ANNUAL REVIEWS

Annual Review of Control, Robotics, and Autonomous Systems

Data-Driven Predictive Control for Autonomous Systems

Ugo Rosolia, Xiaojing Zhang, and Francesco Borrelli

Department of Mechanical Engineering, University of California, Berkeley, California 94720-1740, USA; email: ugo.rosolia@berkeley.edu, xiaojing.zhang@berkeley.edu, fborrelli@berkeley.edu

Annu. Rev. Control Robot. Auton. Syst. 2018. 1:259–86

The *Annual Review of Control, Robotics, and Autonomous Systems* is online at control.annualreviews.org

https://doi.org/10.1146/annurev-control-060117-105215

Keywords

model predictive control, MPC, data-driven control, learning, stochastic predictive control

Abstract

In autonomous systems, the ability to make forecasts and cope with uncertain predictions is synonymous with intelligence. Model predictive control (MPC) is an established control methodology that systematically uses forecasts to compute real-time optimal control decisions. In MPC, at each time step an optimization problem is solved over a moving horizon. The objective is to find a control policy that minimizes a predicted performance index while satisfying operating constraints. Uncertainty in MPC is handled by optimizing over multiple uncertain forecasts. In this case, performance index and operating constraints take the form of functions defined over a probability space, and the resulting technique is called stochastic MPC. Our research over the past 10 years has focused on predictive control design methods that systematically handle uncertain forecasts in autonomous and semiautonomous systems. In the first part of this article, we present an overview of the approach we use, its main advantages, and its challenges. In the second part, we present our most recent results on data-driven predictive control. We show how to use data to efficiently formulate stochastic MPC problems and autonomously improve performance in repetitive tasks. The proposed framework is able to handle a large set of predicted scenarios in real time and learn from historical data.

1. INTRODUCTION

Paraphrasing the definition given in Reference 1, a control system is a device in which sensed quantities are used to generate an autonomous behavior through computation and actuation. In autonomous systems, two assumptions can facilitate the control algorithm design: First, there is no human interaction with the system, and second, there is precise knowledge of the environment that the system interacts with. If both assumptions are satisfied, then control design for autonomous systems is challenged only by the system design itself.

Mass-produced autonomous systems are available today for scenarios satisfying both assumptions. Automated cars on a highway performing lane keeping or lane changing belong to this class of problems. In fact, these systems assume (*a*) that the driver does not touch the pedals or the steering wheel (otherwise, the functionality is disengaged) and (*b*) there is high confidence in the current and predicted positions of surrounding vehicles. These are good assumptions in highway scenarios. When predictions are incorrect (such as in abrupt cut-in scenarios), safety is ensured by low-level active control systems (such as emergency brakes) and the requirement that the driver should pay attention to the road.

DJI drones autonomously moving between two points or landing at the takeoff point is another example of an autonomous functionality delivered in a mass-produced system. During the maneuver, the human does not intervene unless he or she wants to take over. Moreover, the drone uses a simple, obstacle-free environment model that is used to predict only the distance to the ground. Obstacle avoidance is left to the human, who can take over at any time. Other mass-produced automated robots include the iRobot Roomba cleaning robots, painting robots used in car assembly lines, and robots used in semiconductor manufacturing.

This article focuses on the challenges of control design when one or both of the assumptions given above are not satisfied—i.e., when there is interaction between the autonomous systems and a human and/or there is high uncertainty in the environment that the system moves in and interacts with. Walking robots, for example, are difficult to design and control (2, 3). Walking robots that can operate in an unknown environment containing obstacles, slippery surfaces, and doors while handling objects of unknown weight and unknown surface friction are the holy grail of the robotics industry. Autonomous cars in urban driving and autonomous drones in an unknown hostile environment belong to the same class of problems, as they are characterized by high uncertainty in the environment that the system moves in and interacts with. Such problems are challenging, interesting for the academic community, and practically relevant from an industrial point of view.

Human interaction with an autonomous system represents another challenge. In a world with increasing automation, interaction with robots becomes inevitable. An "intelligent" robot needs to predict and understand human intention. Wrong interaction between humans and machines can be fatal.

Undoubtedly, an effective control design for the class of problems described above requires forecasts of human actions and the environment state. Forecasts can be uncertain, and thus the control design can be very challenging. Without forecasts, autonomous systems are slow and dumb. Recently, the intelligence of AlphaGo has been connected to the ability to predict and learn from 30 million human moves (4). Many researchers have embraced the paradigm that the more quickly and accurately a machine can forecast, the smarter it is.

One key challenge is undoubtedly related to the design of the forecasting algorithms. For instance, it is commonly believed that reconstructing a scene in urban driving and predicting the motion of all of its classified objects is the most challenging part of the autonomous driving problem. However, a badly designed control algorithm can easily lead to undesired behavior—e.g.,

as if AlphaGo were able to play billions of games in its "brain" but then played the wrong move in real time. The control design becomes even more challenging if all of the forecasts need to be analyzed in milliseconds on a platform with limited computational resources.

Our research over the past 10 years has focused on control design methods for platforms with limited computational resources that systematically handle uncertain forecasts. Forecast signals provide information about the environment that the system moves in and interacts with, including potential human interaction. In this article, we present the unified framework that we have been using in several applications. In the proposed formulation, uncertainty is introduced in the modeling phase and possibly learned from data. Models are used to make forecasts, and forecasts are used by the autonomous system control algorithm to make decisions in real time. The goal is to provide an overview of the predictive control framework, point to relevant literature, and focus on only one relevant aspect of our recent results: the use of data to improve the safety and performance of the resulting autonomous systems.

As will become clear, data enable two results. First, one can design controllers that are robust to uncertain forecasts without analytically propagating the uncertainty model over the prediction horizon. This results in a simple and effective technique when a large amount of data is available. Second, one can improve controller performance and safety when data from the same or similar tasks are available. Predictive control can easily incorporate the most relevant information contained in past data and use it to continually improve its performance while satisfying operating constraints.

The article is organized as follows: In Section 2, we recall the basic idea behind the predictive control framework. The control design challenges associated with imprecise forecasts of human actions and knowledge of the environment are discussed in Section 3. In Section 4, we describe the data-driven stochastic model predictive control (MPC) framework that exploits data to guarantee safety. Learning model predictive control (LMPC), a framework used to improve the closed-loop performance of a system performing the same task over and over, is discussed in Section 5. Finally, in Section 6, we test the data-driven control frameworks on a simple motivating example.

Some of the material in this article derives from our previous publications. In particular, Section 1 derives from the introduction of the book *Predictive Control for Linear and Hybrid Systems* (5), Section 4 derives from work by Zhang and colleagues (6, 7), and Section 5 derives from articles by Rosolia & Borrelli (8, 9).

2. DYNAMIC OPTIMIZATION AND PREDICTIVE CONTROL

Control design for autonomous systems can often be formulated as a dynamic optimization problem. The basis of a dynamic optimization problem is a dynamic model for the system to automate. For example, $x_{k+1} = f(x_k, u_k, d_k, b_k, e_k),\ x_0 = x(0)$, is a commonly used model and describes the evolution of the state $x_k \in \mathbb{R}^{n_x}$ with time, starting from the initial condition $x(0)$, as it is affected by the manipulated input $u_k \in \mathbb{R}^{n_u}$. The function $f(\cdot)$ here can be an arbitrary nonlinear function, while $d_k \in \mathbb{R}^{n_d}$, $b_k \in \mathbb{R}^{n_b}$, and $e_k \in \mathbb{R}^{n_e}$ are the disturbance, the human action, and the state of the environment at time k, respectively.

The goal is to find a sequence of manipulated inputs $U_T = \{u_0, \ldots, u_{T-1}\}$ that optimizes a given objective function $\sum_{k=0}^{T-1} \ell(x_k, u_k)$ over the autonomous task duration T:

$$\begin{aligned} \min_{u_0,\ldots,u_{T-1}} \quad & \sum_{k=0}^{T-1} \ell(x_k, u_k) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t, d_t, b_t, e_t), \quad \forall t = 0, 1, \ldots, T-1, \\ & x_0 = x(0),\ x_T = x_F, \end{aligned}$$

$$
\begin{aligned}
& g(x_t, u_t, h_t, e_t) \le 0, \quad \forall t = 0, 1, \ldots, T-1, \\
& d_0, h_0, e_0 \ldots, d_T, h_T, e_T \text{ given from forecast models.}
\end{aligned} \tag{1.}
$$

In Equation 1, $x(0)$ and x_F are the initial and terminal goal states, respectively. The function $g(\cdot) \le 0$ describes the state and input constraints that must be satisfied during the control task. Almost all practical autonomous control design problems can be put into this form, and a large number of algorithms and software packages are available to determine the optimal solution vector $U_T^* = \{u_0^*, \ldots, u_{T-1}^*\}$, also known as the optimizer. Indeed, various algorithms exist that exploit the structure of a particular problem (e.g., linearity and convexity), allowing even large problems that are described by complex models and involve many degrees of freedom to be solved efficiently and reliably.

There are two difficulties associated with this idea. First, a long duration T of the autonomous task can easily make it infeasible to solve Equation 1 in real time. Second, in practice the sequence of inputs $u_0, u_1, \ldots$, which is obtained by the procedure described above, cannot be simply applied. This is because the model $f(\cdot)$ used to predict the evolution of the real system can be inaccurate and because the prediction of the external disturbances d_k, human actions h_k, and environment states e_k are uncertain and often wrong in the far future. To alleviate both issues, it is common practice to (*a*) predict over a horizon N shorter than T and (*b*) continuously measure the state of the system (say, once every time step) and then recompute new control sequences with updated information on the disturbance, human action, and environment forecasts. The procedure described above is commonly referred to as model predictive control (MPC) (5, 10–14). At the generic time t, an MPC controller solves the following problem:

MPC controller: the control law represented in Equation 3, where $u_0^*(x(t))$ is the first component of the solution to the optimization problem represented in Equation 2 solved at time t

$$
\begin{aligned}
\min_{u_0, u_1, \ldots, u_{N-1}} \quad & \sum_{t=0}^{N-1} \ell(x_t, u_t) + Q(x_N) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t, d_t, h_t, e_t), \quad \forall t = 0, 1, \ldots, N-1, \\
& x_0 = x(t), \\
& g(x_t, u_t, h_t, e_t) \le 0, \quad \forall t = 0, 1, \ldots, N-1, \\
& g_N(x_N, h_N, e_N) \le 0, \\
& d_0, h_0, e_0 \ldots, d_N, h_N, e_N \text{ given from forecast models,}
\end{aligned} \tag{2.}
$$

where $x(t)$ is the measured state at time t.

Let $U_N^* = \{u_0^*(x(t)), \ldots, u_{N-1}^*(x(t))\}$ be the optimal solution of Equation 2 at time t. Then the first element of U_N^* is applied to the system:[1]

$$
u(t) = u_0^*(x(t)). \tag{3.}
$$

At the next time step $t + 1$, the optimization problem represented in Equation 2 is solved again based on the new state $x_0 = x(t + 1)$, yielding a moving or receding-horizon control strategy.

Compared with Equation 1, Equation 2 is solved over a shorter horizon N and uses a terminal cost $Q(\cdot)$ and terminal constraint $g_N(\cdot)$ to approximate costs and constraints from time N to time T. The choice and the role of $Q(\cdot)$ and $g_N(\cdot)$ are critical in MPC design and are discussed at length in Section 3.1.

[1] We use $u_i^*(x(t))$ to emphasize that the optimal solution depends on the current state $x(t)$. Later in the article, whenever the meaning is obvious, we use the simpler notation u_i^*.

It is important to distinguish between (*a*) the real state $x(t)$ and input $u(t)$ of the system at time t and (*b*) the predicted states x_t and inputs u_t in the optimization problem. Indeed, a more complex notation is often used, where one differentiates between (*a*) the state $x_{j|t}$ at time j predicted at time t and (*b*) the state $x_{j|t+1}$ at time j predicted at time $t+1$. Below, we use the complex notation whenever necessary.

In summary, the sequences $\{x_0, \dots, x_N\}$, $\{e_0, \dots, e_N\}$, and $\{h_0, \dots, h_N\}$ represent the forecasts of the state, environment, and human model from time t to time $t+N$. These forecasts are used by the controller to plan the optimal control sequence $\{u_0^*, \dots, u_{N-1}^*\}$ over the finite horizon N. At each time step t, only the first element of the planned sequence u_0^* is applied to the system.

Before we dive into further technical details, we will use Equation 2 to comment on practical considerations, advantages, and disadvantages of this approach.

2.1. System, Environment, and Human Modeling

Autonomous system models provide a relationship between physical inputs and the states of the system and/or its parts. Choosing the right system model is often an art and is highly dependent on the task complexity as well as the availability of computation power. For real-time computation on platforms with limited resources, for example, objects are often simplified and treated as point masses, and simple kinematic models are preferred over complex dynamic models even if the latter are more accurate.

Environment models are necessary for any autonomous system that requires the ability to navigate and/or manipulate objects. As in system models, the complexity is tightly coupled with the autonomous system task. For instance, autonomous cars on highways do not need models and forecasts for pedestrians, bicycles, or traffic lights. Similarly, painting robots do not need object models if they grasp the same tool and paint the same surface over and over again. Roughly speaking, based on their type of prediction, environment models can be classified into deterministic models, stochastic models, and scenario-based models. Deterministic models provide a single (usually most likely) prediction trajectory. Such models generally cannot capture the significant uncertainty associated with different actions of the agents in the environment, especially over longer periods of time. Stochastic models often rely on standard probability distribution functions to model elements of the environment. They are often implemented by using a set of standard distributions, such as Gaussian distributions or Gaussian mixtures. The general challenge is the difficulty of including interactions between agents in the environment, especially if such interactions are described by a set of rules. Scenario-based models can overcome these limitations by not stating these probability distributions explicitly (15, 16). Rather, the uncertainty is described via a discrete number of possible future scenarios that may be drawn directly from collected real-world data and do not require the fitting of a probabilistic model.

In the case of assisted or interactive automation, it is important to anticipate humans' actions in order to assist their decision and control, giving rise to the need for human models. For example, a collision-avoidance system needs to predict whether a driver will brake or steer in time to avoid an obstacle in order to decide whether to intervene or not (17). Similarly, in robotics, an automated system that must hand over an object to a human needs a model that describes how humans select a grasping configuration to avoid harm (18, 19). Modeling human interaction is often challenging because the actual physical process involves human perception, information processing, decision-making, and physical action execution, all of which are extremely complex to model and not yet fully understood. Several approaches have been proposed to approximate the true process and reduce its complexity. This field is currently receiving a lot of

interest from the automatic control community (20, 21) because control-oriented models of humans can be used by the autonomous systems to deliver better interaction. A thorough literature review goes beyond the scope of this article since the models vary across different fields of robotics. For more details, we refer the interested reader to Reference 22 and the references therein.

Remark 1. In this article, we lump environment and human states into a single variable, e_t, and use $f(\cdot)$ and $m(\cdot)$ to denote the system and environment state update functions, respectively, i.e.,

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, e_k, d_k), \\ e_{k+1} &= m(x_k, u_k, e_k, d_k). \end{aligned} \tag{4.}$$

In Equation 4, d_k describes the uncertainty, which we model as a random variable whose support $\mathcal{D}$ we assume is time invariant.

2.2. Constraint Formulation and Guarantees

A fundamental advantage of using the control formulation represented in Equations 2 and 3 is that system constraints can easily be incorporated in the design stage as $g(x_t, u_t, h_t, e_t) \leq 0$. However, two well-known facts must be considered in this context. First, constraint satisfaction at a given time t does not automatically guarantee constraint satisfaction at time $t + 1$, even in the case of perfect models and forecasts. This issue is known as persistent feasibility, and we will return to it in Section 3.1. Second, with the exception of very simple constraints, the function $g(\cdot)$ in Equation 2 must be formulated in a standard form that is accepted by general-purpose numerical solvers. This can be accomplished either by a software parser [such as YALMIP (23) or CVX (24)] or by the control engineer, which is often computationally more efficient since it is tailored for the specific problem at hand. As an example, let us consider a navigation problem where an autonomous system needs to go from point A to point B while avoiding obstacles along its way. Specifically, if $\mathcal{E}(x_t) \subset \mathbb{R}^3$ is the space occupied by the system at time t, and $\mathcal{O} \subset \mathbb{R}^3$ is the space occupied by the obstacle, then the obstacle-avoidance constraint is given by

$$\mathcal{E}(x_t) \cap \mathcal{O} = \emptyset, \qquad \forall t = 0, 1, \ldots. \tag{5.}$$

Although the constraint represented in Equation 5 describes the obstacle-avoidance problem, Equation 5 cannot be implemented as such since it does not come in a standard form accepted by conventional numerical solvers and must be reformulated. One way of reformulating it is to introduce auxiliary binary variables $\delta_i \in \{0, 1\}$. This method is particularly attractive for linear systems with polytopic constraints since in this case a mixed-integer linear program can be solved using mature off-the-shelf solvers. For nonlinear systems, however, reformulating Equation 5 using binary variables should generally be avoided since mixed-integer nonconvex programs are numerically difficult to handle. To alleviate this issue, one can reformulate Equation 5 as a smooth, albeit nonconvex, constraint function that can be handled with off-the-shelf nonlinear optimization solvers. For example, Zhang et al. (25) recently showed that, if $\mathcal{E}(x_t)$ and $\mathcal{O}$ are polytopes or ellipsoids (or can be decomposed into a finite union of such sets), then the strong duality of convex optimization can be used to reformulate Equation 5 as $\tilde{g}(x_t, \lambda_t) \leq 0$, where λ_t represents auxiliary variables and $\tilde{g}(\cdot)$ is a smooth nonconvex function that can be handled by nonlinear solvers such as IPOPT (Interior Point Optimizer) (26).

2.3. Policy Search

Equation 2 assumes perfect forecasts and looks for a single sequence $\{u_0, \ldots, u_{N-1}\}$ that optimizes the cost and satisfies the constraints for such nominal forecasts. In Section 3, we consider the more general case of uncertain forecasts. In this case, the optimization scheme looks for feedback policies

$$u_t = \pi_t(x_t, e_t)$$

in order to take into account that the control scheme can use a different input sequence for each different forecast. In this case, the cost function and the constraints need to be reformulated in a stochastic or robust sense. Section 3.3 provides more details.

Feedback policy: a function that computes the input given the current system and environment state; also known as a control policy

2.4. Model Predictive Control–Like Approaches

The model-based design principle of Equations 2 and 3 can be found across various research communities working on autonomous systems. Often, the approaches are not called MPC, and the disciplined formalism of Equations 2 and 3 is not employed. For instance, Mueller et al. (27, 28) "solved" Equation 2 for a drone by first heuristically generating thousands of sample trajectories obtained from a model of the form shown in Equation 4 with suitably chosen control inputs, and then picking the best according to a user-defined cost function. The first input of the best trajectory is applied to the system, resulting in a receding-horizon control scheme. A similar idea is the rapidly exploring random trees (RRT) algorithm, which has found its way into motion-planning problems (29, 30). Roughly speaking, RRT "solves" Equation 2 by using a very basic system model to generate a tree of possible solutions from which the best trajectory is chosen. The control actions are then usually performed using a simple proportional–integral–derivative (PID) controller whose goal is to track the chosen trajectory.

Compared with the above-mentioned heuristics, the disciplined approach of Equations 2 and 3 is appealing owing to its conceptual simplicity, especially when it comes to the control design phase. In addition, several mature numerical algorithms exist to efficiently solve Equation 2. When the system is linear and the cost and constraints are convex, Equation 2 is a convex optimization problem and can be solved efficiently and reliably using off-the-shelf numerical solvers, such as the Gurobi Optimizer (31) and MOSEK (32). When the system model is nonlinear and/or the constraints or objective function is nonconvex (as is the case in the obstacle-avoidance problem represented in Equation 5), Equation 2 requires the solution of a nonconvex optimization problem, which can be challenging in general. In addition, solving Equation 2 using general-purpose nonlinear, nonconvex numerical solvers in real time is often seen as unorthodox by traditional control engineers, as one typically has no direct control over how those solvers operate and cannot influence their behavior based on physical considerations.

We close this section by pointing out that one should not underestimate the advantage of having a simple algorithm that does not require additional numerical routines to solve a control design problem. This is the power of the above-mentioned heuristics (such as RRT and A*) compared with MPC. In addition, one should consider the fact that, in the nonconvex case, neither the formal optimization-based approach nor those heuristic approaches have practically useful bounds on the convergence time to compute the control solution.

3. STOCHASTIC PREDICTIVE CONTROL

We begin by detailing one possible formulation of the MPC represented in Equations 2 and 3 for the model represented in Equation 4. Specifically, let us assume that the uncertainties $d_0, d_1, \ldots$

are random variables defined on an abstract probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and consider the following finite-time optimal control problem at time t:

$$\min_{\pi_{t|t}(\cdot),\ldots,\pi_{t+N-1|t}(\cdot)} \quad \sum_{k=t}^{t+N-1} \mathbb{E}\left[\ell(x_{k|t}, u_{k|t})\right] + \mathbb{E}\left[Q(x_{t+N|t})\right] \tag{6a}$$

$$\text{subject to} \quad x_{k+1|t} = f(x_{k|t}, u_{k|t}, e_{k|t}, d_{k|t}), \qquad \forall k = t, \ldots, t+N-1, \tag{6b}$$

$$e_{k+1|t} = m(x_{k|t}, u_{k|t}, e_{k|t}, d_{k|t}), \qquad \forall k = t, \ldots, t+N-1, \tag{6c}$$

$$x_{t|t} = x(t),\ e_{t|t} = e(t), \tag{6d}$$

$$u_{k|t} = \pi_{k|t}(x_{k|t}, e_{k|t}), \qquad \forall k = t, \ldots, t+N-1, \tag{6e}$$

$$\mathbb{P}[g(x_{k|t}, u_{k|t}, d_{k|t}, e_{k|t}) \le 0] \ge 1-\epsilon, \quad \forall k = t, \ldots, t+N-1, \tag{6f}$$

where $\mathbb{E}[\cdot]$ is the expectation with respect to $\mathbb{P}[\cdot]$, $\epsilon \in (0, 1)$ is the so-called violation probability, and $\mathbb{P}[g(x_{k|t}, u_{k|t}, d_{k|t}, e_{k|t}) \le 0]$ denotes the probability that the constraint $g(x_{k|t}, u_{k|t}, d_{k|t}, e_{k|t}) \le 0$ is satisfied. Equation 6 differs from Equation 2 in that the disturbance d_k is no longer assumed to be perfectly known; instead, it is a random variable that gives rise to the expected-value cost represented in Equation 6a and the chance constraint represented in Equation 6f.

As before, $x_{t+k|t}$ denotes the state vector at time $t+k$ predicted at time t obtained by starting from the current state $x_{t|t} = x(t)$ and applying $d_{t|t}, \ldots, d_{t|k-1}$ to the system and environment models and the input sequence $u_{t|t}, \ldots, u_{t+N-1|t}$. The symbol $x_{t+k|t}$ is often read as "the state x at time $t+k$ predicted at time t." Similarly, $u_{t+k|t}$ is read as "the input u at time $t+k$ computed at time t." For instance, $x_{3|1}$ represents the predicted state at time 3 when the prediction is done at time $t = 1$ starting from the current state $x(1)$. It is different, in general, from $x_{3|2}$, which is the predicted state at time 3 when the prediction is done at time $t = 2$ starting from the current state $x(2)$. The uncertainty d and the environment/human model were introduced in Section 2 and Remark 1.

Given the optimal feedback policy $[\pi^*_{t|t}(\cdot), \ldots, \pi^*_{t+N-1|t}(\cdot)]$, the first input

$$u_t = \pi^*_{t|t}(x_{t|t}, e_{t|t}) \tag{7}$$

is applied to the system. At the next time step $t+1$, Equation 6 is resolved based on the new state measurement x_{t+1}, yielding a receding-horizon control strategy. This strategy introduces feedback into the system that can correct the inaccuracy of the system and disturbance model.

Equation 6 is difficult to solve in general because it requires (*a*) a finite-dimensional parameterization of the control polices $\pi(\cdot)$, (*b*) an efficient propagation of the uncertainty d over the system dynamics and the translation of the probabilistic constraints into deterministic constraints, and (*c*) the solution of the resulting mathematical optimization problem. With the exception of linear systems and special classes of distributions (e.g., normal distributions), these steps are nontrivial. In the following, we discuss the design challenges associated with Equation 6 in the context of autonomous systems and provide an overview of existing strategies to address these steps that allow for practical implementation.

3.1. Feasibility and Optimality with Short Horizons

The predictive controller represented in Equations 6 and 7 plans the system's trajectory over the finite time windows of length N, which is usually much smaller than the task duration T in Equation 1 owing to the availability of reliable forecasts and computational resources. For a short N, the controller takes only shortsighted control actions, which may be unsafe or result in poor closed-loop performance. For instance, in autonomous racing, a predictive controller that plans the vehicle's trajectory over a short horizon without accounting for an upcoming curve may

accelerate to the point that safe turning becomes infeasible. In this example, the short planning horizon would force the controller to violate the safety constraints at a certain time instant. Avoiding such situations requires designing recursively feasible controllers, i.e., controllers that are feasible at all time instants despite a short prediction horizon and the presence of uncertainty. This in turn requires designing a terminal set that guarantees the existence of a feasible state and control sequence beyond the prediction horizon.

Recursively feasible controller: a controller that fulfills the constraints at all time instants

Shortsighted control actions may also result in poor closed-loop performance. For instance, a predictive controller for an autonomous agent trying to escape a maze using the shortest path may easily make a suboptimal decision if the prediction horizon is too short. A commonly used solution is to introduce a terminal cost $Q(\cdot)$ and terminal constraint $g_N(\cdot)$ in Equation 2 in order to approximate the cost and constraints from time N to time T of the original Equation 1. The choice and role of $Q(\cdot)$ and $g_N(\cdot)$ are critical in any MPC design; properly chosen $Q(\cdot)$ and $g_N(\cdot)$ ensure feasibility and optimality despite a short-horizon N.

In particular, $g_N(\cdot)$ should be designed such that x_N must belong to a control invariant set $\mathcal{X}_T \subset \mathbb{R}^n$. Control invariant sets are sets of initial states for which there exists a controller $u_k = v(x_k)$ such that system constraints are never violated. Moreover, the terminal function $Q(\cdot)$ must be a Lyapunov function over $\mathcal{X}_T$ for the autonomous system controlled by $u_k = v(x_k)$.

Control invariant sets and Lyapunov functions are very hard to compute for general nonlinear systems (5). In Section 5, we show that historical data can be used to compute both the control invariant set and control Lyapunov function, which guarantee that the MPC design is recursively feasible and does not take shortsighted control actions.

3.2. Policy Approximation

In the presence of uncertainty, it is important to distinguish between open-loop policies and closed-loop policies (5, chap. 15). While the latter are less conservative, they may lead to computationally intractable problems. In practice, compromises are made by parameterizing the control structure and optimizing only over these parameters. The following policy structures are common in the literature:

$$u_t = z_t \qquad \text{open-loop policy,} \tag{8a}$$

$$u_t = z_t + \bar{K} x_t \qquad \text{affine state-feedback policy,} \tag{8b}$$

$$u_t = z_t + \sum_{j=0}^{t-1} Z_{t,j}\, d_j \qquad \text{affine disturbance-feedback policy,} \tag{8c}$$

where the control parameters are $z_t \in \mathbb{R}^{n_u}$ in Equations 8a and 8b and $z_t \in \mathbb{R}^{n_u}$ and $Z_{t,j} \in \mathbb{R}^{n_u \times n_x}$ in Equation 8c. The objective is to determine the optimal parameters z_t and $Z_{t,j}$ that minimize the control objective and satisfy the constraints. Often, the feedback matrix $\bar{K}$ in Equation 8b is fixed to ensure computational tractability. It is easy to see that Equation 8a is a special case of Equation 8b, which in turn is a special case of Equation 8c if the system and constraints are linear (33).

3.3. Chance-Constraint Approximation

As discussed above, a key aspect in the solution of Equation 6 is the appropriate reformulation of the chance constraint. Roughly speaking, the existing approaches dealing with probabilistic constraints can be divided into two categories: cases when $\epsilon = 0$ (robust MPC) and cases when $\epsilon > 0$ (chance-constrained stochastic MPC). Next, we briefly review the main ideas.

Single linear chance constraint: a constraint that takes the form $e^\top x_k + f^\top u_k \leq g$, where e and f are vectors and g is a scalar

Multiple linear chance constraint: a constraint that takes the form $Ex_k + F^\top u_k \leq g$, where E and F are matrices and g is a vector.

3.3.1. Robust model predictive control. One way of satisfying the chance constraints is to require them to be satisfied with probability 1, i.e., $\mathbb{P}[g(x_{k|t}, u_{k|t}, e_{k|t}) \leq 0] = 1$. Under mild regularity assumptions, this is equivalent to requiring

$$g(x_{k|t}, u_{k|t}, e_{k|t}) \leq 0, \quad \forall d_{t|t}, \ldots, d_{k-1|t} \in \mathcal{D}, \tag{9.}$$

where $\mathcal{D}$ is the support of the random variable $d_{k|t}$ that affects both the system $x_{k|t}$ and the environment $e_{k|t}$. Since Equation 9 corresponds to a chance constraint when $\epsilon = 0$, it constitutes a conservative (i.e., inner) approximation of the original chance constraint in Equation 6. Despite its conservatism, robust MPC has received considerable attention because of its conceptual simplicity (information on distribution is neglected, and only the support $\mathcal{D}$ is needed) and because it finds application in safety-critical applications where $\epsilon \approx 0$ is required, and robust constraint satisfaction is a good approximation.

Robust MPC approaches can be roughly categorized based on their policy parameterization. The classical method is to use the state-feedback policy represented in Equation 8b, which results in a gradual tightening of the constraints (34–36). Another method, known as tube MPC, employs an input policy of the form $u_k = \bar{K}(x_k - \bar{x}_k)$, where $\bar{x}_k$ is some nominal state. This parameterization results in a constant constraint tightening along the horizon (13, 37). The main drawback of these approaches is that it is not clear how to choose the feedback matrix $\bar{K}$. To alleviate this issue, approaches based on disturbance-feedback policies of the form shown in Equation 8c have been proposed, which allows the optimal (linear) state-feedback matrix to be chosen (33, 38, 39). Compared with simple state-feedback policies, disturbance-feedback policies have the drawback of increased computational complexity, since the optimization is performed not only over the affine terms z_k but also over the feedback matrices $Z_{t,j}$. Regardless of the chosen policy parameterization, stability and recursive feasibility still need to be ensured by adding an appropriate terminal cost and terminal constraint.

3.3.2. Stochastic constraint satisfaction. As shown in the previous section, the stochastic MPC problem represented in Equation 6 can be conservatively approximated using the theory of robust MPC. The main limitation of this approach is its inherent conservatism, since robust MPC enforces the constraints against all uncertainty realizations, even the rare ones. Indeed, extreme disturbances often limit a controller's performance (in terms of objective function), but the probability of encountering them is often very low, and ignoring them may improve a controller's performance (40). These limitations can be partially overcome by directly reformulating the chance constraints in Equation 6. In the following, we review methods of dealing with chance constraints. To simplify the discussion, we restrict ourselves to linear systems with polyhedral constraints subject to the policy parameterizations in Section 3.2.

The simplest instance of chance constraint arises when d_k is a Gaussian random variable and we have single linear chance constraints. For this special case, the chance constraint can be exactly reformulated as a second-order cone constraint, which is convex and computationally tractable (41). A generalization of this approach is to release the assumption of d_k being Gaussian and consider only its first two moments. In this case, if single linear chance constraints are present, Chebyshev's inequality can be invoked, and the constraints can be reformulated again as second-order cone constraints (42–44). While this method is very general (it can be applied to all random variables with finite first- and second-order moments), it is conservative because only the first two moments of a distribution are considered and because it can be applied only toward single linear chance constraints. Methods based on polynomial chaos theory and Galerkin projection have recently been proposed as tools to handle multiple linear chance constraints (45). The main idea is to analytically propagate the uncertainty through the system dynamics using a series of basis

functions (46–48). The theory of this method can handle very generic constraints and distributions; its main challenge is establishing precise bounds on the approximation quality.

While multiple methods exist for approximating chance constraints, they are often limited to specific distributions (e.g., d is Gaussian) or single linear chance constraints (Chebyshev's inequality), or their approximation quality is unknown (polynomial chaos theory). In Section 4, we show how data-driven sampling-based methods can alleviate some of these limitations.

3.4. Summary: Control Design Challenges

In this section, we have reviewed the basic concepts of predictive control and discussed three challenges associated with it: (*a*) ensuring recursive feasibility and achieving optimality despite a short prediction horizon, (*b*) satisfying input and state constraints in the presence of uncertainty, and (*c*) ensuring computational tractability by properly reformulating constraints and costs and parameterizing control policies. There is no systematic and universal solution to the third challenge, and often the chosen approach is application dependent. In the following, we show how the first and second challenges can be addressed by using data. Specifically, in Section 4 we show how historical data can be used to address the computational challenges associated with the second challenge by using methods from randomized optimization. In Section 5, we show that when a controller repeatedly performs a task, the data from each task execution can be used to address the first challenge.

4. DATA-DRIVEN STOCHASTIC MODEL PREDICTIVE CONTROL

In this section, we show how historical data can be used to overcome the challenges associated with solving the stochastic MPC problem represented in Equation 6, as described in Section 3.3. In particular, we discuss the so-called scenario MPC, also known as randomized MPC, which is a data-driven method for addressing chance-constrained stochastic MPC problems of the form shown in Equation 6 (49–53).

4.1. Control Design

The main idea in scenario MPC is to use data (scenarios) to represent the random variable d_k; these data are then propagated through the system dynamics. More concretely, given the prediction horizon N, let us denote by $\mathbf{d}_t = [d_{t|t}, d_{t+1|t}, \ldots, d_{t+N-1|t}]$ the uncertainty along the prediction horizon N, and let $\{\mathbf{d}_t^{(1)}, \mathbf{d}_t^{(2)}, \ldots, \mathbf{d}_t^{(S)}\}$ be a collection of independent and identically distributed (i.i.d.) samples,[2] where S is called the sample size. The idea now is to propagate each sample $\mathbf{d}_t^{(i)}$ through the state and environmental dynamics represented in Equation 4, resulting in different scenarios of $[x_{t+1|t}^{(i)}, x_2^{(i)}, \ldots, x_{t+N|t}^{(i)}]$ and $[e_{t|t+1}^{(i)}, e_2^{(i)}, \ldots, e_{t+N|t}^{(i)}]$. The goal in scenario MPC is to find a control law that satisfies the system constraints for all sampled scenarios. Hence, the scenario MPC is given by

$$
\begin{aligned}
\min_{\pi_{t|t}(\cdot),\ldots,\pi_{t+N-1|t}(\cdot)} \quad & \sum_{k=t}^{t+N-1} \mathbb{E}[\ell(x_{k|t}, u_{k|t})] + \mathbb{E}[Q(x_{t+N|t})] \\
\text{subject to} \quad & x_{t|t}^{(ik)} = x(t),\ e_{t|t}^{(ik)} = e(t), \\
& x_{k+1|t}^{(ik)} = f(x_{k|t}^{(ik)}, u_{k|t}^{(ik)}, d_{k|t}^{(ik)}, e_{k|t}^{(ik)}),
\end{aligned}
$$

[2] In practice, samples often come from historical or simulated data.

$$
\begin{aligned}
& e^{(i_k)}_{k+1|t} = m(x^{(i_k)}_{k|t}, u^{(i_k)}_{k|t}, d^{(i_k)}_{k|t}, e^{(i_k)}_{k|t}), \\
& u^{(i_k)}_{k|t} = \pi_{k|t}(x^{(i_k)}_{k|t}, e^{(i_k)}_{k|t}), \\
& g(x^{(i_k)}_{k|t}, u^{(i_k)}_{k|t}, e^{(i_k)}_{k|t}) \leq 0, \\
& \forall i_k = 1, \ldots, S_k, \ \forall k = t, \ldots, t+N-1.
\end{aligned} \tag{10.}
$$

For technical reasons, different samples $d^{i_k}_{k|t}$ and sample sizes S_k are required for each prediction stage k. Specifically, for each stage k, a new set of samples $\{\mathbf{d}^{(1)}_{k|t}, \mathbf{d}^{(2)}_{k|t}, \ldots, \mathbf{d}^{(S_k)}_{k|t}\}$, each of cardinality S_k, is extracted, and the constraints associated with the kth stage need to be satisfied only for these samples. The scenario MPC problem represented in Equation 10 is highly intuitive and easy to implement in practice. Furthermore, since it relies only on extracted samples, the scenario MPC approach applies to any probability distribution and is entirely data driven.

Research on scenario MPC has focused mainly on three questions. First, how large do we need to choose the sample sizes S_k such that the solution of Equation 10 indeed satisfies the original chance constraints in Equation 6? This question is of great practical importance since a sample size that is too small may lead to solutions that do not satisfy the original chance constraint. Second, is it possible to implement scenario MPC in real time for systems with fast dynamics? And third, how can we guarantee persistent feasibility and stability when Equation 10 is implemented in a receding-horizon approach? In the following, we provide a brief overview of the first question; for results concerned with the second and third questions, we refer the interested reader to References 54–57 and Reference 52, respectively. To streamline the presentation, we ignore the environment model and consider the special case of linear time-invariant systems with probabilistic state constraints of the form

$$
x_{k+1} = Ax_k + Bu_k + Dd_k, \qquad \mathbb{P}_\delta\,[Gx_k \leq g] \geq 1 - \epsilon, \tag{11.}
$$

where the matrices A, B, D, G, and g are of suitable dimensions.

Scenario optimization, also known as random convex programming, concerns itself with approximating chance-constrained optimization problems of the form $\min_{x\in\mathbb{R}^n}\{c^\top x : \mathbb{P}_\delta[g(x,\delta) \leq 0] \geq 1-\epsilon\}$ by the scenario program $\min_{x\in\mathbb{R}^n}\{c^\top x : g(x,\delta^{(i)}) \leq 0, i = 1, \ldots, S\}$, where $\delta^{(i)}$ are i.i.d. samples randomly extracted according to $\mathbb{P}_\delta$ and S is the so-called sample size. In a learning context, the samples can be interpreted as training data. The scenario approach was originally developed by Calafiore & Campi (58, 59) and then later refined by Campi & Garatti (60) and Calafiore (61). Campi & Garatti (60) showed that, if the constraint function $g(\cdot,\delta)$ is convex for every fixed δ, then a sample size of $S \sim \mathcal{O}(\frac{n}{\epsilon})$ suffices to ensure that the solution of the scenario program is feasible for the original chance-constrained optimal control problem, where n is the dimension of the decision vector x. While the bound $S \sim \mathcal{O}(\frac{n}{\epsilon})$ is tight for the class of convex sampled problems, it can be improved by exploiting additional structural information of the underlying problem (61). This observation has been exploited by, among others, Zhang et al. (6, 62) and Schildbach et al. (63) to derive improved sample sizes tailored toward stochastic MPC problems, as reflected below in Theorems 1 and 2.

4.2. Controller Properties

One of the main challenges in scenario MPC is establishing the required sample sizes S_k such that the solution of the scenario MPC problem represented in Equation 10 satisfies the original stochastic MPC problem represented in Equation 6. Generally speaking, it is desirable to establish sample sizes S_k that are as small as possible, since unnecessarily large sample sizes not only render

the control policy overly conservative (if $S_k \to \infty$, then the solution of the scenario MPC problem converges to that of the robust MPC problem) but also has a negative impact on the computational complexity because each sample in Equation 6 generates a new set of constraints. Next, we show that, depending on the choice of input policy (affine state feedback or affine disturbance feedback; see Section 3.2), different sample sizes are required.

4.2.1. Linear state-feedback policies. Recall from Section 3.2 that affine state-feedback policies take the form $u_k = \bar{K}x_k + z_k$, where $\bar{K}$ is a fixed state-feedback matrix. Then Theorem 1 follows from corollary 5.10 in Reference 7.

Theorem 1. Consider the MPC problem represented in Equation 10 with the linear model represented in Equation 11 and the control law represented in Equation 3, and let $u_k = \bar{K}x_k + z_k$ for all $k = 1, \ldots, N$ and $\beta \in (0, 1)$. If the sample sizes S_k satisfy

$$S_k \geq \frac{2}{\epsilon}\left(\operatorname{rank}(G) - 1 + \log\frac{1}{\beta}\right), \qquad k = 1, \ldots, N, \tag{12.}$$

then, with probability at least $1 - \beta$, the solution of the scenario MPC problem represented in Equation 10 satisfies each chance constraint of Equation 6.

4.2.2. Linear disturbance-feedback policies. Recall from Section 3.2 that affine disturbance-feedback policies take the form $u_k = z_k + \sum_{j<k} Z_{k,j}d_j$, where z_k and $Z_{k,j}$ are parameters that need to be determined. Then we have the following sampling theorem from corollary 5.10 in Reference 7.

Theorem 2. Consider the MPC problem represented in Equation 10 with the linear model represented in Equation 11 and the control law represented in Equation 3, and let $u_k = h_k + \sum_{j<k} K_{k,j}d_j$ for all $k = 1, \ldots, N$ and $\beta \in (0, 1)$. If the sample sizes S_k satisfy

$$S_k \geq \frac{2}{\epsilon}\left(\zeta_k - 1 + \log\frac{1}{\beta}\right), \quad \text{where } \zeta_k = \operatorname{rank}(G)\left[(k-1)n_d + 1\right], \tag{13.}$$

and n_d is the dimension of the uncertainty d_k, then, with probability at least $1 - \beta$, the solution of the scenario MPC problem represented in Equation 10 satisfies each chance constraint of Equation 6.

Comparing Theorems 1 and 2, we find that the sample sizes required by the state-feedback policies are smaller than those required by the disturbance-feedback policies. Intuitively, this can be explained by the fact that, since disturbance-feedback policies are more general than state-feedback policies, they also need more training data to exhibit the same generalization property.

Theorems 1 and 2 are remarkable for two reasons. First, they do not require any assumptions about the distribution of the random variables d_k. The results are distribution free and data driven: The sample sizes hold irrespective of the distribution and do not require knowledge of the distribution of d_k, but only access to samples. Second, for a given violation probability ϵ, the sample sizes S_k depend logarithmically on the so-called confidence β. Therefore, the confidence of obtaining feasible solutions can be chosen to be very high (i.e., β can be chosen close to zero) without a negative impact.

5. LEARNING MODEL PREDICTIVE CONTROL

In this section, we consider iterative control problems where the controller must perform the same task repeatedly. We present the LMPC framework, which uses data from each task execution, often referred to as iteration, to improve the closed-loop performance of the original control problem represented in Equation 1. More detail on recent work on LMPC for autonomous systems can be found in References 8, 9, and 64–66.

5.1. Learning from Data

Consider the original control design problem represented in Equation 1 over the horizon T. In practice, every automated task has a finite duration T, and if the task is repeated at a subsequent iteration j, then its duration T_j might be different from the duration T_i at iteration i. For the sake of simplicity, we set $T = \infty$ in Equation 1 with the implicit understanding that the task is completed at time T_j of iteration j when some convergence condition is satisfied.

At each task iteration, the control input and the related closed-loop trajectory are recorded. In particular, at the jth iteration, the vectors

$$\mathbf{u}^j = [u^j(0),\ u^j(1), \ldots,\ u^j(t), \ldots] \quad \text{and} \quad \mathbf{x}^j = [x^j(0),\ x^j(1), \ldots,\ x^j(t), \ldots] \tag{14.}$$

collect the inputs applied to the system and the corresponding state evolution. In Equation 14, $x^j(t)$ and $u^j(t)$ denote the system state and the control input at time t of the jth iteration. We assume that at each jth iteration, the closed-loop trajectories start from the same initial state x_s, i.e.,

$$x^j(0) = x_S,\ \forall j \geq 0.$$

Next, we introduce the definitions of the sampled safe set and the iteration cost, both of which will be used below to guarantee the stability and feasibility of the LMPC framework.

5.1.1. Sampled safe set. The LMPC framework exploits the iterative nature of the control design. For every kth iteration that successfully steers the system to the terminal point x_F, the state trajectory $\mathbf{x}^k$ is a safe feasible trajectory. Thus, we define the sampled safe set $\mathcal{SS}^j$ at iteration j as

$$\mathcal{SS}^j = \left\{ \bigcup_{i \in M^j} \bigcup_{t=0}^{\infty} x^i(t) \right\}, \quad \text{with} \quad M^j = \left\{ k \in [0, j] : \lim_{t \to \infty} x^k(t) = x_F \right\}. \tag{15.}$$

$\mathcal{SS}^j$ is the collection of all state trajectories at iteration i for $i \in M^j$. M^j in the above equation is the set of indexes k associated with successful iterations k for $k \leq j$.

5.1.2. Iteration cost. At time t of the jth iteration, the cost to go associated with the closed-loop trajectory and input sequence represented in Equation 14 is defined as

$$J^j_{t \to \infty}(x^j(t)) = \sum_{k=t}^{\infty} \ell\left(x^j(k), u^j(k)\right), \tag{16.}$$

where $\ell(\cdot, \cdot)$ is the stage cost of Equation 1. We define the jth iteration cost as the cost represented in Equation 16 of the jth trajectory at time $t = 0$. $J^j_{0 \to \infty}(x^j_0)$ quantifies the controller performance at each jth iteration. We define the function $Q^j(\cdot)$, defined over the sample safe set $\mathcal{SS}^j$, as

$$Q^j(x) = \begin{cases} \min\limits_{(i,t) \in F^j(x)} J^i_{t \to \infty}(x), & \text{if } x \in \mathcal{SS}^j \\ +\infty, & \text{if } x \notin \mathcal{SS}^j \end{cases}, \tag{17.}$$

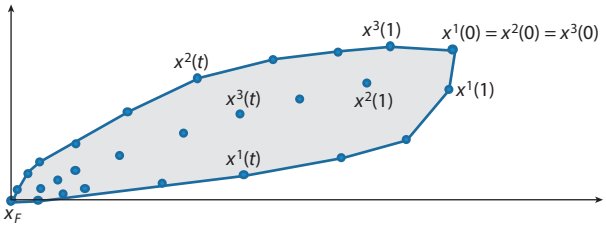

Figure 1

The convex hull of $\mathcal{SS}^j$. The circled dots represent the closed-loop trajectory in a two-dimensional phase plane. Three successful trajectories are shown, and their convex hull (*gray shaded area*) corresponds to the convex safe set $\mathcal{CS}^3$.

where $F^j(\cdot)$ is defined as

$$F^j(x) = \left\{(i,t) : i \in [0,j],\ t \geq 0 \text{ with } x^i(t) = x;\ \text{for } x^i(t) \in \mathcal{SS}^j\right\}.$$

The function $Q^j(\cdot)$ in Equation 17 assigns to every point in the sampled safe set, SS^j, the minimum cost to go along the trajectories in $\mathcal{SS}^j$.

5.1.3. Continuous relaxation. The sampled safe set $\mathcal{SS}^j$ and $Q^j(\cdot)$ are used in the next section to construct the LMPC. Since the sampled safe set $\mathcal{SS}^j$ is a set of discrete points and $Q^j(\cdot)$ is defined over a discrete domain, the LMPC is computationally challenging to solve. In this section, we introduce the continuous relaxation for both the sampled safe set and the $Q^j(\cdot)$ function, which can be used to reduce the computational burden associated with the LMPC. We define the convex safe set $\mathcal{CS}^j$ as the convex hull of $\mathcal{SS}^j$ (shown in **Figure 1**) and the function $P^j(\cdot)$ as the barycentric approximation of $Q^j(\cdot)$. These quantities are convex and can be used in the control design to speed up computations; for more details, we refer the reader to Reference 9.

5.2. Control Design

In this section, we present the design of the LMPC. We exploit the iterative nature of the control task to design a recursively feasible controller that guarantees a nonincreasing iteration cost [i.e., $J^j_{0\to\infty}(\cdot) \leq J^{j-1}_{0\to\infty}(\cdot)$]. Finally, we show that when the iteration cost converges to a steady-state value, the related controller is optimal for the overall control problem represented in Equation 1.

5.2.1. Learning model predictive control formulation. The LMPC solves at time t of iteration j the finite time-constrained optimal control problem

$$J^{\text{LMPC},j}_{t\to t+N}(x^j(t)) = \min_{u_{t|t},\ldots,u_{t+N-1|t}} \left[\sum_{k=t}^{t+N-1} \ell(x_{k|t}, u_{k|t}) + Q^{j-1}(x_{t+N|t})\right] \quad 18a.$$

subject to

$$x_{k+1|t} = f(x_{k|t}, u_{k|t}),\ \forall k \in [t,\ldots,t+N-1], \quad 18b.$$

$$g(x_{k|t}, u_{k|t}) \leq 0,\ \forall k \in [t,\ldots,t+N-1], \quad 18c.$$

$$x_{t+N|t} \in \mathcal{SS}^{j-1}, \quad 18d.$$

$$x_{t|t} = x^j(t), \quad 18e.$$

where Equations 18b and 18e represent the system dynamics and initial condition, respectively. The state and input constraints are given by Equation 18c. Finally, Equation 18d forces the

terminal state into the set $\mathcal{SS}^{j-1}$ defined in Equation 15. Let

$$\mathbf{u}^{*,j}_{t:t+N|t} = [u^{*,j}_{t|t}, \ldots, u^{*,j}_{t+N-1|t}],$$
$$\mathbf{x}^{*,j}_{t:t+N|t} = [x^{*,j}_{t|t}, \ldots, x^{*,j}_{t+N|t}],$$

be the optimal solution of Equation 18 at time t of the jth iteration and $J^{\text{LMPC},j}_{t\to t+N}(x^j_t)$ be the corresponding optimal cost. Then, at time t of the iteration j, the first element of $\mathbf{u}^{*,j}_{t:t+N|t}$ is applied to the system

$$u^j(t) = u^{*,j}_{t|t}. \tag{19.}$$

The finite-time optimal control problem represented in Equation 18 is solved at time $t+1$, based on the new state $x_{t+1|t+1} = x^j(t+1)$.

At iteration $j = 1$, we assume that $\mathcal{SS}^{j-1} = \mathcal{SS}^0$ is a nonempty set and that the trajectory $\mathbf{x}^0 \in \mathcal{SS}^0$ is feasible and convergent to x_F. Furthermore, we assume that the stage cost, $\ell(\cdot,\cdot)$, in Equation 1 is continuous and satisfies

$$\ell(x_F, 0) = 0 \text{ and } \ell(x^j(t), u^j(t)) \succ 0 \; \forall \; x^j(t) \in \mathbb{R}^n \setminus \{x_F\}, u^j(t) \in \mathbb{R}^m \setminus \{0\},$$

where the final state x_F is assumed to be a feasible equilibrium for the unforced system.

In the next section, we present the main fundamental properties of the proposed schema: The LMPC represented in Equations 18 and 19 in a closed loop with Equation 18b guarantees recursive feasibility and stability, and the iteration cost does not increase at each iteration.

5.3. Controller Properties

In this section, we analyze the controller properties. We show that the information from the previous trials, embedded in SS^j and $Q^j(\cdot)$, allow us to guarantee safety and performance improvement for the closed-loop system.

5.3.1. Recursive feasibility and stability. As mentioned in Section 5.1, for every point in the set $\mathcal{SS}^j$ there exists a control sequence that can drive the system to the terminal point x_F. The properties of $\mathcal{SS}^j$ and $Q^j(\cdot)$ guarantee the recursive feasibility and asymptotic stability of the equilibrium point x_F, as stated in the theorem below.

> **Theorem 3.** Consider Equation 18b controlled by the LMPC controller represented in Equations 18 and 19. Let $\mathcal{SS}^j$ be the sampled safe set at iteration j as defined in Equation 15. Then the LMPC represented in Equations 18 and 19 is feasible for all $t \geq 0$ and at every iteration $j \geq 1$. Moreover, the equilibrium point x_F is asymptotically stable for the closed-loop system at every iteration $j \geq 1$.

5.3.2. Convergence properties. Assume that the LMPC represented in Equations 18 and 19 converges to a steady-state trajectory. We can make two statements. First, the jth iteration cost $J^j_{0\to\infty}(\cdot)$ does not increase as j increases. Second, when the learning process has reached steady state (i.e., the behavior of the controller between two consecutive iterations does not change), the controller actions are optimal with respect to the original control problem represented in Equation 1.

> **Theorem 4.** Consider Equation 18b in a closed loop with the LMPC controller represented in Equations 18 and 19. Let $\mathcal{SS}^j$ be the sampled safe set at the jth iteration

as defined in Equation 15. Then the iteration cost $J^j_{0\to\infty}(\cdot)$ does not increase with the iteration index j.

Theorems 3 and 4 show that the proposed control strategy is able to learn from data. Indeed, the experience from the previous trials is used to compensate for the limited forecast availability. The controller uses the information given from the sampled safe set $\mathcal{SS}^j$ and the $Q^j(\cdot)$ function to forecast the evolution of the system in the environment beyond the prediction horizon. This precious information allows the controller to not take shortsighted control action and to improve the closed-loop performance at each task execution.

Finally, it is possible to show that if the learning processes have converged to a steady-state solution, then the controller has learned the best strategy to perform the overall control problem represented in Equation 1 (theorem 3 in Reference 8).

6. AN ILLUSTRATIVE EXAMPLE

In this section, we illustrate the presented control framework using a simple inverted pendulum example, as depicted in **Figure 2**. The system has two controllable inputs: a longitudinal acceleration a_k that is applied to the cart, and a torque T_{c_k} that is applied to the pendulum bar.

This simple system can be used for a fully autonomous demonstration (in this case, a_k and T_{c_k} are the output of the system's controller) or for a semiautonomous control demonstration. In the latter case, the pendulum cart is held by a human, who controls the acceleration a_k, while the system controller manipulates T_{c_k}. In both autonomous and semiautonomous modes, the goal is to move the pendulum in an upright position from point A to point B (**Figure 3**). In semiautonomous mode, the controller assists the human in keeping the pendulum upright by applying only a torque T_{c_k}.

The interested reader can find additional applications of the discussed approaches (scenario MPC and LMPC) to complex real-world examples in References 56, 57, 64, and 66–69.

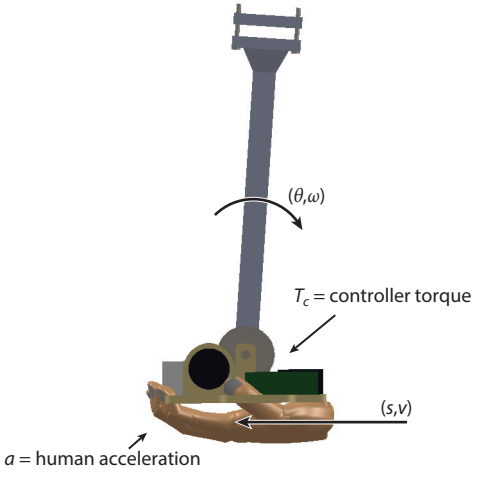

Figure 2

Scheme of the autonomous system used in our illustrative example. An inverted pendulum with an assistive motor is held in the palm of the human's hand.

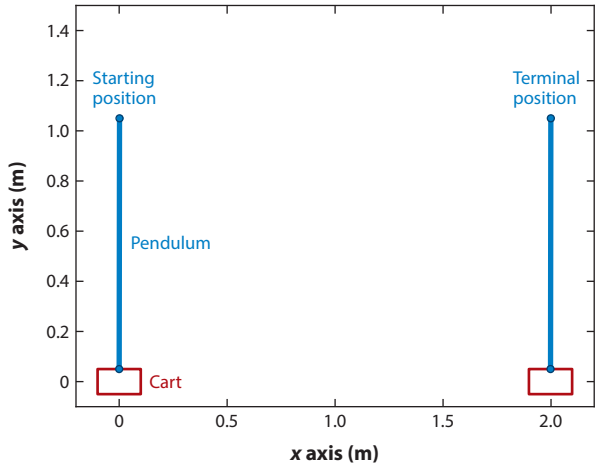

Figure 3

Initial and final goal configurations in our illustrative example. A blue line represents the pendulum, and a red box represents the cart.

6.1. System Modeling

With reference to **Figure 2**, let θ be the angle of the pendulum ($\theta = 0$ if the pendulum points downwards, and $\theta = \pi$ if the pendulum is upright), $\omega = \dot{\theta}$, s be the horizontal position of the cart, $v = \dot{s}$, a be the acceleration applied to the cart, and T_c be the controller torque.[3] The basic principles of physics mean that the dynamics are then given by $\dot{\omega} = \omega + \frac{1}{ml^2}(T_c - mgl\sin(\theta) + mal\cos(\theta) - bv)$, where l is the length of the pendulum, m is the mass on its pinpoint, g is the gravitational constant, and b is the viscous friction coefficient. Choosing $x = (\omega, \theta, v, s)$ as the state vector and employing a forward Euler discretization with time step dt yields the following state-space representation:

$$x_{k+1} = f(x_k, T_{c_k}, a_k) = \begin{bmatrix} \omega_k + \frac{dt}{ml^2}(T_{c_k} - mgl\sin(\theta_k) + ml\cos(\theta_k)a_k - bv_k) \\ \theta_k + \omega_k dt \\ v_k + a_k dt \\ s_k + v_k dt \end{bmatrix}. \quad 20.$$

For the semiautonomous case, the cart position and the position of the human hand that moves the cart coincide. Also, a_k is the acceleration resulting from the human hand, and T_{c_k} is the controller torque. In this notation, a_k can be interpreted as the environment variable e_k used in Equation 4.

6.2. Keeping the Pendulum Upright Using Stochastic Model Predictive Control

In this section, we focus on the semiautonomous case and assume that the controller does not know the exact human movement a, but only that it lies between -6 and 14 m/s^2. The human attempts to move the pendulum from its initial position $s = 0$ to its final position $s = 2$ m, while the controller attempts to keep the pendulum upright—i.e., θ should be $180° \pm 1.5°$ as often as possible. The simulation is terminated as soon as the pendulum has reached $s = 2$ m. For the purposes of this illustration, we assume that a is uniformly distributed, i.e., $a \sim \mathcal{U}[-6, 14]$, and

[3] If x is a variable that depends on time, then $\dot{x}$ denotes its derivative with respect to time.

that the initial state of the system is $x(0) = (0, \pi, 0, 0)$. The discretization time is $dt = 50$ ms and the prediction horizon $N = 5$, which reflects the fact that it is often difficult in practice to accurately predict the human's behavior for a long time. The stochastic MPC problem is now given by

$$\begin{aligned} \min_{T_{c_{t|t}}, \ldots, T_{c_{t+N-1|t}}} \quad & \sum_{k=t}^{t+N-1} T_{c_{k|t}}^2 \\ \text{subject to} \quad & x_{t|t} = x(t), \\ & x_{k+1|t} = f(x_{k|t}, T_{c_{k|t}}, a_{k|t}), \\ & \mathbb{P}[(180^\circ - 1.5^\circ) \leq \theta_{k+1|t} \leq (180^\circ + 1.5^\circ)] \geq 1 - \epsilon, \\ & \forall k = t, \ldots, t + N - 1, \end{aligned} \tag{21.}$$

where $\epsilon > 0$ is a user-chosen desired violation probability and $x(0) = (0, \pi, 0, 0)$. The objective function in Equation 21 reflects the goal that the controller, which actuates the system through T_k, should apply as little torque as necessary. Equation 21 is difficult to solve using conventional analytical methods since the system dynamics are nonlinear, rendering the propagation of the uncertainty a_k a nontrivial task. We circumvent this difficulty by approximating Equation 21 using scenario MPC (discussed in Section 4), which is given by

$$\begin{aligned} \min_{T_{c_{t|t}}, \ldots, T_{c_{t+N-1|t}}} \quad & \sum_{k=t}^{t+N-1} T_{c_{k|t}}^2 \\ \text{subject to} \quad & x_{k+1|t}^{(i)} = f(x_{k|t}^{(i)}, T_{c_{k|t}}, a_{k|t}^{(i)}), \\ & (180^\circ - 1.5^\circ) \leq \theta_{k|t}^{(i)} \leq (180^\circ + 1.5^\circ), \\ & \forall k = t, \ldots, t + N, \ \forall i = 1, \ldots, S. \end{aligned} \tag{22.}$$

In the interest of computational efficiency, the optimization in Equations 21 and 22 is performed over open-loop policies. The use of closed-loop policies such as affine disturbance feedback would introduce additional decision variables and increase computation time.

> **Remark 2.** Equation 22 is, strictly speaking, not an instance of Equation 10 since the latter would require drawing different samples for each stage. In the interest of simplicity, we enforce the same samples in Equation 22, which, in practice, has performed well. Furthermore, we point out that, since system Equation 20 is nonlinear and Equation 21 is nonconvex, the theoretical guarantees on the sample sizes presented in Section 4.2 do not hold. Nevertheless, we will see that Equation 4 can be well approximated by Equation 21 in practice. For a rigorous analysis of sample sizes for stochastic nonlinear MPC problems, we refer the interested reader to References 51 and 70.

Table 1 shows empirical violation probabilities and computation times for solving Equation 22 as a function of the sample size. The numbers are averages obtained from 100

Table 1 Empirical violation probabilities and computation times as a function of the sample size S

Sample size	$S = 1$	$S = 5$	$S = 10$	$S = 30$	$S = 50$	$S = 100$	$S = 200$
Violation probability	31.9%	14.2%	11.2%	9.3%	7.4%	6.9%	5.1%
Computation time	7 ms	18 ms	29 ms	92 ms	131 ms	329 ms	741 ms

Computation times were obtained on a MacBook Pro equipped with a quad-core Intel Core i7 processor at 2.6 GHz and 16 GB of RAM.

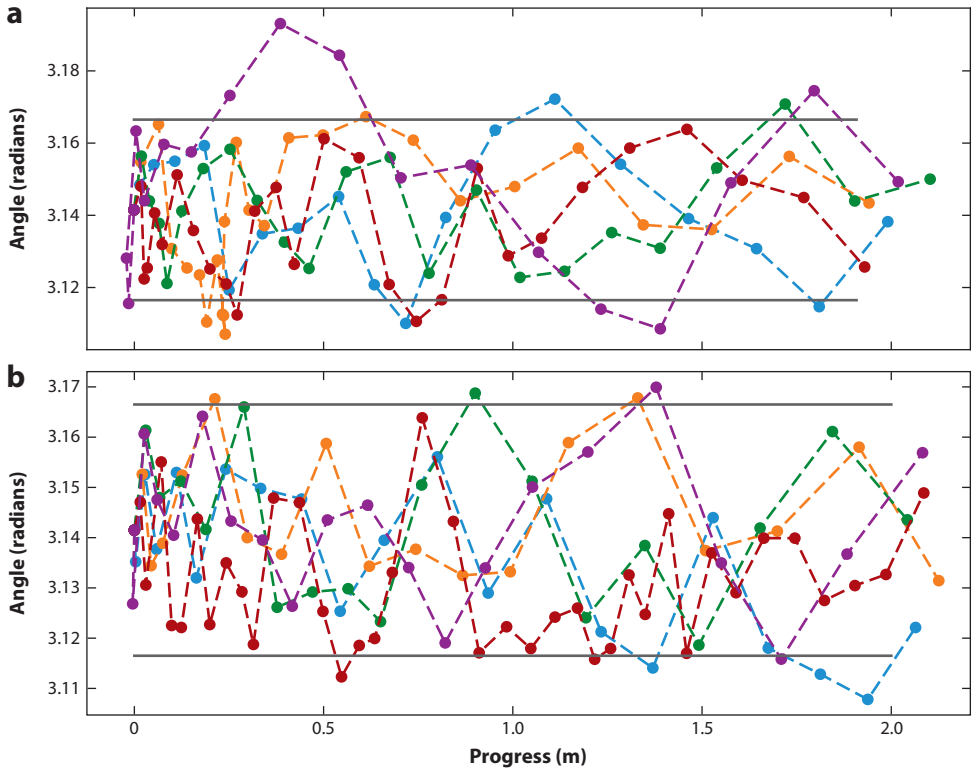

Figure 4

Randomly generated sample trajectories using the controller represented in Equation 22 with (*a*) $S = 10$ and (*b*) $S = 200$. The solid gray lines indicate the $\pm 1.5°$ constraint.

simulation runs. Two interesting observations underline the efficacy of scenario MPC. First, the sample size S can be used as a tuning variable to modulate the robustness of scenario MPC in terms of the violation probability. For small sample sizes (e.g., $S = 10$), scenario MPC is already able to obtain solutions that keep the pendulum within the designated constraints with a high probability, an observation that has also been made previously (57, 67, 71). Second, **Table 1** shows that the optimization problems arising in scenario MPC problems can be solved in real time when the sample size is small. For example, when $S = 10$, the average computation time is 29 ms, which is less than the sampling time of 50 ms. This implies that scenario MPC is also feasible for systems with fast dynamics, as has been previously demonstrated (56, 57).

Figure 4 depicts randomly generated sample trajectories when scenario MPC is used with $S = 10$ and $S = 200$. A large sample size results in both fewer and less severe constraint violations. This is intuitive because when $S \to \infty$, the solution of the scenario MPC problem converges to that of a robust MPC problem, where the constraints are satisfied for all possible uncertainty realizations. Finally, **Figure 5** depicts a snippet of a randomly chosen trajectory for $S = 10$, showing that the controller is indeed able to keep the pendulum in an upright position.

We conclude our first numerical study by pointing out that scenario MPC is a conceptually simple and computationally efficient way to address stochastic MPC problems. It approximates the

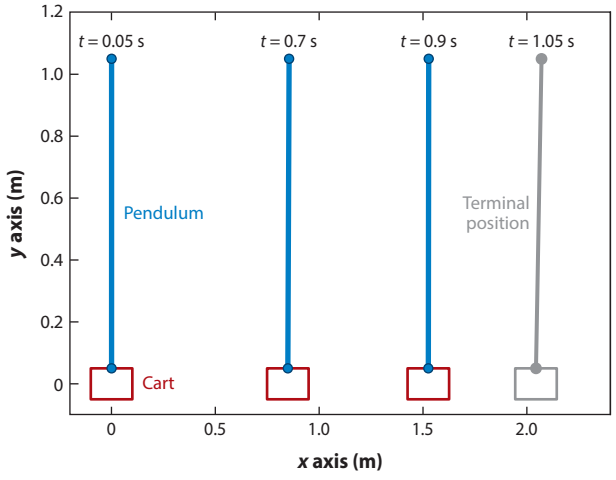

Figure 5

Snippet of one randomly chosen realization with $S = 10$ after 0.05 s, 0.7 s, 0.9 s, and 1.05 s from the task kickoff. The constraint $(180° - 1.5°) \leq \theta \leq (180° + 1.5°)$ is satisfied at $t = 0.05$ s, $t = 0.7$ s, and $t = 0.9$ s but violated at $t = 1.05$ s. The terminal position is represented in light gray.

true distribution of a system using sampled data (training data) and enforces constraint satisfaction only on these discrete data points. Finally, the solution quality is good, even when a small sample size (such as 10 or 50) is chosen.

6.3. Learning Model Predictive Control

In this section, we focus on the fully autonomous case, where the goal of the controller is to learn the best strategy with respect to different objectives, e.g., minimum time control or minimum effort control. The data from each trial are used to improve the closed-loop performance as described in Section 5. One can merge the results of this section with the previous example to design a stochastic LMPC that teaches a human to perform a better control at each iteration, which is an area of ongoing research.

The considered objective is a trade-off between the minimization of the task duration T and the control effort. More formally, the controller's goal is to solve the following finite-time optimal control problem:

$$\begin{aligned} \min_{T,u_0,\dots,u_{T-1}} \quad & \sum_{k=0}^{T} 1 + \alpha ||u_k||_{\bar{Q}} \\ \text{subject to} \quad & x_{k+1} = f(x_k, T_{c_k}, a_k), \\ & x_0 = [0, \pi, 0, 0]^T, \; x_T = x_F, \end{aligned} \tag{23.}$$

where $u_k = [T_{c_k}, a_k]^T$ and $||u_k||_{\bar{Q}} = u_k^T \bar{Q} u_k$, with $\bar{Q} = \text{diag}(10, 0.01)$. Note that, for small values of α, the solution to Equation 23 is close to a minimum-time solution, and for high values of α, the controller's goal is to reach the end point with minimum torque applied to the pendulum.

First, we compute a feasible solution to Equation 23 using an open-loop controller that drives the cart to the final position and a decoupled torque controller that keeps the pendulum upright.

Table 2 Evolution of the closed-loop cost through the iterations

Iteration	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
Cost	1,732,329	22,731	3,590	1,017	258	116	98	97	97

This feasible trajectory is used to construct the convex safe set $\mathcal{CS}^0$ and the terminal cost $P^0(\cdot)$ needed to initialize the first iteration of the LMPC represented in Equations 18 and 19.

The LMPC is implemented with $\alpha = 1$, a horizon $N = 3$, and the cost $\ell(x_k, u_k) = ||x_k||_2^2 / \sqrt{||x_k||_2^4 + 1 + ||u_k||_2^2}$, which approximates the cost in Equation 23. The LMPC with the convex safe set $\mathcal{CS}^j$ and terminal cost $P^j(\cdot)$ is reformulated as a nonlinear optimization problem and is implemented in Julia (72) using the solver IPOPT (26). Furthermore, each jth closed-loop trajectory is used to enlarge the convex safe set used at the $(j+1)$th iteration.

After eight iterations, the LMPC converges to a steady-state solution, as shown in **Table 2**. The iteration cost is nonincreasing over the iterations, and the LMPC improves the closed-loop performance.

Figure 6 shows the evolution of the closed-loop trajectory over the iterations. In particular, **Figure 6*a*** reports the pendulum angle as a function of the distance traveled. After the learning process, the controller swings the pendulum forward in order to limit the control effort needed to reach the final point without overshooting. **Figure 6*b*** shows a snippet of the closed-loop trajectory after 0.5 s, 2 s, and 4 s from the task kickoff. We emphasize that the controller successfully completes all learning iterations without dropping the pendulum.

Table 3 shows the results of several simulations for different values of α. For low values of α, Equation 23 is almost a minimum-time problem. On the other hand, for high values of α, the goal of the controller is to reach the terminal position while minimizing the torque applied to the pendulum. The LMPC improves the closed-loop performance for all values of α.

Finally, we compare the closed-loop trajectory for the first and fifth tunings in **Table 3**. In the first tuning, the controller's goal is mainly to minimize the torque. The fifth tuning is almost

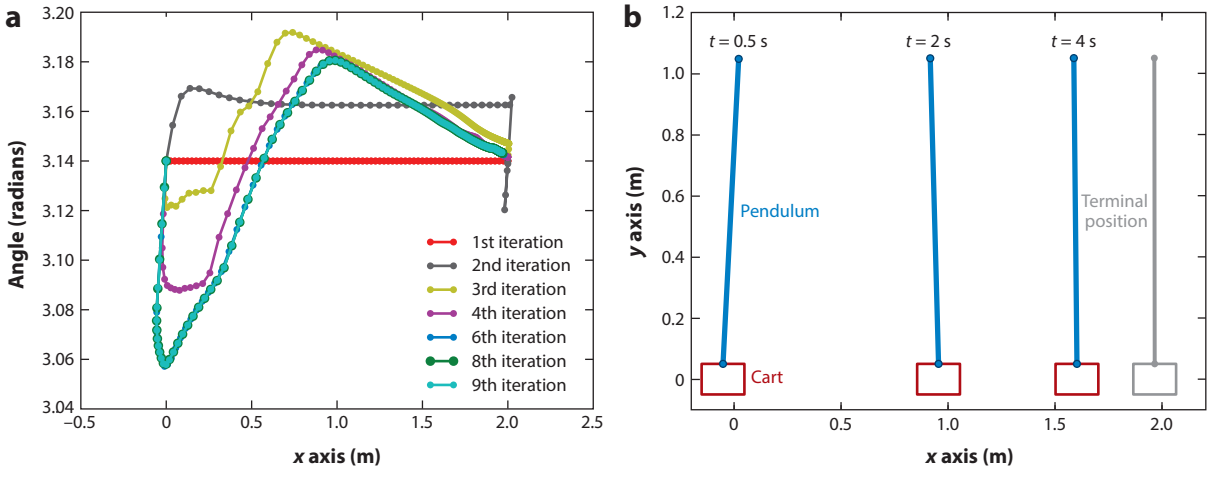

Figure 6

(*a*) Angle evolution through the iterations. After the learning process, the controller swings the pendulum forward to reach the terminal upright position without overshooting. (*b*) Snippet of a steady-state trajectory after 0.5 s, 2 s, and 4 s from the task kickoff. The terminal position is represented in light gray.

Table 3 Simulations for different control tunings

Task	α	$J^0_{0\to T}$	$J^9_{0\to T}$	Improvement
1	1	1.7×10^6	97	99.9%
2	0.1	1.7×10^5	97	99.9%
3	0.01	17,402	100	99.9%
4	0.001	1,812	74	95.8%
5	0.0001	253	51	79.8%

a minimum-time optimal control problem. We emphasize that, for both tunings, the LMPC is initialized with the same data, and therefore $\mathcal{SS}^0$ is the same. This shows that the LMPC algorithm is data efficient and that, depending on the control task, the LMPC explores the state space in different directions. Therefore, the convex safe set converges to different sets, as shown in **Figure 7**. **Figure 8** compares a snippet of the steady-state solution after 0.5 s from the kickoff of the two tunings. In **Figure 8*a***, when the objective is mainly the minimization of the torque, the controller moves the cart backward to swing the pendulum forward. On the other hand, when the controller's goal is to reach the end point in minimum time, the controller immediately moves the cart forward and applies torque to prevent the pendulum from falling. As a result, the inputs corresponding to the steady-state trajectory for the two tunings are different, as shown in **Figure 9**.

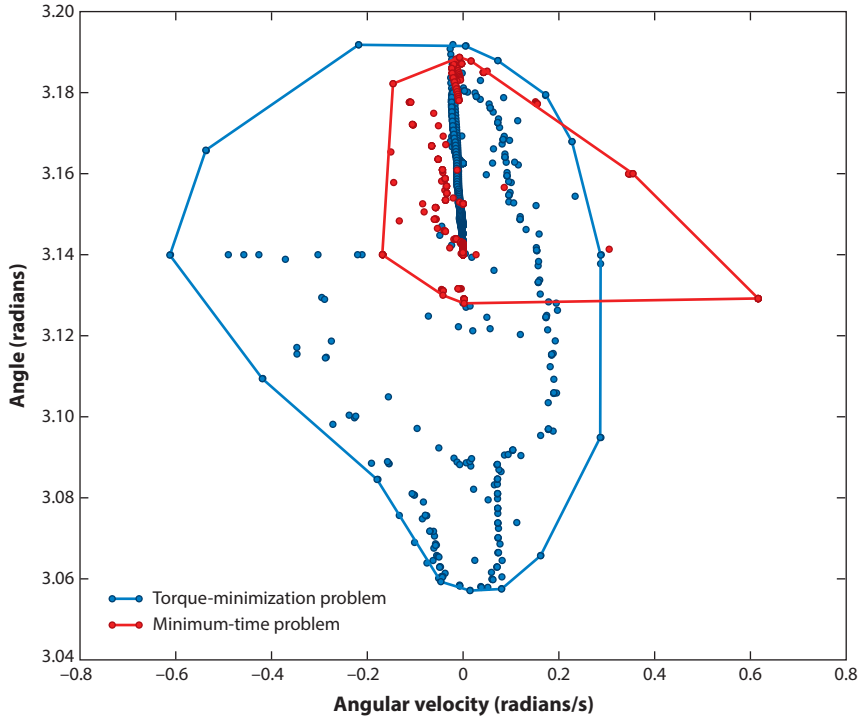

Figure 7

Convex safe set comparison. The convex safe sets for the torque-minimization problem and the minimum-time problem are shown in blue and red, respectively.

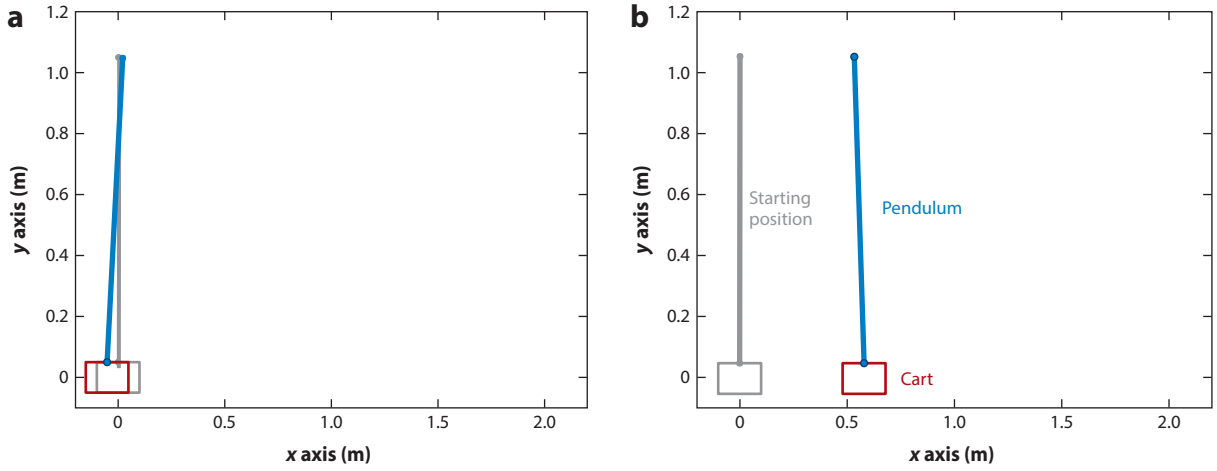

Figure 8

Snippets of the closed-loop trajectories of the two control tunings after 0.5 s from the kickoff. (*a*) Steady-state trajectory for the torque-minimization problem. (*b*) Steady-state trajectory for the minimum-time problem.

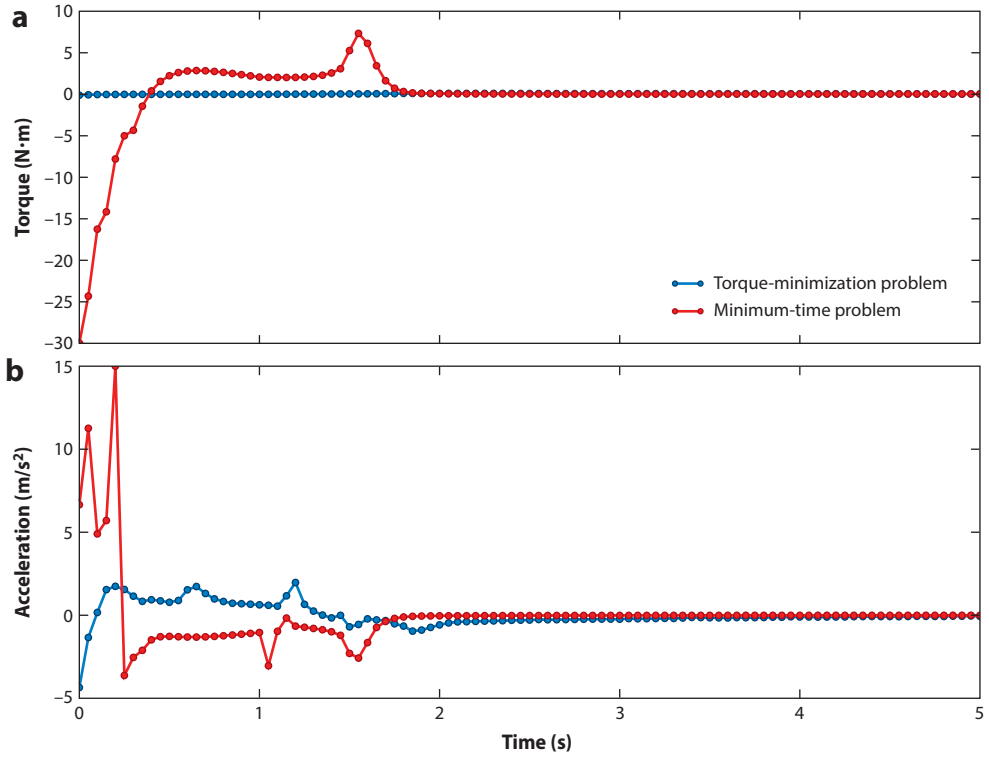

Figure 9

Input comparison for (*a*) the first control tuning and (*b*) the fifth control tuning in **Table 3**. The convex safe sets for the torque-minimization problem and the minimum-time problem are shown in blue and red, respectively.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

Some of the research described in this review was funded by the Hyundai Center of Excellence at the University of California, Berkeley. This work was also sponsored by the Office of Naval Research. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of Naval Research or the US government.

LITERATURE CITED

1. Murray RM, ed. 2003. *Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics, and Systems*. Philadelphia: SIAM
2. Manchester IR, Mettin U, Iida F, Tedrake R. 2011. Stable dynamic walking over uneven terrain. *Int. J. Robot. Res.* 30:265–79
3. Kuindersma S, Deits R, Fallon M, Valenzuela A, Dai H, et al. 2016. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Auton. Robots* 40:429–55
4. Metz C. 2016. How Google's AI viewed the move no human could understand. *Wired*, Mar. 14. **https://www.wired.com/2016/03/googles-ai-viewed-move-no-human-understand**
5. Borrelli F, Bemporad A, Morari M. 2017. *Predictive Control for Linear and Hybrid Systems*. Cambridge, UK: Cambridge Univ. Press
6. Zhang X, Grammatico S, Schildbach G, Goulart P, Lygeros J. 2015. On the sample size of random convex programs with structured dependence on the uncertainty. *Automatica* 60:182–88
7. Zhang X. 2016. *Robust and stochastic control of uncertain systems: from scenario optimization to adjustable uncertainty sets*. PhD Thesis, ETH Zurich
8. Rosolia U, Borrelli F. 2018. Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Trans. Autom. Control.* In press. **https://doi.org/10.1109/TAC.2017.2753460**
9. Rosolia U, Borrelli F. 2017. Learning model predictive control for iterative tasks: a computationally efficient approach for linear system. *IFAC-PapersOnLine* 50:3142–47
10. Garcia CE, Prett DM, Morari M. 1989. Model predictive control: theory and practice—a survey. *Automatica* 25:335–48
11. Morari M, Lee JH. 1999. Model predictive control: past, present and future. *Comput. Chem. Eng.* 23:667–82
12. Mayne DQ, Rawlings JB, Rao CV, Scokaert PO. 2000. Constrained model predictive control: stability and optimality. *Automatica* 36:789–814
13. Rawlings J, Mayne D. 2009. *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill
14. Camacho EF, Bordons C. 2013. *Model Predictive Control*. Berlin: Springer
15. Broadhurst A, Baker S, Kanade T. 2005. Monte Carlo road safety reasoning. In *2005 IEEE Intelligent Vehicles Symposium*, pp. 319–24. New York: IEEE
16. Eidehall A, Petersson L. 2008. Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Trans. Intell. Transp. Syst.* 9:137–47
17. McCall J, Trivedi M. 2007. Driver behavior and situation aware brake assistance for intelligent vehicles. *Proc. IEEE* 95:374–87
18. Bestick A, Bajcsy R, Dragan A. 2016. Implicitly assisting humans to choose good grasps in robot to human handovers. In *2016 International Symposium on Experimental Robotics*, pp. 341–54. Cham, Switz.: Springer
19. Dragan AD. 2017. Robot planning with mathematical models of human state and action. arXiv:170504226D

20. Lam C, Yang A, Driggs-Campbell K, Bajcsy R, Sastry S. 2015. Improving human-in-the-loop decision making in multi-mode driver assistance systems using hidden mode stochastic hybrid systems. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5776–83. New York: IEEE
21. Liu C, Hamrick J, Fisac J, Dragan A, Hedrick K, et al. 2016. Goal inference improves objective and perceived performance in human-robot collaboration. In *AAMAS '16: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 940–48. Richland, SC: Int. Found. Auton. Agents Multiagent Syst.
22. Kehoe B, Patil S, Abbeel P, Goldberg K. 2015. A survey of research on cloud robotics and automation. *IEEE Trans. Autom. Sci. Eng.* 12:398–409
23. Löfberg J. 2004. YALMIP: a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation*, pp. 284–89. New York: IEEE
24. Grant M, Boyd S. 2016. *CVX: Matlab software for disciplined convex programming*. **http: //cvxr.com/cvx**
25. Zhang X, Liniger A, Borrelli F. 2017. Optimization-based collision avoidance. arXiv:1711.03449
26. Pirnay H, López-Negrete R, Biegler L. 2012. Optimal sensitivity based on IPOPT. *Math. Program. Comput.* 4:307–31
27. Mueller MW, Hehn M, D'Andrea R. 2013. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3480–86. New York: IEEE
28. Mueller MW, Hehn M, D'Andrea R. 2015. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Trans. Robot.* 31:1294–310
29. LaValle S. 1998. *Rapidly-exploring random trees: a new tool for path planning*. Tech. Rep., Iowa State Univ., Ames
30. Otte M, Frazzoli E. 2015. RRT^{X}: real-time motion planning/replanning for environments with unpredictable obstacles. In *Algorithmic Foundations of Robotics XI*, ed. HL Akin, NM Amato, V Isler, AF van der Stappen, pp. 461–78. Cham, Switz.: Springer
31. Optim Gurobi. 2017. *Gurobi Optimizer reference manual*. **https://www.gurobi.com/documentation/7.5/refman**
32. MOSEK ApS. 2017. *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (revision 63)*. **http://docs.mosek.com/7.1/toolbox/index.html**
33. Goulart P, Kerrigan E, Maciejowski J. 2006. Optimization over state feedback policies for robust control with constraints. *Automatica* 42:523–33
34. Bemporad A. 1998. Reducing conservativeness in predictive control of constrained systems with disturbances. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pp. 1384–89. New York: IEEE
35. Chisci L, Rossiter J, Zappa G. 2001. Systems with persistent disturbances: predictive control with restricted constraints. *Automatica* 37:1019–28
36. Richards A, How J. 2006. Robust stable model predictive control with constraint tightening. In *2006 American Control Conference*. New York: IEEE. **https://doi.org/10.1109/ACC.2006.1656440**
37. Mayne D, Seron M, Raković S. 2005. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica* 41:219–24
38. Lofberg J. 2003. Approximations of closed-loop minimax MPC. In *42nd IEEE International Conference on Decision and Control*, pp. 1438–42. New York: IEEE
39. Zhang X, Kamgarpour M, Georghiou A, Goulart P, Lygeros J. 2017. Robust optimal control with adjustable uncertainty sets. *Automatica* 75:249–59
40. Batina I, Stoorvogel AA, Weiland S. 2002. Optimal control of linear, stochastic systems with state and input constraints. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pp. 1564–69. New York: IEEE
41. Boyd S, Vandenberghe L. 2004. *Convex Optimization*. Cambridge, UK: Cambridge Univ. Press
42. Kouvaritakis B, Cannon M, Rakovic S, Cheng Q. 2010. Explicit use of probabilistic distributions in linear predictive control. *Automatica* 46:1719–24
43. Cannon M, Cheng Q, Kouvaritakis B, Raković S. 2012. Stochastic tube MPC with state estimation. *Automatica* 48:536–41
44. Farina M, Giulioni L, Magni L, Scattolini R. 2013. A probabilistic approach to model predictive control. In *52nd IEEE Conference on Decision and Control*, pp. 7734–39. New York: IEEE

45. Fagiano L, Khammash M. 2012. Nonlinear stochastic model predictive control via regularized polynomial chaos expansions. In *2012 IEEE 51st Conference on Decision and Control*, pp. 142–47. New York: IEEE
46. Mesbah A, Streif S, Findeisen R, Braatz R. 2014. Stochastic nonlinear model predictive control with probabilistic constraints. In *2014 American Control Conference*, pp. 2413–19. New York: IEEE
47. Paulson J, Mesbah A, Streif S, Findeisen R, Braatz R. 2014. Fast stochastic model predictive control of high-dimensional systems. In *53rd IEEE Conference on Decision and Control*, pp. 2802–9. New York: IEEE
48. Lucia S, Zometa P, Kogel M, Findeisen R. 2015. Efficient stochastic model predictive control based on polynomial chaos expansions for embedded applications. In *2015 54th IEEE Conference on Decision and Control*, pp. 3006–12. New York: IEEE
49. Calafiore G, Fagiano L. 2013. Stochastic model predictive control of LPV systems via scenario optimization. *Automatica* 49:1861–66
50. Schildbach G, Fagiano L, Frei C, Morari M. 2014. The scenario approach for Stochastic Model Predictive Control with bounds on closed-loop constraint violations. *Automatica* 50:3009–18
51. Grammatico S, Zhang X, Margellos K, Goulart P, Lygeros J. 2016. A scenario approach for nonconvex control design. *IEEE Trans. Autom. Control* 61:334–45
52. Lorenzen M, Dabbene F, Tempo R, Allgower F. 2017. Stochastic MPC with offline uncertainty sampling. *Automatica* 81:176–83
53. Vitus MP, Zhou Z, Tomlin CJ. 2016. Stochastic control with uncertain parameters via chance constrained control. *IEEE Trans. Autom. Control* 61:2892–905
54. Margellos K, Goulart P, Lygeros J. 2014. On the road between robust optimization and the scenario approach for chance constrained optimization problems. *IEEE Trans. Autom. Control* 59:2258–63
55. Zhang X, Georghiou A, Lygeros J. 2015. Convex approximation of chance-constrained MPC through piecewise affine policies using randomized and robust optimization. In *2015 54th IEEE Conference on Decision and Control*, pp. 3038–43. New York: IEEE
56. Carrau J, Liniger A, Zhang X, Lygeros J. 2016. Efficient implementation of randomized MPC for miniature race cars. In *2016 European Control Conference*, pp. 957–62. New York: IEEE
57. Liniger A, Zhang X, Aeschbach P, Georghiou A, Lygeros J. 2017. Racing miniature cars: enhancing performance using stochastic MPC and disturbance feedback. In *2017 American Control Conference*, pp. 5642–47. New York: IEEE
58. Calafiore G, Campi MC. 2005. Uncertain convex programs: randomized solutions and confidence levels. *Math. Program.* 102:25–46
59. Calafiore G, Campi M. 2006. The scenario approach to robust control design. *IEEE Trans. Autom. Control* 51:742–53
60. Campi M, Garatti S. 2008. The exact feasibility of randomized solutions of robust convex programs. *SIAM J. Optim.* 19:1211–30
61. Calafiore G. 2010. Random convex programs. *SIAM J. Optim.* 20:3427–64
62. Zhang X, Grammatico S, Schildbach G, Goulart P, Lygeros J. 2014. On the sample size of randomized MPC for chance-constrained systems with application to building climate control. In *2014 European Control Conference*, pp. 478–83. New York: IEEE
63. Schildbach G, Fagiano L, Morari M. 2013. Randomized solutions to convex programs with multiple chance constraints. *SIAM J. Optim.* 23:2479–501
64. Rosolia U, Carvalho A, Borrelli F. 2017. Autonomous racing using learning model predictive control. In *2017 American Control Conference*, pp. 5115–20. New York: IEEE
65. Rosolia U, Zhang X, Borrelli F. 2018. Robust learning model predictive control for uncertain iterative tasks: learning from experience. In *2017 IEEE Conference on Decision and Control*. New York: IEEE. In press
66. Brunner M, Rosolia U, Gonzales J, Borrelli F. 2018. Repetitive learning model predictive control: an autonomous racing example. In *2017 IEEE Conference on Decision and Control*. New York: IEEE. In press
67. Zhang X, Schildbach G, Sturzenegger D, Morari M. 2013. Scenario-based MPC for energy-efficient building climate control under weather and occupancy uncertainty. In *2013 European Control Conference*, pp. 1029–34. New York: IEEE

68. Zhang X, Vrettos E, Kamgarpour M, Andersson G, Lygeros J. 2015. Stochastic frequency reserve provision by chance-constrained control of commercial buildings. In *2015 European Control Conference*, pp. 1134–40. New York: IEEE
69. Parisio A, Varagnolo D, Molinari M, Pattarello G, Fabietti L, Johansson K. 2014. Implementation of a scenario-based MPC for HVAC systems: an experimental case study. *IFAC Proc. Vol.* 47:599–605
70. Zhang X, Grammatico S, Margellos K, Goulart P, Lygeros J. 2014. Randomized nonlinear MPC for uncertain control-affine systems with bounded closed-loop constraint violations. *IFAC Proc. Vol.* 47:1649–54
71. Ioli D, Falsone A, Prandini M. 2016. Energy management of a building cooling system with thermal storage: a randomized solution with feedforward disturbance compensation. In *2016 American Control Conference*, pp. 2346–51. New York: IEEE
72. Bezanzon J, Karpinski S, Shah V, Edelman A. 2012. Julia: a fast dynamic language for technical computing. arXiv:1209.5145