Springer Proceedings in Advanced Robotics 17
Series Editors: Bruno Siciliano · Oussama Khatib

Steven M. LaValle · Ming Lin ·
Timo Ojala · Dylan Shell · Jingjin Yu *Editors*

Algorithmic Foundations of Robotics XIV

Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics

MOREMEDIA
spar
Springer

Springer Proceedings in Advanced Robotics 17

The Springer Proceedings in Advanced Robotics (SPAR) publishes new developments and advances in the fields of robotics research, rapidly and informally but with a high quality.

The intent is to cover all the technical contents, applications, and multidisciplinary aspects of robotics, embedded in the fields of Mechanical Engineering, Computer Science, Electrical Engineering, Mechatronics, Control, and Life Sciences, as well as the methodologies behind them.

The publications within the "Springer Proceedings in Advanced Robotics" are primarily proceedings and post-proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. Also considered for publication are edited monographs, contributed volumes and lecture notes of exceptionally high quality and interest.

An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

Indexed by WTI Frankfurt eG, zbMATH.

More information about this series at http://www.springer.com/series/15556

Steven M. LaValle · Ming Lin · Timo Ojala ·
Dylan Shell · Jingjin Yu
Editors

Algorithmic Foundations of Robotics XIV

Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics

Springer

Editors
Steven M. LaValle
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland

Timo Ojala
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland

Jingjin Yu
Department of Computer Science
Rutgers University
Piscataway, NJ, USA

Ming Lin
Brendan Iribe Center for Computer Science and Engineering
University of Maryland
College Park, MD, USA

Dylan Shell
Department of Computer Science and Engineering
Texas A&M University
College Station, TX, USA

ISSN 2511-1256 ISSN 2511-1264 (electronic)
Springer Proceedings in Advanced Robotics
ISBN 978-3-030-66722-1 ISBN 978-3-030-66723-8 (eBook)
https://doi.org/10.1007/978-3-030-66723-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

At the dawn of the century's third decade, robotics is reaching an elevated level of maturity and continues to benefit from the advances and innovations in its enabling technologies. These all are contributing to an unprecedented effort to bringing robots to human environment in hospitals and homes, factories and schools, in the field for robots fighting fires, making goods and products, picking fruits and watering the farmland, saving time and lives. Robots today hold the promise for making a considerable impact in a wide range of real-world applications from industrial manufacturing to healthcare, transportation, and exploration of the deep space and sea. Tomorrow, robots will become pervasive and touch upon many aspects of modern life.

The *Springer Tracts in Advanced Robotics (STAR)* was launched in 2002 with the goal of bringing to the research community the latest advances in the robotics field based on their significance and quality. During the latest fifteen years, the STAR series has featured publication of both monographs and edited collections. Among the latter, the proceedings of thematic symposia devoted to excellence in robotics research, such as ISRR, ISER, FSR, and WAFR, have been regularly included in STAR.

The expansion of our field as well as the emergence of new research areas has motivated us to enlarge the pool of proceedings in the STAR series in the past few years. This has ultimately led to launching a sister series in parallel to STAR. The *Springer Proceedings in Advanced Robotics (SPAR)* is dedicated to the timely dissemination of the latest research results presented in selected symposia and workshops.

This volume of the SPAR series brings the proceedings of the fourteenth edition of the workshop algorithmic foundations of robotics (WAFR). This edition was supposed to be an in-person event in Oulu, like prior WAFRs. With the rise of the global

COVID-19 crisis, it was postponed with the intention of having a physical meeting in Finland in 2021.

The volume edited by Steve LaValle, Ming Lin, Timo Ojala, Dylan Shell, and Jingjin Yu is a collection of 33 contributions, highlighting cutting-edge algorithmic research in planning, learning, perception, navigation, control, optimality, completeness, and complexity. Applications span a wide variety of areas including multiagent systems, mobile robots, and manipulation.

Rich by topics and authoritative contributors, WAFR culminates with this unique reference on the current developments and new directions in the field of algorithmic foundations. A very fine addition to the series!

November 2020

Bruno Siciliano
Oussama Khatib

Preface

Robots and autonomous systems depend on algorithms to efficiently realize their many functionalities, ranging from perception to decision making, from motion planning to control. These functionalities are but parts of a broader, grander whole: the ambition of bringing intelligent robots to fruition. Such robots must grapple with, as William James says, the "blooming, buzzing confusion" of the world—they must do this via some form of workable description of the circumstances in which they find themselves and must turn that description toward realizing ends. Behaving intelligently in the world involves information processing that can bring prior knowledge (e.g., about constraints and structured properties of the environment) to bear on a stream of evidence (acquired as new data via sensor readings) so as to produce a flow of actions to, ultimately, accomplish aims. As a computational task, the difficulties are immense. Most sophisticated robots are complex physical machines and, accordingly, natural representations of the underlying problems are typically high dimensional and continuous; their environment is often dynamic, evolving of its own accord, so action must be timely lest it risk being irrelevant or erroneous; a robot's own model of its surroundings will inevitably be incomplete and may degraded by other forms of uncertainty that cannot be avoided readily; and there are many more challenges besides.

To contend with these challenges, we have a host of new technologies. Parallel computation is cheap and fast; new sensors (from tactile technologies to event-based imaging) have gained acceptance; established sensors are more affordable now than ever before; new fabrication and manufacturing methods enable better mechanical components and there is the promise of next-generation devices based on novel materials. Some of these represent enhancements that will act as enablers for current approaches, driving further refinement. Others are likely to trigger more fundamental shifts. Part of the work of the roboticist is to lay the course, to identify opportunities for creativity, and to apply the intellect to those difficulties ripe for solution.

Advanced robot algorithms draw on the combined expertise from many fields, including control theory, computational geometry and topology, geometrical and physical modeling, reasoning under uncertainty, probabilistic algorithms, game

theory, and theoretical computer science. It is fair to say that the works collected in this SPAR volume represent the state of the art in algorithmic robotics. They originate from papers accepted to the 14th International Workshop on the Algorithmic Foundations of Robotics (WAFR), traditionally a biannual, single-track meeting of leading researchers in the field of robotics. WAFR has always served as a premiere venue for the publication of some of robotics' most important, fundamental, and lasting algorithmic contributions, ensuring the rapid circulation of new ideas.

For 2020, WAFR had originally been intended to be an in-person meeting in Oulu, Finland. It would have taken the traditional form of an intimate, highly interactive, single-track meeting like prior WAFRs. With the rise of the global COVID-19 crisis, the physical meeting was first postponed and then finally canceled.

The papers collected in this volume represent 33 papers selected from the 59 excellent submissions we received and which were refereed. We thank the program committee and *ad hoc* reviewers who provided insightful, considered, and constructive reviews for all submissions. They were Srinivas Akella, Devin Balkcom, Kostas Bekris, Dmitry Berenson, Abdeslam Boularias, Kyle Jordan Brown, Katie Byl, Nilanjan Chakraborty, Suman Chakravorty, Benjamin Charrow, Howie Choset, Micah Corah, Juan Cortes, Neil Dantam, Jory Denny, Mehmet R. Dogar, Chinwe Ekenna, Brendan Englot, Esra Erdem, Claudia Esteves, Taosha Fan, Aleksandra Faust, Emilio Frazzoli, Jie Gao, Edgar Granados, Jaskaran Singh Grover, Stephen Guy, Dan Halperin, Shuai Han, Geoffrey Hollinger, David Hsu, Volkan Isler, Marcelo Kallmann, Zachary Kingston, Ross Knepper, Sven Koenig, Hadas Kress-Gazit, Torsten Kroeger, Alan Kuntz, Hanna Kurniawati, Jyh-Ming Lien, Maxim Likhachev, Tomas Lozano-Perez, Anirudha Majumdar, Sonia Martinez, Joseph Mitchell, Marco Morales, Todd Murphey, Rafael Murrieta, Songhwai Oh, Jason O'Kane, Michael Otte, Jia Pan, Zherong Pan, Oriana Claudia Peltzer, Florian T. Pokorny, Valentin Polishchuk, Michael Posa, Hazhar Rahmani, Elon Rimon, Alberto Rodriguez, Nicholas Roy, Oren Salzman, Stephen L. Smith, Kiril Solovey, Dezhen Song, Cynthia Sung, Subhash Suri, Lydia Tapia, Shawna Thomas, Pratap Tokekar, Steve Tonneau, Frank van der Stappen, Chee Yap, Sung-Eui Yoon, Wenhao Yu, and Liangjun Zhang. We thank them particularly for devoting the time to carry out this service at an especially challenging time. Among the list, several people found themselves unexpectedly plunged into caregiving roles, including that of minding children round the clock. Please accept our deepest gratitude.

To a degree perhaps greater than ever before, the public is captivated by conceptions of a near future in which robots are more than just labor-saving devices, more than graceless machines, remote, and indifferent. If these mighty hopes

suggest a high peak with immense challenges yet to be met, at least we know we are beyond the foothills. Fortitude all, and *upward!*

Steve LaValle
Ming Lin
Timo Ojala
Dylan A. Shell
Jingjin Yu

Contents

ABC-LMPC: Safe Sample-Based Learning MPC for Stochastic Nonlinear Dynamical Systems with Adjustable Boundary Conditions

Brijen Thananjeyan[1], Ashwin Balakrishna[1(✉)], Ugo Rosolia[2], Joseph E. Gonzalez[1], Aaron Ames[2], and Ken Goldberg[1]

[1] University of California Berkeley, Berkeley, CA 94720, USA
{bthananjeyan,ashwin_balakrishna}@berkeley.edu
[2] California Institute of Technology, Pasadena, CA 91125, USA

Abstract. Sample-based learning model predictive control (LMPC) strategies have recently attracted attention due to their desirable theoretical properties and good empirical performance on robotic tasks. However, prior analysis of LMPC controllers for stochastic systems has mainly focused on linear systems in the iterative learning control setting. We present a novel LMPC algorithm, Adjustable Boundary Condition LMPC (ABC-LMPC), which enables rapid adaptation to novel start and goal configurations and theoretically show that the resulting controller guarantees iterative improvement in expectation for stochastic nonlinear systems. We present results with a practical instantiation of this algorithm and experimentally demonstrate that the resulting controller adapts to a variety of initial and terminal conditions on 3 stochastic continuous control tasks.

Keywords: Model predictive control · Control theory · Imitation learning

1 Introduction

Model predictive control (MPC) has seen significant success in a variety of robotic tasks [1–3], and there is substantial experimental and theoretical work demonstrating that the resulting closed loop system performs well on challenging tasks in stochastic dynamical systems [1,4–6]. In this work, we build on the recently proposed learning model predictive control (LMPC) framework [5–7]. We assume a known stochastic dynamics model and design an iteratively improving MPC-based control strategy by estimating safe sets and value functions from past closed-loop trajectories.

The LMPC framework [5–7] presents a novel class of reference-free control strategies which utilize MPC to iteratively improve upon a suboptimal controller for a goal

B. Thananjeyan and A. Balakrishna—Equal contribution.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-66723-8_1) contains supplementary material, which is available to authorized users.

S. M. LaValle et al. (Eds.): WAFR 2020, SPAR 17, pp. 1–17, 2021.
https://doi.org/10.1007/978-3-030-66723-8_1

directed task. LMPC algorithms typically operate in the iterative learning control setting with fixed initial and terminal conditions, and provide robust guarantees on iterative improvement (in terms of task cost) for stochastic linear systems [5,6] and deterministic nonlinear systems [7] if the MPC problem can be solved exactly. However, while LMPC-based control strategies exhibit a variety of desirable theoretical properties [5–7] and have been shown to work well on practical problems on physical robotic systems [1–8], they have two key limitations: (1) guarantees for stochastic systems are limited to linear systems while practical systems are often stochastic and nonlinear and (2) start states and goal sets are typically assumed to be identical in each iteration.

We address both of these challenges. First, we extend the results in [5] to show iterative improvement guarantees for stochastic nonlinear systems. Second, we present a method to expand the set of feasible start states and goal sets during learning while maintaining these guarantees. Finally, we introduce sample-based approximations to present a practical algorithm to learn safe policies, which reliably complete tasks with varying boundary conditions while satisfying pre-specified constraints. The contributions of this work are (1) a novel multi-start, multi-goal LMPC algorithm, Adjustable Boundary Condition LMPC (ABC-LMPC), which optimizes expected costs subject to robust constraints, with (2) guarantees on expected performance, robust constraint satisfaction, and convergence to the goal for stochastic nonlinear systems, (3) a practical algorithm for expanding the allowed set of initial states and goal sets during learning, and (4) simulated continuous control experiments demonstrating that the learned controller can adapt to novel start states and goal sets while consistently and efficiently completing tasks during learning.

2 Related Work

Model Predictive Control: There has been a variety of prior work on learning based strategies for model predictive control in the reinforcement learning [1–3] and controls communities [9–14]. Prior work in learning for model predictive control has focused on estimating the following three components used to design MPC policies: i) a model of the system [1–3,8,10,12,13,15], ii) a safe set of states from which the control task can be completed using a known safe policy [16–19] and iii) a value function [1,5,6,20], which for a given safe policy, maps each state of the safe set to the closed-loop cost to complete the task. The most closely related works, both by Rosolia et al. [5,6], introduce the learning MPC framework for iterative learning control in stochastic linear systems. Here, MPC is used to iteratively improve upon a suboptimal demonstration by estimating a safe set and a value function from past closed loop trajectories. Robust guarantees are provided for iterative controller improvement if the MPC problem can be solved exactly. Furthermore, Thananjeyan et al. [1] propose a practical reinforcement learning algorithm using these strategies to learn policies for nonlinear systems. However, [1,6] are limited to the iterative learning control setting, and although [5] presents a strategy for controller domain expansion, the method is limited to linear systems and requires the user to precisely specify an expansion direction. In this work, we build on this framework by (1) extending the theoretical results to prove that under similar assumptions, LMPC based controllers yield iterative improvement in expectation under

certain restrictions on the task cost function and (2) providing a practical and general algorithm to adapt to novel start states and goal sets while preserving all theoretical guarantees on controller performance.

Reinforcement Learning: There has been a variety of work from the reinforcement learning (RL) community on learning policies which generalize across a variety of initial and terminal conditions. Curriculum learning [21–23] has achieved practical success in RL by initially training agents on easier tasks and reusing this experience to accelerate learning of more difficult tasks. Florensa et al. [21] and Resnick et al. [22] train policies initialized near a desired goal state, and then iteratively increase the distance to the goal state as learning progresses. While these approaches have achieved practical success on a variety of simulated robotic and navigation tasks, the method used to expand the start state distribution is heuristic-based and requires significant hand-tuning. We build on these ideas by designing an algorithm which expands the start state distribution for an MPC-based policy by reasoning about reachability, similar to Ivanovic et al. [24]. However, [24] provides a curriculum for model free RL algorithms and does not provide feasibility or convergence guarantees, while we present an MPC algorithm which expands the set of allowed start states while preserving controller feasibility and convergence guarantees. There is also recent interest in goal-conditioned RL [25,26]. The most relevant prior work in this area is hindsight experience replay [27], which trains a goal-conditioned policy using imagined goals from past failures. This strategy efficiently reuses data to transfer to new goal sets in the absence of dense rewards. We use a similar idea to learn goal-conditioned safe sets to adapt to novel goal sets by reusing data from past trajectories corresponding to goal sets reached in prior iterations.

Motion Planning: The domain expansion strategy of the proposed algorithm, ABC-LMPC, bears a clear connection to motion planning in stochastic dynamical systems [28,29]. Exploring ways to use ABC-LMPC to design motion planning algorithms which can efficiently leverage demonstrations is an exciting avenue for future work, since the receding horizon planning strategy could prevent the exponential scaling in complexity with time horizon characteristic of open-loop algorithms [30].

3 Problem Statement

In this work, we consider nonlinear, stochastic, time-invariant systems of the form:

$$x_{t+1} = f(x_t, u_t, w_t) \tag{3.0.1}$$

where $x_t \in \mathbb{R}^n$ is the state at time t, $u_t \in \mathbb{R}^m$ is the control, $w_t \in \mathbb{R}^k$ is a disturbance input, and x_{t+1} is the next state. The disturbance w_t is sampled i.i.d. from a known distribution over a bounded set $\mathcal{W} \subseteq \mathbb{R}^p$. We denote Cartesian products with exponentiation, e.g. $\mathcal{W}^2 = \mathcal{W} \times \mathcal{W}$. We consider constraints requiring states to belong to the feasible state space $\mathcal{X} \subseteq \mathbb{R}^n$ and controls to belong to $\mathcal{U} \subseteq \mathbb{R}^m$. Let x_t^j, u_t^j, and w_t^j be the state, control input, and disturbance realization sampled at time t of iteration j respectively. Let $\pi^j : \mathbb{R}^n \to \mathbb{R}^m$ be the control policy at iteration j that maps states to controls (i.e. $u_t^j = \pi^j(x_t^j)$).

Unlike [5], in which the goal of the control design is to solve a robust optimal control problem, we instead consider an expected cost formulation. Thus, instead of optimizing for the worst case noise realization, we consider control policies which optimize the given cost function in expectation over possible noise realizations. To do this, we define the following objective function with the following Bellman equation and cost function $C(\cdot,\cdot)$:

$$J^{\pi^j}(x_0^j) = \mathop{\mathbb{E}}_{w_0^j}\left[C(x_0^j, \pi^j(x_0^j)) + J^{\pi^j}(f(x_0^j, u_0^j, w_0^j))\right] \tag{3.0.2}$$

However, we would like to only consider policies that are robustly constraint-satisfying for all timesteps. Thus, the goal of the control design is to solve the following infinite time optimal control problem:

$$\begin{aligned} J^{j,*}_{0\to\infty}(x_0^j) = \min_{\pi^j(.)} \quad & J^{\pi^j}(x_0^j) \\ \text{s.t. } & x_{t+1}^j = f(x_t^j, u_t^j, w_t^j) \\ & u_t^j = \pi^j(x_t^j) \\ & x_t^j \in \mathcal{X}, u_t^j \in \mathcal{U} \\ & \forall w_t^j \in \mathcal{W}, t \in \{0,1,\ldots\} \end{aligned} \tag{3.0.3}$$

In this paper, we present a strategy to iteratively design a feedback policy $\pi^j(.) : \mathcal{F}_\mathcal{G}^j \subseteq \mathcal{X} \to \mathcal{U}$, where $\mathcal{F}_\mathcal{G}^j$ is the domain of π^j for goal set $\mathcal{G}$ (and also the set of allowable initial conditions). Conditioned on the goal set $\mathcal{G}$, the controller design provides guarantees for (i) robust satisfaction of state and input constraints, (ii) convergence in probability of the closed-loop system to $\mathcal{G}$, (iii) iterative improvement: for any $x_0^j = x_0^l$ where $j < l$, expected trajectory cost is non-increasing ($J^{\pi^j}(x_0^j) \geq J^{\pi^{j+1}}(x_0^{j+1})$), and (iv) exploration: the domain of the control policy does not shrink over iterations ($\mathcal{F}_\mathcal{G}^j \subseteq \mathcal{F}_\mathcal{G}^{j+1}$ for all goal sets $\mathcal{G}$ sampled up to iteration j). In Sect. 4.3, we describe how to transfer to a new goal set $\mathcal{H}$ by reusing data from prior iterations while maintaining the same properties.

We adopt the following definitions and assumptions:

Assumption 1. ***Costs:*** *We consider costs which are zero within the goal set $\mathcal{G}$ and greater than some $\varepsilon > 0$ outside the goal set: $\exists \varepsilon > 0$ s.t. $C(x,u) \geq \varepsilon \mathbb{1}_{\mathcal{G}^C}(x)$ where $\mathbb{1}$ is an indicator function and $\mathcal{G}^C$ is the complement of $\mathcal{G}$.*

Definition 1. ***Robust Control Invariant Set:*** *As in Rosolia et al. [5], we define a robust control invariant set $\mathcal{A} \subseteq \mathcal{X}$ with respect to dynamics $f(x,u,w)$ and policy class Π as a set where $\forall x \in \mathcal{A}$, $\exists \pi \in \Pi$ s.t. $f(x,\pi(x),w) \in \mathcal{A}, \pi(x) \in \mathcal{U}$, $\forall w \in \mathcal{W}$.*

Assumption 2. ***Robust Control Invariant Goal Set:*** *$\mathcal{G} \subseteq \mathcal{X}$ is a robust control invariant set with respect to the dynamics and the set of state feedback policies Π.*

4 Preliminaries

Here we formalize the notion of safe sets, value functions, and how they can be conditioned on specific goals. We also review standard definitions and assumptions.

4.1 Safe Set

We first recall the definition of a robust reachable set as in [5]:

Definition 2. ***Robust Reachable Set:*** *The robust reachable set $\mathcal{R}_t^{\pi}(x_0^j)$ contains the set of states reachable in t-steps by the system (3.0.1) in closed loop with π at iteration j:*

$$\mathcal{R}_{t+1}^{\pi}(x_0^j) = \left\{ x_{t+1} \in \mathbb{R}^n \,|\, \exists w_t \in \mathcal{W},\ x_t \in \mathcal{R}_t^{\pi}(x_0^j),\ x_{t+1} = f(x_t, \pi(x_t), w_t) \right\} \tag{4.1.1}$$

where $\mathcal{R}_0^{\pi}(x_0^j) = x_0^j$. We define $\mathcal{R}_{t+1}^{\pi}$ similarly when the input is a set and for time-varying policies.

Now, we define the safe set at iteration j for the goal set $\mathcal{G}$ as in [5].

Definition 3. ***Safe Set:*** *The safe set $\mathcal{SS}_{\mathcal{G}}^j$ contains the full evolution of the system at iteration j,*

$$\mathcal{SS}_{\mathcal{G}}^j = \left\{ \bigcup_{t=0}^{\infty} \mathcal{R}_t^{\pi^j}(x_0^j) \bigcup \mathcal{G} \right\}. \tag{4.1.2}$$

Note that (4.1.2) is robust control invariant by construction [5]. We could set $\mathcal{SS}_{\mathcal{G}}^0 = \mathcal{G}$ or initialize the algorithm with a nominal controller π^0. This enables the algorithm to naturally incorporate demonstrations to speed up training.

Definition 4. ***Expected Cost:*** *The expected cost of π^j from start state x_0^j is defined as*

$$J^{\pi^j}(x_0^j) = \mathop{\mathbb{E}}_{w^j} \left[\sum_{t=0}^{\infty} C(x_t^j, \pi^j(x_t^j)) \right] \tag{4.1.3}$$

4.2 Value Function

Definition 5. ***Value Function:*** *Recursively define the value function of π^j in closed-loop with (3.0.3) as:*

$$L_{\mathcal{G}}^{\pi^j}(x) = \begin{cases} \mathop{\mathbb{E}}_{w} \left[C(x, \pi^j(x)) + L_{\mathcal{G}}^{\pi^j}(f(x, \pi^j(x), w)) \right] & x \in \mathcal{SS}_{\mathcal{G}}^j \\ +\infty & x \notin \mathcal{SS}_{\mathcal{G}}^j \end{cases} \tag{4.2.4}$$

Let $V_{\mathcal{G}}^{\pi^j}(x) = \min_{k \in \{0,\dots j\}} L_{\mathcal{G}}^{\pi^k}(x)$, which is the expected cost-to-go of the best performing prior controller at x.

Observe that $L_{\mathcal{G}}^{\pi^j}$ is defined only on $\mathcal{SS}_{\mathcal{G}}^j$, and $J^{\pi^j} = L_{\mathcal{G}}^{\pi^j}$ on $\mathcal{SS}_{\mathcal{G}}^j$. In the event a nominal controller π^0 is used, we require the following assumption on the initial safe set $\mathcal{SS}_{\mathcal{G}}^0$, which is implicitly a restriction on π^0 for start state x_0^0.

Assumption 3. ***Safe Set Initial Condition:*** *If a nominal controller π^0 is used, then $\forall x \in \mathcal{SS}_{\mathcal{G}}^0$, $L_{\mathcal{G}}^{\pi^0}(x) < \infty$.*

This assumption requires that the nominal controller is able to robustly satisfy constraints and converge in probability to $\mathcal{G}$. If no nominal controller is used, then this assumption is not required. In that case, we let $\mathcal{SS}_{\mathcal{G}}^0 = \mathcal{G}$ and $L_{\mathcal{G}}^{\pi^0}(x) = 0\ \forall x \in \mathcal{SS}_{\mathcal{G}}^0$.

4.3 Transfer to Novel Goal Sets

While Rosolia et al. [5] studies tasks with fixed goal sets, here we show how the safe set and value function can be modified to transfer the learned controller at iteration $j+1$ to a new robust control invariant goal set $\mathcal{H}$ and reuse data from the earlier iterations to accelerate learning.

Definition 6. ***Goal Conditioned Safe Set:*** *Define the goal conditioned safe set by collecting the prefixes of all robust reachable sets until they robustly fall in $\mathcal{H}$ as follows:*

$$\mathcal{SS}^j_{\mathcal{H}} = \begin{cases} \bigcup_{k=0}^{k^*} \mathcal{R}_k^{\pi^j} \bigcup \mathcal{H} & \max_{k \in \mathbb{N}} \mathbb{1}\{\mathcal{R}_k^{\pi^j}(x_0^j) \subseteq \mathcal{H}\} = 1 \\ \mathcal{H} & \textit{otherwise} \end{cases} \tag{4.3.5}$$

where $k^* = \arg\max_{k \in \mathbb{N}} \mathbb{1}\{\mathcal{R}_k^{\pi^j}(x_0^j) \subseteq \mathcal{H}\}$

We also redefine the value function as follows:

Definition 7. ***Goal Conditioned Value Function:*** *Recursively define the goal-conditioned value function of π^j in closed-loop with (3.0.3) as:*

$$L^{\pi^j}_{\mathcal{H}}(x) = \begin{cases} \mathop{\mathbb{E}}_{w}\left[C(x,\pi^j(x)) + L^{\pi^j}(f(x,\pi^j(x),w))\right] & x \in \mathcal{SS}^j_{\mathcal{H}} \setminus \mathcal{H} \\ 0 & x \in \mathcal{H} \\ +\infty & x \notin \mathcal{SS}^j_{\mathcal{H}} \end{cases} \tag{4.3.6}$$

Define $V^{\pi^j}_{\mathcal{H}}(x) = \min_{k \in \{0,\ldots j\}} L^{\pi^k}_{\mathcal{H}}(x)$ *as before.*

This new value function is for a policy that executes π^j but switches to a policy that keeps the system in $\mathcal{H}$ upon entry.

5 Controller Design

Here we describe the controller design for optimizing the task cost function while satisfying state and input constraints (Sect. 5.1), and discuss how this can be extended to iteratively expand the controller domain (Sect. 5.2). We consider a fixed goal set $\mathcal{G}$ for clarity, but note that the same formulation holds for other goal sets if the safe set and value function are appropriately defined as in Definitions 6 and 7. See Fig. 1 for an illustration of the full ABC-LMPC controller optimization procedure.

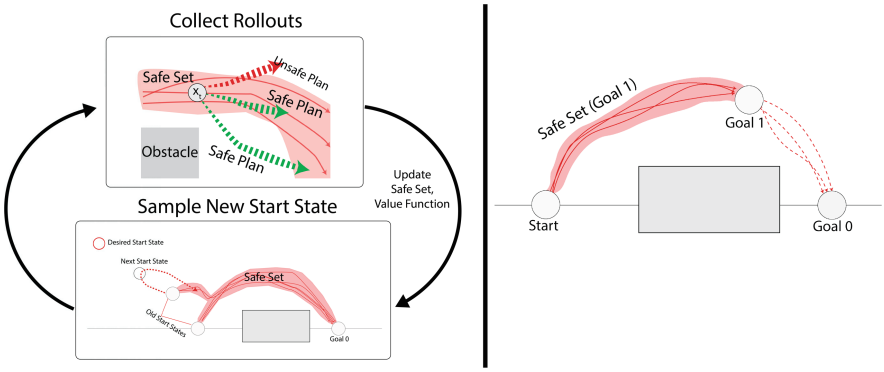


Fig. 1. ABC-LMPC Iterative Algorithm (Left): ABC-LMPC alternates between (1) collecting rollouts under the current policy π^j given $\mathcal{SS}^j_{\mathcal{G}_0}$ and $L^{\pi^j}_{\mathcal{G}_0}$ (by optimizing (5.1.1)), (2) updating $\mathcal{SS}^{j+1}_{\mathcal{G}_0}$ and $L^{\pi^{j+1}}_{\mathcal{G}_0}$ given the new rollouts, and (3) expanding the controller domain towards a desired start state (by optimizing (5.2.4)); **Goal Set Transfer (Right):** When a new goal set $\mathcal{G}_1$ is supplied, trajectories to goal $\mathcal{G}_0$ can be reused to estimate a new safe set for a new goal $\mathcal{G}_1$ ($\mathcal{SS}^j_{\mathcal{G}_1}$) and associated value function ($L^{\pi^j}_{\mathcal{G}_1}$).

5.1 Task Driven Optimization

At time t of iteration j with goal set $\mathcal{G}$, the controller solves the following receding-horizon trajectory optimization problem with planning horizon $H > 0$:

$$
\begin{aligned}
J^j_{t\to t+H}(x^j_t) = \min_{\pi_{t:t+H-1|t}\in\Pi^H} \quad & \mathbb{E}_{w^j_{t:t+H-1}}\left[\sum_{i=0}^{H-1} C(x^j_{t+i|t}, \pi_{t+i|t}(x^j_{t+i|t})) + V^{\pi^{j-1}}_{\mathcal{G}}(x^j_{t+H|t})\right] \\
\text{s.t.} \quad & x^j_{t+i+1|t} = f(x^j_{t+i|t}, \pi_{t+i|t}(x^j_{t+i}), w_{t+i}) \ \forall i \in \{0,\dots,H-1\} \\
& x^j_{t+H|t} \in \bigcup_{k=0}^{j-1} \mathcal{SS}^k_{\mathcal{G}}, \ \forall w^j_{t:t+H-1} \in \mathcal{W}^H \\
& x^j_{t:t+H|t} \in \mathcal{X}^{H+1}, \ \forall w^j_{t:t+H-1} \in \mathcal{W}^H
\end{aligned}
\tag{5.1.1}
$$

where $\pi_{t+i|t}$ is the i-th policy in the planning horizon conditioned on x^j_t and $\pi_{t:t+H-1|t} = \{\pi_{t|t},\dots,\pi_{t+H-1|t}\}$ (likewise for other optimization variables). Let the minimizer of (5.1.1) be $\pi^{*,j}_{t:t+H-1|t}$. Then, execute the first policy at x^j_t:

$$
u^j_t = \pi^j(x^j_t) = \pi^{*,j}_{t|t}(x^j_t) \tag{5.1.2}
$$

Solving 5.1.1 is typically intractable in practice, so we discuss practical approximations we make to the designed algorithm in Sect. 7.

5.2 Start State Expansion

We now describe the control strategy for expanding the controller domain. If there exists a policy π for which the H-step robust reachable set for the start states sampled at

iteration j is contained within the current safe set for goal set $\mathcal{G}$, then we can define the feasible set/domain for the controller at iteration j. The domain of π^j for $\mathcal{G}$ is computed by collecting the set of all states for which there exists a sequence of policies which robustly keep the system in $\bigcup_{k=0}^{j-1} \mathcal{SS}_{\mathcal{G}}^k$. Precisely, we define the controller domain as follows:

$$\mathcal{F}_{\mathcal{G}}^j = \{x \mid \exists \pi_{0:H-1} \in \Pi^H \text{ s.t. } \mathcal{R}_H^{\pi_{0:H-1}}(x) \subseteq \bigcup_{k=0}^{j-1} \mathcal{SS}_{\mathcal{G}}^k\} \tag{5.2.3}$$

This set defines the states from which the system can robustly plan back to $\bigcup_{k=0}^{j-1} \mathcal{SS}_{\mathcal{G}}^k$. Note that the controller domain is a function of the goal set $\mathcal{G}$. While any start state sampled from $\mathcal{F}_{\mathcal{G}}^j$ will ensure feasibility and convergence for goal set $\mathcal{G}$ (proven in Sect. 6), we present a method to compute states from $\mathcal{F}_{\mathcal{G}}^j \setminus \bigcup_{k=0}^{j-1} \mathcal{SS}_{\mathcal{G}}^k$ to expand $\mathcal{F}_{\mathcal{G}}^j$ towards a desired start state, which may not be added to the domain through task-directed exploration. Computing this set is intractable for general nonlinear stochastic systems, so we introduce the following method to approximate this.

At the end of iteration j, we sample a start state $x_S^j \in \bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$ and seek to execute a sequence of H' exploration policies $\pi_{E,0:H'-1}^j$ which carry the system outside of $\bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$ and then robustly back into $\bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$, for all noise realizations $w_{0:H'-2} \in \mathcal{W}^{H'-1}$ where $H' \geq 0$. The sequence of policies $\pi_{E,0:H'-1}^j$ is computed by solving an H'-step optimization problem with a cost function $C_E^j(x,u)$ that encourages exploration outside of $\bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$ while enforcing that the controller terminates in some state $x_{H'}^j \in \bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$. In Sect. 7, we discuss some possibilities for $C_E^j(x,u)$, implement one instantiation, and demonstrate that it enables adaptation to novel start states while maintaining controller feasibility. The sequence of controllers can computed by solving the following 1-step trajectory optimization problem:

$$\begin{aligned} \pi_{E,0:H'-1}^j = \underset{\pi_{0:H'-1} \in \Pi^{H'}}{\text{argmin}} \quad & \mathbb{E}_{w_{0:H'-2}^j} \left[\sum_{i=0}^{H'-1} C_E^j(x_i^j, \pi_i(x_i^j)) \right] \\ \text{s.t.} \quad & x_{i+1}^j = f(x_i^j, \pi_i(x_i^j), w_i), \ \forall i \in \{0, \ldots, H'-1\} \\ & x_{H'}^j \in \bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k, \ \forall w_{0:H'-2} \in \mathcal{W}^{H'-1} \\ & x_{0:H'}^j \in \mathcal{X}^{H'+1}, \ \forall w_{0:H'-2} \in \mathcal{W}^{H'-1} \end{aligned} \tag{5.2.4}$$

Let $\mathcal{M} = \left(\mathcal{R}_i^{\pi_{E,0:H'-1}^j}(x_S^j) \right)_{i=0}^{H'}$, the set of all states reachable in H' steps by π_E and let $\mathcal{M}_H = \left(\mathcal{R}_i^{\pi_{E,0:H'-1}^j}(x_S^j) \right)_{i=\max(H'-H,\,0)}^{H'}$. Note that $\forall x \in \mathcal{M}_H$, the controller initialized at x can be robustly guided to $\bigcup_{k=0}^{j} \mathcal{SS}_{\mathcal{G}}^k$ in H steps. At iteration $j+1$, feasible start states can be sampled from $\mathcal{M}_H$ to guide the policy's domain toward a desired target start state. An MPC policy π_E^j could be executed instead to generate these future start

states. We could also use the exploration policy to explicitly augment the value function $L^{\pi^j}_{\mathcal{G}}$ and safe set $\mathcal{SS}^j_{\mathcal{G}}$ and thus $\mathcal{F}^j_{\mathcal{G}}$. This could be used for general domain expansion instead of directed expansion towards a desired start state.

6 Properties of ABC-LMPC

In this section, we study the properties of the controller constructed in Sect. 5. For analysis, we will assume a fixed goal set $\mathcal{G}$, but note that if the goal set is changed at some iteration, the same properties still apply to the new goal set $\mathcal{H}$ by the same proofs, because all of the same assumptions hold for $\mathcal{H}$. See Appendix ? for all proofs.

Lemma 1. ***Recursive Feasibility:*** *Consider the closed-loop system* (5.1.1) *and* (5.1.2)*. Let the safe set $\mathcal{SS}^j_{\mathcal{G}}$ be defined as in* (4.1.2)*. If Assumptions 1–3 hold and $x^j_0 \in \mathcal{F}^j_{\mathcal{G}}$, then the controller induced by optimizing* (5.1.1) *and* (5.1.2) *is feasible almost surely for $t \geq 0$ and $j \geq 0$. Equivalently stated, $\underset{w^j_{0:H-1}}{\mathbb{E}} [J^j_{t\to t+H}(x^j_t)] < \infty, \forall t, j \geq 0$.*

Lemma 1 shows that the controller is guaranteed to satisfy state-space constraints for all timesteps t in all iterations j given the definitions and assumptions presented above. Equivalently, the expected planning cost of the controller is guaranteed to be finite. The following lemma establishes convergence in probability to the goal set given initialization within the controller domain.

Lemma 2. ***Convergence in Probability:*** *Consider the closed-loop system defined by* (5.1.1) *and* (5.1.2)*. Let the sampled safe set $\mathcal{SS}^j_{\mathcal{G}}$ be defined as in* (4.1.2)*. Let Assumptions 1–3 hold and $x^j_0 \in \mathcal{F}^j_{\mathcal{G}}$. If the closed-loop system converges in probability to $\mathcal{G}$ at iteration 0, then it converges in probability at all subsequent iterations. Stated precisely, at iteration j: $\lim_{t\to\infty} P(x^j_t \notin \mathcal{G}) = 0$.*

Theorem 1. ***Iterative Improvement:*** *Consider system* (3.0.1) *in closed-loop with* (5.1.1) *and* (5.1.2)*. Let the sampled safe set $\mathcal{SS}^j$ be defined as in* (4.1.2)*. Let Assumptions 1–3 hold, then the expected cost-to-go* (4.1.3) *associated with the control policy* (5.1.2) *is non-increasing in iterations for a fixed start state. More formally:*

$$\forall j \in \mathbb{N},\ x^j_0 \in \mathcal{F}^j_{\mathcal{G}},\ x^{j+1}_0 \in \mathcal{F}^{j+1}_{\mathcal{G}} \implies J^{\pi^j}(x^j_0) \geq J^{\pi^{j+1}}(x^{j+1}_0)$$

Furthermore, $\{J^{\pi^j}(x^j_0)\}_{j=0}^{\infty}$ is a convergent sequence.

Theorem 1 extends prior results [5], which guarantee robust iterative improvement for stochastic linear systems with convex costs and convex constraint sets. Here we show iterative improvement in *expectation* for ABC-LMPC for stochastic nonlinear systems with costs as in Assumption 1. The following result implies that the controller domain is non-decreasing.

Lemma 3. ***Controller domain expansion:*** *The domain of π^j is an non-decreasing sequence of sets: $\mathcal{F}^j_{\mathcal{G}} \subseteq \mathcal{F}^{j+1}_{\mathcal{G}}$.*

7 Practical Implementation

ABC-LMPC alternates between two phases at each iteration: the first phase performs the task by executing π^j and the second phase runs the exploration policy $\pi^j_{E,0:H'-1}$. Only data from π^j is added to an approximation of $\mathcal{SS}^j_{\mathcal{G}}$, on which the value function L^{π^j} is fit, but in principle, data from $\pi^j_{E,0:H'-1}$ can also be used. Although the task (5.1.1) and exploration (5.2.4) objectives are generally intractable, we present a simple algorithm which introduces sample-based approximations to expand the policy's domain $\mathcal{F}^j_{\mathcal{G}}$ while approximately maintaining theoretical properties in practice. Here, we describe how each component in the controller design is implemented and how optimization is performed. See Appendix ? for further implementation details.

7.1 Sample-Based Safe Set

In practice, as in [5], we approximate the safe set $\mathcal{SS}^j_{\mathcal{G}}$ using samples from the closed loop system defined by (5.1.1) and (5.1.2). To do this, we collect R closed-loop trajectories at iteration j, each of length T as in [5] where T is the task horizon.

Thus, given the ith disturbance realization sequence collected at iteration j, given by $\mathbf{w}^j_i = [w^j_{0,i}, \ldots w^j_{T,i}]$, we define the closed loop trajectory associated with this sequence as in [5]: $\mathbf{x}^j(\mathbf{w}^j_i) = \left[x^j_0(\mathbf{w}^j_i), \ldots, x^j_T(\mathbf{w}^j_i)\right]$. As in [5], we note that $x^j_k(\mathbf{w}^j_i) \in \mathcal{R}^{\pi^j}_k(x^j_0)$, so R rollouts from the closed-loop system provides a sample-based approximation to $\mathcal{R}^{\pi^j}_k(x^j_0)$ as follows: $\tilde{\mathcal{R}}^{\pi^j}_k(x^j_0) = \bigcup_{i=1}^R x^j_k(\mathbf{w}^j_i) \subseteq \mathcal{R}^{\pi^j}_k(x^j_0)$. Similarly, we can define a sample-based approximation to the safe set as follows: $\tilde{\mathcal{SS}}^j_{\mathcal{G}} = \left\{\bigcup_{k=0}^{\infty} \tilde{\mathcal{R}}^{\pi^j}_k(x^j_0) \cup \mathcal{G}\right\}$.

While $\tilde{\mathcal{SS}}^j_{\mathcal{G}}$ is not robust control invariant, with sufficiently many trajectory samples (i.e. R sufficiently big), this approximation becomes more accurate in practice [5]. To obtain a continuous approximation of the safe set for planning, we use the same technique as [1], and fit density model $\rho^{\mathcal{G}}_{\alpha}$ to $\bigcup_{k=0}^{j-1} \tilde{\mathcal{SS}}^k_{\mathcal{G}}$ and instead of enforcing the terminal constraint by checking if $x_{t+H} \in \bigcup_{k=0}^{j-1} \tilde{\mathcal{SS}}^k_{\mathcal{G}}$, ABC-LMPC instead enforces that $\rho^{\mathcal{G}}_{\alpha}(x_{t+H}) > \delta$, where α is a kernel width parameter. We implement a tophat kernel density model using a nearest neighbors classifier with tuned kernel width α and use $\delta = 0$ for all experiments. Thus, all states within Euclidean distance α from the closest state in $\bigcup_{k=0}^{j-1} \tilde{\mathcal{SS}}^k_{\mathcal{G}}$ are considered safe under $\rho^{\mathcal{G}}_{\alpha}$.

7.2 Start State Expansion Strategy

To provide a sample-based approximation to the procedure from Sect. 5.2, we sample states x^j_S from $\bigcup_{k=0}^{j} \tilde{\mathcal{SS}}^k$ and execute $\pi^j_{E,0:H'-1}$ for R trajectories of length H', which approximate $\mathcal{M}$. We repeat this process until an x^j_S is found such that all R sampled trajectories satisfy the terminal state constraint that $x^j_{H'} \in \bigcup_{k=0}^{j} \tilde{\mathcal{SS}}^k_{\mathcal{G}}$ (Sect. 5.2). Once such a state is found, a state is sampled from the last H steps of the corresponding trajectories to serve as the start state for the next iteration, which approximates sampling from $\mathcal{M}_H$. We utilize a cost function which encourages controller domain expansion

towards a specific desired start state x^*, although in general any cost function can be used. This cost function is interesting because it enables adaptation of a learning MPC controller to desired specifications while maintain controller feasibility. Precisely, we optimize a cost function which simply measures the discrepancy between a given state in a sampled trajectory and x^*, ie. $C_E^j(x,u) = D(x,x^*)$. This distance measure can be tuned on a task-specific basis based on the appropriate distance measures for the domain (Sect. 8.3). However, we remark that this technique requires: (1) an appropriate distance function $D(\cdot,\cdot)$ and (2) a reverse path from the goal to the start state, that may differ from the optimal forward path, along which the goal is robustly reachable.

7.3 Goal Set Transfer

We practically implement the goal set transfer strategy in Sect. 4.3 by fitting a new density model $\rho_\alpha^{\mathcal{H}}$ on the prefixes of prior trajectories that intersect some new user-specified goal set $\mathcal{H}$. If $\mathcal{H}$ is chosen such that $\tilde{\mathcal{SS}}_{\mathcal{H}}^j$ contains many states, the controller can seamlessly transfer to $\mathcal{H}$. If this is not the case, the controller domain for $\mathcal{H}$ must be expanded from $\mathcal{H}$ until it intersects many trajectories in the original domain.

7.4 ABC-LMPC Optimization Procedure

As in prior work on MPC for nonlinear control [1,2], we solve the MPC optimization problem in (5.1.1) over sampled open loop sequences of controls using the cross entropy method (CEM) [31]. In practice, we implement the terminal safe set constraints and state-space constraints in (5.1.1) and (5.2.4) by imposing a large cost on sampled action sequences which violate constraints when performing CEM. We use a probabilistic ensemble of 5 neural networks to approximate $L_{\mathcal{G}}^{\pi^j}(x)$ as in [1]. In contrast to [1], a separate $L_{\mathcal{G}}^{\pi^j}(x)$ is fit using data from each iteration instead of fitting a single function approximator on all data. We utilize Monte Carlo estimates of the cost-to-go values when fitting $L_{\mathcal{G}}^{\pi^j}(x)$. Each element of the ensemble outputs the parameters of a conditional axis-aligned Gaussian distribution and are trained on bootstrapped samples from the training dataset using a maximum likelihood [2].

8 Experiments

We evaluate whether ABC-LMPC can enable (1) iterative improvement in expected performance for stochastic nonlinear systems, (2) adaptation to new start states and (3) transfer to new goal sets on 3 simulated continuous control domains. In Sect. 8.1 we describe the experimental domains, in Sect. 8.2, we evaluate the controller with fixed start states and goal sets, in Sect. 8.3, we expand the controller domain iteratively toward a desired start state far from the goal set, in Sect. 8.4, we switch the goal set during learning, and finally in Sect. 8.5 we utilize both start state expansion and the goal set transfer technique to control a pendulum to an upright position. In all experiments, we use $C(x,u) = \mathbb{1}\{x \notin \mathcal{G}\}$ as in [1]. Note that for this cost function, the maximum trajectory cost is the task horizon T, and the resulting objective corresponds to minimum time optimal control. We include comparisons to the minimum trajectory cost

achieved by the state-of-the-art demonstration augmented model-based reinforcement learning algorithm, SAVED [1] after 10 iterations of training to evaluate the quality of the learned controller. For all experiments, we use $R = 5$ closed-loop trajectories from the current controller to estimate $\tilde{\mathcal{S}}\tilde{\mathcal{S}}^j_{\mathcal{G}}$ and perform start state expansion. Experimental domains have comparable stochasticity to those in [5]. See Appendix ? for further details about experimental, optimization, and environment parameters.

8.1 Experimental Domains

See Fig. 2.

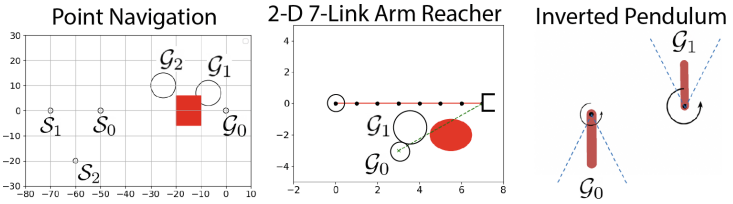


Fig. 2. Experimental Domains: We evaluate ABC-LMPC on three stochastic domains: a navigation domain with an obstacle, a 2D 7-link arm reacher domain with an obstacle, and an inverted pendulum domain. In the first two domains, suboptimal demonstrations are provided, while no demonstrations are provided for the inverted pendulum task.

Point Mass Navigation: We consider a 4-dimensional (x, y, v_x, v_y) navigation task as in [1], in which a point mass is navigating to a goal set (a unit ball centered at the origin unless otherwise specified). The agent exerts force (f_x, f_y), $\|(f_x, f_y)\| \leq 1$, in each cardinal direction and experiences drag coefficient ψ. We introduce truncated Gaussian process noise $z_t \sim \mathcal{N}(0, \sigma^2 I)$ in the dynamics with domain $[-\sigma, \sigma]$. We include a large obstacle in the center of the environment that the robot must navigate around to reach the goal. While this task has linear dynamics, the algorithm must consider non-convex state space constraints and stochasticity.

7-Link Arm Reacher: Here, we consider a 2D kinematic chain with 7 joints where the agent provides commands in delta joint angles. We introduce truncated Gaussian process noise $z_t \sim \mathcal{N}(0, \sigma^2 I)$ in the dynamics with domain $[-\sigma, \sigma]$ and build on the implementation from [32]. The goal is to control the end effector position to a 0.5 radius circle in $\mathbb{R}^2$ centered at $(3, -3)$. We do not model self-collisions but include a circular obstacle of radius 1 in the environment which the kinematic chain must avoid.

Inverted Pendulum: This environment is a noisy inverted pendulum task adapted from OpenAI Gym [33]. We introduce truncated Gaussian process noise in the dynamics.

8.2 Fixed Start and Goal Conditions

We first evaluate ABC-LMPC on the navigation and reacher environments with a fixed start state and goal set. In the navigation domain, the robot must navigate from

$\mathcal{S}_0 = (-50, 0, 0, 0)$ to the origin ($\mathcal{G}_0$) while in the reacher domain, the agent must navigate from a joint configuration with the end effector at $(7, 0)$ to one with the end effector at $(3, -3)$ ($\mathcal{G}_1$). For optimization parameters and other experimental details, see Appendix ?. The controller rapidly and significantly improves upon demonstrations for both domains (Fig. 3). The controller achieves comparable cost to SAVED for both tasks and never violates constraints during learning.

Fig. 3. Fixed Start, Single Goal Set Experiments: Learning curves for ABC-LMPC averaged over $R = 5$ rollouts per iteration on simulated continuous control domains when the start state and goal set is held fixed during learning. Performance of the demonstrations is shown at iteration 0, and the controller performance is shown thereafter. **Point Mass Navigation:** The controller immediately improves significantly upon the demonstration performance within 1 iteration, achieving a mean trajectory cost of around 20 while demonstrations have mean trajectory cost of 42.58. **7-Link Arm Reacher:** The controller significantly improves upon the demonstrations, achieving a final trajectory cost of around 18 while demonstrations achieve a mean trajectory cost of 37.77. In all experiments, the controller quickly converges to the best cost produced by SAVED.

8.3 Start State Expansion

ABC-LMPC is now additionally provided a target start state which is initially outside its domain and learns to iteratively expand its domain toward the desired start state. We report the sequence of achieved start states over iterations in addition to the mean and standard deviation trajectory cost. ABC-LMPC is able to maintain feasibility throughout learning and achieve comparable performance to SAVED at the final start state when SAVED is supplied with 100 demonstrations from the desired state. To ensure that results are meaningful, we specifically pick desired start states such that given 100 demonstrations from the original start state, ABC-LMPC is never able to accomplish the task after 30 iterations of learning. A different $C_E(x, u)$ is used for start state expansion based on an appropriate distance metric for each domain.

We first consider a navigation task where 100 suboptimal demonstrations are supplied from $(-25, 0, 0, 0)$ with average trajectory cost of 44.76. The goal is to expand the controller domain in order to navigate from start states $\mathcal{S}_1 = (-70, 0, 0, 0)$ and $\mathcal{S}_2 = (-60, -20, 0, 0)$. $C_E(x, u)$ measures the Euclidean distance between the positions of x and those of the desired start state. After 20 iterations, the controller reaches the desired start state while consistently maintaining feasibility during learning (Table 1).

We then consider a similar Reacher task using the same suboptimal demonstrations from Sect. 8.2. The desired start end effector position is $(-1,0)$, and $C_E(x,u)$ measures the Euclidean distance between the end effector position of states x in optimized trajectories and that of the desired start state. Within 16 iterations of learning, the controller is able to start at the desired start state while maintaining feasibility during learning (Table 1). On both domains, the controller achieves comparable performance to SAVED when trained with demonstrations from that start state and the controller successfully expands its domain while rapidly achieving good performance at the new states. Constraints are never violated during learning for all experiments.

Table 1. Start State Expansion Experiments: Pointmass Navigation: Start State Expansion towards position $(-70,0)$ (left) and $(-60,-20)$ (center). Here we see that ABC-LMPC is able to reach the desired start state in both cases while consistently maintaining controller feasibility throughout learning. Furthermore, the controller achieves competitive performance with SAVED, which achieves a minimum trajectory cost of 21 from $(-70,0)$ and 23 from $(-60,-20)$; **7-link Arm Reacher:** Here we expand the start state from that corresponding to an end effector position of $(7,0)$ to that corresponding to an end effector position of $(-1,0)$ (right). Again, we see that the controller consistently maintains feasibility during learning and achieves trajectory costs comparable to SAVED, which achieves a minimum trajectory cost of 24. The trajectory costs are presented in format: mean $\pm$ standard deviation over $R = 5$ rollouts.

Point Navigation (-70, 0)

Iteration	Start Pos (x, y)	Trajectory Cost
4	$(-42.3, 1.33)$	23.0 ± 0.89
8	$(-54.1, 0.08)$	22.8 ± 1.67
12	$(-61.2, 2.70)$	25.0 ± 2.37
16	$(-70.3, -0.26)$	32.6 ± 5.08
20	$(-70.4, 0.12)$	29.4 ± 2.33

Point Navigation (-60, -20)

Iteration	Start Pos (x, y)	Trajectory Cost
4	$(-42.6, -8.76)$	19.6 ± 4.22
8	$(-54.6, -14.2)$	25.6 ± 5.23
12	$(-58.8, -20.3)$	27.2 ± 12.0
16	$(-60.6, -20.2)$	21.0 ± 0.63
20	$(-60.5, -19.6)$	22.4 ± 1.85

7-Link Reacher

Iteration	Start EE Position	Trajectory Cost
4	$(-1.28, -0.309)$	31.6 ± 8.04
8	$(-0.85, -0.067)$	30.8 ± 15.7
12	$(-0.95, -0.014)$	20.2 ± 1.83
16	$(-1.02, -0.023)$	19.4 ± 4.03

8.4 Goal Set Transfer

ABC-LMPC is trained as in Sect. 8.2, but after a few iterations, the goal set is changed to a new goal set that is in the controller domain. In the navigation domain, the robot is supplied a new goal set centered at $\mathcal{G}_1 = (-25,10,0,0)$ or $\mathcal{G}_2 = (-7,7,0,0)$ with radius 7 after 2 iterations of learning on the original goal set. We increase the radius so more prior trajectories can be reused for the new goal-conditioned value function. Results are shown in Fig. 4 for both goal set transfer experiments. We also perform a goal set transfer experiment on the 7-link Reacher Task in which the robot is supplied a new goal set centered at $\mathcal{G}_1 = (4,0.2)$ with radius 1 after 2 iterations of training. Results are shown in Fig. 4. In both domains, ABC-LMPC seamlessly transfers to the new goal by leveraging prior experience to train a new set of value functions.

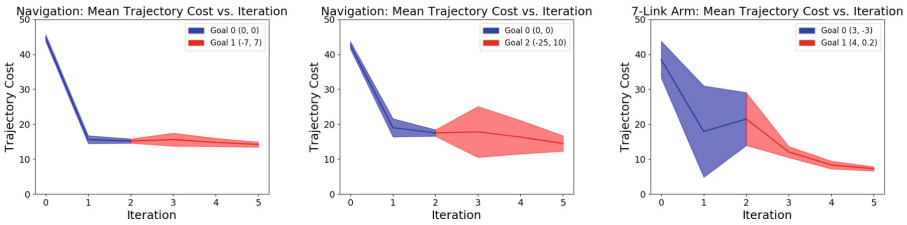


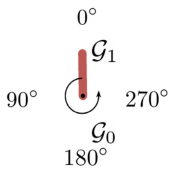
Fig. 4. Goal Set Transfer Learning: In this experiment, the goal set is switched to a new goal set at iteration 3 and we show a learning curve which indicates performance on both the first goal (blue) and new goal (red). The controller is re-trained as in Sect. 7.3 to stabilize to the new goal. The controller immediately is able to perform the new task and never hits the obstacle. Results are plotted over $R = 5$ controller rollouts per iteration.

8.5 Inverted Pendulum Swing-Up Task

In this experiment, we incorporate both the start state optimization procedure and goal set transfer strategies to balance a pendulum in the upright position, but without any demonstrations. We initialize the pendulum in the downward orientation ($\mathcal{G}_0$), and the goal of the task is to eventually stabilize the system to the upright orientation ($\mathcal{G}_1$). We iteratively expand the controller domain using the start state expansion strategy with initial goal $\mathcal{G}_0$ until the pendulum has swung up sufficiently close to the upright orientation. Once this is the case, we switch the goal set to $\mathcal{G}_1$ to stabilize to the upright position. The controller seamlessly transitions between the two goal sets, immediately transitioning to $\mathcal{G}_1$ while completing the task (convergence to either $\mathcal{G}_0$ or $\mathcal{G}_1$ within the task horizon) on all iterations (Table 2). $C_E(x,u)$ measures the distance between the pendulum's orientation and the desired start state's orientation.

Table 2. Pendulum Swing Up Experiment: We iteratively expand the controller domain outward from a goal set centered around the downward orientation ($\mathcal{G}_0$) towards the upward orientation until the controller domain includes a goal set centered around the upward orientation ($\mathcal{G}_1$). Then, the goal set is switched to $\mathcal{G}_1$. The resulting controller maintains feasibility throughout and seamlessly transitions to $\mathcal{G}_1$. The trajectory costs are presented as: mean $\pm$ standard deviation over $R = 5$ rollouts. The upward orientation corresponds to a pendulum angle of 0° and the angle (degrees) increases counterclockwise from this position until 360°.

Iteration	Start Angle	Goal Set	Trajectory Cost
3	200.3	$\mathcal{G}_0$	30.2 ± 1.47
6	74.3	$\mathcal{G}_0$	35.0 ± 0.00
9	53.9	$\mathcal{G}_0$	34.4 ± 0.49
12	328.1	$\mathcal{G}_1$	36.0 ± 0.63
15	345.1	$\mathcal{G}_1$	13.8 ± 7.03
18	0.6	$\mathcal{G}_1$	0.00 ± 0.00



9 Discussion and Future Work

We present a new algorithm for iteratively expanding the set of feasible start states and goal sets for an LMPC-based controller and provide theoretical guarantees on iterative improvement in expectation for non-linear systems under certain conditions on the cost function and demonstrate its performance on stochastic linear and nonlinear continuous control tasks. In future work, we will explore synergies with sample based motion planning to efficiently generate asymptotically optimal plans. We will also integrate the reachability-based domain expansion strategies of ABC-LMPC with model-based RL to learn safe and efficient controllers when dynamics are learned from experience.

Acknowledgements. This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab. Authors were also supported by the Scalable Collaborative Human-Robot Learning (SCHooL) Project, a NSF National Robotics Initiative Award 1734633, and in part by donations from Google and Toyota Research Institute. Ashwin Balakrishna is supported by an NSF GRFP. This article solely reflects the opinions and conclusions of its authors and does not reflect the views of the sponsors. We thank our colleagues who provided helpful feedback and suggestions, especially Michael Danielczuk, Daniel Brown and Suraj Nair.

References

1. Thananjeyan, B., Balakrishna, A., Rosolia, U., Li, F., McAllister, R., Gonzalez, J.E., Levine, S., Borrelli, F., Goldberg, K.: Safety augmented value estimation from demonstrations (SAVED): safe deep model-based RL for sparse cost robotic tasks. IEEE Robot. Autom. Lett. **5**(2), 3612–3619 (2020)
2. Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Proceedings of Advances in Neural Information Processing Systems (2018)
3. Nagabandi, A., Konolige, K., Levine, S., Kumar, V.: Deep dynamics models for learning dexterous manipulation. In: Conference on Robot Learning (CoRL) (2019)
4. Balakrishna, A., Thananjeyan, B., Lee, J., Li, F., Zahed, A., Gonzalez, J.E., Goldberg, K.: On-policy robot imitation learning from a converging supervisor. In: Conference on Robot Learning (CoRL) (2019)
5. Rosolia, U., Borrelli, F.: Sample-based learning model predictive control for linear uncertain systems. CoRR (2019). arXiv: 1904.06432
6. Rosolia, U., Zhang, X., Borrelli, F.: Robust learning model predictive control for iterative tasks: learning from experience. In: Annual Conference on Decision and Control (CDC) (2017)
7. Rosolia, U., Borrelli, F.: Learning model predictive control for iterative tasks. A data-driven control framework. IEEE Trans. Autom. Control **63**(7), 1883–1896 (2018)
8. Rosolia, U., Borrelli, F.: Learning how to autonomously race a car: a predictive control approach. In: IEEE (2019)
9. Aswani, A., Gonzalez, H., Sastry, S., Tomlin, C.: Provably safe and robust learning-based model predictive control. Automatica **49**(5), 1216–1226 (2013)
10. Kocijan, J., Murray-Smith, R., Rasmussen, C.E., Girard, A.: Gaussian process model based predictive control (2004)
11. Koller, T., Berkenkamp, F., Turchetta, M., Krause, A.: Learning based model predictive control for safe exploration and reinforcement learning (2018)

12. Hewing, L., Liniger, A., Zeilinger, M.: Cautious NMPC with gaussian process dynamics for autonomous miniature race cars (2018)
13. Terzi, E., Fagiano, L., Farina, M., Scattolini, R.: Learning-based predictive control for linear systems: a unitary approach. Automatica **108**, 108473 (2019)
14. Hewing, L., Kabzan, J., Zeilinger, M.N.: Cautious model predictive control using Gaussian process regression. IEEE Trans. Control Syst. Technol. (2019)
15. Kocijan, J., Murray-Smith, R., Rasmussen, C.E., Girard, A.: Gaussian process model based predictive control. In: Proceedings of the 2004 American Control Conference (2004)
16. Bacic, M., Cannon, M., Lee, Y.I., Kouvaritakis, B.: General interpolation in MPC and its advantages. IEEE Trans. Autom. Control **48**(6), 1092–1096 (2003)
17. Brunner, F.D., Lazar, M., Allgöwer, F.: Stabilizing linear model predictive control: on the enlargement of the terminal set. In: 2013 European Control Conference (ECC) (2013)
18. Wabersich, K.P., Zeilinger, M.N.: Linear model predictive safety certification for learning-based control. In: 2018 IEEE Conference on Decision and Control (CDC) (2018)
19. Blanchini, F., Pellegrino, F.A.: Relatively optimal control and its linear implementation. IEEE Trans. Autom. Control **48**(12), 2151–2162 (2003)
20. Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., Mordatch, I.: Plan online, learn offline: efficient learning and exploration via model-based control. In: Proceedings of International Conference on Machine Learning (2019)
21. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. In: Conference on Robot Learning (CoRL) (2017)
22. Resnick, C., Raileanu, R., Kapoor, S., Peysakhovich, A., Cho, K., Bruna, J.: Backplay: "man muss immer umkehren". CoRR (2018). arXiv: 1807
23. Narvekar, S., Stone, P.: Learning curriculum policies for reinforcement learning. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (2019)
24. Ivanovic, B., Harrison, J., Sharma, A., Chen, M., Pavone, M.: BaRC: backward reachability curriculum for robotic reinforcement learning. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2019)
25. Nair, A.V., Pong, V., Dalal, M., Bahl, S., Lin, S., Levine, S.: Visual reinforcement learning with imagined goals. In: Proceedings Advances in Neural Information Processing Systems (2018)
26. Schaul, T., Horgan, D., Gregor, K., Silver, D.: Universal value function approximators. In: Proceedings of International Conference on Machine Learning (2015)
27. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O.P., Zaremba, W.: Hindsight experience replay. In: Advances in Neural Information Processing Systems (2017)
28. van den Berg, J., Abbeel, P., Goldberg, K.Y.: LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. Int. J. Robot. Res. **30**(7), 895–913 (2011)
29. Lee, A., Patil, S., Schulman, J., McCarthy, Z., Berg, J., Goldberg, K., Abbeel, P.: Gaussian belief space planning for imprecise articulated robots. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2013)
30. Kurniawati, H., Du, Y., Hsu, D., Lee, W.S.: Motion planning under uncertainty for robotic tasks with long time horizons. Int. J. Robot. Res. **30**(3), 308–323 (2011)
31. Botev, Z.I., Kroese, D.P., Rubinstein, R.Y., Faculty of Industrial Engineering: The Cross-Entropy Method for Optimization (2013)
32. Sakai, A., Ingram, D., Dinius, J., Chawla, K., Raffin, A., Paques, A.: PythonRobotics: a Python code collection of robotics algorithms (2018)
33. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016). eprint: arXiv:1606.01540