Journal Title
XX(X):1–21
©The Author(s) 2025
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

# CC-VPSTO: Chance-Constrained Via-Point-Based Stochastic Trajectory Optimisation for Safe and Efficient Online Robot Motion Planning

**Lara Brudermüller[1], Guillaume O. Berger[2], Julius Jankowski[3], Raunak Bhattacharyya[1], Sylvain Calinon[3], Raphaël M. Jungers[2], and Nick Hawes[1]**

## Abstract

Reliable robot autonomy hinges on decision-making systems that account for uncertainty without imposing overly conservative restrictions on the robot's action space. We introduce *Chance-Constrained Via-Point-Based Stochastic Trajectory Optimisation (CC-VPSTO)*, a real-time capable framework for generating task-efficient robot trajectories that satisfy constraints with high probability by formulating stochastic control as a chance-constrained optimisation problem. Since such problems are generally intractable, we propose a deterministic surrogate formulation based on Monte Carlo sampling, solved efficiently with gradient-free optimisation. To address bias in naïve sampling approaches, we quantify approximation error and introduce padding strategies to improve reliability. We focus on three challenges: (i) sample-efficient constraint approximation, (ii) conditions for surrogate solution validity, and (iii) online optimisation. Integrated into a receding-horizon MPC framework, CC-VPSTO enables safe, reactive control under uncertainty. The strengths of our approach lie in its generality, *i.e.*, no assumptions on the underlying uncertainty distribution, system dynamics, cost function, or the form of inequality constraints; and its applicability to online robot motion planning. We demonstrate the validity and efficiency of our approach in both simulation and on a Franka Emika robot. Videos and additional material are made available here.

## Keywords

Chance-Constrained Optimisation, Stochastic Model Predictive Control, Trajectory Optimisation

## 1 Introduction

Uncertainty is inherent to most real-world robotics applications, arising from noisy sensors, imprecise actuators, and incomplete or evolving knowledge of the environment. Effectively leveraging this uncertainty is essential for achieving reliable and efficient robot behaviour, particularly in online motion planning tasks that require fast adaptation to new information. In this work, we adopt a *chance-constrained* perspective, aiming to satisfy task-specific constraints, such as avoiding collisions (cf. Figure 1), limiting interaction forces, or achieving task success, with high probability (Prékopa 2013; Dai et al. 2019). While traditional robust control methods (Köhler et al. 2023; Majumdar and Tedrake 2017; Badings et al. 2023) optimise for the worst-case scenario under *bounded uncertainty*, they can be overly conservative or unrealistic. Instead, chance-constrained optimisation allows us to model uncertainty *probabilistically*, enabling the use of more general uncertainty models (Margellos et al. 2014; Schildbach et al. 2014). Crucially, we do not assume that hard constraint violations are catastrophic; instead, our objective is to trade off constraint satisfaction and task performance (*e.g.*, the efficiency of a motion trajectory) in a principled manner that avoids unnecessary conservatism in an online motion planning setting. While traditional Model Predictive Control (MPC) approaches can implicitly provide some robustness through high-frequency re-planning, their reliance on deterministic assumptions often leads to myopic or brittle behaviour in stochastic settings. Our goal is to incorporate available probabilistic information into the control pipeline, enabling more flexible and robust decision-making in uncertain environments. Despite their promise, chance-constrained formulations are generally difficult to solve exactly and, in the worst case, intractable (Blackmore et al. 2010). One common strategy is to approximate the chance constraint and reformulate the problem as a deterministic surrogate that can be addressed using standard optimisation techniques. However, identifying a suitable approximation is often non-trivial and may introduce significant conservatism at the cost of task efficiency.

Towards this end, we propose *Chance-Constrained Via-Point-Based Stochastic Trajectory Optimisation (CC-VPSTO)*, a practical and tractable Monte Carlo approach for chance-constrained optimisation problems that is suitable for real-time robot motion planning. By sampling from

---

[1] Oxford Robotics Institute, University of Oxford, UK
[2] ICTEAM, UCLouvain, Belgium
[3] Idiap Research Institute & Ecole Polytechnique Fédérale de Lausanne (EPFL), CH

**Corresponding author:**
Lara Brudermüller, Oxford Robotics Institute,
23 Banbury Rd, Oxford OX2 6NN, UK
Email: larab@robots.ox.ac.uk

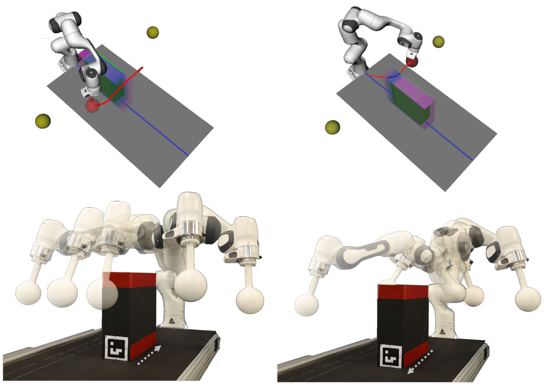

**Figure 1.** ***Robot experiment.*** The robot is tasked to move its ball-shaped end effector from a start point on one side to a goal point on the other side of a conveyor belt (indicated by the yellow balls in the right simulation view). Meanwhile, the ball end effector has to avoid the box obstacle on a moving conveyor belt which is controlled according to a stochastic policy. Depending on the anticipated box movement, the robot can either pass the box in front (left images) or behind (right images). This problem setting requires the robot to be *reactive* whilst being able to plan *safe* motions in real-time. It further poses a trade-off between safety and performance, *i.e.*, reaching the other side of the conveyor belt in minimum time.

the uncertainty distribution, we construct a deterministic surrogate problem that can be efficiently solved using gradient-free optimisation. A key challenge in this approach is that naive Monte Carlo formulations can bias the solution toward regions that appear feasible on the sampled data but violate the true chance constraint (Homem-de Mello and Bayraksan 2014). To address this, we quantify the extent of this bias and introduce padding techniques that mitigate its effect. Our method focuses on three main challenges: *i)* selecting an appropriate number of samples to accurately approximate the chance constraint, *ii)* establishing when a solution to the surrogate problem is likely to satisfy the true chance constraint, and *iii)* solving the resulting surrogate problem efficiently in real time.

In summary, the main contributions of this work are:

1. A new surrogate formulation to chance constraints focusing on sample efficiency, while accounting for the increased approximation error.
2. Theoretical and empirical evaluations of the correctness of the approach. Under the assumption of independence between the solution and the samples, we show that the solution to the surrogate problem satisfies the true chance constraint, *i.e.*, the chance constraint that is intractable, with high confidence.
3. A detailed discussion of the conditions under which the independence assumption holds in theory and the use of the surrogate formulation as a heuristic in practice.
4. The integration of the chance-constrained optimisation problem into a Model Predictive Control (MPC) framework for *online* reactive robot control Jankowski et al. (2023), enabling safe receding-horizon control under uncertainty.

We demonstrate the validity and efficiency of our approach in a motion planning task with stochastic dynamic obstacles in simulation, and in real-robot experiments with a Franka Emika robot arm. The key advantages of our approach are: *i)* flexibility to handle arbitrary uncertainty distributions, *ii)* compatibility with real-time MPC via parallelisable sampling and optimisation, and *iii)* support for general inequality constraints, such as collision avoidance, force limits, or performance objectives.

## 2 Related Work

Risk-averse planning and control methods in robotics aim to enforce constraints in the presence of uncertainty. These methods typically enforce these constraints by formulating them as chance constraints (CCs) or by using risk measures, such as Conditional Value-at-Risk (CVaR). When employed in a *online* receding horizon control scheme, these methods fall into the category of Stochastic Model Predictive Control (SMPC) (Heirung et al. 2018; Mesbah 2016). SMPC addresses optimal control problems for dynamical systems with stochastic uncertainty subject to chance constraints.

The literature distinguishes between two types of chance constraints. Constraint violation probabilities can either be formulated *point-wise*, *i.e.*, independently for each time step, or *jointly*, *i.e.*, over the entire (finite) time horizon. We focus on joint chance constraints in this work, as they are preferable in robotics where it is important to account for the cumulative effect of uncertainty over time. Many approaches bound the joint probability using either an *additive* approach, summing over all point-wise probabilities via Boole's inequality (*e.g.*, Ono and Williams (2008); Priore and Oishi (2023); Castillo-Lopez et al. (2020)), or a *multiplicative* approach, which involves explicitly constraining the product of the complements of point-wise probabilities (*e.g.*, Sun et al. (2016); Van Den Berg et al. (2011)), as discussed in Janson et al. (2017). Yet, both of these strategies do not account for the time correlations of uncertainty, and thus may lead to over- or underestimation of the joint probability of constraint violation. We thus adopt the time-wise supremum approach from Lew et al. (2023), which evaluates only the maximum constraint violation over the entire time horizon for a given trajectory, thereby capturing time correlations effectively.

The main challenge in SMPC is evaluating the probability of constraint violation over the planning horizon. This requires computing an expectation integral over time and space, which is typically intractable for general uncertainty distributions and constraint structures (Peña-Ordieres et al. 2020). As a result, SMPC must address two key questions: *i)* how to approximate or bound the probability of constraint violation in a tractable way, and *ii)* how to solve the resulting optimization problem with minimal computational overhead for online control. Previous approaches to these questions proposed semidefinite programming formulations (Jasour et al. 2015) or constraint tightening (Alcan and Kyrki 2022; Ono and Williams 2008; Parsi et al. 2022). While having proven to be effective in providing probabilistic guarantees on the satisfaction of chance constraints, they are typically tailored to very specific types of constraints, uncertainty distributions and/or system dynamics, thereby limiting their applicability to real robotics problems. As noted in Lew et al. (2023), there is still a lack of formulations and solution algorithms that are capable of truly capturing different sources of uncertainty as well as different types of constraints in a unified framework.

In contrast, sample-based methods offer a more general approach for approximating chance constraints, as they do not require any assumptions on the underlying probability distributions, as long as the number of samples is sufficiently large. In the sample-based setting we can distinguish between *scenario optimisation* (Schildbach et al. 2014; de Groot et al. 2023) and *Monte Carlo* methods (Blackmore et al. 2010; Schmerling and Pavone 2016; Blackmore 2006). Both use samples (aka. scenarios) of the uncertainty. Scenario optimisation synthesises controls satisfying the constraint for each of the samples and relies on a well-established theory to identify the right sample size for a given confidence level (Calafiore and Campi 2006). However, these theoretical bounds are mostly limited to convex or quasi-convex problems (Calafiore 2010; Berger et al. 2021) and solutions are typically overly conservative, *i.e.*, they require much larger sample sizes than identified by empirical tests (Schildbach et al. 2014). Monte Carlo methods typically approximate the probability of constraint violation from the samples, rooted in the *sample average approximation (SAA)* approach (Shapiro et al. 2021; Shapiro 2003; Pagnoncelli et al. 2009). They are generally less conservative and can be used with arbitrary constraints and uncertainty models. However, without further adjustments, they do not provide finite-sample guarantees, but only asymptotic guarantees (Blackmore 2006), implying the requirement of large sample sets, and higher computational resources. The need for large sample sets is reinforced when the desired probability of constraint violation is low, as is commonly targeted in robotics applications. A remedy to this can be importance sampling (Schmerling and Pavone 2016), or data reduction methods based on parameter estimation of sample statistics, *e.g.*, through computing moments of the probability distribution of the uncertainty (Wang et al. 2020; Priore and Oishi 2023; Blackmore et al. 2006; Yan et al. 2018). Yet, the propagation of moments can be complex, and requires restrictive assumptions, such as Gaussianity. Alternatively, for collision avoidance, Trevisan et al. (2025) propose a naive Monte Carlo approach that approximates the chance constraint using a fixed number of samples, increasing sample density by constraining the collision region across time steps. However, this method is tailored specifically to collision avoidance and does not account for the approximation error introduced by the finite sample size.

Other approaches have used SAA to approximate constraints on risk metrics like conditional Value-at-Risk (CVaR) instead (Lew et al. 2023; Yin et al. 2023; Nemirovski and Shapiro 2007). CVaR constraints are more conservative than chance constraints, as they account for tail events, but the resulting reformulation is smooth and convex, which enables the use of off-the-shelf optimisation tools, such as sequential convex programming (SCP). For instance, Lew et al. (2023) provide a general framework for risk-averse SMPC based on the combination of the SAA of CVaR constraints and concentration inequalities to bound the approximation error. However, their approach relies on strong continuity assumptions on the objective function and constraints, which may not hold in practice. Yin et al. (2023) use the SAA of CVaR constraints within a Model Predictive Path Integral (MPPI) controller but do not account for errors in the approximation of the CVaR constraint and limit the source of uncertainty to process noise. Finally, the work of Peña-Ordieres et al. (2020) reformulates the chance constraint as a quantile function and uses SAA to approximate it, which results in a formulation that is amenable to gradient-based optimisation methods. However, similar to Lew et al. (2023), their approach relies on continuity and differentiability assumptions of the constraint function.

Our main observation across chance constrained optimisation approaches is that they are typically tailored to specific constraint models (*e.g.*, collision avoidance constraints, or reaching polytopic target sets) with smoothness and continuity assumptions, or specific types of uncertainty (*e.g.*, Gaussian or bounded uncertainty). These restrictive assumptions limit the applicability of these approaches in real-world robotics problems. In contrast, we provide a general framework for chance-constrained finite-horizon optimal control problems with generic chance constraints and generic uncertainty distributions. Building upon the SAA approach, we propose a Monte Carlo method that actively accounts for the approximation error caused by the number of samples used in the approximation by adjusting the threshold for constraint violation based on a fixed confidence level. This allows us to use a small number of samples in order to efficiently solve the resulting deterministic problem with the stochastic trajectory optimisation framework VP-STO (Jankowski et al. 2023).

## 3 Problem Formulation

### 3.1 Preliminaries

Let $\Delta$ be a random variable that models the uncertainty of the system. This can include stochasticity in the dynamics, the environment, and the sensor measurements. A realization $\boldsymbol{\delta}$ of $\Delta$, denoted by $\boldsymbol{\delta} \sim \Delta$, will be referred to as an *uncertainty scenario*, or a *particle*.

### 3.2 Chance-Constrained Optimisation

In the following, we introduce the general chance-constrained optimisation problem. The goal is to find a solution $\boldsymbol{x}$ that minimizes a cost $J(\boldsymbol{x}) \in \mathbb{R}$ while satisfying a set of constraints. In our work, we consider inequality constraints, *i.e.*, constraints that can be formulated as a function $g$ being negative at $\boldsymbol{x}$, *i.e.*, $g(\boldsymbol{x}) \leq 0$. For instance, $g$ can encode a deterministic collision-avoidance constraint on the robot's distance to obstacles.

Chance-constrained optimisation generalises the above by allowing constraints that depend on a random variable. More precisely, the constraints have the form $g(\boldsymbol{x}, \boldsymbol{\delta}) \leq 0$, where $g$ depends on $\boldsymbol{x}$ and the realization $\boldsymbol{\delta}$ of the uncertainty variable $\Delta$. For instance, $g(\boldsymbol{x}, \boldsymbol{\delta}) \leq 0$ can encode a collision-avoidance requirement of a stochastic system in state $\boldsymbol{x}$ given a particular uncertainty realisation $\boldsymbol{\delta}$. Requiring that $g(\boldsymbol{x}, \boldsymbol{\delta}) \leq 0$ holds for *all* realizations of $\boldsymbol{\delta}$ is often overly conservative, or even infeasible. This is especially true if the distribution of $\Delta$ has unbounded support (such as Gaussian noise). Therefore, chance-constrained optimisation relaxes the constraint into a soft constraint, allowing violation of the constraint with a bounded probability $\eta$. It thus requires that the probability of a realisation $\boldsymbol{\delta} \sim \Delta$ to satisfy $g(\boldsymbol{x}, \boldsymbol{\delta}) > 0$ is smaller than $\eta$. A general chance-constrained optimisation problem can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{x} \in X} \quad & J(\boldsymbol{x}) \\ \text{s.t.} \quad & P_{\boldsymbol{\delta} \sim \Delta}[g(\boldsymbol{x}, \boldsymbol{\delta}) > 0] \leq \eta, \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is the decision variable, constrained in some domain $X$ (e.g., $X \subseteq \mathbb{R}^n$), $J : X \to \mathbb{R}$ is the objective function, and $\eta \in [0, 1]$ is a user-provided threshold for the probability of violating $g$. We assume that the probability distribution of $\Delta$ is known or that we have a generative model for $\Delta$ from which we draw samples, *i.e.*, we can draw an arbitrary number of *independent* samples $\boldsymbol{\delta}_1 \sim \Delta, \ldots, \boldsymbol{\delta}_N \sim \Delta$. We do not make any additional assumptions on the uncertainty distribution, *i.e.*, it can be of any type and is not restricted to additive noise formulations. However, note that state-dependent uncertainties are outside the scope of this work. The following example illustrates possible sources of uncertainty in a simplified robot motion planning problem.

*Example 1. System with uncertain initial condition, actuation noise and uncertain obstacle dynamics.* Consider a simple kinodynamic system in discrete time: $s(t+1) = s(t)(u(t) + w(t))$, where $s(t) \in \mathbb{R}$ is the system state, $u(t) \in \mathbb{R}$ the control input, and $w(t) \in \mathbb{R}$ is a random variable representing the actuation noise at time $t$. For each $t$, $w(t)$ follows a known distribution, say a uniform distribution between $-1$ and 1. In addition, $s(0)$ is a random variable with a known distribution as well, say a standard Gaussian distribution. Finally, there is a randomly moving obstacle covering the interval $[z(t), \infty)$ where $z(t)$ is a random variable with a known distribution, say an exponential distribution. The objective is to minimize the sum of squared inputs over two time steps, while avoiding the obstacles with high probability over two time steps. Following the formulation of (1), we have $\boldsymbol{x} = (u(0), u(1))$, $\boldsymbol{\delta} = (s(0), w(0), w(1), z(0), z(1), z(2))$, $J(\boldsymbol{x}) = u(0)^2 + u(1)^2$, and $g(\boldsymbol{x}, \boldsymbol{\delta}) = \max_{t=0,1,2} s(t) - z(t)$, where $s(t)$ follows the dynamics introduced above.

As mentioned above, the chance constraint is satisfied at $\boldsymbol{x}$ if the probability of violating the constraint $g$ at $\boldsymbol{x}$ is at most $\eta$. Computing this probability for a given $\boldsymbol{x}$ is often challenging. For this reason, chance-constrained optimisation problems are often very hard, if not impossible, to solve exactly. Therefore we contribute a tractable approximation of such problems, along with an analysis of the soundness of the approximation.

*Remark 1.* Note that Eq. (1) can be generalized to multiple chance constraints $P_{\boldsymbol{\delta} \sim \Delta}[g_i(\boldsymbol{x}, \boldsymbol{\delta}) > 0] \leq \eta_i$, for $i = 1, \ldots, L$, with different violation thresholds $\eta_i$. However, in this work, we will focus on a single joint constraint ($L = 1$) for simplicity.

### 3.3 Constraint Satisfaction as a Binary Random Variable

In this subsection, we reformulate the chance constraint in Eq. (1) as a constraint on a binary random variable obtained from $\Delta$. The motivation for doing this is that we will use this formulation to define a *Monte Carlo approximation* of the chance constraint in the next section. Concretely, given $\boldsymbol{x}$, we introduce a binary random variable $G_{\boldsymbol{x}} = \mathbf{1}_{g(\boldsymbol{x},\Delta)>0}$, wherein $\mathbf{1}_{(\cdot)}$ denotes the indicator function, *i.e.*, for any $\boldsymbol{\delta}$, if $\Delta = \boldsymbol{\delta}$, then $G_{\boldsymbol{x}} = 1$ if $g(\boldsymbol{x}, \boldsymbol{\delta}) > 0$; otherwise, $G_{\boldsymbol{x}} = 0$.

$G_{\boldsymbol{x}}$ is a random variable since it depends on the random uncertainty variable $\Delta$. Thus, we are interested in the probability distribution of the value of $G_{\boldsymbol{x}}$. By definition, this can be obtained from the probability distribution of $\Delta$: namely, $P[G_{\boldsymbol{x}} = 1] = P_{\boldsymbol{\delta} \sim \Delta}[g(\boldsymbol{x}, \boldsymbol{\delta}) > 0]$. Hence, the chance constraint in Eq. (1) can be rewritten as

$$P[G_{\boldsymbol{x}} = 1] \leq \eta. \tag{2}$$

*Remark 2.* If we know the probability density function $p_\Delta$ of $\Delta$, then $P[G_{\boldsymbol{x}} = 1]$ can be obtained by computing the integral

$$P[G_{\boldsymbol{x}} = 1] = \int_{\mathcal{D}} \mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta})>0} \, p_\Delta(\boldsymbol{\delta}) \, \mathrm{d}\boldsymbol{\delta}, \tag{3}$$

where the integration domain $\mathcal{D}$ consists of all realizations $\boldsymbol{\delta}$ of $\Delta$. However, computing this integral is generally intractable in practice, especially when the dimension of $\Delta$ is large.

### 3.4 Relationship between Chance Constraints and Conditional Value-at-Risk

In the following, we introduce the concept of *conditional value-at-risk* (CVaR) (Majumdar and Pavone 2020) and its relationship to chance constraints in the context of binary indicator functions. This relationship will be helpful to understand subsequent results comparing the two formulations. In contrast to formulating chance constraints, the concept of constraining the CVaR depends on the topology of constraint function $g$ with respect to the realization of the disturbance. In general, Lew et al. (2023) shows that using a chance constraint as in Eq. (1) is equivalent to constraining the *value-at-risk* (VaR) with

$$\mathrm{VaR}(g(\boldsymbol{x}, \boldsymbol{\delta})) = \inf_{\lambda \in \mathbb{R}} \{\lambda \mid P_{\boldsymbol{\delta} \sim \Delta}[g(\boldsymbol{x}, \boldsymbol{\delta}) > \lambda] \leq \eta\} \leq 0. \tag{4}$$

Furthermore, it can be shown that constraining the CVaR is strictly more conservative, *i.e.*, Eq. (4) holds if $\mathrm{CVaR} \leq 0$ (Lew et al. 2023).

In the following, we show that constraining the CVaR of a binary indicator function corresponds to a tighter chance constraint with a lower effective chance threshold. As the CVaR corresponds to the expected value of the constraint function for $g > \mathrm{VaR}$, the CVaR of a binary indicator function can be reformulated in terms of the probability of violating the constraint $g$, *i.e.*,

$$\mathrm{CVaR}(G_{\boldsymbol{x}}) = \begin{cases} \frac{P[G_{\boldsymbol{x}}=1]}{\eta}, & \text{if} P[G_{\boldsymbol{x}} = 1] < \eta, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The CVaR of the binary indicator function in Eq. (5) is in the range $[0, 1]$. Next, we may define a threshold $\mathrm{CVaR}_{\max} \in [0, 1]$ in order to construct a constraint based on the CVaR with $\mathrm{CVaR}(G_{\boldsymbol{x}}) \leq \mathrm{CVaR}_{\max}$. By using Eq. (5), it follows that constraining the CVaR yields another threshold on the probability of violating the constraint, *i.e.*,

$$P[G_{\boldsymbol{x}} = 1] \leq \eta \, \mathrm{CVaR}_{\max} \iff \mathrm{CVaR}(G_{\boldsymbol{x}}) \leq \mathrm{CVaR}_{\max}. \tag{6}$$

Note that for any $\mathrm{CVaR}_{\max} \in [0, 1)$, the resulting constraint is more conservative than the VaR constraint in Eq. (4). For $\mathrm{CVaR}_{\max} = 1$, the resulting constraint is equivalent to the VaR constraint.

## 4 Monte Carlo Approximation

Because of the challenges in computing the probability in Eq. (2) exactly (see Remark 2), a tractable approximation is required. A popular approach is the *particle-based* approximation proposed by Blackmore et al. (2010). The core concept is to draw a finite set of i.i.d. uncertainty samples, or particles, and approximate $P[G_{\boldsymbol{x}} = 1]$ as an average over the particles. This approach is justified by the law of large numbers: as the number of particles tends to infinity, the average converges to $P[G_{\boldsymbol{x}} = 1]$. In the remainder of this work, we refer to this as a *Monte Carlo approximation*.

Formally, consider a set $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$ of $N$ i.i.d. particles drawn from $\Delta$. Based on $D$, a Monte Carlo approximation of $P[G_{\boldsymbol{x}} = 1]$ can be computed as follows:

$$P[G_{\boldsymbol{x}} = 1] \approx \frac{1}{N} \underbrace{\sum_{i=1}^{N} \mathbf{1}_{g(\boldsymbol{x}, \boldsymbol{\delta}_i) > 0}}_{s_N(\boldsymbol{x}; D)} . \tag{7}$$

This is equivalent to counting the number $s_N(\boldsymbol{x}; D)$ of particles $\boldsymbol{\delta}_i$ in $\{\boldsymbol{\delta}_i\}_{i=1}^{N}$ that violate the constraint $g$ at $\boldsymbol{x}$ and dividing it by the total number of particles $N$. Note that given $\boldsymbol{x}$ and $\boldsymbol{\delta}$, determining whether $g(\boldsymbol{x}, \boldsymbol{\delta}) \leq 0$ and computing $s_N(\boldsymbol{x}; D)$ for a given solution $\boldsymbol{x}$ is generally much cheaper than computing the integral in Eq. (3).

We can now use the approximation Eq. (7) to construct a surrogate constraint of the intractable chance constraint in the optimisation problem Eq. (1). A naive approach is to simply replace $P[G_{\boldsymbol{x}} = 1]$ in Eq. (2) by its approximation, *i.e.*, require that $\frac{1}{N} s_N(\boldsymbol{x}; D) \leq \eta$, or equivalently that $s_N(\boldsymbol{x}; D) \leq \eta N$. When the number of particles approaches infinity, the feasible set of this surrogate constraint asymptotically converges to that of the original chance constraint (cf. Eq. (2)). However, when using a finite number of particles, satisfaction of the surrogate constraint does not guarantee that the original chance constraint is satisfied. The reason is that we need to account for the approximation error in Eq. (7). This can be achieved through strengthening the surrogate constraint: by requiring that

$$s_N(\boldsymbol{x}; D) \leq k_{\mathrm{thresh}} \tag{8}$$

for some $k_{\mathrm{thresh}} < \eta N$. The precise value of $k_{\mathrm{thresh}}$ depends on two parameters: *i)* the number of particles $N$, and *ii)* the level of *confidence* that we want on the *soundness* of the surrogate constraint Eq. (8). Determining suitable values for $k_{\mathrm{thresh}}$ is the main contribution of this paper.

We formulate the following surrogate optimisation problem to the original chance-constrained optimisation problem Eq. (1) as follows:

$$\begin{aligned} \min_{\boldsymbol{x} \in X} \quad & J(\boldsymbol{x}) \\ \text{s.t.} \quad & s_N(\boldsymbol{x}; D) \leq k_{\mathrm{thresh}}. \end{aligned} \tag{9}$$

The term *confidence* above refers to the probability that we sample $N$ particles $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$ for which satisfying the surrogate constraint in Eq. (8) implies satisfaction of the original chance constraint Eq. (2). Although the highest confidence of 1 (100 %) would be desirable, this in general only achievable in the limit, *i.e.*, when $N \to \infty$. Indeed, when $N$ is finite, there is in general a non-zero probability of sampling an unrepresentative set of particles, so that the true chance constraint Eq. (2) may be violated even though the surrogate constraint Eq. (8) is satisfied. However, we can leverage the confidence to establish a connection between the number of particles $N$ and the threshold $k_{\mathrm{thresh}}$ in the surrogate constraint Eq. (8) to account for the approximation error arising from the finite number of samples.

### 4.1 Monte Carlo Approximation as a Bernoulli Process

The approximation Eq. (7) of $P[G_{\boldsymbol{x}} = 1]$ can be interpreted as a Bernoulli process, *i.e.*, the act of drawing $N$ independent samples from a given binary random variable $G$. This connection allows us to derive suitable values for $k_{\mathrm{thresh}}$ as a function of $N$ and the confidence $1 - \beta$, which we will discuss in the subsequent Sections 4.2 and 4.3.

*Bernoulli process* A *Bernoulli process* is a sequence of $N$ i.i.d. binary random variables $G_1, \ldots, G_N$, *i.e.*, Bernoulli variables, that all follow the same Bernoulli distribution $G$*. Hence, every variable in the sequence is associated with a Bernoulli trial that has a binary outcome following the given Bernoulli distribution $G$. The resulting sum of the outcomes of the Bernoulli trials, *i.e.*, $S_N = \sum_{i=1}^{N} G_i$, is a random variable that follows a *Binomial distribution* (Taboga 2017), *i.e.*, for all $k = 0, \ldots, N$,

$$P[S_N = k] = \binom{N}{k} p^k (1-p)^{N-k}, \tag{10}$$

where $p = P[G = 1]$.

In the Monte Carlo approximation Eq. (7), the binary random variable $G$ is $G_{\boldsymbol{x}}$ and the corresponding Bernoulli trials are given by $\mathbf{1}_{g(\boldsymbol{x}, \boldsymbol{\delta}_i) > 0}$, $\boldsymbol{\delta}_i \sim \Delta$, for each $i = 1, \ldots, N$. Since with a fixed $\boldsymbol{x}$ and $\{\boldsymbol{\delta}_i\}_{i=1}^{N}$ being i.i.d., the trials are independent, it holds that for all $k = 0, \ldots, N$,

$$P_{\boldsymbol{\delta}_1 \sim \Delta, \ldots, \boldsymbol{\delta}_N \sim \Delta}[s_N(\boldsymbol{x}; D) = k] = \binom{N}{k} p^{\ell} (1-p)^{N-k},$$

where $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$ and $p = P[G_{\boldsymbol{x}} = 1]$.

The above yields a closed-form expression of *confidence* through the *cumulative distribution function (CDF)* $C(k; N, p)$ of the binomial distribution with parameters $N$ and $p$, defined for all $k = 0, \ldots, N$ by

$$C(k; N, p) = \sum_{\ell=0}^{k} \binom{N}{\ell} p^{\ell} (1-p)^{N-\ell}. \tag{11}$$

---

*Note that in the more general definition the sequence of a Bernoulli process can also be infinite.

## 4.2 Confidence-Bounded Surrogate Constraint

In the following, we leverage the Bernoulli formulation of the Monte Carlo approximation in Eq. (8) and the closed-form expression of the CDF in Eq. (11) to determine a threshold $k_{\text{thresh}} = k_{\text{binom}}(\beta, N, \eta)^{\dagger}$. We set this threshold such that the true chance constraint Eq. (2) is satisfied with a user-defined *confidence* $1 - \beta \in (0, 1)$ (where typically, $\beta \ll 1$). This confidence level applies to any solution $\boldsymbol{x}$ that adheres to the surrogate constraint

$$s_N(\boldsymbol{x}; D) \le k_{\text{binom}}(\beta, N, \eta). \tag{12}$$

on the sampled set of uncertainty particles $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$. We refer to Eq. (12) as a *confidence-bounded surrogate constraint* to the original chance constraint in Eq. (2). In addition, for simplicity of notation, we also define $\eta_{\text{binom}} = k_{\text{binom}}/N$.

**Proposition 1.** *Let* $\beta \in (0, 1)$, $N \in \mathbb{N}_{>0}$, $\eta \in (0, 1)$ *and let*

$$k_{\text{binom}}(\beta, N, \eta) = \max \left\{ k \in \mathbb{N} \mid C(k; N, \eta) \le \beta \right\} \tag{13}$$

*Let* $\boldsymbol{x}_{\text{reject}}$ *be a solution that violates the chance constraint in Eq. (12), i.e., such that* $P[G_{\boldsymbol{x}_{\text{reject}}} = 1] > \eta$. *It holds that*

$$P_{\boldsymbol{\delta}_1 \sim \Delta, \ldots, \boldsymbol{\delta}_N \sim \Delta}[s_N(\boldsymbol{x}_{\text{reject}}; D) > k_{\text{thresh}}] \ge 1 - \beta, \tag{14}$$

*where* $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$ *and* $k_{\text{thresh}} = k_{\text{binom}}(\beta, N, \eta)$.

The inequality in Eq. (14) is a lower bound on the probability of *correctly rejecting* a candidate solution $\boldsymbol{x}$ by means of the surrogate constraint in Eq. (12).

**Proof.** Given $\boldsymbol{x}_{\text{reject}}$ as in the proposition, we look at the probability that $s_N(\boldsymbol{x}_{\text{reject}}; D) \le k_{\text{thresh}}$, *i.e.*, the probability that we do not reject $\boldsymbol{x}_{\text{reject}}$ when using the surrogate constraint in Eq. (12). Since $G_{\boldsymbol{x}_{\text{reject}}}$ is a binary random variable with probability $p = P[G_{\boldsymbol{x}_{\text{reject}}} = 1]$ and $D = \{\boldsymbol{\delta}_i\}_{i=1}^{N}$ are independent, the sum $s_N(\boldsymbol{x}_{\text{reject}}; D)$ follows a Binomial distribution with parameters $N$ and $p$, for which the CDF is given by Eq. (11). This implies that the probability that $s_N(\boldsymbol{x}_{\text{reject}}; D) \le k_{\text{binom}}$ is equal to $C(k_{\text{binom}}; N, p)$. We build on the fact that the Binomial distribution is monotonic with respect to $p$ (Taboga 2017), *i.e.*, $p_1 < p_2$ implies $C(k; N, p_1) > C(k; N, p_2)$. Hence, it holds that $P_{\boldsymbol{\delta}_1 \sim \Delta, \ldots, \boldsymbol{\delta}_N \sim \Delta}[s_N(\boldsymbol{x}_{\text{reject}}; D) \le k_{\text{binom}}]$ is smaller than or equal to $C(k_{\text{binom}}; N, \eta)$ since $p > \eta$. Now, by definition of $k_{\text{binom}}$, it holds that $C(k_{\text{binom}}; N, \eta) \le \beta$, so that the probability of not rejecting $\boldsymbol{x}_{\text{reject}}$ is at most $\beta$.

In Fig. 2, we show the CDF of the binomial distribution for different values of $N$ and $p$. From the plots, we can obtain $k_{\text{binom}}(\beta, N, p)$ by looking at the intersection of the CDF with the horizontal line $y \equiv \beta$.

The key insight from the derivation of the above framework is that using the naive Monte Carlo approach, *i.e.*, $k_{\text{thresh}} = \eta N$, corresponds to a confidence $\beta \approx 0.5$. Indeed, in Fig. 2, we can see that $k_{\text{binom}}(0.5, N, p) \approx pN$ because the horizontal line $y \equiv 0.5$ intersects the curves roughly at $k/N = p$. The naive approach, therefore, exhibits greater presumed certainty than the confidence-bounded approach. However, this presumption often proves ill-founded, as it leads to a high probability ($\approx 50\,\%$) of accepting an infeasible candidate. This is further illustrated in Sec. 11 and Fig. 13 in the Appendix.

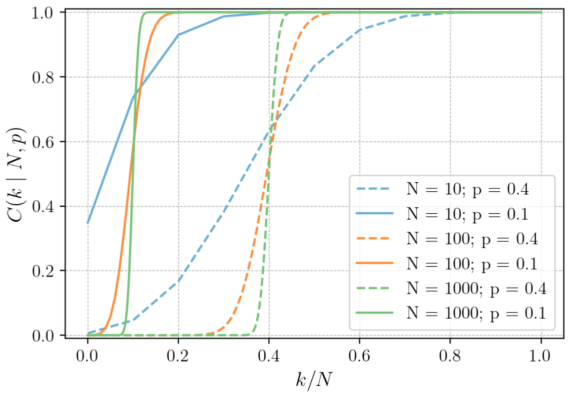

**Figure 2.** CDF of the binomial distribution for different values of $N$ and $p$, given $k/N$ on the x-axis. The CDF values map to our confidence in observing $k/N$ constraint violations under the assumption that the true probability of constraint violation is $p$.

The idea of expressing a chance constraint in terms of a sample-based variable $s_N(\boldsymbol{x}; D)$ and the inverse CDF, *i.e.*, $\text{CDF}^{-1}(\beta)$ is not new; see, *e.g.*, Heirung et al. (2018); Peña-Ordieres et al. (2020). In fact, it has been used in the past to derive theoretical bounds on the number of samples needed to ensure constraint satisfaction in scenario optimisation approaches; see, *e.g.*, Campi and Garatti (2011). Yet, to the best of our knowledge, it has never been used within the Boolean formulation of a chance constraint and its Monte Carlo approximation. Instead, it has only been used for simple continuous constraints with tractable distributions for which the CDF could be derived analytically.

## 4.3 Limitation of the Approximation

In summary, the confidence-bounded surrogate constraint Eq. (12) is based on the formulation of the Monte Carlo approximation as a Bernoulli process. However, a crucial assumption in Proposition 1 is that the Bernoulli variables $G_{\boldsymbol{x},i} = \mathbf{1}_{g(\boldsymbol{x}, \boldsymbol{\delta}_i) > 0}$ for $i = 1, \ldots, N$, are independent, such that the sum of the Bernoulli variables $s_N(\boldsymbol{x}; D)$ follows a Binomial distribution. However, when applied to the output of (9), although the samples $\{\boldsymbol{\delta}_i\}_{i=1}^{N}$ themselves are independent, the Bernoulli variables $G_{\boldsymbol{x},i}$ are generally not independent, as the solution $\boldsymbol{x}$ depends on these samples through the optimisation scheme. Hence, only if $\boldsymbol{x}$ is independent from the samples $\{\boldsymbol{\delta}_i\}_{i=1}^{N}$, does $s_N(\boldsymbol{x}; D)$ follow a binomial distribution, and Proposition 1 holds. Consequently, without additional assumptions that ensure independence, the confidence-bounded threshold $k_{\text{binom}}$ is a *heuristic*.

## 4.4 Overcoming the Limitation

In the following, we present two applications of the optimisation framework (9) for which a form of independence can be assumed to hold. In the first application, independence of the binary trials holds by construction. The second application

---

[†] When clear from the context, we will drop the arguments $\beta$, $N$ and $\eta$ in $k_{\text{binom}}(\beta, N, \eta)$.

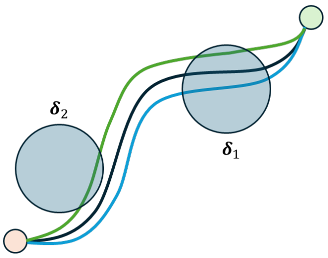

**Figure 3.** When $\boldsymbol{\delta} = \boldsymbol{\delta}_1$, the obstacle collides with all three paths, whereas when $\boldsymbol{\delta} = \boldsymbol{\delta}_2$, the obstacle collides only with the green path. Assumption 1 states that $\boldsymbol{\delta}$ takes a value for which a situation like $\boldsymbol{\delta}_2$ occurs with probability at most $\epsilon$.

can be seen as an extension of the first one for which a form independence holds under smoothness assumptions.

*4.4.1 A-posteriori validation* In the first application, we use Eq. (9) as an *a-posteriori* validation step. Given a candidate solution $\hat{\boldsymbol{x}}$ (*e.g.*, obtained by solving Eq. (9) with some sample set $\hat{D}$), we re-evaluate the constraint $g$ at $\hat{\boldsymbol{x}}$ for a new set of $N$ i.i.d. samples $D = \{\boldsymbol{\delta}_i\}_{i=1}^N$. If $s_N(\hat{\boldsymbol{x}}; D) > k_{\text{binom}}$, then we reject the candidate solution and compute a new one. This process is repeated until the *a-posteriori* validation is successful. This setting falls in the scope of Proposition 1 because $\hat{\boldsymbol{x}}$ and $D$ are independent; consequently, if $\hat{\boldsymbol{x}}$ violates the chance constraint, Proposition 1 tells us that it will be rejected with probability $1-\beta$ during the *a-posteriori* validation step. In other words, if $\hat{\boldsymbol{x}}$ passes the validation step, our confidence that it satisfies the chance constraint is of at least $1-\beta$.

*4.4.2 Receding-horizon optimisation* In the second application, we use Eq. (9) in a receding-horizon optimisation scheme, such as Model Predictive Control (MPC). In this setting, at each MPC step $m = 1, \ldots, M$, we compute a new solution $\boldsymbol{x}_m$ with a new sample set $D = \{\boldsymbol{\delta}_{m,i}\}_{i=1}^N$. Each solution is executed for a few milliseconds, until we re-sample and compute a new solution in the next MPC step. In general, the solution $\boldsymbol{x}_m$ computed at step $m$ is *not very different* from the solution $\boldsymbol{x}_{m-1}$ computed at step $m-1$ (if needed, this can also be enforced as an explicit constraint of the MPC). In this case, provided $g$ varies smoothly with $\boldsymbol{x}$, we can make the assumption that the binary trials $\mathbf{1}_{g(\boldsymbol{x}_m, \boldsymbol{\delta}_{m,i})>0}$ are independent because $\{\boldsymbol{\delta}_{m,i}\}_{i=1}^N$ is i.i.d. and $\boldsymbol{x}_m \approx \boldsymbol{x}_{m-1}$. We formalize this with an assumption and a proposition:

**Assumption 1.** There is $\epsilon > 0$ such that with probability $1-\epsilon$ on $\boldsymbol{\delta}$, if there is $\boldsymbol{x} \in X$ such that $g(\boldsymbol{x}, \boldsymbol{\delta}) > 0$, then for all $\boldsymbol{x} \in X$, it holds that $g(\boldsymbol{x}, \boldsymbol{\delta}) > 0$.

See Figure 3 for an illustration of Assumption 1.

**Proposition 2.** *Under Assumption 1, it holds that*

$$P_{\boldsymbol{\delta}_1 \sim \Delta, \ldots, \boldsymbol{\delta}_N \sim \Delta}[P[G_{\boldsymbol{x}} = 1] \leq \eta + \epsilon] \geq 1 - \beta, \tag{15}$$

*where $\boldsymbol{x}$ is the solution of the surrogate optimisation problem in Eq. (9) with $D = \{\boldsymbol{\delta}_i\}_{i=1}^N$ and $k_{\text{thresh}} = k_{\text{binom}}(\beta, N, \eta)$.*

**Proof.** Let $\boldsymbol{x}$ be the solution of Eq. (9) with $D = \{\boldsymbol{\delta}_i\}_{i=1}^N$. Assume that $P[G_{\boldsymbol{x}} = 1] > \eta + \epsilon$. By Assumption 1, this implies that $P_{\boldsymbol{\delta} \sim \Delta}[\min_{\boldsymbol{x}' \in X} g(\boldsymbol{x}', \boldsymbol{\delta}) > 0] > \eta$. Furthermore,

$$s_N^{\min}(D) \triangleq \sum_{i=1}^{N} \min_{\boldsymbol{x}' \in X} \mathbf{1}_{g(\boldsymbol{x}', \boldsymbol{\delta}_i)>0} \leq s_N(\boldsymbol{x}; D) \leq k_{\text{binom}}.$$

Since the variables $\min_{\boldsymbol{x}' \in X} \mathbf{1}_{g(\boldsymbol{x}', \boldsymbol{\delta}_i)>0}$ for $i = 1, \ldots, N$, are i.i.d., it follows that $s_N^{\min}(D)$ is a Bernoulli process with $N$ trials and $p = P_{\boldsymbol{\delta} \sim \Delta}[\min_{\boldsymbol{x}' \in X} g(\boldsymbol{x}', \boldsymbol{\delta}_i) > 0]$. Hence, the probability that $s_N^{\min}(D) \leq k_{\text{binom}}$ is equal to $C(k_{\text{binom}}; N, p)$. The rest follows in the same way as in the proof of Proposition 1.

*Remark 3.* In a receding-horizon optimisation scheme, under the assumption that the solution does not vary too much from one step to the next one, the samples used at the previous steps also provide indication that the solution at the current step is valid. This information is *not* used in the confidence bound that we can derive from Proposition 2, since it is for a single MPC-step. This is why in practice the real value of $P[G_{\boldsymbol{x}_m} = 1]$ is often smaller than $\eta$, as we will show in the experiments (see Sec. 6). In future work, we plan to use and quantify this information to obtain even less conservative bounds on $P[G_{\boldsymbol{x}_m} = 1]$.

## 5 Stochastic Trajectory Optimisation with Chance Constraints

Due to the non-smooth nature of the uncertainty dynamics and the resulting non-smooth surrogate chance constraint with respect to the optimization variable in Eq.(12), we approach the optimization problem with a gradient-free, i.e. zero-order, evolutionary optimization technique. Building upon our previous work *Via-Point-Based Stochastic Trajectory Optimisation (VP-STO)* (Jankowski et al. 2023), we introduce *chance-constrained VP-STO (CC-VPSTO)* for finding robot trajectories that minimise a given task-related objective *while satisfying a given chance constraint.*

### 5.1 Preliminaries on VP-STO

VP-STO builds on stochastic optimisation in order to find robot trajectories that minimise a given task-related objective in dynamic environments.

*Trajectory Representation* In VP-STO the decision variable $\boldsymbol{x}$ for an optimisation problem, such as the one in Eq. (1), is a set of $S$ via-points $\boldsymbol{q}_{\text{via}} = (q_{\text{via},1}, \ldots, q_{\text{via},s})$, *i.e.*, $\boldsymbol{x} = \boldsymbol{q}_{\text{via}}$. For a given set of via-points, VP-STO synthesises a time-continuous and smooth trajectory that satisfies the boundary conditions, such as initial and final state and velocity, and kinodynamic constraints, such as velocity and acceleration limits[‡]. The advantage of the approach lies in the low-dimensional representation of the trajectory, which allows for efficient optimisation in a low-dimensional space.

---

[‡] For more information on how we generate the continuous trajectories from via-points, we refer the reader to Sec. 9 in the Appendix and to (Jankowski et al. 2023, 2022).

*Optimisation Algorithm* VP-STO uses the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen 2016) to find the optimal set of via-points that map to a trajectory that minimises the given objective function. CMA-ES iteratively updates the mean and covariance of a Gaussian distribution that represents the search space of the optimisation problem, *i.e.*, the set of via-points. In each iteration $j$, the algorithm samples candidate solutions from this distribution, *i.e.*, $\mathcal{N}\left({}^{j}\boldsymbol{\mu}_{\text{via}}, {}^{j}\boldsymbol{\Sigma}_{\text{via}}\right)$, evaluates them on the given objective function, and updates the distribution based on the evaluation results. The algorithm converges to the optimal solution in a few iterations, making it suitable for real-time applications.

### 5.2 Chance-Constrained VP-STO

VP-STO has been shown to be effective in generating robot trajectories in real-time for dynamic environments, outperforming state-of-the-art sampling-based MPC methods (in applications without non-holonomic constraints), which is based on the well-known algorithm MPPI (Williams et al. 2017). Yet, in it's original form, VP-STO does not consider uncertainty, but instead assumes a deterministic environment. In this work, we extend the VP-STO framework to consider uncertainty in the environment, *i.e.*, we introduce the *chance-constrained VP-STO (CC-VPSTO)* framework.

CC-VPSTO optimises the optimisation problem in (9) using the surrogate constraint in Eq. (12). We enforce the constraint through a penalty-based approach, *i.e.*, we include the constraint $k \leq k_{\text{thresh}}$ in the objective function as a penalty term. This can be seen as a discontinuous barrier function that adds a very high penalty term $J_{\text{pen}}$ to the objective function if the constraint is violated, *i.e.*, when the observed number of constraint violations $k > k_{\text{thresh}}$. The closed-form formulation of this penalty term is as follows:

$$J_{\text{pen}} = \mathbf{1}[k > k_{\text{thresh}}] \cdot (J_{\text{pen,min}} + a \cdot (k - k_{\text{thresh}} - 1)).$$

We choose the minimum penalty term $J_{\text{pen,min}}$ to be much larger than the maximum cost objective without constraint violations. Moreover, we add a piecewise linear term to the minimum penalty term that grows linearly with the extra number of violations compared to $k_{\text{thresh}}$. This term makes the constraint landscape smoother and gives the optimiser a direction towards feasible solutions without violations. The overall algorithm for CC-VPSTO is summarised in Alg. 1, where the approximation of the chance constraint, *i.e.*, counting the number of particles that cause the solution to violate the constraint, is encapsulated in the `evaluate` function.

In our previous work, we demonstrated the suitability of the VP-STO framework for real-time robot motion planning in dynamic environments (Jankowski et al. 2023). Similarly, the CC-VPSTO framework, *i.e.*, the above algorithm, can be used in a receding horizon MPC scheme to generate robot trajectories in real-time. Yet, we note that the constraint evaluation in the `evaluate` function will be computationally more expensive than in the original VP-STO framework, as it requires $N$ Monte Carlo simulations per candidate trajectory $\boldsymbol{\xi}$. This implies that the reactivity of our framework now depends on the number of particles $N$ used for the approximation. The advantage of our approximation is that it allows us to choose the number of samples and then use confidence levels to determine an appropriate threshold. In contrast, a scenario-optimisation approach is typically tied to a large number of samples, derived from conservative bounds. Instead, the number of samples can be selected based on the target execution frequency of the MPC scheme.

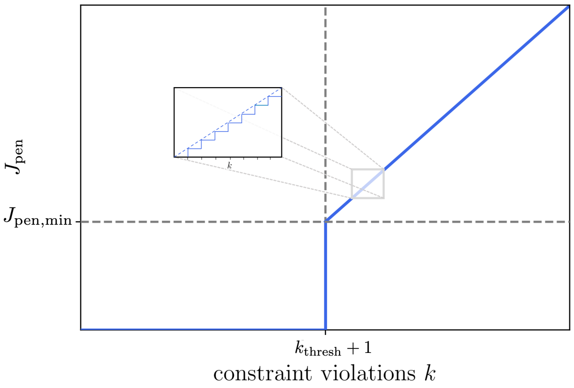

**Figure 4.** Graph of the penalty function used in CC-VPSTO. When observing more than $k_{\text{thresh}}$ constraint violations in the $N$ Monte Carlo simulations, the penalty function takes value $J_{\text{pen,min}}$ plus a quantity proportional to the number of extra constraint violations compared to $k_{\text{thresh}}$. Note that we chose the minimum penalty term $J_{\text{pen,min}}$ to be much larger than the largest cost objective without constraint violations.

**Algorithm 1:** CC-VPSTO

```
Input: $\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T, \dot{\boldsymbol{q}}_{\text{min}}, \dot{\boldsymbol{q}}_{\text{max}}, \ddot{\boldsymbol{q}}_{\text{min}}, \ddot{\boldsymbol{q}}_{\text{max}}, N_{via},$
    $maxIter, S, H, \eta, \beta, N$
/* $N_{via}$: no. of via-points,                      */
/* $maxIter$: max. no. of CMA-ES iterations,          */
/* $S$: no. of sampled candidate trajs.               */
/* $H$: horizon                                       */
/* $\eta$: chance constraint threshold                */
/* $\beta$: confidence threshold                      */
/* $N$: no. of particles                              */
Output: Robot trajectory $\boldsymbol{\xi}^*_{0:H}$
${}^{0}\boldsymbol{\mu}_{\text{via}}, {}^{0}\boldsymbol{\Sigma}_{\text{via}} \leftarrow \text{init}(N_{via})$
$j \leftarrow 0$
Sample $\Delta \leftarrow \{\boldsymbol{\delta}_i \sim p_\Delta\}_{i=1}^{N}$
$k_\beta \leftarrow C(k \mid N, \eta)^{-1}(\beta)$
while $j < maxIter$ do
    $\{\boldsymbol{q}_{\text{via}}\}_{s=1}^{S} \leftarrow \text{sample}\left({}^{j}\boldsymbol{\mu}_{\text{via}}, {}^{j}\boldsymbol{\Sigma}_{\text{via}}\right)$    // via-points
    $\{\boldsymbol{\xi}\}_{s=1}^{S} \leftarrow \text{synthesise}\left(\{\boldsymbol{q}_{\text{via}}\}_{s=1}^{S}\right)$    // trajectories
    $\{c\}_{s=1}^{S} \leftarrow \text{evaluate}\left(\{\boldsymbol{\xi}\}_{s=1}^{S}, k_\beta, \Delta\right)$    // cost
    $\boldsymbol{\mu}_{\text{via}}^{j+1}, \boldsymbol{\Sigma}_{\text{via}}^{j+1} \leftarrow \text{CMA-ES}\left(\{\boldsymbol{q}_{\text{via}}, c\}_{s=1}^{S}\right)$
    $j \leftarrow j + 1$
end
$\boldsymbol{\xi}^*_{0:H} \leftarrow \text{synthesise}\left(\boldsymbol{\mu}_{\text{via}}^{j}\right)$
```

## 6 Experiments

We evaluate our framework, *i.e.*, Algorithm 1, with and without the MPC-scheme, in simulations and a real-world experiment with a Franka Emika robot arm. The simulation experiments allow us to make claims about the empirical performance of our system across different settings and parameterisations. The robot experiment allows us to evaluate the real-time applicability of our approach. The supplementary video includes videos from both, simulated and real experiments. These can also be found on our website https://sites.google.com/oxfordrobotics.institute/cc-vpsto.

## 6.1 Experimental Setup

*6.1.1 Joint probability of constraint violation* In all our experiments the chance constraint is formulated on the collision probability with obstacles in the robot's environment. We encode this as a *joint* chance constraint, *i.e.*, enforcing *trajectory-wise* constraint satisfaction with high probability (cf. Sec. 2 for more detail). A joint formulation is the more meaningful interpretation for robot behaviour, in contrast to evaluating the safety constraint independently at each time step. This means that we would not consider a trajectory to be safe if it avoids collisions in one time step with a very high probability, but collides in the next time step. We thus consider correlation over time in the chance constraint, *i.e.*, the first collision in a trajectory renders the whole trajectory unsafe and all subsequent collisions do not add any additional risk (Lew et al. 2023).

*6.1.2 Collisions* In the case of a single obstacle, we consider a robot trajectory to be in collision if the robot would collide with the obstacle at *any point in time*. For multiple obstacles, we extend this definition to say a robot trajectory is in collision if the robot would collide with *any* obstacle *at any point in time*. By this, we avoid double counting collisions. One particle can only be counted as one collision, even in cases where it might collide with several obstacles at different points in time.

*6.1.3 Particles* Before we outline our experiments in more detail, we want to clarify our use of particles for *i)* our algorithm and *ii)* its evaluation. As outlined above, in our algorithm, we use particles to approximate the chance constraint in the optimisation scheme. Moreover, we also use particles to approximate the distribution of test environments the planned trajectory could encounter in the experiments. For both cases, the algorithm and the evaluation, the definition of a particle is the same: One particle always maps to one particular belief state of how the specific environment is going to evolve or looks like. In the case of one static uncertain obstacle, a particle would therefore directly map to the obstacle's position. While in the case of one dynamically moving obstacle, one particle would map to a trajectory, *i.e.*, the prediction of where that obstacle is going to be in the next time steps. And finally, in the case of multiple dynamic obstacles, one particle maps to $M$ trajectory predictions for $M$ obstacles. Thus, the general approach taken in each of the experiments is to first draw a set of particles that we use within the optimisation scheme to generate a solution trajectory and then, second, to draw another set of particles from the same distribution, to evaluate the given solution.

## 6.2 Simulation Experiments

All simulation experiments are conducted in a bounded 2D environment with a circular holonomic robot, as shown in Fig. 6. In Sec. 6.2.1 we perform offline planning experiments, where we run CC-VPSTO only once to compute a trajectory over the full horizon from start to goal. In Sec. 6.2.3 we evaluate the online planning case, where we follow a receding horizon approach using CC-VPSTO to re-plan the trajectory at every MPC step. The results of the offline experiments will also be relevant for the online setting, as each online replanning step can be seen as solving a new offline optimisation problem. In both experiment settings, the uncertainty stems from obstacles in the environment, with no uncertainty in the robot dynamics[§]. For the offline planning case, we assume that the obstacles are static with uncertain positions, which reflects measurement noise. For the online receding horizon setting, we assume the obstacles are moving according to a random walk model. In addition, their velocities are inverted when they hit the boundaries, keeping them inside the workspace and rendering their dynamics non-linear. In all the experiments, the obstacles are assumed to be circular with different radii, but they could be replaced with more complex shapes, as we do not require any kind of convexity in our optimisation scheme.

*6.2.1 Offline Planning (Gaussian Uncertainty)* In order to show the properties of CC-VPSTO, we use a simple offline setting with a single static obstacle. The obstacle's uncertain position follows a Gaussian distribution, as shown in Fig. 6 (see Sec. 6.2.2 for results on a non-Gaussian distribution). Therefore in this offline setting a particle, as explained in more detail in Sec. 6.1, refers to a possible (static) position of the obstacle.

For every combination of values of $N$ (100, 1000) and $\eta$ (0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.6, 0.8), we run $N_{\text{exp}} = 10^5$ experiments. For $i = 0, \dots, N_{\text{exp}}$ sample sets of $N$ i.i.d. particles, we compute the trajectory $\xi_i$ using CC-VPSTO with $k_\beta(\beta, \eta, N)$. Then, for each trajectory $\xi_i$, we evaluate its probability $\hat{\eta}_i$ of colliding with the static uncertain obstacle. We use a new set of $N_{\text{eval}} = 10^4$ i.i.d. particles, sampled from the same distribution of possible obstacle locations, and count the number of particles that collide with $\xi_i$. The ratio of this number by $N_{\text{eval}}$ is $\hat{\eta}_i$. We use $\hat{\eta}_i$ to compute the following three metrics:

1. ***Mean collision probability***:
   $\hat{\eta}_{\text{avg}} = \sum_{i=0}^{N_{\text{exp}}} \hat{\eta}_i / N_{\text{exp}}$
2. ***(1-$\beta$)-percentile of the collision probability***:
   $\hat{\eta}_{(1-\beta)} = \text{percentile}(\{\hat{\eta}_i\}_{i=0}^{N_{\text{exp}}}, 1-\beta)$
3. ***Probability of chance constraint violation***:
   $P(\hat{\eta}_i > \eta) = \hat{\beta} = \left(\sum_{i=0}^{N_{\text{exp}}} \mathbf{1}(\hat{\eta}_i > \eta)\right) / N_{\text{exp}}$

Note, that we denote *empirical* values with a hat, *e.g.*, $\hat{\eta}$. For the proposed heuristic bound to be a good approximation, a share of maximum $\beta$ of the solutions can be in collision. This is because we set our confidence threshold to $1-\beta$, thereby permitting the acceptance of "poor" approximations in a fraction of $\beta$ instances. A trajectory $\xi_i$ is deemed to be in collision, *i.e.*, violate the chance constraint, if its estimated value $\hat{\eta}_i$ exceeds $\eta$.

---

[§] This is for simplicity only. The extension to process noise and external disturbances is straightforward given the proposed approach.

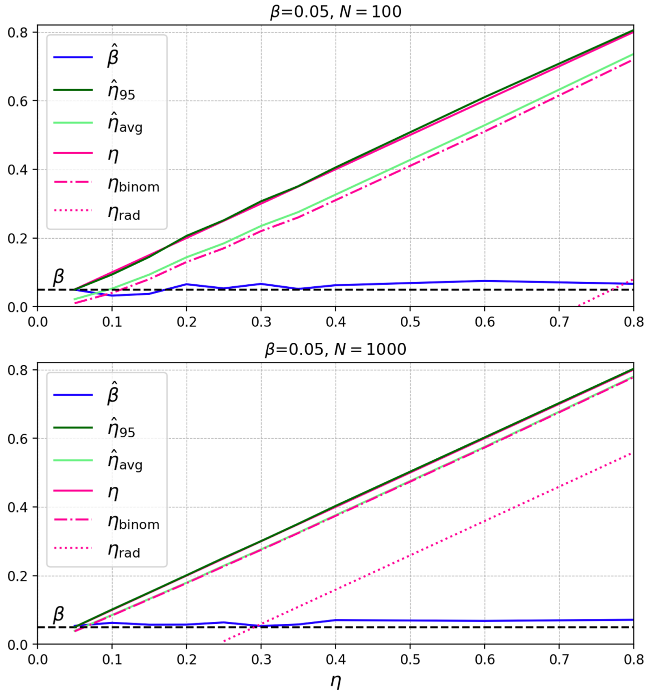

**Figure 5.** ***Offline Planning Experiment.*** We evaluate our bound $\eta_{\text{binom}}$ for different values of $\eta$ and numbers of particles $N$ (100, 1000) by running CC-VPSTO $N_{\text{exp}} = 10^5$ times and evaluating the solutions on a set of $N_{\text{eval}} = 10^4$ new unseen samples. We report the *mean collision probability* $\hat{\eta}_{\text{avg}}$, the $(1-\beta)$*-percentile* of the collision probabilities (*i.e.*, $\hat{\eta}_{(1-\beta)}$ across experiments), and the *empirical probability of chance constraint violation* $\hat{\beta}$ (*i.e.*, the share of experiments that had a collision probability $\hat{\eta}_i > \eta$). Theoretical values of $\eta_{\text{binom}}$ and the Rademacher-based baseline $\eta_{\text{rad}}$ are shown for comparison.

*Baseline* In the course of this work, we developed an alternative approach to approximate the chance constraint in Eq. (1) based on the *Rademacher complexity* from statistical learning theory (Shalev-Shwartz and Ben-David 2014; Mohri et al. 2018). Computing a suitable $k_{\text{thresh}}$ for the surrogate optimisation problem in Eq. (9) can be approached by computing an upper bound on the Rademacher complexity of the associated set of functions. However, despite the theoretical attractivity of the this approach, computing an upper bound on the Rademacher complexity can be very challenging in general, and there is usually no closed-from expression for such bounds. Yet, we found a tight bound $k_{\text{rad}}$ for a special case of collision-avoidance problem. This bound is more theoretically complete, as it does not require the independence of the Bernoulli variables, but we also note that it is more conservative, computationally expensive, and, less general since it is limited to a specific motion planning problem. Consequently, we use this bound as a baseline for our simulation experiments. The full derivation, as well as the closed-form expression for $k_{\text{rad}}$, and the necessary proofs and assumptions, can be found in Appendix 10.

*Results* The numerical results of our offline analysis are summarised in Fig. 5. The pink dotted curves show the theoretical values of $\eta_{\text{binom}}$ and $\eta_{\text{rad}}$, and the green curves show the empirical values of $\hat{\eta}_{\text{avg}}$ and $\hat{\eta}_{(1-\beta)}$. In addition, we plot the empirical probability of chance constraint violation $\hat{\beta}$ in blue against the user-defined value of $\beta$. Note that we also provide the exact numerical results from Fig. 5 in Tab. 2 in the Appendix. We observe that our proposed bound $\eta_{\text{binom}}$ provides a sufficient value for $k_{\text{thresh}}$ since the observed $\hat{\eta}_{0.95}$ is close to the target $\eta$ (or equivalently, $\hat{\beta}$ is close to $\beta$). This means that empirically the probability of collision is below $\eta$, with confidence 95 %. We also observe that when $N$ is larger, $\eta_{\text{binom}}$ and $\hat{\eta}_{\text{avg}}$ are closer to $\eta$, implying that the surrogate optimisation problem becomes less conservative as the number of particles increases, given the same user-defined confidence-level. This is expected since more particles provide a better approximation of the distribution. Last, the figure also shows that the baseline bound $\eta_{\text{rad}}$ is much more conservative, as it is significantly smaller than $\eta_{\text{binom}}$. Moreover, the offset from $\eta_{\text{binom}}$ increases substantially when decreasing the number of particles $N$. In addition to the quantitative results, we visualize the mean trajectories for the two local optima found by CC-VPSTO for different values of $\eta$ in Fig. 6. Note, that we only show the solutions for the experiments with $N = 100$ in the optimisation, as the solutions for $N = 1000$ are visually indistinguishable. In the legend of Fig. 6, we also show the average motion duration of the trajectories across experiments for the different values of $\eta$. This qualitative analysis shows that with higher values of $\eta$, CC-VPSTO finds more efficient, but also less “padded” trajectories, as the mean trajectories are closer to the obstacle, since they allow for a higher probability of collision.

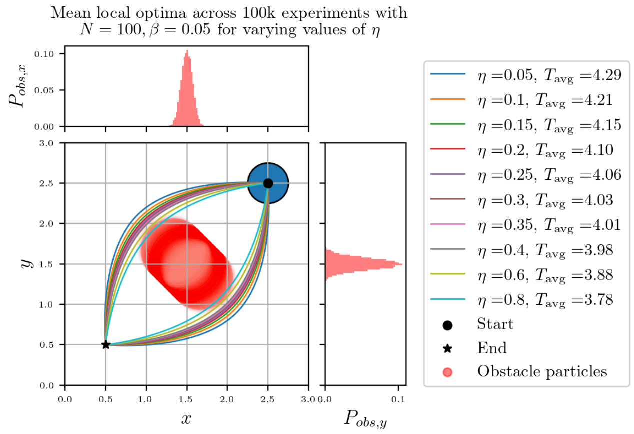

**Figure 6.** ***Offline Planning Experiment (Gaussian Uncertainty).*** We show $N_{\text{eval}} = 10^4$ red circles for the uncertain obstacle position and the mean trajectory for the two local optima from CC-VPSTO, which used $N = 100$ particles in the optimisation, for varying values of $\eta$ across $N_{\text{exp}} = 10^5$ experiments. The blue circle shows the robot's radius and starting position.

*6.2.2 Offline Planning (Multimodal Uncertainty)* CC-VPSTO does not make any assumptions about the uncertainty distribution and is able to handle arbitrary distributions. To illustrate this, we provide an additional offline motion planning experiment where we replace the Gaussian distribution over the obstacle position in the previous experiment from Sec. 6.2.1 with a Gaussian mixture distribution with three modes. Fig. 7 illustrates the qualitative results for this scenario with a multimodal distribution. It shows that CC-VPSTO is able to find a timing-optimal trajectory given the non-Gaussian uncertainty over the obstacle position depending on the user-defined chance

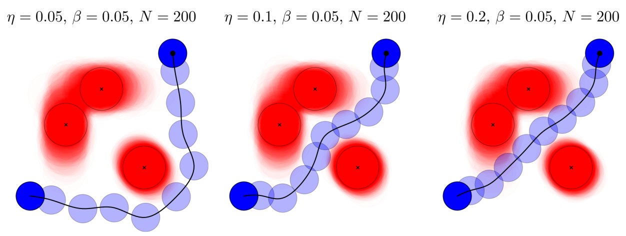

**Figure 7.** ***Offline Planning Experiment (Multimodal Uncertainty).*** We show $N_{\text{eval}} = 10^4$ red circles for the uncertain obstacle position. The black crosses indicate the means of the three Gaussian modes. The black trajectory and the blue circles illustrate the optimal trajectory computed with CC-VPSTO, which used $N = 200$ particles in the optimisation, for varying values of $\eta$.

**Table 1.** *Results of the Offline Planning Experiment (Multimodal Uncertainty).*

| | $\eta = 0.05$ | $\eta = 0.1$ | $\eta = 0.2$ |
|---|---|---|---|
| $\hat{\eta}_{\text{avg}}$ | 0.026 | 0.086 | 0.164 |
| $\hat{\beta}\,(\beta = 0.05)$ | 0.026 | 0.228 | 0.069 |
| $T_{\text{avg}}$ | 4.702 | 3.164 | 2.752 |

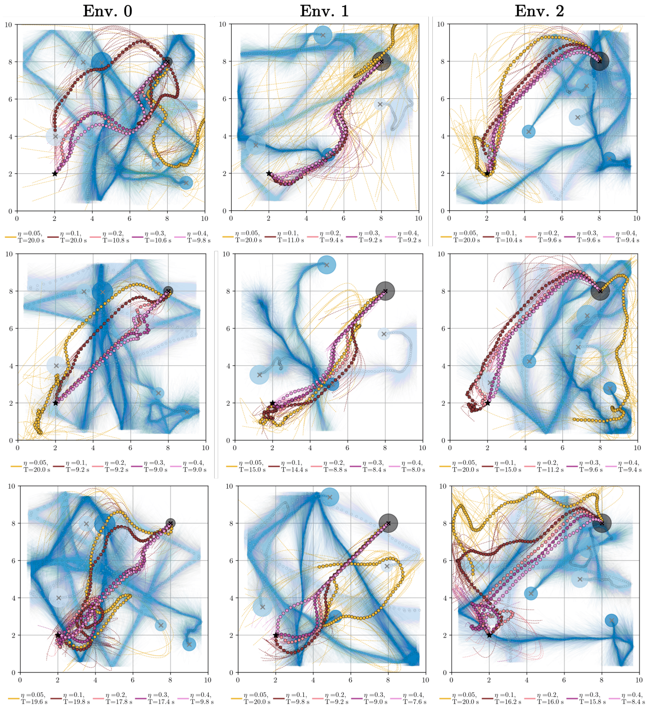

**Figure 8.** ***Overview of the environments used for the MPC experiments.*** In each column, one per environment configuration, we show three examples of how the obstacle rollouts can look like, given the same environment configuration. The initial obstacle positions and their radii are shown by the blue circles. The smaller circles along the trajectories indicate the ground truth MPC updates, whilst the opaque trajectories plotted from each smaller circle show the particle predictions at that MPC-step. In addition, we plot the solutions of the online-CC-VPSTO controller for varying $\eta$-values. The start and goal is shown by the dark grey circle, which also reflects the robot radius, and the star, respectively. In addition, we plot the current solution for that MPC step, *i.e.*, the trajectory segment from the current robot position planned over a receding horizon, shown as dashed lines.

threshold $\eta$ and confidence threshold $1 - \beta$. Yet, note that a more complex distribution as the one given by the Gaussian mixture distribution by default requires more samples in the approximation, which is why we used $N = 200$ samples in this experiment. Table 1 provides numerical results on the average probability of collision of the solution ($\hat{\eta}_{\text{avg}}$), the empiric estimation of $\beta$, and the average duration of the solution trajectory $T$ (i.e. the optimisation objective). These statistical results are computed from 1000 experiments performed for each $\eta$ and evaluated on $10^4$ new samples from the Gaussian mixture distribution. We observe that on average the chance constraint is satisfied. A higher chance threshold, *i.e.*, allowing a higher probability of colliding with the obstacle, results in lower trajectory durations.

*6.2.3 Online Planning (MPC)* The online planning experiments correspond to a receding horizon/model predictive control (MPC) approach, where a new robot trajectory is planned at every MPC step $t_{i,\text{MPC}} = t_{i-1,\text{MPC}} + \Delta_{\text{MPC}}$ with $1/\Delta_{\text{MPC}}$ being the run frequency of the MPC controller. At each MPC step, the robot gets a position update of the $M$ obstacles in the environment, which is assumed to be exact, *i.e.*, no measurement uncertainty. As in the offline planning experiment, we assume that CC-VPSTO has access to an accurate generative model that generates predictions of the future obstacle motions. In our online our online experiments we use a random walk model, parameterised to match the simulation environment. In a real-world setting, this could be replaced by a generative model learned from real-world data, *e.g.*, a model similar to Jiang et al. (2023). In our optimisation scheme, given a position update, new obstacle trajectories are sampled from the random walk model and rolled out for a fixed time horizon $T > \Delta_{\text{MPC}}$. As described in Sec. 6.1, one particle in this experiment corresponds to one belief state of how the obstacles are going to evolve in the next time steps, *i.e.*, one particle maps to $M$ trajectory predictions for $M$ obstacles. Hence, when using $N$ particles, we generate $M \cdot N$ obstacle trajectories, *i.e.*, the MC simulations, of duration $T$. At each MPC-step, after sampling these new trajectory predictions, we use them to re-run CC-VPSTO.

We evaluate online CC-VPSTO on four metrics across three environment configurations with four or five obstacles. Each obstacle is initialised with varying start position, velocity, and acceleration variance in the random walk model. The trajectory we evaluate is the trajectory that the robot executes, *i.e.*, the concatenation of the first $\Delta_{\text{MPC}}$ time steps of each of the solutions across MPC steps. One single experiment produces one trajectory from the initial position to the goal for the given environment instantiation and $\eta$-value. Fig. 8 shows three example rollouts and the respective CC-VPSTO solutions for different values of $\eta$ for each of them. Note, that the length of the plotted obstacle rollouts was not fixed across the three examples, but depended on the maximum duration of the generated solutions for the given example. All solutions depicted are collision-free. Additional details about the configurations can be found in the appendix in Sec. 12.1. For each environment configuration and for different values of $\eta$ (0.05, 0.2, 0.4, 0.6, 0.8), we run 1000 experiments.

The evaluation metrics for the experiments are:

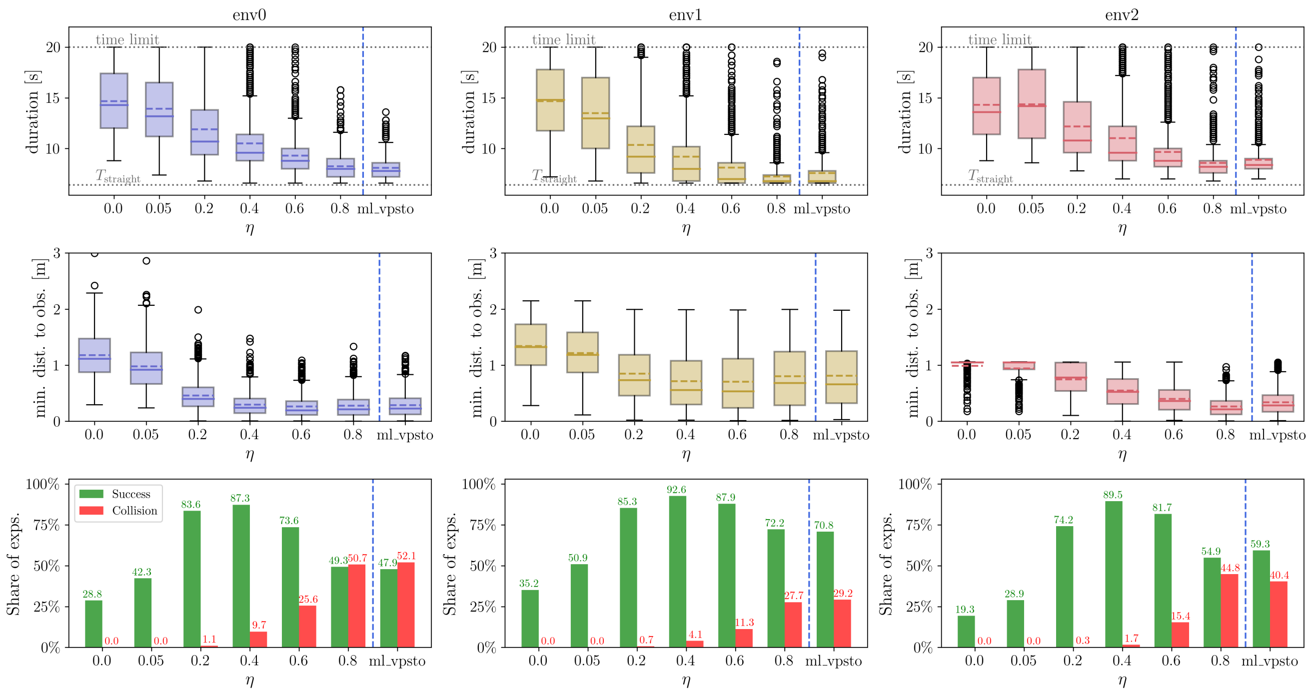

**Figure 9.** ***Simulation: MPC experiments.*** Evaluating *motion duration*, *success rate*, *collision rate* and the *minimum distance to obstacles* across 1000 experiments on 3 different environments. One experiment corresponds to running online-CC-VPSTO until reaching the goal or until a maximum number of 100 MPC steps is reached. Goal and start location remain fixed across all experiments and environments, whilst the obstacle trajectories vary across experiments and environments. Each environment is initialized with different start positions and velocities of the obstacles, as well as different variance on the acceleration used in the random walk model. The boxplots include the mean (dashed line) and median (solid line) across all experiments.

1. ***Motion duration (time to until goal reached):*** This translates to the number of MPC-steps needed until the robot reaches the goal. In the experiments, we set a maximum number of 100 MPC steps. We only report the duration for successful experiments.

2. ***Success rate:*** The fraction of experiments, where the generated trajectory reaches the goal within the maximum number of MPC steps. An experiment is further only considered to be successful if the executed robot trajectory does not collide at any point of time with any of the obstacles *and* if the goal is reached.

3. ***Collision rate:*** This rate reflects the share of experiments where the respective trajectories were in collision at least once with any of the obstacles across the entire motion.

4. ***Minimum distance to obstacles:*** Per experiment, we measure the closest distance of the robot to any of the obstacles across the entire motion. This metric is only reported for successful experiments.

Note, that these metrics are different from the metrics used in the offline evaluation. This is because we aim to show the properties of the MPC trajectory given the guarantees from the offline experiments (which corresponds to a single MPC step in the online case).

*Baselines* We compare the use of our confidence-based bound $k_{\text{binom}}$ as a surrogate for the chance constraint (cf. Eq. (8) and Eq. (12)) against two alternative approximations: *i)* the MC approximation of the original chance constraint, and *ii)* the CVaR-based formulation described in Sec. 3.4. As the main contribution of this work lies in the derivation a new confidence-bounded sample average approximation of a chance constraint, we do not compare against other methods of solving the resulting optimisation problem, such as MPPI. For a more thorough comparison of VP-STO against MPPI, please reference our previous work (Jankowski et al. 2023). In addition, we compare our approach to a baseline, that we abbreviate with "ML-VPSTO", where ML stands for maximum likelihood. Instead of computing the probability of constraint violation based on particles, ML-VPSTO uses the same particles to compute mean obstacle trajectories and uses standard VPSTO to generate a solution that avoids these trajectories. In addition, running CC-VPSTO with $\eta = 0$ can also be seen as a baseline, as this is comparable to using a hard collision avoidance constraint within VPSTO. For all MPC simulation experiments, we assumed a replanning frequency of 4 Hz with a time step of 0.05 seconds, while setting the planning horizon $T_{\text{MPC}}$ to 5 seconds (mapping to 100 time steps for the rollouts), the maximum number of MPC iterations to 100 and the number of particles $N$ to 100.

*Results* The results of the online experiment are a key insight of this paper, as they demonstrate the effects of combining *reactivity* (the MPC setting) with *safety* (the chance constraints). Fig. 9 summarises the results across the 1000 experiments for each of the 3 different environment configurations. We observe that CC-VPSTO in an MPC-loop is able to generate trajectories that are entirely collision-free for eta-values of up to 5%. In the given experimental setting, ML-VPSTO is approximately equivalent to permitting collisions with a probability as high as 80% in CC-VPSTO. In the most challenging environment configuration (`env0`) both approaches lead to a situation where 50% of the experiments are in collision. This indicates that employing average obstacle prediction for collision avoidance is inadequate, as a 50% collision rate is generally not an acceptable outcome in the majority of robotic applications. Moreover, the dependency of the safety/performance trade-off on the value of $\eta$ is reflected in the motion duration. The higher the value of $\eta$, the shorter the duration of the trajectory. While ML-VPSTO produces the quickest trajectories, it is also the least safe approach. For reference, we also include $T_{\text{straight}}$ in the duration plots, which is the duration of the straight-line trajectory from start to goal (ignoring obstacles). When looking at the minimum distance to obstacles across trajectories and experiments, it seems like the expressiveness of this metric depends on the environment configuration. For the first and second environment configuration, there is a decreasing trend, until it plateaus at values of $\eta > 0.4$ for the first environment. For the second environment configuration, after a downward trend, the minimum distance to obstacles increases again for values of $\eta > 0.4$. This can be explained by CC-VPSTO being more risk-taking and probably choosing a more direct path to the goal, which can possibly lead to more collisions, but in the case of no collision, the distance to obstacles might actually be bigger, as the motion is also quicker and some obstacles might not have had time to move closer to the robot. Last, the small variance in the distances for small $\eta$-values in the third environment can be explained by the initial configuration of obstacles, as they are already very close to the robot, which is then probably already the minimum distance across the entire motion. Overall, for this experiment setting, it seems like CC-VPSTO with $\eta = 0.4$ offers a good trade-off between safety and performance, as it is able to generate safe trajectories with a high success rate, whilst also being able to generate trajectories that are efficient in their motion duration.

Last, we evaluate the effect of the number of particles used in the MC approximation on the success and collision rates depending on the surrogate constraint that is used in CC-VPSTO. We do this across 1000 MPC experiments for each particle count and environment configuration. We evaluate three different surrogate constraints:

1. *Confidence:* $s_N(\boldsymbol{x}, D) \leq \eta_{\text{binom}}(\eta, \beta)$
2. *Value at Risk (VaR):* $s_N(\boldsymbol{x}, D) \leq \eta$
3. *Conditional VaR (CVaR):* $s_N(\boldsymbol{x}, D) \leq \text{CVaR}_{\max}$, with $\text{CVaR}_{\max}$ set to 0.6.

For all experiments we fixed $\eta$ to 0.2 and the confidence threshold to 0.99. The results are shown in Fig. 10. The numbers are averaged across three different environment configurations. For a 100 particles the experiments are equivalent to the results shown in Fig. 9. With more particles, the collision rate converges to zero due to the MPC setting. The results show that the confidence-bounded approximation is the only approximation that is able to maintain high success and low collision rates with a small number of particles. This is in contrast to the other methods which do not account for the number of particles used in the approximation. With an increasing number of particles, CVaR and our approximation converge to the same success and collision rates, which is expected as the number of particles increases, as shown in Sec. 3.4. Since it takes into account the number of samples used in the approximation, the results of CC-VPSTO show less of a dependence on the number of particles used (e.g. the variation across N is the smallest out of evaluated methods). However we note that in low particle regimes the approximation error cannot fully be eliminated.

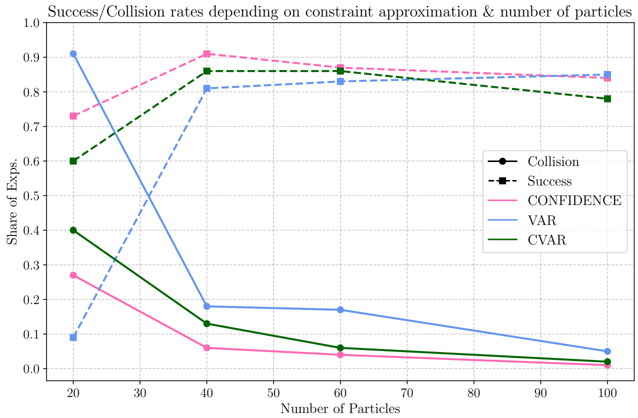

**Figure 10.** Comparison of the confidence-bounded chance constraint approximation to CVaR and a standard VaR approximation in terms of success and collision rates across MPC experiments depending on the number of particles used in the MC approximation.

## 6.3 Robot Experiment

We further demonstrate CC-VPSTO on a real robot for the scenario shown in Fig. 11. The robot is tasked to move from one side to the other of the conveyor belt, whilst avoiding a box which is controlled according to a stochastic policy. This requires the robot to be *reactive* whilst being able to plan *safe* motions in real time. The possible motions, also illustrated in Fig. 1, are to either move behind or in front of the box, as the robot is not allowed to simply move over the box. Moreover, besides the candidate trajectories that we synthesise from the sampled via-points in CC-VPSTO, we add a "waiting" trajectory to the set of final candidate trajectories sampled in the final optimisation step. A waiting trajectory is a trajectory repeating the current robot position for the entire planning horizon, *i.e.*, keeping the robot stationary. This is to allow the robot to wait for the box to pass, which is a safe, but not very efficient solution. Without these waiting trajectories, CC-VPSTO would keep the robot moving at all times, but this is not always necessary.

**Setup.** The experiment is performed on a Franka Emika robot arm. The framework was run on Ubuntu 20.04 with an Intel Core i7-8700 CPU@3.2GHz and 16GB of RAM. The ground truth box position is tracked using an Intel RealSense camera and a barcode detection pipeline. In every MPC step the robot is given the current position of the box and then plans a new trajectory using CC-VPSTO[¶]. With this setup, we are able to run the framework at a frequency of 3 Hz, using $N = 100$ particles and a planning horizon of $T_{\text{MPC}} = 3$

---

[¶] Note we ignore measurement noise in this setup.

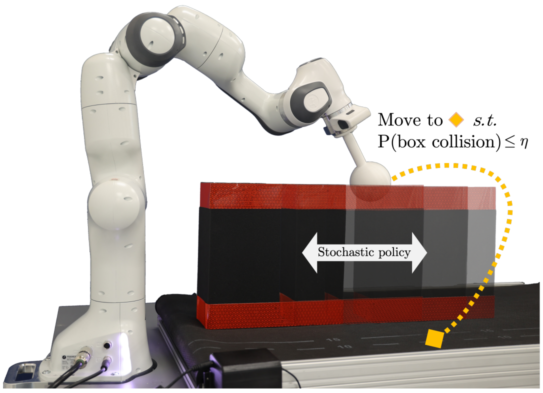

**Figure 11.** ***Robot experiment setup.*** The robot task is to move from one side to the other side of the conveyor belt while assuring that the probability of colliding with the box obstacle is below a user-defined threshold $\eta$. The motion of the box obstacle is stochastic, as the conveyor belt is actuated with constant velocity, but the rate of direction change follows an exponential distribution.

seconds (mapping to 60 time steps for the rollouts, as we use a time step of 0.05 seconds).

**Stochastic conveyor belt policy.** In this experiment, the uncertainty stems from the movement of the box on the conveyor belt, which serves as an obstacle for a robot to navigate around. The conveyor belt is velocity controlled, where the magnitude of the velocity is fixed to $0.05\frac{\mathrm{m}}{\mathrm{sec}}$ but its direction is governed by the probability density function $f(x;\alpha) = \alpha\exp(-\alpha x)$, where the probability density where $x$ is the time since the last direction change and $\alpha$ is a parameter influencing the rate of direction change. We describe our implementation of this stochastic model in more detail in the appendix in Sec. 12.3.

*Results* Similar to the MPC experiments in simulation, we evaluate our real-world robot experiment on *i)* the motion duration per run, *i.e.*, the time taken to go from one side of the conveyor belt to the other side, *ii)* the share of experiments that collide with the box, and *iii)* the minimum distance to the box across 70 runs for different values of $\eta$, as shown in Fig. 12. We do not compare our approach to "ML-VPSTO" as for the given stochastic model, the mean over the exponential distribution is not meaningful. However, we include $\eta = 0.0$ as a baseline, which corresponds to a VPSTO approach with a hard collision avoidance constraint. Overall, the results support the insights gained from the simulation experiments. We observe that the higher the value of $\eta$, the shorter the duration of the trajectory. This is because with higher values of $\eta$, CC-VPSTO is allowed to generate trajectories that are more efficient, but also less safe. Moreover, we also observe the trend that higher values of $\eta$ indeed lead to a higher share of experiments that collide with the box. However, we also observe that the share of experiments that collide with the box is still very low, even for very high values of $\eta$. This is an interesting insight from combining MPC with chance-constrained trajectory optimisation. In addition, we observe in this experiment that a value of $\eta = 0.2$ outperforms $\eta = 0.1$ in terms of the share of experiments that collide with the box. We anticipate that additional experiments will reduce this variability, as the current variance in the results is still quite high. Last, in terms of the minimum distance to the box, we cannot observe a clear trend across different values of $\eta$. This can be explained by the use of waiting trajectories that do not move the robot at all. The robot can choose to wait very close to the box, which is a safe, but not very efficient solution. This results in higher durations, but not higher minimum distances.

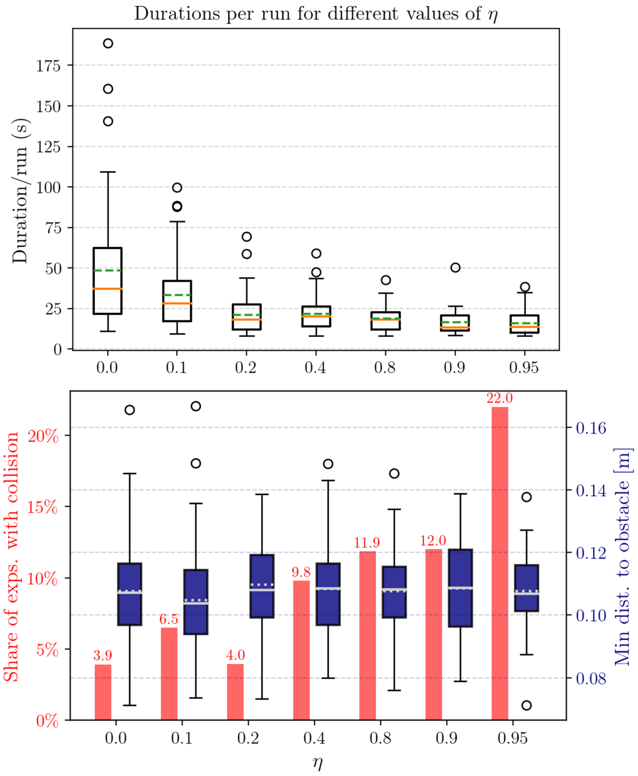

**Figure 12.** ***Robot experiment results.*** Similar to the metrics evaluated in the MPC simulation experiments, we evaluate the *motion duration*, *minimum distance* to the box and the *share of experiments that collide* with the box across 70 experiments for different values of $\eta$. Means in the boxplots are shown as dashed lines and medians as solid lines.

## 7 Discussion and Future Work

Our experiments show that CC-VPSTO is able to generate task-efficient motions while consistently bounding the probability of constraint violation, *e.g.*, collision with stochastic obstacles. While it is typically more challenging to deal with safety constraints over entire trajectories (as opposed to constraints per time step), our Monte Carlo formulation allows us to do this in a straightforward way by simulating trajectories of the obstacles and the robot then checking for collisions between them at any time over a given horizon. This is made possible by the flexibility of our approach which makes no assumption on the distribution of the uncertainty, but only requires sampling access to it. Hence, this can also be a joint distribution across all sources of uncertainty, *e.g.*, several obstacles.

As anticipated by the theory, the collision rate in the experiments was consistently below the threshold $\eta$. Nevertheless, we observe a gap between the collision rate and the threshold $\eta$. This gap is much bigger in the online planning (MPC) than in the offline planning experiments. This can be explained by the fact that at each MPC step we optimise trajectories for a longer horizon than just the time steps that we actually execute on the robot. Hence, the anticipation of potential collisions in the future makes us more conservative, resulting in lower actual rate of collision than that which was imposed. In future work, we plan to use discounted probabilities in the chance constraint, such as in Yan et al. (2018), to allow for larger probabilities of collision for time steps far in the future, knowing that the control input that we will apply then will be recomputed with stronger constraints in the meantime. Besides the introduction of discounted chance constraints, another direction could be to explore how to adapt the parameters $\eta$ and $\beta$ during online execution, *e.g.*, based on the current state of the system, the current uncertainty distribution, or the current cost function. This could result in a more robust and adaptive approach, which would be more suitable for real-world applications.

Another interesting direction for future research is about what to do in the case of an actual constraint violation, since we allow this with some probability. In our framework, since we do open-loop planning, there is no explicit way to deal with such cases. Implicitly in an MPC setting, we replan the motion given an update of the uncertain environment in the next MPC step. Yet, an interesting direction for future research is to explore closing the loop and switching the objective for the motion planner in the case of constraint violation. For instance, the objective could be turned into a purely safety maximizing formulation. Moreover, our work does not ensure recursive feasibility, which enforces that there always exists a solution to the optimization problem. Other works such as Köhler et al. (2023), have addressed this issue by enforcing that the predicted nominal state lands in a terminal set while not taking the measured state into account. However, this again comes at the cost of being restricted to linear systems with linear inequality constraints and convex objectives. For chance constraints this means that they would need to be linearized into half-space constraints, yet if we took the collision avoidance examples from this work and view them as a system that is augmented by the obstacle states, the non-collision constraint itself is non-linear in the augmented state as it has a quadratic relation through the distance measure. Yet, it would be an interesting direction for future research on how to be able to ensure recursive feasibility by with a less constrained problem formulation.

A core assumption in our approach is that we are given an *accurate* model of the uncertainty, from which we can take samples. Yet, this might be a limitation in practice, as our model might not capture the true underlying distribution with sufficient accuracy, relating to *epistemic uncertainty*. However, that can be addressed in future work, by either extending the approach to be *distributionally robust*, such as in Hakobyan and Yang (2021), or by using generative data-driven models that can be adapted online as the robot acquires more data, similar to the work of Thorpe et al. (2022).

In terms of computational efficiency, we have demonstrated the applicability of our algorithm to an MPC-setting, where we achieve frequencies of 3 Hz on a real robot using 100 particles. The biggest computational bottleneck lies in the rollouts of the uncertainty dynamics. We therefore believe that the reported control rate can be improved by adding parallelization and GPU acceleration, which we did not leverage in the given experiments. Moreover, we are aware that multimodal distributions require already by default more samples in the approximation, which might further limit current control rates. We believe that future work could explore methods to efficiently generate representative sample sets, possibly leveraging learned generative models. The advantage of our formulation is that the user can actively choose the number of particles while considering computational resources and requirements of minimum control rates. We also believe that there is still room for improvement in our implementation, as the particle rollouts for the stochastic box model have not been parallelised, as was done in the simulation experiments. Possibly, replacing the Monte Carlo simulations by a learned generative model from which predicted trajectories can be directly sampled could also lead to greater computational efficiency.

## 8 Conclusion

In this work, we explored the problem of safe and efficient robot motion planning for stochastic control problems. We proposed a novel approach that combines the benefits of chance-constrained optimisation with the benefits of sampling-based MPC. The strength of our approach lies in its generality, as it does not require any specific assumptions on the underlying uncertainty distribution, the dynamics of the system, the cost function or the specific form of inequality constraints. While we focused on the problem of collision avoidance in this work, our approach is not limited to this problem, as it can be applied to any type of stochastic control problem, as long as we can sample from the uncertainty distribution. For instance, in future work we aim to extend this framework to include constraints on interaction forces in the context of contact-rich manipulation tasks and physical human-robot interaction. We showed that our approach is able to generate safe and efficient trajectories in a variety of scenarios, including a real-world robot experiment.

### References

Alcan G and Kyrki V (2022) Differential dynamic programming with nonlinear safety constraints under system uncertainties. *IEEE Robotics and Automation Letters* 7(2): 1760–1767.

Badings T, Romao L, Abate A, Parker D, Poonawala HA, Stoelinga M and Jansen N (2023) Robust control for dynamical systems with non-gaussian noise via formal abstractions. *Journal of Artificial Intelligence Research* 76: 341–391.

Berger GO, Jungers RM and Wang Z (2021) Chance-constrained quasi-convex optimization with application to data-driven switched systems control. In: *Learning for Dynamics and Control*. PMLR, pp. 571–583.

Blackmore L (2006) A probabilistic particle control approach to optimal, robust predictive control. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. p. 6240.

Blackmore L, Li H and Williams B (2006) A probabilistic approach to optimal robust path planning with obstacles. In: *2006 American Control Conference*. IEEE, pp. 7–pp.

Blackmore L, Ono M, Bektassov A and Williams BC (2010) A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics* 26(3): 502–517.

Calafiore GC (2010) Random convex programs. *SIAM Journal on Optimization* 20(6): 3427–3464.

Calafiore GC and Campi MC (2006) The scenario approach to robust control design. *IEEE Transactions on automatic control* 51(5): 742–753.

Campi MC and Garatti S (2011) A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *Journal of optimization theory and applications* 148(2): 257–280.

Castillo-Lopez M, Ludivig P, Sajadi-Alamdari SA, Sanchez-Lopez JL, Olivares-Mendez MA and Voos H (2020) A real-time approach for chance-constrained motion planning with dynamic obstacles. *IEEE Robotics and Automation Letters* 5(2): 3620–3625.

Dai S, Schaffert S, Jasour A, Hofmann A and Williams B (2019) Chance constrained motion planning for high-dimensional robots. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8805–8811.

de Groot O, Ferranti L, Gavrila D and Alonso-Mora J (2023) Scenario-based motion planning with bounded probability of collision. *arXiv preprint arXiv:2307.01070* .

Hakobyan A and Yang I (2021) Wasserstein distributionally robust motion control for collision avoidance using conditional value-at-risk. *IEEE Transactions on Robotics* 38(2): 939–957.

Hansen N (2016) The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* .

Heirung TAN, Paulson JA, O'Leary J and Mesbah A (2018) Stochastic model predictive control—how does it work? *Computers & Chemical Engineering* 114: 158–170.

Homem-de Mello T and Bayraksan G (2014) Monte carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science* 19(1): 56–85.

Jankowski J, Brudermüller L, Hawes N and Calinon S (2023) VP-STO: Via-point-based stochastic trajectory optimization for reactive robot behavior. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 10125–10131.

Jankowski J, Racca M and Calinon S (2022) From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control. *IEEE Robotics and Automation Letters* 7(2): 3242–3249.

Janson L, Schmerling E and Pavone M (2017) Monte carlo motion planning for robot trajectory optimization under uncertainty. In: *Robotics Research: Volume 2*. Springer, pp. 343–361.

Jasour AM, Aybat NS and Lagoa CM (2015) Semidefinite programming for chance constrained optimization over semialgebraic sets. *SIAM Journal on Optimization* 25(3): 1411–1440.

Jiang C, Cornman A, Park C, Sapp B, Zhou Y, Anguelov D et al. (2023) Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9644–9653.

Köhler J, Geuss F and Zeilinger MN (2023) On stochastic mpc formulations with closed-loop guarantees: Analysis and a unifying framework. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 6692–6699.

Lew T, Bonalli R and Pavone M (2023) Risk-averse trajectory optimization via sample average approximation. *IEEE Robotics and Automation Letters* .

Majumdar A and Pavone M (2020) How should a robot assess risk? towards an axiomatic theory of risk in robotics. In: *Robotics Research: The 18th International Symposium ISRR*. Springer, pp. 75–84.

Majumdar A and Tedrake R (2017) Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research* 36(8): 947–982.

Margellos K, Goulart P and Lygeros J (2014) On the road between robust optimization and the scenario approach for chance constrained optimization problems. *IEEE Transactions on Automatic Control* 59(8): 2258–2263.

Mesbah A (2016) Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine* 36(6): 30–44.

Mohri M, Rostamizadeh A and Talwalkar A (2018) *Foundations of machine learning*. MIT press.

Nemirovski A and Shapiro A (2007) Convex approximations of chance constrained programs. *SIAM Journal on Optimization* 17(4): 969–996.

Ono M and Williams BC (2008) Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In: *2008 47th IEEE Conference on Decision and Control*. IEEE, pp. 3427–3432.

Pagnoncelli BK, Ahmed S and Shapiro A (2009) Sample average approximation method for chance constrained programming: theory and applications. *Journal of optimization theory and applications* 142(2): 399–416.

Parsi A, Anagnostaras P, Iannelli A and Smith RS (2022) Computationally efficient robust mpc using optimized constraint tightening. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, pp. 1770–1775.

Peña-Ordieres A, Luedtke JR and Wächter A (2020) Solving chance-constrained problems via a smooth sample-based nonlinear approximation. *SIAM Journal on Optimization* 30(3): 2221–2250.

Prékopa A (2013) *Stochastic programming*, volume 324. Springer Science & Business Media.

Priore S and Oishi M (2023) Chance constrained stochastic optimal control based on sample statistics with almost surely probabilistic guarantees. *arXiv preprint arXiv:2303.16981* .

Schildbach G, Fagiano L, Frei C and Morari M (2014) The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations. *Automatica* 50(12): 3009–3018.

Schmerling E and Pavone M (2016) Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. *arXiv preprint arXiv:1609.05399* .

Shalev-Shwartz S and Ben-David S (2014) *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shapiro A (2003) Monte carlo sampling approach to stochastic programming. In: *ESAIM: proceedings*, volume 13. EDP Sciences, pp. 65–73.

Shapiro A, Dentcheva D and Ruszczynski A (2021) *Lectures on stochastic programming: modeling and theory*. SIAM.

Sun W, Torres LG, Van Den Berg J and Alterovitz R (2016) Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In: *Robotics Research: The 16th International Symposium ISRR*. Springer, pp. 685–701.

Taboga M (2017) Lectures on probability theory and mathematical statistics. *(No Title)* .

Thorpe A, Lew T, Oishi M and Pavone M (2022) Data-driven chance constrained control using kernel distribution embeddings. In: *Learning for Dynamics and Control Conference*. PMLR, pp. 790–802.

Trevisan E, Mustafa KA, Notten G, Wang X and Alonso-Mora J (2025) Dynamic risk-aware mppi for mobile robots in crowds via efficient monte carlo approximations. *arXiv preprint arXiv:2506.21205* .

Van Den Berg J, Abbeel P and Goldberg K (2011) Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research* 30(7): 895–913.

Wang A, Jasour A and Williams B (2020) Moment state dynamical systems for nonlinear chance-constrained motion planning. *arXiv preprint arXiv:2003.10379* .

Williams G, Wagener N, Goldfain B, Drews P, Rehg JM, Boots B and Theodorou EA (2017) Information theoretic mpc for model-based reinforcement learning. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 1714–1721.

Yan S, Goulart P and Cannon M (2018) Stochastic model predictive control with discounted probabilistic constraints. In: *2018 European Control Conference (ECC)*. IEEE, pp. 1003–1008.

Yin J, Zhang Z and Tsiotras P (2023) Risk-aware model predictive path integral control using conditional value-at-risk. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7937–7943.

Zhang Z, Tomlinson J and Martin C (1997) Splines and linear control theory. *Acta Math. Appl* 49: 1–34.

# APPENDIX

## 9 Trajectory Representation

The way we represent trajectories is based on previous work showing that the closed-form solution to the following optimisation problem

$$\begin{aligned} \min \quad & \int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds \\ \text{s.t.} \quad & \boldsymbol{q}(s_n) = \boldsymbol{q}_n, \quad n = 1, ..., N \\ & \boldsymbol{q}(0) = \boldsymbol{q}_0, \boldsymbol{q}'(0) = \boldsymbol{q}'_0,\ \boldsymbol{q}(1) = \boldsymbol{q}_T, \boldsymbol{q}'(1) = \boldsymbol{q}'_T \end{aligned} \tag{16}$$

is given by cubic splines (Zhang et al. 1997) and that it can be formulated as a weighted superposition of basis functions (Jankowski et al. 2022). Hence, the robot's configuration is defined as $\boldsymbol{q}(s) = \boldsymbol{\Phi}(s)\boldsymbol{w} \in \mathbb{R}^D$, with $D$ being the number of degrees of freedom. The matrix $\boldsymbol{\Phi}(s)$ contains the basis functions which are weighted by the vector $\boldsymbol{w}$[‖]. The trajectory is defined on the interval $\mathcal{S} = [0, 1]$, while the time $t$ maps to the phase variable $s = \frac{t}{T} \in \mathcal{S}$ with $T$ being the total duration of the trajectory. Consequently, joint velocities and accelerations along the trajectory are given by $\dot{\boldsymbol{q}}(s) = \frac{1}{T}\boldsymbol{\Phi}'(s)\boldsymbol{w}$ and $\ddot{\boldsymbol{q}}(s) = \frac{1}{T^2}\boldsymbol{\Phi}''(s)\boldsymbol{w}$, respectively[**]. The basis function weights $\boldsymbol{w}$ include the trajectory constraints consisting of the boundary condition parameters $\boldsymbol{w}_{\text{bc}} = [\boldsymbol{q}_0^\top, \boldsymbol{q}_0'^\top, \boldsymbol{q}_T^\top, \boldsymbol{q}_T'^\top]^\top$ and $N$ via-points the trajectory has to pass through $\boldsymbol{q}_{\text{via}} = [\boldsymbol{q}_1^\top, \ldots, \boldsymbol{q}_N^\top]^\top \in \mathbb{R}^{DN}$, such that $\boldsymbol{w} = [\boldsymbol{q}_{\text{via}}^\top, \boldsymbol{w}_{\text{bc}}^\top]^\top$. Throughout this paper, the via-point timings $s_n$ are assumed to be uniformly distributed in $\mathcal{S}$. Note, that boundary velocities map to boundary derivatives w.r.t. $s$ by multiplying them with the total duration $T$, *i.e.*, $\boldsymbol{q}_0' = T\dot{\boldsymbol{q}}_0$ and $\boldsymbol{q}_T' = T\dot{\boldsymbol{q}}_T$. Furthermore, the optimisation problem in Eq. (16) minimizes not only the objective $\boldsymbol{q}''(s)$, but also the integral over accelerations, since $\boldsymbol{q}''(s) = T^2\ddot{\boldsymbol{q}}(s)$ and thus the objective $\int_0^1 \ddot{\boldsymbol{q}}(s)^\top \ddot{\boldsymbol{q}}(s) ds$ directly maps to $\frac{1}{T^4}\int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds$, corresponding to the control effort. It is minimal *iff* the objective in Eq. (16) is minimal. As a result, this trajectory representation provides a linear mapping from via points, boundary conditions and the movement duration to a time-continuous and smooth trajectory.

CC-VPSTO, analogously to VP-STO, exploits this explicit parameterisation with via-points and boundary conditions by optimizing *only the via-points* while keeping the predefined boundary condition parameters fixed.

## 10 Baseline for Offline Simulation Experiments: Derivation and Background

We present an alternative approach to approximate the chance constraint in Eq. (1) for the special case of obstacle collision avoidance, which we use as a baseline. For this, we leverage statistical learning theory (Shalev-Shwartz and Ben-David 2014; Mohri et al. 2018) to obtain an alternative confidence-based bound for $k_{\text{thresh}}$, which we call $k_{\text{rad}}$. This bound is more theoretically complete, as it does not require the independence of the Bernoulli variables, but we also note that it is more conservative, computationally expensive, and, less general since it is limited to a specific motion planning problem.

### *10.1 Preliminaries on Statistical Learning Theory*

We introduce the concept of *Rademacher complexity* which is a measure used in statistical learning theory to quantify the complexity of a class of functions with respect to a given dataset. The intuition is as follows: if the "chance-constraint" function $g$ is "simple" (*e.g.*, a constant or linear function), then the complexity of the class $\mathcal{F}$, defined later based on $g$, will be low. As established by Proposition 3 in Mohri et al. (2018), this implies that the "generalization property" of $\mathcal{F}$ is good. In our case, this means that if a solution $\boldsymbol{x}$ has a small rate of violating constraint $g$ with respect to $N$ i.i.d. samples, then there is a good chance that the actual probability of constraint violation of $\boldsymbol{x}$ is small too.

First, we introduce the notion of *Rademacher complexity* and the associated *generalization* result:

**Definition 1.** If $\mathcal{F}$ is a (possibly infinite) set of functions from a set $\mathcal{D}$ to $\mathbb{R}$, *i.e.*, $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{D}}$, and $D = \{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_N\}_{i=1}^N$

---

[‖] A more detailed explanation of the basis functions and their derivation can be found in the appendix of Jankowski et al. (2022).

[**] We use the notation $f'(s)$ for derivatives w.r.t. $s$ and the notation $\dot{f}(s)$ for derivatives w.r.t. $t$.

is a set of $N$ elements of $\mathcal{D}$, then the *Rademacher complexity of $\mathcal{F}$ with respect to $D$* is defined by

$$R_D(\mathcal{F}) = E_{\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_N}\left[\sup_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\sigma}_i f(\boldsymbol{\delta}_i)\right],$$

where $\{\boldsymbol{\sigma}_i\}_{i=1}^N$ are sampled independently uniformly at random in $\{-1,1\}$. Furthermore, if $\Delta$ is a random variable with values in $\mathcal{D}$, then the *Rademacher complexity of $\mathcal{F}$ with respect to $\Delta$ with $N$ samples* is defined by

$$R_{\Delta,N}(\mathcal{F}) = E_{\boldsymbol{\delta}_1\sim\Delta,\ldots,\boldsymbol{\delta}_N\sim\Delta}[R_D(\mathcal{F})]$$

wherein $D$ stands for $\{\boldsymbol{\delta}_i\}_{i=1}^N$.

A well-known result in statistical learning states that if $D = \{\boldsymbol{\delta}_i\}_{i=1}^N$ is a set of $N$ independent samples $\boldsymbol{\delta}_1 \sim \Delta,\ldots,\boldsymbol{\delta}_N \sim \Delta$, then with confidence $1-\beta$ on the sampling of $D$, it holds that for every $f \in \mathcal{F}$, $\frac{1}{N}\sum_{i=1}^N f(\boldsymbol{\delta}_i)$ is "close" to $E_{\boldsymbol{\delta}\sim\Delta}[f(\boldsymbol{\delta})]$, where "close" is quantified with a quantity that depends on $\beta$, $N$ and $R_{\Delta,N}(\mathcal{F})$. Formally, we have:

**Proposition 3.** (Mohri et al. 2018, Theorem 3.3). *It holds that*

$$P_{\boldsymbol{\delta}_1\sim\Delta,\ldots,\boldsymbol{\delta}_N\sim\Delta}\left[\max_{f\in\mathcal{F}}\left\{E_{\boldsymbol{\delta}\sim\Delta}[f(\boldsymbol{\delta})] - \frac{1}{N}\sum_{i=1}^N f(\boldsymbol{\delta}_i)\right\} \le 2R_{\Delta,N}(\mathcal{F}) + \sqrt{\frac{\log(\frac{1}{\beta})}{2N}}\right] \ge 1-\beta. \tag{17}$$

## 10.2 Rademacher Complexity for Surrogate Constraint

We apply the result in Proposition 3 to the surrogate optimisation problem in Eq. (9). For that, we define $\mathcal{D}$ as the domain of $\Delta$ and $\mathcal{F} = \{\boldsymbol{\delta} \mapsto \mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta})\le 0} \mid \boldsymbol{x} \in X\} \subseteq \mathbb{R}^{\mathcal{D}}$. It holds that for each $\boldsymbol{x} \in X$ and $D = \{\boldsymbol{\delta}_i\}_{i=1}^N \subseteq \mathcal{D}$, $E_{\boldsymbol{\delta}\sim\Delta}[\mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta})\le 0}] = P[G_{\boldsymbol{x}} = 1]$ and $\sum_{i=1}^N \mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta}_i)\le 0} = s_N(\boldsymbol{x}; D)$. Hence, we get the following:

**Corollary 1.** *With $\mathcal{F}$ defined as above, it holds that*

$$P_{\boldsymbol{\delta}_1\sim\Delta,\ldots,\boldsymbol{\delta}_N\sim\Delta}\left[\max_{\boldsymbol{x}\in X}\left\{P[G_{\boldsymbol{x}} = 1] - \frac{1}{N}s_N(\boldsymbol{x}; D)\right\} \le 2R_{\Delta,N}(\mathcal{F}) + \sqrt{\frac{\log(\frac{1}{\beta})}{2N}}\right] \ge 1-\beta, \tag{18}$$

*wherein $D$ stands for $\{\boldsymbol{\delta}_i\}_{i=1}^N$.*

Corollary 1 tells us that with confidence $1-\beta$ on the sampling of $D = \{\boldsymbol{\delta}_i\}_{i=1}^N$ with $N$ i.i.d. samples from $\Delta$, *any* solution $\boldsymbol{x}$ that is feasible for the surrogate optimisation problem Eq. (9) with

$$k_{\text{thresh}} \le \left(\eta - 2R_{\Delta,N}(\mathcal{F}) - \sqrt{\frac{\log(\frac{1}{\beta})}{2N}}\right) N \tag{19}$$

is feasible for the original optimisation problem Eq. (1).

In view of Eq. (19), computing a suitable $k_{\text{thresh}}$ for the surrogate optimisation problem in Eq. (9) can be approached by computing an upper bound on the Rademacher complexity of the associated set of functions $\mathcal{F}$. Despite the theoretical attractivity of the this approach, computing an upper bound on the Rademacher complexity can be very challenging in general, and there is usually no closed-from expression for such bounds. Nevertheless, we present here a tight bound for a special case of collision-avoidance problem.

## 10.3 A Special Case of Collision Avoidance

We consider a robot motion planning problem, where a ball-shaped robot has to avoid $m$ ball-shaped obstacles with high probability across time instants $t_1,\ldots,t_H$.

For the sake of simplicity, we first focus on the case with one obstacle ($m = 1$) and one time step ($H = 1$), before generalising. Thus, we consider the problem of finding the position $\boldsymbol{x} \in X$ ($X \subseteq \mathbb{R}^n$) of the center of the robot at time $t_1$ such that $P_{\boldsymbol{\delta}\sim\Delta}(\|\boldsymbol{x} - \boldsymbol{p}(\boldsymbol{\delta})\| \ge r) \ge 1-\eta$, where $r > 0$ is the combined radius of the obstacle and the robot, and $\boldsymbol{p}(\boldsymbol{\delta}) \in \mathbb{R}^n$ is the position of the center of the obstacle at time $t_1$ under scenario $\boldsymbol{\delta}$. In the formulation of Eq. (1), we have

$$g(\boldsymbol{x},\boldsymbol{\delta}) = r - \|\boldsymbol{x} - \boldsymbol{p}(\boldsymbol{\delta})\|,$$

*i.e.*, $g(\boldsymbol{x},\boldsymbol{\delta}) \le 0 \Leftrightarrow \|\boldsymbol{x} - \boldsymbol{p}(\boldsymbol{\delta})\| \ge r$.

Let $\mathcal{F}$ be defined as in the previous subsection, *i.e.*, $\mathcal{F} = \{\mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta})\le 0} \mid \boldsymbol{x} \in X\}$, with $X$ and $g$ as above. In the subsequent section with the additional proofs (Appendix 10.4), we bound $R_{\Delta,N}(\mathcal{F})$ as follows:

$$R_{\Delta,N}(\mathcal{F}) \le \sqrt{\frac{d\log\left(\frac{eN}{d}\right)}{2N}}, \tag{20}$$

where $d = n+1$ and $e$ is Euler's number. We obtain the following:

**Proposition 4.** *In the setting defined above with $m = H = 1$, if we define $k_{\text{rad}}(\beta,N,\eta)$ as*

$$k_{\text{rad}}(\beta,N,\eta) = \eta N - \sqrt{2dN\log\left(\frac{eN}{d}\right)} - \sqrt{\frac{N\log(\frac{1}{\beta})}{2}},$$

*then any feasible solution of the surrogate optimisation problem in Eq. (9) with $k_{\text{thresh}} = k_{\text{rad}}(\beta,N,\eta)$ is feasible for the original optimisation problem in Eq. (2) with confidence $1-\beta$ on the sampling of $D$.*

**Proof.** Consequence of Eqs. (19) and (20).

We now discuss the case of $m \in \mathbb{N}_{\ge 1}$ obstacles and $H \in \mathbb{N}_{\ge 1}$ time steps. In this case,

$$g(\boldsymbol{x},\boldsymbol{\delta}) = \max_{\substack{j=1,\ldots,m\\ k=1,\ldots,H}} r - \|\boldsymbol{q}(\boldsymbol{x},t_k) - \boldsymbol{p}_j(\boldsymbol{\delta},t_k)\|,$$

*i.e.*, $g(\boldsymbol{x},\boldsymbol{\delta}) \le 0 \Leftrightarrow \forall j\,\forall k\ \|\boldsymbol{x}(t_k) - \boldsymbol{p}_j(\boldsymbol{\delta},t_k)\| \ge r$, where $\boldsymbol{x}(t_k)$ ($X \subseteq \mathbb{R}^{nH}$) is the position of the center of the robot at time $t_k$ and $\boldsymbol{p}_j(\boldsymbol{\delta},t_k) \in \mathbb{R}^n$ is the position of the center of the $j^{\text{th}}$ obstacle at time $t_k$ under scenario $\boldsymbol{\delta}$. We show in Appendix 10.4 that $R_{\Delta,N}(\mathcal{F})$ can be bounded as follows:

$$R_{\Delta,N}(\mathcal{F}) \le mH\sqrt{\frac{d\log\left(\frac{eN}{d}\right)}{2N}}, \tag{21}$$

where $d = n+1$ and $e$ is Euler's number. Similarly to the above, we get the following:

**Proposition 5.** *In the setting defined above with $m \in \mathbb{N}_{\geq 1}$ and $H \in \mathbb{N}_{\geq 1}$, if we define $k_{\text{rad}}(\beta, N, \eta)$ as*

$$k_{\text{rad}}(\beta, N, \eta) = \eta N - mH\sqrt{2dN\log\left(\frac{eN}{d}\right)} - \sqrt{\frac{N\log(\frac{1}{\beta})}{2}},$$

*then any feasible solution of the surrogate optimisation problem in Eq. (9) with $k_{\text{thresh}} = k_{\text{rad}}(\beta, N, \eta)$ is feasible for the original optimisation problem in Eq. (2) with confidence $1-\beta$ on the sampling of $D$.*

**Proof.** Consequence of Eqs. (19) and (21).

*Remark 4.* Note that, unlike other approaches in the literature, Proposition 5 does not rely on *Boole's inequality* to bound the joint probability of collision avoidance. Indeed, the use of Boole's inequality would amount to set the collision avoidance probability for *each time step and each obstacle* to $\eta' = \frac{\eta}{mH}$, so that the probability of collision *with at least one obstacle at at least one time step* is bounded from above by $\eta = \sum_{j,k} \eta'$. Furthermore, we would need to set the confidence for *each time step and each obstacle* to $1-\beta'$ with $\beta' = \frac{\beta}{mH}$, in order to guarantee with confidence $1-\beta = 1-\sum_{j,k}\beta'$ that the chance constraint holds for each of them simultaneously. This would result in a value of $k_{\text{thresh}}$ as follows:

$$k'_{\text{rad}}(\beta, N, \eta) = \frac{\eta N}{mH} - \sqrt{2dN\log\left(\frac{eN}{d}\right)} - \sqrt{\frac{N\log(\frac{mH}{\beta})}{2}}.$$

This can be rewritten as

$$k'_{\text{rad}}(\beta, N, \eta) = \frac{\eta N - mH\sqrt{2dN\log\left(\frac{eN}{d}\right)} - mH\sqrt{\frac{N\log(\frac{mH}{\beta})}{2}}}{mH},$$

The above shows that for any values of $\beta$, $N$ and $\eta$ for which $k_{\text{rad}}(\beta, N, \eta) \geq 0$, it holds that

$$k'_{\text{rad}}(\beta, N, \eta) \leq \frac{1}{mH} k_{\text{rad}}(\beta, N, \eta).$$

Hence, using $k_{\text{thresh}} = k_{\text{rad}}(\beta, N, \eta)$ in Eq. (9) is less conservative (by a "factor" $mH$) than using $k_{\text{thresh}} = k'_{\text{rad}}(\beta, N, \eta)$.

### *10.4 Additional Proofs*

We start with Eq. (20), reminded in the proposition below:

**Proposition 6.** *In the setting defined in above with $m = H = 1$, it holds that*

$$R_{\Delta,N}(\mathcal{F}) \leq \sqrt{\frac{d\log\left(\frac{eN}{d}\right)}{2N}}, \tag{22}$$

*wherein $d = n+1$ and $e$ is Euler's number.*

**Proof.** Consider the set of functions $\mathcal{H} = \{2f - 1 \mid f \in \mathcal{F}\}$, *i.e.*,

$$\mathcal{H} = \{2 \cdot \mathbf{1}_{\|\boldsymbol{x}-\boldsymbol{p}(\boldsymbol{\delta})\| \leq r} - 1 \mid \boldsymbol{x} \in X\}, \tag{23}$$

which is essentially the same as $\mathcal{F}$ except that the functions take values in $\{-1, 1\}$ instead of $\{0, 1\}$,[††] and the quantity

$$\Pi_{\mathcal{H}}(N) = \max_{(\boldsymbol{\delta}_1,\ldots,\boldsymbol{\delta}_N)\in\mathcal{D}^N} \#\{(h(\boldsymbol{\delta}_1),\ldots,h(\boldsymbol{\delta}_N)) \mid h \in \mathcal{H}\}.$$

By (Mohri et al. 2018, Corollary 3.8), it holds that

$$R_{\Delta,N}(\mathcal{H}) \leq \sqrt{\frac{2\log\Pi_{\mathcal{H}}(N)}{N}}.$$

We will bound $\Pi_{\mathcal{H}}(N)$ by $\left(\frac{eN}{d}\right)^d$ by using Eq. (23). For that, we consider the set of functions $\mathcal{H}' = \{h_{\boldsymbol{x},r} \mid (\boldsymbol{x}, r) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0}\} \subseteq \mathbb{R}^{\mathcal{P}}$ where $\mathcal{P} = \mathbb{R}^n$ and $h_{\boldsymbol{x},r}(\boldsymbol{p}) = 2 \cdot \mathbf{1}_{\|\boldsymbol{x}-\boldsymbol{p}\| \leq r} - 1$, which contains all *ball classifiers* in $\mathbb{R}^n$ since $h_{\boldsymbol{x},r}(\boldsymbol{p}) = 1$ if $\boldsymbol{p}$ is in the ball of center $\boldsymbol{x}$ and radius $r$, and $-1$ otherwise. It follows that the VC dimension of $\mathcal{H}'$, defined as the largest $N$ for which $\Pi_{\mathcal{H}'} = 2^N$, is $d = n+1$.[‡‡] Hence, by (Mohri et al. 2018, Corollary 3.18), it follows that $\Pi_{\mathcal{H}'}(N) \leq \left(\frac{eN}{d}\right)^d$. It is also straightforward to show that $\Pi_{\mathcal{H}}(N) \leq \Pi_{\mathcal{H}'}(N)$ since for every $\boldsymbol{x} \in X$ and every $\boldsymbol{\delta} \in \mathcal{D}$, $2 \cdot \mathbf{1}_{\|\boldsymbol{x}-\boldsymbol{p}(\boldsymbol{\delta})\| \leq r} - 1 = h_{\boldsymbol{x},r}(\boldsymbol{p}(\boldsymbol{\delta}))$. Hence,

$$R_{\Delta,N}(\mathcal{H}) \leq \sqrt{\frac{2d\log(eN/d)}{N}}.$$

Finally, from (Shalev-Shwartz and Ben-David 2014, Lemma 26.9), we get that

$$R_{\Delta,N}(\mathcal{F}) \leq \frac{1}{2}R_{\Delta,N}(\mathcal{H}) \leq \sqrt{\frac{d\log(eN/d)}{2N}},$$

concluding the proof.

Next, we consider Eq. (21) and prove the following:

**Proposition 7.** *In the setting defined in Sec.?? with $m \in \mathbb{N}_{\geq 1}$ and $H \in \mathbb{N}_{\geq 1}$, it holds that*

$$R_{\Delta,N}(\mathcal{F}) \leq mH\sqrt{\frac{d\log\left(\frac{eN}{d}\right)}{2N}}, \tag{24}$$

*wherein $d = n+1$ and $e$ is Euler's number.*

**Proof.** Note that by definition of $g$, it holds that

$$\mathbf{1}_{g(\boldsymbol{x},\boldsymbol{\delta})>0} = \max_{j,k} \mathbf{1}_{\|\boldsymbol{x}(t_k)-\boldsymbol{p}_j(\boldsymbol{\delta},t_k)\| \leq r} - 1. \tag{25}$$

For each $j = 1, \ldots, m$ and $k = 1, \ldots, H$, let

$$\mathcal{F}_{j,k} = \{\mathbf{1}_{\|\boldsymbol{x}(t_k)-\boldsymbol{p}_j(\boldsymbol{\delta},t_k)\| \leq r} - 1 \mid \boldsymbol{x} \in X\},$$

and let $\mathcal{F}' = \{\max_{j,k} f_{j,k} \mid \forall j\, \forall k\; f_{j,k} \in \mathcal{F}_{j,k}\}$. By Eq. (25), it holds that $\mathcal{F} \subseteq \mathcal{F}'$. By (Mohri et al. 2018,

---

[††] This is done to stick to the classical theory of *binary classification* learning for which many results on the Rademacher complexity have been derived (Shalev-Shwartz and Ben-David 2014; Mohri et al. 2018).

[‡‡] See for instance Sec. 15.5.2 in `https://ti.inf.ethz.ch/ew/lehre/CG12/lecture/Chapter%2015.pdf` (last consulted: July 30, 2024).

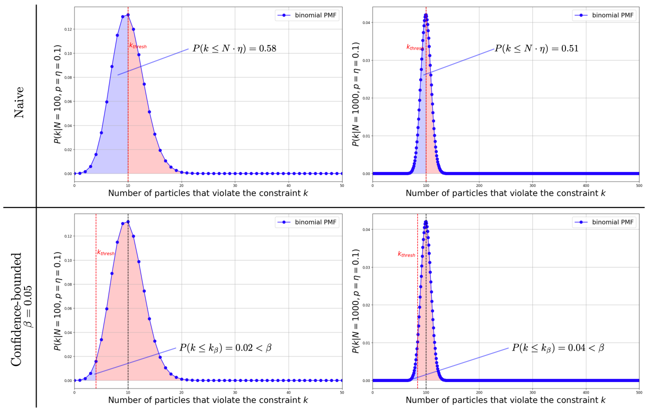

**Figure 13.** Analysis of the binomial distribution with $N = 100$ (left column) and $N = 1000$ (right column) Bernoulli experiments, which correspond to the number of particles used to approximate the chance constraint in the optimisation. The top row shows the values $k_{\text{thresh}}$ takes for the naive formulation of setting $k_{\text{thresh}} = \eta N$, where $\eta$ is the user-defined maximum probability of violation. The bottom row shows the values $k_{\text{thresh}}$ takes for the confidence-bounded formulation of the chance constraint, *i.e.*, $k_{\text{thresh}} = k_{\text{binom}}(\beta; N, \eta)$ for $\beta = 0.05$.

Ex. 3.8), it holds that $R_{\Delta,N}(\mathcal{F}') = \sum_{j,k} R_{\Delta,N}(\mathcal{F}_{j,k})$. Furthermore, by Proposition 6, we know that for each $j = 1, \dots, m$ and $k = 1, \dots, H$, $R_{\Delta,N}(\mathcal{F}_{j,k})$ is bounded from above by the right-hand side of Eq. (22). By summing over $j$ and $k$, we get that $R_{\Delta,N}(\mathcal{F}')$ is bounded from above by the right-hand side term in Eq. (24). Since $\mathcal{F} \subseteq \mathcal{F}'$, we have that $R_{\Delta,N}(\mathcal{F}) \leq R_{\Delta,N}(\mathcal{F}')$, concluding the proof.

## 11 Additional Details on Naive vs. Confidence-Bounded Surrogate Constraint

We use Fig. 13 to illustrate the difference between the naive formulation that only considers the maximum violation threshold $\eta$ by setting $k_{\text{thresh}} = \eta N$ and the confidence-bounded formulation using $k_\beta$. For this purpose we analyse the binomial distribution with parameters $N$ and $p = \eta = 0.1$ for different values of $N$, which correspond to the number of particles $\boldsymbol{\delta}_i \sim p_\Delta$ in the optimisation scheme. The plots in Fig. 13 shows the binomial distribution with $N = 100$ on the left and $N = 1000$ on the right. The blue shaded areas under the curve corresponds to the value of the CDF for $k_{\text{thresh}}$, *i.e.*, $P(K \leq k_{\text{thresh}} \mid N, \eta)$. The red-colored area under the curve corresponds to the the probability that $k > k_{\text{thresh}}$. The top row shows the naive formulation, *i.e.*, setting $k_{\text{thresh}} = \eta N$. The bottom row shows the confidence-bounded formulation, *i.e.*, setting $k_{\text{thresh}} = C(k \mid N, p)^{-1}(\beta)$ for $\beta = 0.05$.

## 12 Experiment Details

### *12.1 MPC Experiments: Environment Details*

In this section, we provide additional details about the environments used for the MPC experiments in Sec. 6.2. We used three different environment configurations for which we generated the parameters randomly. Tab. 3 shows the specifications of the environments, *i.e.*, the number of obstacles $N_{obs}$, the initial obstacle positions $\boldsymbol{x}_0$, the initial obstacle velocities $\dot{\boldsymbol{x}}_0$, the obstacle radii and the variance of the obstacle accelerations $\ddot{\boldsymbol{x}}$, when sampling from a zero-mean Gaussian distribution in the random-walk model. The environment size was chosen to be consistent across all environments on a 10 by 10 grid.

**Table 2.** Offline Planning Experiments for $\beta = 0.05$

| $\eta$ | $\eta_{\text{rad}}$ | $\eta_{\text{binom}}$ | $\hat{\eta}_{\text{avg}}$ | $\hat{\eta}_{1-\beta}$ | $\bar{\beta}$ |
|---|---|---|---|---|---|
| 0.05 | n/a | 0.01 | 0.0218 | 0.0499 | 0.0497 |
| | n/a | 0.038 | 0.0393 | 0.0503 | 0.0539 |
| 0.1 | n/a | 0.04 | 0.0525 | 0.0936 | 0.0325 |
| | n/a | 0.084 | 0.0854 | 0.1012 | 0.0627 |
| 0.15 | n/a | 0.08 | 0.0934 | 0.1449 | 0.0375 |
| | n/a | 0.131 | 0.1322 | 0.1508 | 0.0572 |
| 0.2 | n/a | 0.13 | 0.1438 | 0.2060 | 0.0656 |
| | n/a | 0.178 | 0.1794 | 0.2010 | 0.0574 |
| 0.25 | n/a | 0.17 | 0.1842 | 0.2515 | 0.0535 |
| | 0.009 | 0.227 | 0.2288 | 0.2518 | 0.0638 |
| 0.3 | n/a | 0.22 | 0.2350 | 0.3068 | 0.0667 |
| | 0.059 | 0.275 | 0.2764 | 0.3005 | 0.0533 |
| 0.35 | n/a | 0.26 | 0.2755 | 0.3507 | 0.0517 |
| | 0.109 | 0.324 | 0.3251 | 0.3510 | 0.0578 |
| 0.4 | n/a | 0.31 | 0.3262 | 0.4053 | 0.0627 |
| | 0.159 | 0.374 | 0.3760 | 0.4029 | 0.0705 |
| 0.6 | n/a | 0.51 | 0.5282 | 0.6101 | 0.0754 |
| | 0.359 | 0.573 | 0.5748 | 0.6025 | 0.0683 |
| 0.8 | 0.158 | 0.72 | 0.7359 | 0.8053 | 0.0668 |
| | 0.559 | 0.778 | 0.7798 | 0.8024 | 0.0714 |

### *12.2 Detailed Results on Offline-CC-VPSTO*

The numerical results for the offline planning experiments are shown in Tab. 2.

### *12.3 Robot Experiment: Implementation of Stochastic Model*

In this section, we provide additional details about the implementation of the stochastic model for the robot experiment in Sec. 6.3 describing the motion of the box obstacle on the conveyor belt. As our approach is Monte Carlo-based, in every MPC-step we simulate the motion of the box obstacle for $N_{\text{sim}}$ particles. The particles are initialised with the same position and velocity as the box obstacle at the beginning of the MPC-step. The particles are then propagated through the conveyor belt dynamics for the duration of the MPC-step. The conveyor belt dynamics are modelled as a probabilistic system, where the probability of changing direction increases over time. A particle at time step $k$ is modeled by state vector $\mathbf{s} = [x_k, \dot{x}_k, p_k]$ where $x_k$ is the position, $\dot{x}_k$ is the velocity, and $p_k$ is the probability of changing direction at time step $k$. The dynamics of this system for each time step $\Delta t$ can be described as follows:

1. *Update the Probability of Direction Change:*

$$p_{k+1} = p_k \cdot (1 - \alpha) \tag{26}$$

   where $\alpha$ is the rate at which the probability of a direction change increases over time.

2. *Determine the Direction Change:*

**Table 3.** MPC Environment Specifications

| Env. | 0 | 1 | 2 |
|---|---|---|---|
| $N_{obs}$ | 5 | 4 | 5 |
| Robot radius | 0.25 | 0.5 | 0.5 |
| $\boldsymbol{x}_0$ | $\left[\begin{bmatrix}2.0\\4.0\end{bmatrix},\begin{bmatrix}3.5\\8.0\end{bmatrix},\begin{bmatrix}7.5\\2.5\end{bmatrix},\begin{bmatrix}9.0\\1.5\end{bmatrix},\begin{bmatrix}4.5\\8.0\end{bmatrix}\right]$ | $\left[\begin{bmatrix}7.9\\5.7\end{bmatrix},\begin{bmatrix}1.3\\3.5\end{bmatrix},\begin{bmatrix}4.9\\9.4\end{bmatrix},\begin{bmatrix}5.2\\3.0\end{bmatrix}\right]$ | $\left[\begin{bmatrix}2.1\\3.1\end{bmatrix},\begin{bmatrix}6.8\\5.0\end{bmatrix},\begin{bmatrix}7.3\\6.7\end{bmatrix},\begin{bmatrix}4.2\\4.2\end{bmatrix},\begin{bmatrix}8.5\\2.8\end{bmatrix}\right]$ |
| $\dot{\boldsymbol{x}}_0$ | $\left[\begin{bmatrix}0.7\\0.0\end{bmatrix},\begin{bmatrix}0.25\\-0.5\end{bmatrix},\begin{bmatrix}-0.5\\0.5\end{bmatrix},\begin{bmatrix}-0.1\\0.1\end{bmatrix},\begin{bmatrix}0.0\\-1.0\end{bmatrix}\right]$ | $\left[\begin{bmatrix}0.6\\0.1\end{bmatrix},\begin{bmatrix}0.0\\0.2\end{bmatrix},\begin{bmatrix}-0.4\\0.1\end{bmatrix},\begin{bmatrix}-0.2\\0.0\end{bmatrix}\right]$ | $\left[\begin{bmatrix}0.5\\-0.2\end{bmatrix},\begin{bmatrix}0.5\\0.0\end{bmatrix},\begin{bmatrix}0.0\\-0.2\end{bmatrix},\begin{bmatrix}0.4\\0.6\end{bmatrix},\begin{bmatrix}0.2\\-0.3\end{bmatrix}\right]$ |
| Radii | $[0.5, 0.4, 0.3, 0.35, 0.55]$ | $[-0.32, 0.51, 0.49, 0.34]$ | $[0.54, 0.45, 0.55, 0.35, 0.34]$ |
| $var(\ddot{\boldsymbol{x}})$ | $[0.5, 0.75, 0.65, 0.8, 0.6]$ | $[0.54, 0.64, 0.51, 0.8]$ | $[0.64, 0.66, 0.62, 0.57, 0.75]$ |

- Sample a random number $r$ from a uniform distribution between 0 and 1.
- If $r < p_{k+1}$ or if the projected position $x_k + \dot{x}_k \Delta t$ is outside the boundaries of the conveyor belt, a direction change occurs.

3. *Update State based on Direction Change:*

$$\dot{x}_{k+1} = \begin{cases} -\dot{x} & \text{if direction change occurs} \\ \dot{x} & \text{otherwise} \end{cases} \tag{27}$$

$$p_{k+1} = \begin{cases} \alpha & \text{if direction change occurs} \\ p_{k+1} & \text{otherwise} \end{cases} \tag{28}$$

4. *Update Position:*

$$x_{k+1} = x + \dot{x}_{k+1} \Delta t \tag{29}$$

Therefore, the updated state vector after each time step is:

$$\mathbf{s}_{k+1} = [x_{k+1},\, \dot{x}_{k+1},\, p_{k+1}] \tag{30}$$

In summary, the above models the probabilistic dynamics of one box particle on the conveyor belt, where the direction of motion can change randomly influenced by the parameter $\alpha$ and the physical constraints of the system.