism for SMPC frameworks that achieves a similar objective as in [9] by formulating an auxiliary problem that calculates the optimal slack. An approximate solution of such auxiliary problems is learned offline and evaluated online, in a user-defined priority order, to guide constraint relaxation used in the main SMPC problem. Our approach enables flexibility and applicability to SMPC formulations, at the cost of limited online computational overhead. We validate our method in an autonomous driving scenario, demonstrating its ability to handle unforeseen constraints and maintain safety, making it a promising solution for real-world deployment.

A. Notation

Consider a discrete-time linear system described by

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \tag{1}$$

Received 17 March 2025; revised 15 May 2025; accepted 1 June 2025. Date of current version 7 July 2025. This work was supported in part by the Fordonsstrategisk Forskning och Innovation Program of Vinnova under Grant 2018-05005, and in part by the European Union—Next Generation Eu—under the National Recovery and Resilience Plan (NRRP), within the Italian National Program Ph.D. Programme in Autonomous Systems (DAuSy) under Project D93C23000450005. Recommended by Senior Editor C. Manzie. *(Corresponding author: Ying Shuai Quan.)*

Ying Shuai Quan is with the Mechatronics Group, Department of Electrical Engineering, Chalmers University of Technology, 41258 Gothenburg, Sweden (e-mail: quany@chalmers.se).

Mohammad Jeddi is with the Dipartimento di Ingegneria "Enzo Ferrari," Università di Modena e Reggio Emilia, 41125 Modena, Italy, and also with the Department of Electrical and Information Engineering, Italian National Ph.D. DAUSY, Polytechnic of Bari, 70126 Bari, Italy (e-mail: mohammad.jeddi@unimore.it).

Francesco Prignoli is with the Dipartimento di Ingegneria "Enzo Ferrari," Università di Modena e Reggio Emilia, 41125 Modena, Italy (e-mail: francesco.prignoli@unimore.it).

Paolo Falcone is with the Mechatronics Group, Department of Electrical Engineering, Chalmers University of Technology, 41258 Gothenburg, Sweden, and also with the Dipartimento di Ingegneria "Enzo Ferrari," Università di Modena e Reggio Emilia, 41125 Modena, Italy (e-mail: paolo.falcone@unimore.it).

Digital Object Identifier 10.1109/LCSYS.2025.3580494

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k \in \mathbb{R}^{n_u}$ are the state and input vectors at time k, respectively. We denote the predicted state, input and disturbance at time n given the information available at time k by $\mathbf{x}_{n|k}$, $\mathbf{u}_{n|k}$ and $b_{g_{n|k}}$, respectively. The system is subject to two groups of convex state and input constraints: constraints $h(\mathbf{x}, \mathbf{u}) \leq 0$ with

$$h\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) = C_h \mathbf{x}_{n|k} + D_h \mathbf{u}_{n|k} - \mathbf{b}_h, \tag{2}$$

and *hard* constraints $g(\mathbf{x}, \mathbf{u}, b_g) \leq 0$ with

$$g\Big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}\Big) = C_g \mathbf{x}_{n|k} + D_g \mathbf{u}_{n|k} - b_{g_{n|k}}. \tag{3}$$

While the proposed framework works for an arbitrary number of constraints, for simplicity, we consider the case $h(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{n_h}$, and $g(\mathbf{x}, \mathbf{u}, b_g) \in \mathbb{R}^1$. The constraints $h(\cdot, \cdot)$ are defined offline at the problem formulation stage, while the constraint $g(\cdot, \cdot, \cdot)$ is constructed online as the system evolves, and is driven by an external input. For example, in the context of mobile autonomous systems, $g(\cdot, \cdot, \cdot)$ arises from collision-avoidance constraints imposed by the interactions with the environment and the disturbance $b_{g_{n|k}}$ is the predicted motion of surrounding moving obstacles.

We use the notation $\mathbb{I}_a^b \coloneqq \{a, a+1, \ldots, b\}$ to denote a set of integers. We denote the Euclidean norm as $\|x\| = \sqrt{x^T x}$ whereas the Euclidean norm w.r.t. $P = P^T \succ 0$ is denoted by $\|x\|_P^2 = x^T P x$. We define $\mathbf{x}_+^{\mathbf{u}}$ as the next state under control input $\mathbf{u}$: $\mathbf{x}_+^{\mathbf{u}} \coloneqq A\mathbf{x} + B\mathbf{u}$.

II. Safe MPC

We define safety, which is the primary objective of our controller:

Definition 1 (Safety): A controller $\mathbf{u}_k = \kappa(\mathbf{x}_k)$ is safe for the system (1) in a set $\mathcal{S} \subseteq \mathbb{R}^{n_x}$ if $\forall\, \mathbf{x} \in \mathcal{S}$, the control inputs $\mathbf{U} = \{\kappa(\mathbf{x}_0), \ldots, \kappa(\mathbf{x}_\infty)\}$ and the corresponding state trajectories $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_\infty\}$ are such that $h(\mathbf{x}_k, \mathbf{u}_k) \leq 0$ and $g(\mathbf{x}_k, \mathbf{u}_k, b_{g_k}) \leq 0$, $\forall\, k \geq 0$.

Defining the stage and terminal cost functions as $q_{\mathbf{r}}$ and $p_{\mathbf{r}}$, which penalize the deviations from the reference trajectory $\mathbf{r} = (\mathbf{r}^{\mathbf{x}}, \mathbf{r}^{\mathbf{u}})$, we make the following assumptions:

Assumption 1: The pair (A, B) is stabilizable. The stage cost $q_{\mathbf{r}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$, and terminal cost $p_{\mathbf{r}}{:}\mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$, are continuous at the origin and satisfy $q_{\mathbf{r}}(\mathbf{r}_k^{\mathbf{x}}, \mathbf{r}_k^{\mathbf{u}}) = 0$, and $p_{\mathbf{r}}(\mathbf{r}_k^{\mathbf{x}}) = 0$. Additionally, $q_{\mathbf{r}}(\mathbf{x}_k, \mathbf{u}_k) \geq \alpha_1(\|\mathbf{x}_k - \mathbf{r}_k^{\mathbf{x}}\|)$ for all feasible $\mathbf{x}_k$, $\mathbf{u}_k$, and $p_{\mathbf{r}}(\mathbf{x}_N) \leq \alpha_2(\|\mathbf{x}_N - \mathbf{r}_N^{\mathbf{x}}\|)$, where α_1 and α_2 are $\mathcal{K}_\infty$-functions.

Assumption 2: The reference is feasible for the system dynamics, i.e., $\mathbf{r}_{k+1}^{\mathbf{x}} = A\mathbf{r}_k^{\mathbf{x}} + B\mathbf{r}_k^{\mathbf{u}}$, and satisfies the known constraints $h(\mathbf{r}_k^{\mathbf{x}}, \mathbf{r}_k^{\mathbf{u}}) \leq 0$, for all $k \in \mathbb{I}_0^\infty$.

Assumption 3: There exists a parametric stabilizing terminal set $\mathcal{X}_{\mathbf{r}}^{\mathrm{s}}$ and a terminal control law $\kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})$ yielding $p_{\mathbf{r}}(\mathbf{x}_+^{\kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})}) - p_{\mathbf{r}}(\mathbf{x}) \leq -q_{\mathbf{r}}(\mathbf{x}, \kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x}))$, $\mathbf{x}_+^{\kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})} \in \mathcal{X}_{\mathbf{r}}^{\mathrm{s}}$, and $h(\mathbf{x}, \kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})) \leq 0$, for all $\mathbf{x} \in \mathcal{X}_{\mathbf{r}}^{\mathrm{s}}$.

Assumptions 1-3 are commonly used in standard MPC to enforce stability, as noted in [10]. In addition, the following assumption requires the existence of a safe set, which helps ensuring that the controller's recursive feasibility holds.

Assumption 4: There exists a robust invariant set denoted $\mathcal{X}_{\mathrm{safe}} \subseteq \mathcal{X}_{\mathbf{r}}^{\mathrm{s}}$ such that for all $\mathbf{x} \in \mathcal{X}_{\mathrm{safe}}$ there exists a safe control set $\mathcal{U}_{\mathrm{safe}} \subseteq \mathbb{R}^{n_u}$ entailing that $\mathbf{x}_+^{\mathbf{u}_{\mathrm{safe}}} \in \mathcal{X}_{\mathrm{safe}}$, and $h(\mathbf{x}, \mathbf{u}_{\mathrm{safe}}) \leq 0$, for all $\mathbf{u}_{\mathrm{safe}} \in \mathcal{U}_{\mathrm{safe}}$. Moreover, $\mathcal{X}_{\mathrm{safe}}$ is constructed to ensure that $g(\mathbf{x}, \mathbf{u}_{\mathrm{safe}}, b_{g_{n|k}}) \leq 0$ holds for all $\mathbf{x} \in \mathcal{X}_{\mathrm{safe}}$ and $\mathbf{u}_{\mathrm{safe}} \in \mathcal{U}_{\mathrm{safe}}$.

Then based on the Safety Definition 1, we formulate the following SMPC control problem.

$$\begin{aligned}
\min_{\mathbf{u}} \quad & \sum_{n=k}^{k+N-1} q_{\mathbf{r}}\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) + p_{\mathbf{r}}\big(\mathbf{x}_{k+N|k}\big) \\
\text{s.t.} \quad & \mathbf{x}_{k|k} = \mathbf{x}_k, && \text{(4a)} \\
& \mathbf{x}_{n+1|k} = A\mathbf{x}_{n|k} + B\mathbf{u}_{n|k}, \qquad n \in \mathbb{I}_k^{k+M-1} && \text{(4b)} \\
& h\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) \leq 0, \qquad n \in \mathbb{I}_k^{k+M-1} && \text{(4c)} \\
& g\Big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}\Big) \leq 0, \quad n \in \mathbb{I}_k^{k+M-1} && \text{(4d)} \\
& \mathbf{x}_{k+n|k} \in \mathcal{X}_{\mathbf{r}}^{\mathrm{s}}, \qquad n \in \mathbb{I}_{k+N}^{k+M-1} && \text{(4e)} \\
& \mathbf{x}_{k+M|k} \in \mathcal{X}_{\mathrm{safe}}, && \text{(4f)}
\end{aligned}$$

where k is the current time, N is the prediction horizon, while $M \geq N$ is an extended prediction horizon. The constraint (4a) ensures that predictions of the system dynamics in (4b) start from the current system state $\mathbf{x}_k$.

To ensure safety, we introduce the following assumption:

Assumption 5: The hard constraint function satisfies

$$g\Big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k+1}}\Big) \leq g\Big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}\Big), \tag{5}$$

for all $n \geq k$.

Assumption 5 requires that the hard constraints $g(\mathbf{x}_k, \mathbf{u}_k, b_{g_k}) \leq 0$ are *consistent*, that is, they do not become "more restrictive" as the system (1) evolves. In the context of autonomous mobile robots, Assumption 5 requires that, at time $k+1$, a moving obstacle does not get "much" closer to the robot than what has been predicted at time k.

Finally, based on Assumptions 1-5, the following result guarantees that the system

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\kappa(\mathbf{x}_k), \tag{6}$$

with $\kappa(\mathbf{x}_k) = \mathbf{u}_{n|k}^*$ and $\{\mathbf{u}_{n|k}^*\}_{n=k}^{k+M-1}$ solution of (4), is safe according to Definition 1:

Proposition 1 [1]: Suppose that Assumptions 1-5 hold, and that Problem (4) is feasible for the initial state $\mathbf{x}_k$. Then, system (1) in a closed loop with the solution of (4) applied in receding horizon is safe (recursively feasible) at all times.

Proof: Readers are referred to [1]. ■

III. Priority-Driven Softened Safe MPC

Assumptions 1-5 enable strong safety guarantees [1]. However, Assumption 5 can be overly restrictive and often violated in practice. In this section, we design a learning-based, disturbance-responsive SMPC built upon a constraint relaxation mechanism that reacts to the disturbance b_g, which causes the violation of Assumption 5. The extent of the violation of the Assumption 5 is described by $\Delta_{g_{n|k+1}} \in \mathbb{R}^+$, which is defined as

$$\Delta_{g_{n|k+1}} = b_{g_{n|k}} - b_{g_{n|k+1}}, \tag{7}$$

and used to drive the selective relaxation of the constraints $h(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}) \leq 0$ to ensure the persistent satisfaction of the hard constraints $g(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}) \leq 0$. As shown in Section III-A, the proposed constraints relaxation mechanism can be computationally demanding. Hence, we develop a learning-based algorithm that approximately determines the necessary constraint relaxation at the cost of limited computational overhead.

A. Uncertainty-Responsive SMPC

Given the current hard constraint $g(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}})$, we solve the following *auxiliary* optimization problem:

$$\begin{aligned}
\min_{\mathbf{u},\Delta_h} \quad & \sum_{n=k}^{k+N-1} q_h\big(\Delta_{hn|k}\big) + p_h\big(\Delta_{hk+N|k}\big) \\
\text{s.t.} \quad & \text{(4a), (4b), (4d), (4f)} && \text{(8a)} \\
& 0 \le \Delta_{hn|k}, \qquad n \in \mathbb{I}_k^{k+M-1} && \text{(8b)} \\
& h\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) \le E\Delta_{hn|k}, \qquad n \in \mathbb{I}_k^{k+M-1} && \text{(8c)} \\
& \big(\mathbf{x}_{n|k}, \Delta_{hn|k}\big) \in \mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}, \qquad n \in \mathbb{I}_{k+N}^{k+M-1} && \text{(8d)}
\end{aligned}$$

where $\Delta_{hn|k}$ and $\mathbf{u}_{n|k}$ are decision variables. Assuming that $\Delta_h \in \mathbb{R}^{n_\Delta}$, $E \in \mathbb{R}^{n_h \times n_\Delta}$ is a user-defined matrix that selects those constraints that can be relaxed, where $E_{ij} = 1$ if the i-th row can be relaxed with j-th element of Δ_h, and $E_{ij} = 0$ otherwise. Furthermore, $\sum_{j=1}^{n_\Delta} E_{ij} = 1$ holds $\forall\ i \in \{1, \ldots, n_h\}$. We impose the terminal dynamics $\Delta_{h+} = M_h \Delta_h$, with $M_h \in \mathbb{R}^{n_\Delta \times n_\Delta}$ such that $\Delta_h \in \mathbb{R}_+^{n_\Delta} \to M_h\Delta_h \in \mathbb{R}_+^{n_\Delta}$ and $\max \|\lambda(M_h)\| < 1$. Then by solving $M_h^\top P_h M_h - P_h \preceq -Q_h$ with $Q_h \in \mathbb{R}^{n_\Delta \times n_\Delta}, Q_h = Q_h^\top \succ 0$, the stage and terminal cost functions are defined as $q_h(\Delta_{hn|k}) := \|\Delta_{hn|k}\|_{Q_h}$ and $p_h(\Delta_{hk+N|k}) := \|\Delta_{hk+N|k}\|_{P_h}$. Note that the terminal dynamics for Δ_h is not included as explicit constraints in (8) but is implicitly enforced through the terminal cost and the terminal set $\mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}$, which is constructed to force the desired evolution of Δ_h over time [5]. The introduction of the terminal dynamics for Δ_h is intended to induce desirable terminal and asymptotic behaviors of the relaxation terms $\Delta_{hn|k}$. For this extended terminal set $\mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}$, we make the following assumption based on Assumption 3:

Assumption 6: There exists a terminal set $\mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}$ such that with the terminal control law $\kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})$, $\forall (\mathbf{x}, \Delta_h) \in \mathcal{Z}_{\mathbf{r}}^{\mathrm{s}} \Rightarrow (\mathbf{x}_+^{\kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})}, M_h\Delta_h) \in \mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}$, and $h(\mathbf{x}, \kappa_{\mathbf{r}}^{\mathrm{s}}(\mathbf{x})) \le 0$.

Denoting the relaxation of $h(\mathbf{x}_{n|k}, \mathbf{u}_{n|k})$ by $\Delta_{hn|k}^\star$, obtained as solution of (8), we formulate the following uncertainty-responsive SMPC control problem:

$$\begin{aligned}
\min_{\mathbf{u}} \quad & \sum_{n=k}^{k+N-1} q_{\mathbf{r}}\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) + p_{\mathbf{r}}\big(\mathbf{x}_{k+N|k}\big) \\
\text{s.t.} \quad & \text{(4a), (4b), (4d), (4f)} && \text{(9a)} \\
& h\big(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\big) \le E\Delta_{hn|k}^\star, \qquad n \in \mathbb{I}_k^{k+M-1} && \text{(9b)} \\
& \big(\mathbf{x}_{n|k}.\Delta_{hn|k}^\star\big) \in \mathcal{Z}_{\mathbf{r}}^{\mathrm{s}}, \qquad n \in \mathbb{I}_{k+N}^{k+M-1} && \text{(9c)}
\end{aligned}$$

The proposed disturbance-responsive SMPC framework requires solving problems (8) and (9) sequentially at each time step. As solving both problems online may be computationally demanding, we propose to replace (8) with a learning-based approximation.

B. Learning Uncertainty-Responsive SMPC

In this section we formulate the design and training problem of a neural network (NN) that approximates the solution of problem (8). The input and output signals of the NN are defined as

$$\boldsymbol{\theta}_k = \begin{bmatrix} \big[b_{g_{n|k}}\big]_{n \in \mathbb{I}_k^{k+M-1}} \\ \mathbf{x}_k \end{bmatrix}, \boldsymbol{\eta}_k = \big[\Delta_{hn|k}\big]_{n \in \mathbb{I}_k^{k+M-1}}. \tag{10}$$

We introduce the following assumption

Assumption 7: There exists a nonempty compact set $\Theta = \{\boldsymbol{\theta}$: (8) is feasible$\}$.

Notice that the feasible set Θ can be computed by projecting the constraint-defined feasible region onto the parameter space. This projection can be performed, for example, using the Multi-Parametric Toolbox (MPT3) toolbox [11].

1) NN With Bounded Slope Nonlinearity: We introduce an l-layer NN described by the following recursive equations:

$$\begin{aligned}
& \boldsymbol{s}^0 = \boldsymbol{\theta}, \boldsymbol{s}^{i+1} = \phi_i\big(W^i \boldsymbol{s}^i + b^i\big), i \in \mathbb{I}_0^{l-1}, \\
& \boldsymbol{\eta} = f(\boldsymbol{\theta}) = W^l \boldsymbol{s}^l + b^l,
\end{aligned} \tag{11}$$

where $W^i \in \mathbb{R}^{n_{i+1} \times n_i}$, $b^i \in \mathbb{R}^{n_{i+1}}$, and $n_0, \ldots, n_{l+1}$ are the dimension of the input, the neurons in the hidden layers, and the output. The function ϕ_i is the vector of activation functions at each layer $\phi_i(z^i) = \big[\psi(z_1^i) \cdots \psi(z_n^i)\big]^\top$ with continuous nonlinear activation functions $\psi : \mathbb{R} \to \mathbb{R}$, with bounded slope nonlinearity:

$$\alpha \le \frac{\psi(x) - \psi(y)}{x - y} \le \beta \quad \forall x, y \in \mathbb{R}. \tag{12}$$

With a diagonal weighting matrix $T \in \mathcal{D}_n := \{T = \sum_{i=1}^n \lambda_{ii} e_i e_i^T, \lambda_{ii} \ge 0\}$ where e_i denotes the i-th standard basis vector and $\lambda_{ii} \ge 0$ are nonnegative scalar weights, a quadratic constraint can be used to characterize all the stacked activation functions [12]:

$$\begin{bmatrix} \tilde{x} - \tilde{y} \\ \phi(\tilde{x}) - \phi(\tilde{y}) \end{bmatrix}^T F(T) \begin{bmatrix} \tilde{x} - \tilde{y} \\ \phi(\tilde{x}) - \phi(\tilde{y}) \end{bmatrix} \ge 0 \tag{13}$$

with $F(T) = \begin{bmatrix} -2\alpha\beta T & (\alpha+\beta)T \\ (\alpha+\beta)T & -2T \end{bmatrix}$ for all $T \in \mathcal{D}_n, \tilde{x}, \tilde{y} \in \mathbb{R}^n$ with $n = \sum_{i=1}^l n_i$. Here, the inputs to all neurons in the l-layer NN are stacked into one vector $\tilde{x}, \tilde{y}$, respectively, and the activation function for all layers $\phi : \mathbb{R}^n \to \mathbb{R}^n, \phi(\tilde{x}) = \big[\phi_1(\tilde{x}^1)^T \cdots \phi_l(\tilde{x}^l)^T\big]$ is applied to the concatenated vector.

2) Lipschitz Continuity Analysis: Denoting each element of η as $\eta_j, j \in \mathbb{I}_1^{M-1}$, we define $\eta_j = f_j(\boldsymbol{\theta}) = W^{l_j} \boldsymbol{s}^{l_j} + b^{l_j}$, with $W^{l_j} \in \mathbb{R}^{1 \times n_l}$ and $b^{l_j} \in \mathbb{R}$ the j-th row of W^l and b^l. We characterize the NN without the output layer using:

$$\Sigma = \begin{bmatrix} W^0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W^{l-1} & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & I_n \end{bmatrix}. \tag{14}$$

Then, we introduce Lipschitz continuity of each f_j.

Definition 2: A function $f_j : \mathbb{R}^n \to \mathbb{R}^m$ is globally Lipschitz continuous if there exists an $L_j \ge 0$ such that:

$$\|f_j(x) - f_j(y)\| \le L_j \|x - y\| \ \forall x, y \in \mathbb{R}^n \tag{15}$$

The smallest L_j such that condition (15) holds is called the Lipschitz constant $L_j^\star$, which provides an upper bound on how much the output f_j can change when the input varies from x to y. To find the smallest $L_j \ge L_j^\star$, we firstly introduce the following theorem:

Theorem 1 [13]: If there exists $L_j^2 > 0, T \in \mathcal{D}_n$ such that:

$$\begin{bmatrix} \Sigma \\ V \end{bmatrix}^T F(T) \begin{bmatrix} \Sigma \\ V \end{bmatrix} + \begin{bmatrix} -L_j^2 I & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & W^{l_j \top} W^{l_j} \end{bmatrix} \preceq 0 \tag{16}$$

then each f_j is globally continuous with a Lipschitz constant $L_j \ge L_j^\star$. For a detailed proof, refer to [13].

The smallest upper bound for each Lipschitz constant is obtained by solving the following semidefinite programs:

$$\min_{L_j^2, T} L_j^2 \quad \text{s.t. (16)} \tag{17}$$

where L_j^2 and T are the decision variables and $j \in \mathbb{I}_1^{M-1}$.

The bound $\bar{\Delta}_h$ on the constraints relaxation $\Delta_{hn|k}$ induces a corresponding upper bound $\bar{\eta}_j \in \mathbb{R}_+$ for each η_j. To ensure that the NN output η_j remains within the prescribed bound $\bar{\eta}_j$, we derive a corresponding bound on L_j:

$$L_j \le \bar{L}_j, \text{ with } \bar{L}_j = \frac{\bar{\eta}_j}{\bar{\Delta}_g + \bar{\Delta}_{\mathbf{x}}}, \tag{18}$$

where $\bar{\Delta}_g$ specifies the maximum variation of the disturbance b_g and $\bar{\Delta}_{\mathbf{x}} = \max_{\mathbf{x}\in \text{Proj}_{\Theta}, \mathbf{u}\in\mathcal{U}} \left\| \mathbf{x}_+^{\mathbf{u}} - \mathbf{x} \right\|$. Utilizing the Lipschitz continuity of each f_j with constant L_j, we obtain

$$\eta_j \le L_j \left\| \begin{bmatrix} \left[\Delta_{g_{n|k+1}}\right]_{n\in\mathbb{I}_k^{k+M-1}} \\ \mathbf{x}_{k+1} - \mathbf{x}_k \end{bmatrix} \right\| \le L_j\left(\bar{\Delta}_g + \bar{\Delta}_{\mathbf{x}}\right) \le \bar{\eta}_j,$$

which guarantees the NN outputs a slack variable $\Delta_h \le \bar{\Delta}_h$ whenever $\left\| \left[\Delta_{g_{n|k+1}}\right]_{n\in\mathbb{I}_k^{k+M-1}} \right\| \le \bar{\Delta}_g$ holds. The inequality (18) can be verified after the NN training phase or enforced during its training using the method proposed in [13]. Denoting the output of the NN as $\hat{\Delta}_h n|k$, we assume the NN satisfies the following assumption.

Assumption 8: The approximation error is uniformly bounded element-wise for the trained regressor, i.e.,

$$\left\| \Delta_h^\star n|k - \hat{\Delta}_h n|k \right\| \le \epsilon \mathbf{1}. \tag{19}$$

We then redefine the notion of safety.

Definition 3 (Safety): A controller $\mathbf{u}_k = \kappa(\mathbf{x}_k)$ is safe for the system (1) in a set $\mathcal{S} \subseteq \mathbb{R}^{n_x}$ if $\forall \mathbf{x} \in \mathcal{S}$, the control inputs $\mathbf{U} = \{\kappa(\mathbf{x}_0), \ldots, \kappa(\mathbf{x}_\infty)\}$ and the corresponding state trajectories $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_\infty\}$ are such that $h(\mathbf{x}_k, \mathbf{u}_k) \le E\bar{\Delta}_h$ and $g(\mathbf{x}_k, \mathbf{u}_k, b_{g_k}) \le 0$, $\forall k \ge 0$.

Finally, we formulate the learning-based uncertainty-responsive SMPC problem as follows:

$$\min_{\mathbf{u}} \sum_{n=k}^{k+N-1} q_{\mathbf{r}}\left(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\right) + p_{\mathbf{r}}\left(\mathbf{x}_{k+N|k}\right)$$

$$\text{s.t. (4a), (4b), (4d), (4f)} \tag{20a}$$

$$h\left(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}\right) \le E\left(\hat{\Delta}_h n|k + \epsilon\mathbf{1}\right), \quad n \in \mathbb{I}_k^{k+M-1} \tag{20b}$$

$$\left(\mathbf{x}_{n|k}.\hat{\Delta}_h n|k\right) \in \mathcal{Z}_{\mathbf{r}}^{s}, \qquad n \in \mathbb{I}_{k+N}^{k+M-1} \tag{20c}$$

C. Relaxed Consistency Condition and Recursive Feasibility

When Assumption 5 does not hold, feasibility can no longer be guaranteed by Proposition 1. To overcome such a limitation, we introduce the following relaxed assumption.

Assumption 9: The hard constraint function satisfies

$$g\left(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k+1}}\right) = g\left(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}\right) + \Delta_{g_{n|k+1}}, \tag{21}$$

where

$$\left\| \left[\Delta_{g_{n|k+1}}\right]_{n\in\mathbb{I}_k^{k+M-1}} \right\| \le \bar{\Delta}_g k + 1, \tag{22}$$

with $\bar{\Delta}_g k + 1 = \min_j \frac{\bar{\eta}_j - \epsilon}{\bar{L}_j} - \|\mathbf{x}_{k+1} - \mathbf{x}_k\|$.

Assumption 9 bounds the *inconsistency* $\Delta_{g_{n|k+1}}$ that causes the violation of the Assumption 5 and is applied in Case 2 of the proof of the following Theorem 2, which addresses the scenario where Assumption 5 no longer holds.

Theorem 2: Suppose that Assumptions 1-4, and 6-9 hold, and that Problem (4) is feasible for the initial state $\mathbf{x}_k$. Then, system (1) in closed loop with the solution of (20) applied in receding horizon is safe at all times.

Proof: Feasibility at time k ensures that there exist $\mathbf{U}_k = \{\mathbf{u}_{k|k}, \ldots, \mathbf{u}_{k+M-1|k}\}$, $\mathbf{X}_k = \{\mathbf{x}_{k|k}, \ldots, \mathbf{x}_{k+M|k}\}$ and $\tilde{\Delta}_h k = \{\Delta_h k|k, \ldots, \Delta_h k + M - 1|k\}$ that satisfy $h(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}) \le E\Delta_{hn|k}$, and $g_{n|k}(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}) \le 0$ for all $n \in \mathbb{I}_k^{k+M-1}$. At time $k+1$, we consider two cases:

1) Assumption 5 holds.
2) Assumption 5 does not but Assumption 9 holds.

Case 1: We verify that the trajectories $\mathbf{X}_{k+1}^s = \{\mathbf{x}_{k+N+1|k}, \ldots, \mathbf{x}_{k+M|k}, f(\mathbf{x}_{k+M|k}, \mathbf{u}_{\text{safe}})\}$, $\mathbf{U}_{k+1}^s = \{\mathbf{u}_{k+N+1|k}, \ldots, \mathbf{u}_{k+M-1|k}, \mathbf{u}_{\text{safe}}\}$, and $\tilde{\Delta}_h^s k + 1 = \{M_h\Delta_h k + N + 1|k, \ldots, M_h\Delta_h k + M - 1|k\}$ satisfy $h(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}) \le EM_h\Delta_{hn|k}$, and $g(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k+1}}) \le g_{n|k}(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}}) \le 0$, $\forall n \in \mathbb{I}_{k+N}^{k+M-1}$. Since $\mathbf{x}_{k+M|k} \in \mathcal{X}_{\text{safe}}$, Assumption 4 ensures $h(\mathbf{x}_{k+M|k}, \mathbf{u}_{\text{safe}}) \le 0$, $g_{k+M|k+1}(\mathbf{x}_{k+M|k}, \mathbf{u}_{\text{safe}}, b_{g_{k+M|k+1}}) \le 0$, and $f(\mathbf{x}_{k+M|k}, \mathbf{u}_{\text{safe}}) \in \mathcal{X}_{\text{safe}}$. Thus, there exist control inputs $\mathbf{u} := \mathbf{u}_{k+N|k}$ ensuring $(f(\mathbf{x}_{k+N|k}, \mathbf{u}), M_h\Delta_{hk+N|k}) \in \mathcal{Z}_{\mathbf{r}}^s$. Recursive feasibility follows from Assumption 5, ensuring that $\mathbf{U}_k$, $\mathbf{X}_k$, $\tilde{\Delta}_h k$ and their prolongation to infinite time satisfy constraints h and g.

Case 2: Assumption 9 ensures that $\eta_{j_{n|k+1}} \le L_j(\bar{\Delta}_g + \|\mathbf{x}_{k+1} - \mathbf{x}_k\|) \le \bar{\eta}_j - \epsilon$, i.e., $\hat{\Delta}_h n|k + 1 + \epsilon\mathbf{1} \le \bar{\Delta}_h$. At the same time, Assumptions 7 and 8 ensure that $\Delta_{hn|k+1}^\star$ exists and satisfies $\Delta_{hn|k+1}^\star \le \hat{\Delta}_h n|k + 1 + \epsilon\mathbf{1} \le \bar{\Delta}_h$ such that (8) is feasible, and the corresponding solutions $\mathbf{U}_{k+1}^\star = \{\mathbf{u}_{k+1|k+1}^\star, \ldots, \mathbf{u}_{k+M|k+1}^\star\}$ and $\mathbf{X}_{k+1}^\star = \{\mathbf{x}_{k+1|k+1}^\star, \ldots, \mathbf{x}_{k+M+1|k+1}^\star\}$ satisfying $h_{n|k+1}(\mathbf{x}_{n|k+1}^\star, \mathbf{u}_{n|k+1}^\star) \le E\Delta_h^\star n|k+1 \le E(\hat{\Delta}_h n|k + \epsilon\mathbf{1}) \le E\bar{\Delta}_h$, $g_{n|k+1}(\mathbf{x}_{n|k+1}^\star, \mathbf{u}_{n|k+1}^\star, b_{g_{n|k+1}}) \le 0$, $\forall n \in \mathbb{I}_{k+1}^{k+M}$, therefore guaranteeing the safety of the system by Definition 3. ■

D. Priority-Driven Soft-Constrained MPC

This section introduces an MPC approach that relax, on a priority basis, the constraints defined by the matrix E in (8c), such that the constraints (4d) are satisfied.

Let E^i, $i \in \mathbb{I}_1^l$, denote the i-th relaxation choice, ranked from 1 (highest priority) to l (lowest priority). For each E^i, two NNs are trained: the first NN, $\text{NN}_{E^i}^1$, is trained with $\boldsymbol{\theta} \in \Theta$, and predicts the constraint softening parameter $\hat{\Delta}_h$. The second network, $\text{NN}_{E^i}^2$ is trained using both feasible $\boldsymbol{\theta} \in \Theta$ and infeasible $\boldsymbol{\theta} \notin \Theta$ samples and serves as an infeasibility indicator, predicting whether the SMPC problem (9) is feasible under relaxation E^i. The output of $\text{NN}_{E^i}^2$ is denoted by $\mathcal{F}^i$, defined as:

$$\mathcal{F}^i = \begin{cases} 0, \text{ if (9) with } E = E^i \text{ is feasible,} \\ 1, \text{ if (9) with } E = E^i \text{ is infeasible.} \end{cases} \tag{23}$$

Algorithm 1: Priority-Driven Soft-Constrained MPC

Initialize: Set $\mathbf{x}_0$.
for *each time step* k **do**
  Obtain $\mathbf{x}_k$, evaluate $g_{n|k}(\mathbf{x}_{n|k}, \mathbf{u}_{n|k}, b_{g_{n|k}})$.
  if *(5) holds* **then**
    Solve (4).
  else if *(22) holds* **then**
    Compute $\mathcal{F}^i$ via $\mathrm{NN}^2_{E^i}$ for $i \in \mathbb{I}^l_1$. **for** *lowest-priority* i *with* $\mathcal{F}^i = 0$ **do**
      Solve (20) with $E = E^i$, $\hat{\Delta}_h$ from $\mathrm{NN}^1_{E^i}$.
  else
    Report failure, exit.

We summarize the proposed priority-driven soft-constrained MPC in Algorithm 1. At each time step, the controller first solves (4) if Assumption 5 holds. Otherwise, if the relaxed condition in Assumption 9 is satisfied, the controller evaluates a ranked set of constraint relaxations and applies the feasible option E^i with the lowest priority, using $\hat{\Delta}_h$ provided by $\mathrm{NN}^1_{E^i}$.

IV. Simulation Result

In this section, we present an autonomous driving example focusing on collision avoidance with a pedestrian at a crosswalk. The simulations are performed on a laptop computer using MATLAB, with the CasADi software package [14] and the IPOPT solver [15].

A. Vehicle Model and System Constraints

We model the vehicle longitudinal dynamics by defining the system state as $\mathbf{x} = [p\ v\ a]^T$ and the control input as $\mathbf{u} = a^{\mathrm{req}}$, where p represents the longitudinal position, v is the longitudinal velocity, a is the acceleration, and a^{req} is the requested acceleration. The system dynamics are defined as $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -t_{\mathrm{acc}} \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 0 \\ t_{\mathrm{acc}} \end{bmatrix}$ with $t_{\mathrm{acc}} = 1.8$ s a time constant that defines the acceleration dynamics. We assume that system (1) is subject to the following a-priori known box constraints:

$$0 \le v \le \bar{v}, \underline{a} \le a \le \bar{a}, \underline{a} \le a^{\mathrm{req}} \le \bar{a}, \underline{j} \le j \le \bar{j}, \underline{j} \le j^{\mathrm{req}} \le \bar{j},$$

where $j = a_+ - a$ and $j^{\mathrm{req}} = a^{\mathrm{req}}_+ - a^{\mathrm{req}}$ represent the actual and required jerks. The constraint (4d) is formulated to prevent collisions, thus leading to $C_g = [1\ 0\ 0]$, $D_g = 0$, and $b_g = p^{\mathrm{obs}}$ where p^{obs} represents the position of an obstacle, which may vary based on real-time observations. We express the safe set as all states such that the vehicle has come to a full stop, formulated as $\mathcal{X}_{\mathrm{safe}} := \{\mathbf{x} \mid \mathbf{x} = A\mathbf{x} + B\mathbf{u}, h(\mathbf{x}, \mathbf{u}) \le 0, v_{k+M|k} = 0\}$. The MPC controller is designed with a sampling time $t_s = 0.05$s and horizons $N = 20$ and $M = 100$. The dynamics in (6) were discretized using a fourth-order Runge-Kutta method with five steps per control interval.

B. Data Collection and Training

We define two relaxation modes. One with E^1 relaxing only $\underline{j}$ with δ_j, and the other one with E^2 relaxing both $\underline{j}$ with δ_j, and $\underline{a}$ with δ_a. For each mode E^i, two NNs, $\mathrm{NN}^1_{E^i}$ and $\mathrm{NN}^2_{E^i}$, are trained. The input for each NN, $\mathrm{NN}^{1,2}_{E^i}$, is $\boldsymbol{\theta} = [p^{\mathrm{obs}} - p\ v\ a]^\top$. The output of $\mathrm{NN}^1_{E^1}$ is $\eta = \delta_j$, while the output of $\mathrm{NN}^1_{E^2}$ is $\eta = [\delta_j\ \delta_a]^\top$. The NNs are trained offline on datasets generated by solving (8) in various driving scenarios. Each dataset is constructed by varying the NN inputs over predefined ranges. The velocity v ranges from 0 to 5.5m s^{-1}, and the acceleration a spans from -3.5 to 0.1m s^{-2}. The relative distance $p - p^{\mathrm{obs}}$ is sampled between 0.1 and 5m, while the requested acceleration a^{req} varies from -3.7 to 2.5m s^{-2}. Each parameter is incremented in uniform steps of 0.1, providing a detailed dataset that captures a wide range of possible conditions for the NN training. The computational complexity depends on the number of relaxable constraints and the horizon length, which affect dataset size and training phase. In our example, we use a two-layer NN with 50 and 30 neurons and hyperbolic tangent activation functions. The choice of NN architecture and size vary with real-world scenarios. Further discussion on scalability is provided in our extended work [16].

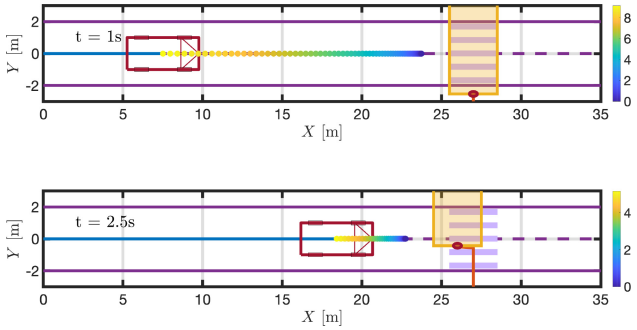

Fig. 1. Two time instances of the scenario. Yellow shaded region denotes the predicted position of a pedestrian. Sensing the pedestrian, ego vehicle plans a trajectory (yellow/green/blue line indicating high/medium/low speed) within the sensing range.

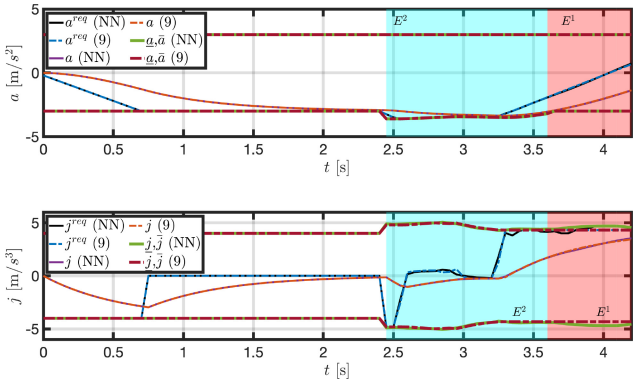

Fig. 2. Closed-loop trajectories with NNs and (9). Light blue and red regions indicate relaxation choices E^2 and E^1, respectively.

C. Simulation Scenario

In this scenario, at $t_1 = 2.5$s, a sudden change in pedestrian position is detected with $\Delta_g = 1$m, as shown in Fig. 1. In Fig. 2, adjusting only $\underline{j}$ is insufficient and will lead to an infeasible solution at $t_1 = 2.5$s. To restore feasibility, $\mathrm{NN}^1_{E^2}$ is deployed, updating both $\underline{j}$ and $\underline{a}$ to maintain feasibility.

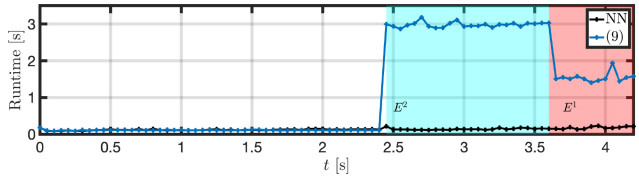

Fig. 3. Computation time of applying NN and (9).

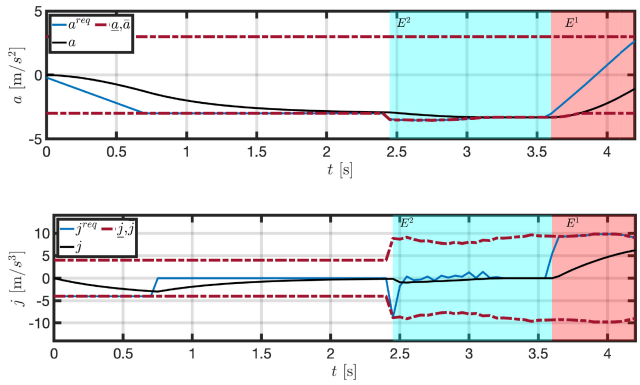

Fig. 4. Closed-loop trajectories with softened MPC.

However, around $t_2 = 3.6$s, $\mathcal{F}^1$ signals that feasibility can be maintained using applying E^1, prompting a switch to $\text{NN}^1_{E^1}$.

We simulate the same scenario using the solution of (9). The close-loop trajectories obtained by applying the NNs and (9) are represented in Fig. 2. At $t_1 = 2.5$s, infeasibility is detected when solving (9) with $E = E^1$, requiring a switch to $E = E^2$ until feasibility with $E = E^1$ is restored. As seen in Fig. 2, both methods result in the ego vehicle decelerating to a complete stop to avoid a collision with the pedestrian. However, the proposed method achieves this while reducing computational execution time, highlighting its potential for real-time applications, as shown in Fig. 3.

We further compare the results with those obtained by applying a softened MPC approach that introduces δ_j and δ_a as slack variables to relax constraints on $\underline{j}$ and $\underline{a}$, respectively. As shown in Fig. 4, with appropriate tuning ($R_{\delta_j} = 10^1$, $R_{\delta_a} = 1.5 \times 10^3$), softened MPC prioritizes relaxing δ_j, demonstrating that adjusting penalty weights enables some control over which constraint is relaxed more, producing results comparable to our approach. However, strict prioritization cannot be guaranteed through tuning alone. Moreover, as the number of slack variables increases, identifying an appropriate tuning strategy that effectively enforces the desired prioritization becomes increasingly challenging [6].

V. Conclusion

In this letter, we proposed a priority-driven constraint softening for SMPC that dynamically adjusts constraint relaxations in response to external disturbances. By integrating NNs to approximate the solution of the proposed auxiliary optimization problem, our approach ensures safe and feasible control with reduced computational complexity. The simulation results of an autonomous driving scenario demonstrated the effectiveness of the proposed method in handling unexpected constraints while preserving safety. While in this letter we have considered the case where the external disturbance enters the constraints, in future extensions we will consider systems with perturbed and uncertain dynamics.

References

[1] I. Batkovic, M. Ali, P. Falcone, and M. Zanon, "Safe trajectory tracking in uncertain environments," *IEEE Trans. Autom. Control*, vol. 68, no. 7, pp. 4204–4217, Jul. 2023.

[2] I. Batkovic, A. Gupta, M. Zanon, and P. Falcone, "Experimental validation of safe MPC for autonomous driving in uncertain environments," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 5, pp. 2027–2042, Sep. 2023.

[3] Z. Liu and O. Stursberg, "Recursive feasibility and stability of MPC with time-varying and uncertain state constraints," in *Proc. 18th Eur. Control Conf. (ECC)*, 2019, pp. 1766–1771.

[4] T. Manrique, M. Fiacchini, T. Chambrion, and G. Millérioux, "MPC tracking under time-varying polytopic constraints for real-time applications," in *Proc. Eur. Control Conf. (ECC)*, 2014, pp. 1480–1485.

[5] S. V. Raković, S. Zhang, H. Sun, and Y. Xia, "Model predictive control for linear systems under relaxed constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 369–376, Jan. 2023.

[6] E. Kerrigan and J. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," presented at United Kingdom Autom. Control Council, Sep. 2000.

[7] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press.

[8] E. Kerrigan and J. Maciejowski, "Designing model predictive controllers with prioritised constraints and objectives," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Design*, Glasgow, U.K., 2002, pp. 33–38.

[9] J. Vada, O. Slupphaug, T. A. Johansen, and B. A. Foss, "Linear MPC with optimal prioritized infeasibility handling: Application, computational issues and stability," *Automatica*, vol. 37, no. 11, pp. 1835–1843, 2001.

[10] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Madison, WI, USA: Nob Hill Publishing, 2017.

[11] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. Eur. Control Conf. (ECC)*, 2013, pp. 502–510.

[12] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–17.

[13] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," *IEEE Control Syst. Lett.*, vol. 6, pp. 121–126, 2021.

[14] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, pp. 25–57, Mar. 2006.

[16] F. Prignoli, Y. S. Quan, M. Jeddi, J. Sjöberg, and P. Falcone, "Priority-driven safe model predictive control approach to autonomous driving applications," 2025, *arXiv:2505.05933*.