# Uncertainty Quantification and Robustification of Model-based Controllers using Conformal Prediction

**Kong Yao Chee** CKONGYAO@SEAS.UPENN.EDU
**Thales C. Silva** SCTHALES@SEAS.UPENN.EDU
**M. Ani Hsieh** M.HSIEH@SEAS.UPENN.EDU
**George J. Pappas** PAPPASG@SEAS.UPENN.EDU
*University of Pennsylvania, Philadelphia, PA 19104*

**Editors:** A. Abate, K. Margellos, A. Papachristodoulou

## Abstract

In modern model-based control frameworks such as model predictive control or model-based reinforcement learning, machine learning has become a ubiquitous class of techniques deployed to improve the accuracy of the dynamics models. By leveraging expressive architectures such as neural networks, these frameworks aim to improve both the model accuracy and the control performance of the system, through the construction of accurate data-driven representations of the system dynamics. Despite achieving significant performance improvements over their non-learning counterparts, there are often little or no guarantees on how these model-based controllers with learned models would perform in the presence of uncertainty. In particular, under the influence of modeling errors, noise and exogenous disturbances, it is challenging to ascertain the accuracy of these learned models. In some cases, constraints may even be violated, rendering the controllers unsafe. In this work, we propose a novel framework that can be applied to a large class of model-based controllers and alleviates the above mentioned issues by robustifying the model-based controllers in an online and modular manner, with provable guarantees on the model accuracy and constraint satisfaction. The framework first deploys conformal prediction to generate finite-sample, provably valid uncertainty regions for the dynamics model in a distribution-free manner. These uncertainty regions are incorporated into the constraints through a dynamic constraint tightening procedure. Together with the formulation of a predictive reference generator, a set of *robustified* reference trajectories are generated and incorporated into the model-based controller. Using two practical case studies, we demonstrate that our proposed methodology not only produces well-calibrated uncertainty regions that establish the accuracy of the models, but also enables the closed-loop system to satisfy constraints in a robust yet non-conservative manner.

**Keywords:** Learning-based control, Model-based control, Uncertainty quantification

## 1. Introduction

There is an uptrend in the application of model-based controllers across a wide range of domains due to recent advances in nonlinear optimization frameworks and an increase in the availability of computational resources, *e.g.*, for temperature control in buildings (Yao and Shekhar (2021)), for autonomous vehicles (Wu et al. (2022)) and in quadrotor control (Chee et al. (2023a)). The proliferation of machine learning methods has concurrently led to an increase in the development of learning-enhanced, model-based control frameworks that leverage learning tools to improve control performance through an improvement in the dynamics models, *e.g.* Jiahao et al. (2023). Despite the surge in these developments, the question of how these learning-based control frameworks would perform in the presence of uncertainty, remains an active research topic (Mesbah et al. (2022); Brunke et al. (2022)). In this work, we approach this problem by proposing a novel framework that systematically allows the model-based controller to satisfy constraints robustly, under the collective influence of model mismatch, noise and external disturbances.

*Related Works*: There are a number of methods in the literature that provide constraint satisfaction through the modification of the control inputs. Examples include methods that utilize control barrier functions, *e.g.*, Marvi and Kiumarsi (2021) and Ames et al. (2019), and predictive safety filters, such as Chen et al. (2023) and Wabersich and Zeilinger (2021). Our approach in this work is fundamentally different from these methods since we aim to augment the reference trajectories given to the model-based controller, rather than the output of the controller. There are works that use either neural networks or Gaussian processes to characterize the uncertainty of the system dynamics such as in Aswani et al. (2013); Berkenkamp and Schoellig (2015); Soloperto et al. (2018). However, these are often limited to linear nominal models. For a survey of methods that consider safety from various aspects, readers are referred to Brunke et al. (2022). Reference and command governors (Garone et al. (2017)) are methods, which at first glance, have some resemblance to our approach. However, our pipeline is more holistic as it includes an uncertainty quantification procedure and the distribution-free nature of conformal prediction (CP) allows us to quantify uncertainties of any form. Furthermore, our method is general as it can be applied and is catered to nonlinear systems. There is a growing interest in the application of CP within control and planning frameworks, with recent works given in Lindemann et al. (2023) and Dixit et al. (2023). In this work, we not only adopt a state-of-the-art variant of CP that does not require the data samples to be exchangeable, but we also delineate a principled approach of incorporating the uncertainties into a predictive reference generator, which enables robust constraint satisfaction for a given model-based controller.

*Contributions*: Our contributions in this work are three-fold. First, by considering a receding horizon variant of weighted conformal prediction (WCP), we show how the uncertainty regions computed by WCP can be incorporated into a dynamic constraint tightening procedure. Second, we formulate a novel methodology, which integrates uncertainty quantification and constraint tightening with a predictive reference generator in an online manner. The predictive reference generator computes a set of *robustified* reference trajectories, which can be incorporated to a large class of model-based control frameworks. By tracking these robustified trajectories, the proposed framework guarantees robust constraint satisfaction with high probability. Finally, through two case studies, we show that our proposed framework enhances the robustness of the given model-based control schemes, without inducing additional feasibility issues. To the best of our knowledge, this is the first work that integrates weighted conformal prediction with constraint tightening in a receding horizon procedure and leverages them to enhance the robustness of model-based controllers.

## 2. Problem Setup and Preliminaries

We consider a discrete-time, nonlinear system with the following dynamics,

$$x^+ = f\left(x, u, w, d(x, u)\right), \tag{1}$$

where the function $f : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^n$ denotes the system dynamics, the vectors $x, x^+ \in \mathbb{R}^n$ are the current and successor states, $u \in \mathbb{R}^{n_u}$ is the control input to the system, $w \in \mathbb{R}^p$ is the process noise acting on the system. The function $d : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^q$ maps (possibly unknown) external disturbances to the system dynamics. To control the system, we are given a model-based control framework that generates a sequence of control inputs, which is applied to the system during deployment, described in the following section.

### 2.1. Model-based Control with Learned Dynamics Models

In many model-based control frameworks that use machine or deep learning techniques, architectures such as neural networks are trained to obtain an accurate model of the system dynamics. In this work, we assume that we are given a (possibly partially unknown) model-based control

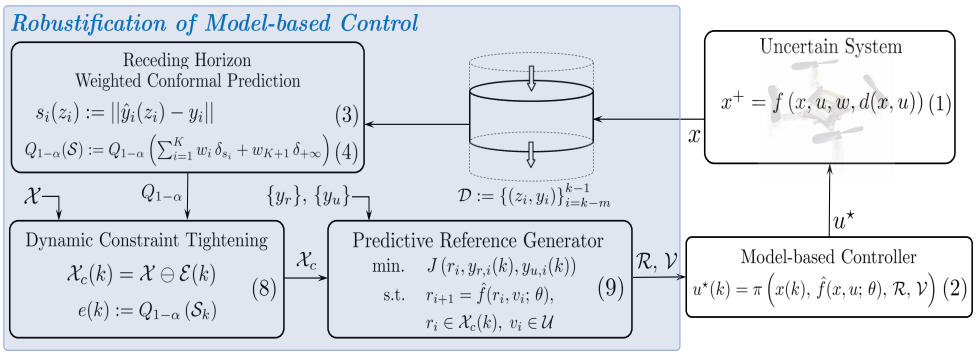


Figure 1: Schematic of the proposed framework, highlighted in blue. Quadrotor image: Bitcraze.

framework to control the system (1). In particular, at each time step $k$, the model-based controller $\pi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{(N+1)n} \times \mathbb{R}^{Nn} \to \mathbb{R}^{n_u}$ produces a control input $u^\star(k) \in \mathbb{R}^{n_u}$, which is given as

$$u^\star(k) = \pi\left(x(k),\, \hat{f}(x, u;\, \theta),\, \mathcal{R},\, \mathcal{V}\right), \tag{2}$$

where $x(k) \in \mathbb{R}^n$ denotes the current measurement of the state, the sequences $\mathcal{R} := \{r_i\}_{i=1}^{N+1}$ and $\mathcal{V} := \{v_i\}_{i=1}^{N}$ denote the reference state and control trajectories that the model-based controller is required to track, and $N \in \mathbb{N}_+$ denotes the length of the reference trajectories. The function $\hat{f} : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ denotes the dynamics model with learned parameters $\theta \in \mathbb{R}^b$. While we may not know the full details of the control framework, we assume to have access to the learned model $\hat{f}(x, u;\, \theta)$. Many existing frameworks fall within this general class of control frameworks. For instance, recent methods in learning-based model predictive control (MPC), *e.g.*, Salzmann et al. (2023), Chee et al. (2023c), lie within this class of controllers. Model-based reinforcement learning techniques such as those formulated by Lambert et al. (2019) and Williams et al. (2016) fall within this category as well. Iterative LQR and differential dynamic programming methods, such as those described in Li and Todorov (2004) and Tassa et al. (2014) can also be subsumed under this class.

*Problem 1*: Given the model $\hat{f}(x, u;\, \theta)$, we would like to quantify the uncertainty region between the model and the true system dynamics with provable guarantees on the model accuracy, and utilize these uncertainty regions to ensure that constraints are robustly satisfied for a given model-based controller $\pi(x, \hat{f}, \mathcal{R}, \mathcal{V})$, in the presence of modeling errors, noise and external disturbances.

Our approach to solving *Problem 1* is depicted in Fig. 1. Before describing our approach in Section 3, we first recall some key concepts in conformal prediction.

### 2.2. Conformal Prediction

Conformal prediction (CP) is an uncertainty quantification technique that is used in tandem with a given regression model to produce provably valid prediction intervals, as described in Vovk et al. (2005). This method assumes that the data samples used to compute the prediction intervals are either *i.i.d.* or exchangeable. For time series data, this is a strong assumption. In this work, we use a state-of-the-art variant of CP, known as weighted CP (WCP), developed by Barber et al. (2023), to alleviate this assumption.

Consider a dataset of $K$ samples, $\mathcal{D} := \{(z_i,\, y_i)\}_{i=1}^{K}$, where $\{z_i\}_{i=1}^{K}$ and $\{y_i\}_{i=1}^{K}$ are the features and labels of the dataset. For time series data, these features and labels are the states and successor states of the system respectively, but for clarity, we maintain the notation of $z$ and $y$ in our exposition of WCP. The objective is to construct a conformalized uncertainty interval $\mathcal{Q}(z)$ such that for a test data sample $z$ that is not within $\mathcal{D}$, its label $y$ lies within $\mathcal{Q}(z)$ with a pre-defined probability. The algorithm first uses the samples in the dataset to compute a set of nonconformity scores $\mathcal{S} := \{s_i\}_{i=1}^{K}$, where

$$s_i(z_i) := ||\hat{y}_i(z_i) - y_i||_1 \tag{3}$$

and $\{\hat{y}_i(z_i)\}_{i=1}^{K}$ are the predictions computed using the model $\hat{f}$. These nonconformity scores account for the deviation between the predictions given by the model and the true values in a natural

way - the smaller the nonconformity scores, the more accurate the predictions. Next, given a miscoverage rate $\alpha \in (0, 1)$, we compute the weighted $(1-\alpha)^{\text{th}}$ empirical quantile of $\mathcal{S}$, $Q_{1-\alpha}(\mathcal{S})$,

$$Q_{1-\alpha}(\mathcal{S}) := Q_{1-\alpha}\left(\sum_{i=1}^{K} w_i\, \delta_{s_i} + w_{K+1}\, \delta_{+\infty}\right), \tag{4}$$

where $\{w_i\}_{i=1}^{K+1}$ are normalized weights, $Q_{1-\alpha}(\cdot)$ denotes the $(1-\alpha)^{\text{th}}$ quantile of the argument and $\delta_a$ denotes a probability point mass for a real number $a$ on the extended real line (Barber et al. (2023)). Finally, the conformalized interval is attained by extending symmetrically from the predictions of the learned model,

$$\mathcal{Q}(z) := [\hat{y}(z) - Q_{1-\alpha}(\mathcal{S}),\, \hat{y}(z) + Q_{1-\alpha}(\mathcal{S})]\,. \tag{5}$$

Qualitatively, this procedure uses the dataset $\mathcal{D}$ to compute the half-width of the intervals $Q_{1-\alpha}$. The choice of weights $\{w_i\}_{i=1}^{K+1}$ plays a role on how the exchangeability assumption is alleviated. For details on how the weights are chosen, the reader is referred to Barber et al. (2023).

## 3. Robustification of Model-based Controllers

### 3.1. Online Uncertainty Quantification

To quantify the uncertainty between the model and true system dynamics in an efficient manner, we consider a method to perform WCP in a receding horizon manner. We follow a procedure similar to that described in Section 2.2 by using the data samples within a window of size $m$. In particular, at each time step, the most recent nonconformity score is added and the oldest one is removed from the set of scores. This approach promotes recency, since we only use the most recent data samples to compute the uncertainty intervals. Details of this procedure are given in Algorithm 1. An inherited attribute of Algorithm 1 is that it makes no assumptions on the data distribution, and hence can be applied to any dynamics model to quantify any form of uncertainty between the model and the true system, such as those caused by modeling errors, process noise or external disturbances. Although not necessary, we highlight that using learning methods can help to bridge the gap between the model and the true system substantially, and this often leads to uncertainty intervals with relatively small magnitudes as compared to the magnitudes of the states. Instead of choosing the most recent data points, other sampling methods can be applied. For instance, it is possible to maintain a buffer and choose data that are representative of the errors between the model and true system. This may lead to smaller intervals and is related to the notion of persistent excitation, often discussed in adaptive control (Narendra and Annaswamy (2012)). Formally, the uncertainty intervals computed by Algorithm 1 satisfy the following guarantees.

**Algorithm 1** A receding horizon variant of WCP

**Input:** Current state and control input $x(k)$, $u(k)$, learned model $\hat{f}(\theta)$, window size $m$, weights $\{w_i\}_{i=1}^{m}$, miscoverage rate $\alpha$

**Output:** Conformalized uncertainty interval $\mathcal{Q}$

**Initialize:** $\mathcal{S} \leftarrow \{0\}$
At each time step $k$,
Compute model prediction $\hat{y}_k$ using $\hat{f}(x(k), u(k);\, \theta)$
**if** $k \geq 1$ **then**
  Calculate latest nonconformity score $s_k$ element-wise and add to set, $\mathcal{S} \leftarrow \{s_k\} \cup \mathcal{S}$
  **if** $k > m$ **then**
    Compute $Q_{1-\alpha}(\mathcal{S})$ and $\mathcal{Q}$ in (4) and (5) using $\{w_i\}_{i=1}^{m}$, $\alpha$ and $\hat{y}_{k-1}$, with $\mathcal{S} := \{s_i\}_{i=k-m}^{k-1}$
    Remove oldest nonconformity score from set, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{s_{k-m}\}$
  **end if**
**end if**

**Lemma 1** *(Adapted from Barber et al. (2023)) Given the dataset* $\mathcal{D} := \{(z_i, y_i)\}_{i=k-m}^{k-1}$ *and the dynamics model* $\hat{f}$*, let the predictions computed by the dynamics model at time* $t \geq k$ *be denoted by* $\hat{y}(z_t)$*. Then, for the miscoverage rate* $\alpha \in (0, 1)$*, the following inequality holds for the label* $y_t$ *and the width of the uncertainty interval* $Q_{1-\alpha}(\mathcal{S})$ *computed by Algorithm 1,*

$$\mathbb{P}\left\{||y_t - \hat{y}(z_t)|| \leq Q_{1-\alpha}(\mathcal{S})\right\} \geq 1 - \delta_t, \tag{6}$$

*where* $\delta_t := \alpha + \sum_{i=1}^{m} w_i \, d_{TV}\left(\mathcal{S}, \mathcal{S}^i\right)$*. The metric* $d_{TV}$ *denotes the total variation between two distributions. The set* $\mathcal{S}^i$ *consists of the scores when the data sample* $(z_t, y_t)$ *is replaced with the* $i^{th}$ *sample in* $\mathcal{S}$*, as defined in Algorithm 1.*

**Proof** For any data sample $(z_t, y_t)$, $t \geq k$, applying Theorem 2 in Barber et al. (2023) gives $\mathbb{P}\left\{y_t \in \mathcal{Q}\left(z_t\right)\right\} \geq 1 - \delta_t$. With $\mathcal{Q}(z_t)$ as in (5), $y_t \in \mathcal{Q}\left(z_t\right) \Leftrightarrow ||y_t - \hat{y}(z_t)|| \leq Q_{1-\alpha}(S)$ and that gives the required result. ■

Lemma 1 implies that for any feature $z_t$, the corresponding label $y_t$ is guaranteed to be within $\mathcal{Q}$ with probability of $1 - \delta_t$. Importantly, the data samples do not have to be *i.i.d.* or exchangeable. This result implies the following for a sequence of states predicted by the dynamics model.

**Proposition 2** *Consider an arbitrary state trajectory of length* $N$ *at time* $k$*, denoted by* $\{x_i\}_{i=k}^{k+N}$*, and let* $\delta := N\delta_t$*. For a predicted state trajectory* $\{\hat{x}_i\}_{i=k+1}^{k+N}$ *computed by the dynamics model* $\hat{f}$ *starting from* $x_k$*, we have*

$$\mathbb{P}\left\{||x_i - \hat{x}_i|| \leq Q_{1-\alpha}(\mathcal{S}),\ \forall i \in \{k+1, \ldots, k+N\}\right\} \geq 1 - \delta \tag{7}$$

**Proof** We use a technique similar to that in Theorem 1 of Lindemann et al. (2023). We first apply Lemma 1 with $z_t := x_{i-1}$ and $y_t := x_i$. For each $i \in \{k+1, \ldots, k+N\}$, we have

$$\mathbb{P}\left\{||x_i - \hat{x}_i|| \leq Q_{1-\alpha}(\mathcal{S})\right\} \geq 1 - \delta_t \Leftrightarrow \mathbb{P}\left\{||x_i - \hat{x}_i|| > Q_{1-\alpha}(\mathcal{S})\right\} < \delta_t.$$

By applying the union bound, we have

$$\begin{aligned}\mathbb{P}\left\{\exists i \in \{k+1, \ldots, k+N\} \,\middle|\, ||x_i - \hat{x}_i|| > Q_{1-\alpha}(\mathcal{S})\right\} &= \mathbb{P}\left\{\bigcup_{i=k+1}^{k+N} ||x_i - \hat{x}_i|| > Q_{1-\alpha}(\mathcal{S})\right\} \\ &\leq \sum_{i=k+1}^{k+N} \mathbb{P}\left\{||x_i - \hat{x}_i|| > Q_{1-\alpha}(\mathcal{S})\right\} < N\delta_t = \delta.\end{aligned}$$

The result in (7) is obtained by considering the probability of the complementary event. ■

Proposition 2 provides a certificate that the sequence of state predictions given by the learned model would be close to the true system states, with high probability. We note that even though the second term of $\delta_t$ may be challenging to compute, the effect of this term can be controlled through a careful selection of weights, $w_i$. In Section 5, we show that as the intervals are computed online and with an accurate model, the intervals are not overly conservative. This paves the way for these intervals to be used in constraint tightening, which enables robust constraint satisfaction.

### 3.2. Dynamic Constraint Tightening

To ensure that the constraints are satisfied in a robust yet non-conservative manner, our approach is to tighten the constraint sets dynamically during deployment, using the uncertainty intervals computed by Algorithm 1. In general, given the constraint set $\mathcal{X}$, we would like to find a set of operations, denoted by $\ominus$, such that it tightens the constraint set in an online manner, *i.e.*, $\mathcal{X}_c(k) := \mathcal{X} \ominus \mathcal{E}(k)$, where $\mathcal{E}(k)$ denote the uncertainty intervals. In this work, we consider a common, yet important, type of constraints to exemplify the role of the uncertainty intervals in the dynamic constraint tightening procedure. Consider constraints that are represented by the polyhedral set, $\mathcal{X} := \{x \in \mathbb{R}^n : Ax \leq b\}$, where $A \in \mathbb{R}^{n_\mathcal{X} \times n}$ and $b := [b_1, \ldots, b_{n_\mathcal{X}}]^\top \in \mathbb{R}^{n_\mathcal{X}}$. The rows of $A$ are given by $\{a_i^\top\}_{i=1}^{n_\mathcal{X}}$, $a_i \in \mathbb{R}^n$. Next, we define an uncertainty vector $e(k) :=$

$[e(k)_{x_1}, \ldots, e(k)_{x_n}]^\top \in \mathbb{R}^n$, where $e(k)_{x_i} := Q_{1-\alpha}(\mathcal{S}_{x_i})$ is the width of the interval $\mathcal{Q}(\mathcal{S}_{x_i})$ computed for the $i^{\text{th}}$ element of $x$ at time step $k$. With that, we have the following tightened constraints,

$$a_i^\top x \leq b_i - a_i^\top e(k) := c_i(k), \quad \forall i \in \{1, \ldots, n_{\mathcal{X}}\}, \tag{8}$$

and $\mathcal{X}_c(k) := \{x \in \mathbb{R}^n \,:\, Ax \leq c(k)\}$, where $c(k) := [c_1(k), \ldots, c_{n_{\mathcal{X}}}(k)]^\top$. This procedure can be generalized for the case where the uncertainty intervals are asymmetric about the predictions, *e.g.*, with methods in Romano et al. (2019), by formulating the constraints and $e(k)$ appropriately.

This proposed procedure of computing the uncertainty intervals with Algorithm 1 and incorporating them into a dynamic constraint tightening procedure is also useful as a standalone component. It can be applicable in related contexts such as in robust MPC (Rawlings et al. (2017)) and in trajectory planning schemes, *e.g.*, Romero et al. (2022). In the next section, we incorporate this procedure into a predictive reference generator, and that allows us to construct a set of reference state and control trajectories for a given controller.

### 3.3. Predictive Reference Generator

To generate reference trajectories that enable the model-based controller to satisfy constraints robustly, we first consider the following finite-horizon optimization problem at each time step $k$,

$$\begin{aligned}
\underset{\{r_i\},\{v_i\}}{\text{minimize}} \quad & \sum_{i=0}^{M-1} ||r_i - y_{r,i}(k)||_Q^2 + ||v_i - y_{u,i}(k)||_R^2 + ||r_M - y_{r,M}(k)||_P^2 && \text{(9a)}\\
\text{subject to} \quad & r_{i+1} = \hat{f}(r_i, v_i;\, \theta), \quad i \in \{0, \ldots, M-1\} && \text{(9b)}\\
& r_i \in \mathcal{X}_c(k), \quad v_i \in \mathcal{U}, \quad i \in \{0, \ldots, M-1\} && \text{(9c)}\\
& r_M = y_{r,M}(k), && \text{(9d)}\\
& r_0 = x_k, && \text{(9e)}
\end{aligned}$$

where $M \in \mathbb{N}_+$ is the reference horizon and $\mathcal{X}_c(k) \subseteq \mathbb{R}^n$, $\mathcal{U}(k) \subseteq \mathbb{R}^{n_u}$ denote the tightened state and control input constraints respectively. The matrices $Q$, $P \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{n_u \times n_u}$ penalize the stage and terminal costs, and the control input cost respectively. The sequences $\{y_{r,i}(k)\}_{i=0}^{M}$ and $\{y_{u,i}(k)\}_{i=0}^{M-1}$ denote the desired states and control inputs, which are not necessarily dynamically feasible nor adhere to the constraints. For instance, the desired states can be step commands. The vector $x_k \in \mathbb{R}^n$ is the state measurement obtained at time step $k$. Upon solving (9) at each time step, we obtain a set of optimal state and control trajectories, $\{r_i^\star\}_{i=k}^{k+M}$ and $\{v_i^\star\}_{i=k}^{k+M-1}$. These reference trajectories are dynamically feasible and are *robustified*, as they adhere to both the dynamics and tightened constraints. The predictive reference generator, and details on how the trajectories $\mathcal{R}$ and $\mathcal{V}$ are extracted, are given in Algorithm 2.

A few remarks are in order. First, the terminal equality constraints (9d) play an essential role, as it allows the reference trajectories to be extrapolated in a smooth manner, when necessary (see lines 13-16 of Algorithm 2). Second, the problem (9) is solved in a receding horizon manner, albeit at a lower sampling rate, as compared to the operating frequency of the given controller. Downsampling allows the reference generator to consider a longer horizon, while maintaining moderate computational costs. Third, the reference horizon $M$ is of a different length, as compared to that of the given controller, and should be chosen such as it is longer than the latter. This allows the reference generator to account for the effect of the tightened constraint sets on the trajectories over a longer horizon. We demonstrate the importance of this design choice in Section 5. We highlight that the estimated uncertainties are not propagated along the prediction horizon and this helps to promote feasibility when solving (9). Instead, we rely on the tracking performance of the model-based controller to follow the reference trajectory well enough such that constraints can be satisfied. With Proposition 2 and the dynamic constraint tightening procedure, we have the following result for the state trajectories obtained from solving (9).

**Theorem 3** *Consider a set of reference state and control trajectories obtained from the solution of Problem (9) at time $k$, denoted by $\{r_i^\star\}_{i=k}^{k+N}$ and $\{v_i^\star\}_{i=k}^{k+N-1}$. For a state trajectory $\{x_i\}_{i=k}^{k+N}$ that starts at an initial state $x_k \in \mathcal{X}$ and is obtained by applying $\{v_i^\star\}_{i=k}^{k+N-1}$, it holds that*

$$\mathbb{P}\{x_i \in \mathcal{X}\} \geq 1-\delta, \quad \forall i \in \{k, \ldots, k+N\}. \tag{10}$$

**Proof** First, for each time step $k$, the reference trajectories, $\{r_i^\star\}_{i=k}^{k+N}$ and $\{v_i^\star\}_{i=k}^{k+N-1}$, comply with the model $\hat{f}$, since they satisfy the constraints (9b). Next, by applying Proposition 2, the following holds with probability of at least $1-\delta$,

$$||x_i - r_i^\star|| \leq Q_{1-\alpha}(\mathcal{S}) = e(k) \Leftrightarrow x_i \leq r_i^\star + e(k), \ \forall i \in \{k+1, \ldots, k+N\}, \tag{11}$$

with $e(k)$ defined in Section 3.2. Next, since the reference states satisfy the tightened constraints (9c), we have

$$r_i^\star \in \mathcal{X}_c(k) \overset{(8)}{\Longleftrightarrow} Ar_i^\star \leq b - Ae(k) \Leftrightarrow A(r_i^\star + e(k)) \leq b, \ \forall i \in \{k+1, \ldots, k+N\}$$

Combining this with (11), and together with the hypothesis $x_k \in \mathcal{X}$, we get the required result, where

$$Ax_i \leq A(r_i^\star + e(k)) \leq b \ \Leftrightarrow \ x_i \in \mathcal{X}, \forall i \in \{k, \ldots, k+N\},$$

holds with a probability of at least $1-\delta$. ■

**Algorithm 2** Robustification of Model-based Control

**Input:** Current state $x(k)$, model $\hat{f}(\theta)$, constraint sets $\mathcal{X}$,$\mathcal{U}$, desired trajectories $\{y_r\}$, $\{y_u\}$, down-sampling count $p$, reference horizon $M$, control horizon $N$

**Output:** Output state trajectories $\mathcal{R}$ and control input trajectories $\mathcal{V}$

**Initialize:** $j \leftarrow 0$, $\{r_s\} \leftarrow \{0\}, \{v_s\} \leftarrow \{0\}, \{r_c\} \leftarrow \{0\}, \{v_c\} \leftarrow \{0\}$
At each (control) time step $k$,
Compute uncertainty intervals $\mathcal{Q}(k)$ with Algorithm 1
**if** $k \bmod p = 0$ **then**
  Compute tightened constraint set $\mathcal{X}_c(k)$ using $\mathcal{Q}(k)$, $\mathcal{X}$ and (8)
  Solve problem (9) with $x(k)$, $\hat{f}$, $\mathcal{X}_c(k)$, $\mathcal{U}$ and $\{y_r, y_u\}$ to get $\{r_i^\star\}_{i=k}^{k+M}$ and $\{v_i^\star\}_{i=k}^{k+M-1}$
  Extract output trajectories $\{r_i^\star\}_{i=k}^{k+N}$, $\{v_i^\star\}_{i=k}^{k+N-1}$ from $\{r_i^\star\}_{i=k}^{k+M}$, $\{v_i^\star\}_{i=k}^{k+M-1}$, $M \gg N$
  Store reference trajectories $\{r_c\}, \{v_c\} \leftarrow \{r_i^\star\}_{i=k}^{k+M}, \{v_i^\star\}_{i=k}^{k+M-1}$
  Store time index $j \leftarrow k$
**else**
  **if** $k+N \leq j+M$ **then**
    Extract output trajectories $\{r_i^\star\}_{i=k}^{k+N}$, $\{v_i^\star\}_{i=k}^{k+N-1}$ from latest ref. trajectory $\{r_c\}$, $\{v_c\}$
  **else**
    Remove oldest element and append with last element in latest ref. trajectory,
    $\{r_i^\star\}_{i=k}^{k+N} \leftarrow (r_s \setminus r_{s,0}) \cup r_{j+M}^\star, \quad \{v_i^\star\}_{i=k}^{k+N-1} \leftarrow (v_s \setminus v_{s,0}) \cup v_{j+M-1}^\star$
  **end if**
**end if**
Store output trajectories $\{r_s\}, \{v_s\} \leftarrow \{r_i^\star\}_{i=k}^{k+N}, \{v_i^\star\}_{i=k}^{k+N-1}$
Apply output trajectories $\mathcal{R} := \{r_i^\star\}_{i=k}^{k+N}$ and $\mathcal{V} := \{v_i^\star\}_{i=k}^{k+N-1}$ to model-based controller $\pi$

We note that while this result is applicable when the proposed framework is incorporated into a model-based controller, this is considered an open-loop result. The closed-loop performance is primarily dependent on how well the controller track the reference trajectories, which falls beyond the scope of this paper and will be explored in future work. In the following sections, we show that the framework performs well, when integrated with contemporary model-based controllers.

## 4. Case Studies

We consider two practical systems to illustrate the efficacy of the proposed methodology - the cart-pole and the quadrotor system.

### 4.1. Cartpole with Learning-based MPC

The cartpole dynamics are given as $\ddot{\theta} = (g\sin\theta - \cos\theta(F + m_p l\dot{\theta}^2\sin\theta))/(l(\frac{4}{3} - \frac{m_p\cos^2\theta}{m_c+m_p}))$ and $\ddot{x}_c = (F + m_p l(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta))/(m_c + m_p)$, where $x_c$ is the cart position and $\theta$ is the angle between the pole and the vertical (Barto et al. (1983)). The masses and the pole length are denoted by $m_c$, $m_p$ and $2l$. Gravity is denoted by $g$ and $F$ denotes an external force. By defining the state $x := [x_c \ \dot{x}_c \ \theta \ \dot{\theta}]^\top$ and the control input $u := F$, the discretized dynamics form the nominal dynamics. The cart has a true mass of 1.5kg, while the mass within the nominal dynamics is set at 1kg. We consider additive Gaussian process noise $w$ with zero mean and standard deviations of $\{0.01\text{m}, 0.1\text{m/s}, 2^\text{o}, 10^\text{o}\text{/s}\}$ added to the dynamics. The cart is to track a sequence of step commands. The state constraints are $\mathcal{X} := A_x x \leq b_x$ , where $A_x := \left[\begin{smallmatrix} \mathbf{0} & A_1 \\ \mathbf{0} & A_2 \end{smallmatrix}\right]$, $A_1 := \left[\begin{smallmatrix} 1 & 0 \\ -1 & 0 \end{smallmatrix}\right]$ and $A_2 := \left[\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right]$, and $b_x := [1.7 \ 1.7 \ 4.2 \ 3.8]^\top$. For Alg. 1, the window size $m := 150$ and runs at the same rate as the given model-based controller of 10Hz. The miscoverage rate $\alpha := 0.01$. The computation of the uncertainty intervals is activated after $t = 15s$, since $m = 150$. For the constraint tightening procedure, we increase the tightening by a factor of two to enhance the robustness of reference trajectories. The horizon in the reference generator is $M := 100$. The cost matrices in (9a) are set as $Q := \mathbb{I}_4$, $R := 0.5$ and $P := 10\,\mathbb{I}_4$. The problem (9) is formulated in CasADi (Andersson et al. (2019)) and solved using IPOPT (Wächter and Biegler (2006)). For the model-based controller, we deploy learning-based MPC from our previous work (Chee et al. (2022)). The parameters for this scheme are given as $Q := \{1, 0.01, 1, 0.01\}$, $R := 0.5$, $P := \{1, 10, 10, 10\}$ and $N := 25$ and we highlight that these are not known to the proposed framework. The learned model $\hat{f}$ uses the equations of motion as the nominal dynamics and a neural ODE to account for residual dynamics.

### 4.2. Quadrotor with Nonlinear MPC

Next, we consider a high-dimensional quadrotor, whose dynamics are written as $\ddot{r} = g + R\eta/m$ and $\dot{\omega} = J^{-1}(\tau - \omega \times J\omega)$, where $r, \omega \in \mathbb{R}^3$ are the position and angular rates of the quadrotor, $\eta$ and $\tau \in \mathbb{R}^3$ are the motor thrust and moments (Mellinger and Kumar (2011)). We denote gravity and the matrix that maps $\eta$ to its acceleration by $g$ and $R$ respectively. The mass and moments of inertia are given as $m$ and $J$. The state and controls are $x := [r^\top \ \dot{r}^\top \ q^\top \ \omega^\top]^\top$ and $u := [\eta \ \tau^\top]^\top$.The quadrotor is tasked to track a 3D path and move towards an end position, while avoiding an cylindrical obstacle, as depicted in Fig. 4. The cross section of the obstacle has an origin at $(x_o, y_o, z_o) = (5, 0, 0)$m with radius $r_o = 1$m and extends to $(-4, 4)$m in the z axis. The tightened state constraint is given by $\sqrt{(x - x_o)^2 + (y - y_o)^2} \geq r_o + Q_{1-\alpha}(\mathcal{S}_r)$, where $Q_{1-\alpha}(\mathcal{S}_r)$ is computed using Algorithm 1, with the scores $s_k$ computed using the 1-norm of the position $r$. Additive Gaussian process noise $w$ with zero mean and standard deviations of $\{0.5\text{m}, 0.5\text{m/s}, 3^\text{o}, 20^\text{o}\text{/s}\}$ are included in the dynamics. The window size $m$ is set to be 150 and $\alpha := 0.02$. The reference generator has the parameters $M := 250$, $Q := \text{blkdiag}\{\mathbb{I}_3, 5\mathbb{I}_3, 0.1\mathbb{I}_4, 0.5\mathbb{I}_3\}$, $R := 0.0001\mathbb{I}_4$ and $P := \text{blkdiag}\{100\mathbb{I}_3, 5\mathbb{I}_3, 0.1\mathbb{I}_4, 0.5\mathbb{I}_3\}$. We do not consider external disturbances and deploy a nonlinear MPC control scheme as the model-based controller. The parameters $Q$, $P$ and $R$ are the same as those of the reference generator, while $N := 50$. The model-based controller and problem (9) are formulated in CasADi and IPOPT. The setup for both examples are based on implementations from our previous work (Chee et al. (2023b,c)).

## 5. Results and Discussion

For the cartpole system described in Section 4.1, we conduct 30 runs with different process noise profiles. First, we examine the intervals computed by Algorithm 1–they are plotted in Fig. 2. The narrow widths of the intervals observed in Fig. 2 highlight the benefit of having an accurate learned model $\hat{f}(x, u, \theta)$, as compared to a possibly less accurate non-learning model. Specifically, using an

accurate learned model accounts for most of the mismatch between the true system and the nominal model. This allows the remaining discrepancies to be potentially smaller, resulting in relatively small uncertainty intervals. Consequently, the dynamic constraint tightening procedure can produce useful tightened constraints for downstream tasks. In our formulation, there is noise in the system and the model may not be perfect. Hence, the case in which the nonconformity scores are zero is not applicable.

We compare our approach against a nominal case where only the given model-based controller is used and constraints are not included. The time histories of the states across 30 runs are plotted in Fig. 3. Due to the presence of modeling errors and noise, there are instances where the states violate the constraints for the nominal case. On the other hand, the incorporation of the proposed robustification allows the state trajectories to stay within the constraints.

Next, we consider two test cases where either the nominal or tightened constraints are directly included in the given controller. This provides a straightforward way of satisfying the constraints, but may encounter infeasibility during the runs. It is observed that 25 and 2 out of the 30 runs did not encounter feasibility issues for the cases with the nominal and tightened constraint sets respectively (83.3% and 6.7% success rate). In contrast, our proposed approach achieved of 100% success rate and did not encounter any feasibility issues in any of the runs. Apart from illustrating the brittleness of this direct approach, these test cases lead to an interesting perspective and an important feature of our framework. As the predictive reference generator in Algorithm 2 considers a longer horizon as compared to that of the underlying model-based controller, it is easier to obtain a feasible trajectory that satisfies the constraints robustly. This however comes at the expense of a higher computational cost, but that can be traded off with a lower sampling rate, as described in Algorithm 2. Our framework can thus be interpreted as a dual-timescale architecture with the reference generator working in tandem with the model-based controller such that the system tracks a sequence of desired states and control inputs, while ensuring robust constraint satisfaction.

For the quadrotor system, the paths under a nominal MPC scheme, with and without tightening, as well as with the proposed robustification, are depicted in Fig. 4. With the proposed robustification, the path taken is more conservative. The tightened constraints are satisfied with a larger margin, as compared to the case where the tightened constraints are included in nominal MPC. In the case where the constraints are not tightened, there are instances where the path violates the constraints, as shown in the bottom right subplot. The uncertainty intervals computed are relatively small in magnitude, as compared to the magnitudes of the states. One possible reason could be that the nonconformity scores are computed using one-step model predictions, and that the state measurements are obtained at a sufficiently high sampling rate. These allow the nonconformity scores to be closer to being exchangeable, as compared to the case where these scores are computed using multi-step model predictions, or when the measurements are obtained less frequently. This interesting observation remains to be studied in future work.

## 6. Conclusion

We present a novel robustification scheme that can be incorporated into a wide range of model-based controllers to enhance their ability to satisfy constraints robustly. The framework quantifies the uncertainties that arise from modeling errors, process noise and external disturbance in a non-conservative manner, by utilizing WCP. Through dynamic constraint tightening and the predictive reference generator, a set of robustified trajectories are computed and serve as inputs to the model-based controller. We demonstrate through the case studies that the proposed framework allows the system to adhere to the constraints with robust margins, without inducing infeasibility issues.

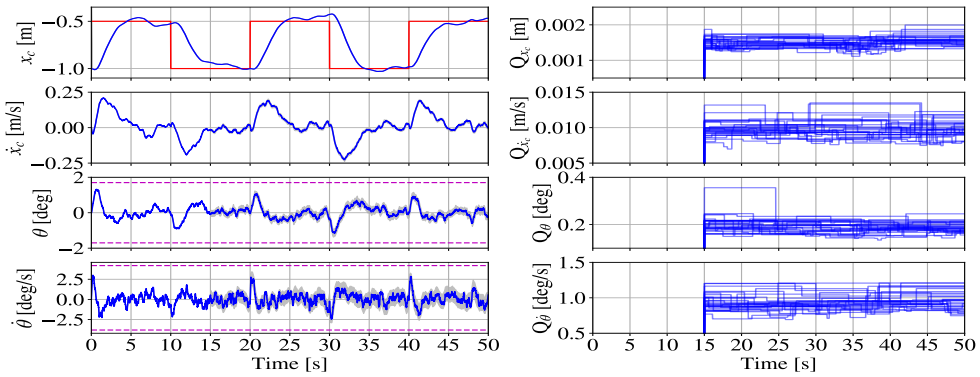


Figure 2: **Left**: Time histories of the cartpole states in a single run. The grey regions are the uncertainty intervals computed by Alg. 1. The red lines in the first subplot are the commands given to the cart to track. The dashed magenta lines depict the constraints. **Right**: The time-varying widths of the uncertainty intervals $Q_{1-\alpha}(\mathcal{S})$ for each state that are computed during deployment, for the 30 runs.

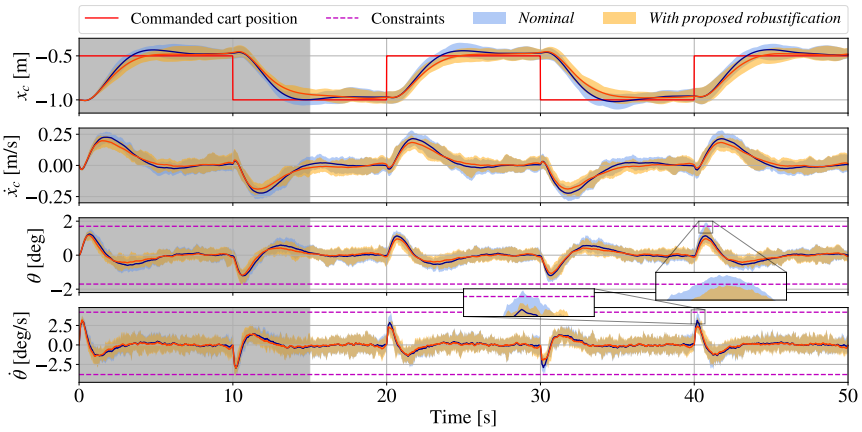


Figure 3: Time histories of the cartpole states for the nominal and robustified schemes across 30 runs. The grey region denotes the period in which conformal prediction has not been activated. The colored regions denote the spread of the trajectories. The transparency of these regions has been adjusted for visibility. The dark blue and orange lines denote the mean trajectories, taken across 30 runs. The zoomed-in plots depict constraint violation for the nominal controller.

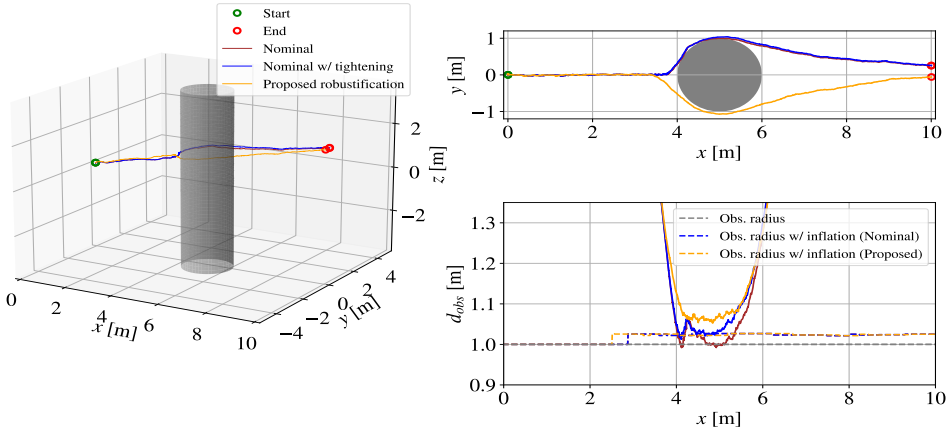


Figure 4: **Left**: The 3D path of the quadrotor with a cylindrical obstacle between the start and end points. **Right**: The top subplot shows the top view of the quadrotor path. The solid lines in the bottom subplot indicate the distance of the quadrotor to the obstacle, *i.e.* $d_{\text{obs}} := \sqrt{(x - x_{\text{o}})^2 + (y - y_{\text{o}})^2}$, while the dashed lines depict the radius of the obstacle, with and without inflation.

## References

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Math. Program. Computation*, 11(1):1–36, 2019.

Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2):816–845, 2023.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Syst., Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Felix Berkenkamp and Angela P Schoellig. Safe and robust learning control with gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501. IEEE, 2015.

Bitcraze. Crazyflie 2.1. URL https://www.bitcraze.io/products/crazyflie-2-1/.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Kong Yao Chee, Tom Z Jiahao, and M Ani Hsieh. KNODE-MPC: A Knowledge-Based Data-Driven Predictive Control Framework for Aerial Robots. *IEEE Robotics and Automation Letters*, 7(2): 2819–2826, 2022.

Kong Yao Chee, M Ani Hsieh, and Nikolai Matni. Learning-enhanced nonlinear model predictive control using knowledge-based neural ordinary differential equations and deep ensembles. In *Learning for Dynamics and Control Conference*, pages 1125–1137. PMLR, 2023a.

Kong Yao Chee, M. Ani Hsieh, and George J. Pappas. Uncertainty quantification for learning-based mpc using weighted conformal prediction. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 342–349, 2023b. doi: 10.1109/CDC49753.2023.10383587.

Kong Yao Chee, Thales C Silva, M Ani Hsieh, and George J Pappas. Enhancing sample efficiency and uncertainty compensation in learning-based model predictive control for aerial robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9435–9441. IEEE, 2023c.

Shaoru Chen, Kong Yao Chee, Nikolai Matni, M Ani Hsieh, and George J Pappas. Safety filter design for neural network systems via convex optimization. *arXiv preprint arXiv:2308.08086*, 2023.

Anushri Dixit, Lars Lindemann, Skylar X Wei, Matthew Cleaveland, George J Pappas, and Joel W Burdick. Adaptive conformal prediction for motion planning among dynamic agents. In *Learning for Dynamics and Control Conference*, pages 300–314. PMLR, 2023.

Emanuele Garone, Stefano Di Cairano, and Ilya Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

Tom Z Jiahao, Kong Yao Chee, and M Ani Hsieh. Online dynamics learning for predictive control with an application to aerial robots. In *Conference on Robot Learning*, pages 2251–2261. PMLR, 2023.

Nathan O Lambert, Daniel S Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer SJ Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 4(4):4224–4230, 2019.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.

Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J Pappas. Safe planning in dynamic environments using conformal prediction. *IEEE Robotics and Automation Letters*, 2023.

Zahra Marvi and Bahare Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *International Journal of Robust and Nonlinear Control*, 31(6):1923–1940, 2021.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

Ali Mesbah, Kim P Wabersich, Angela P Schoellig, Melanie N Zeilinger, Sergio Lucia, Thomas A Badgwell, and Joel A Paulson. Fusion of machine learning and mpc under uncertainty: What advances are on the horizon? In *2022 American Control Conference (ACC)*, pages 342–357. IEEE, 2022.

Kumpati S Narendra and Anuradha M Annaswamy. *Stable adaptive systems*. Courier Corporation, 2012.

James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022.

Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

Raffaele Soloperto, Matthias A Müller, Sebastian Trimpe, and Frank Allgöwer. Learning-based robust model predictive control with state-dependent uncertainty. *IFAC-PapersOnLine*, 51(20): 442–447, 2018.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1): 25–57, 2006.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

Jingda Wu, Zhiyu Huang, and Chen Lv. Uncertainty-aware model-based reinforcement learning: Methodology and application in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 8(1):194–203, 2022.

Ye Yao and Divyanshu Kumar Shekhar. State of the art review on model predictive control (mpc) in heating ventilation and air-conditioning (hvac) field. *Building and Environment*, 200:107952, 2021.