MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Robust Optimization for Trajectory-Centric Model-based Reinforcement Learning

Jha, Devesh K.; Kolaric, Patrik; Romeres, Diego; Raghunathan, Arvind; Benosman, Mouhacine; Nikovski, Daniel N.

TR2019-156 December 18, 2019

Abstract

This paper presents a method to perform robust trajectory optimization for trajectory-centric Model-based Reinforcement Learning (MBRL). We propose a method that allows us to use the uncertainty estimates present in predictions obtained from a model-learning algorithm to generate robustness certificates for trajectory optimization. This is done by simultaneously solving for a time-invariant controller which is optimized to satisfy a constraint to generate the robustness certificate. We first present a novel formulation of the proposed method for the robust optimization that incorporates use of local sets around a trajectory where the closed-loop dynamics of the system is stabilized using a time-invariant policy. The method is demonstrated on an inverted pendulum system with parametric uncertainty. A Gaussian process is used to learn the residual dynamics and the uncertainty sets generated by the Gaussian process are then used to generate the trajectories with the local stabilizing policy.

NeurIPS Workshop on Safety and Robustness in Decision Making

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Robust Optimization for Trajectory-Centric Model-based Reinforcement Learning

Patrik Kolaric
Univ. of Texas at Arlington
Fort Worth, TX
patrik.kolaric@mavs.uta.edu

Devesh K. Jha
MERL
Cambridge, MA
jha@merl.com

Diego Romeres
MERL
Cambridge, MA
romeres@merl.com

Arvind U. Raghunathan
MERL
Cambridge, MA
raghunathan@merl.com

Mouhacine Benosman
MERL
Cambridge, MA
benosman@merl.com

Daniel Nikovski
MERL
Cambridge, MA
nikovski@merl.com

Abstract

This paper presents a method to perform robust trajectory optimization for trajectory-centric Model-based Reinforcement Learning (MBRL). We propose a method that allows us to use the uncertainty estimates present in predictions obtained from a model-learning algorithm to generate robustness certificates for trajectory optimization. This is done by simultaneously solving for a time-invariant controller which is optimized to satisfy a constraint to generate the robustness certificate. We first present a novel formulation of the proposed method for the robust optimization that incorporates use of local sets around a trajectory where the closed-loop dynamics of the system is stabilized using a time-invariant policy. The method is demonstrated on an inverted pendulum system with parametric uncertainty. A Gaussian process is used to learn the residual dynamics and the uncertainty sets generated by the Gaussian process are then used to generate the trajectories with the local stabilizing policy.

1 Introduction

Reinforcement learning (RL) is a learning framework that addresses sequential decision-making problems, wherein an 'agent' or a decision maker learns a policy to optimize a long-term reward by interacting with the (unknown) environment. At each step, the RL agent obtains evaluative feedback (called reward or cost) about the performance of its action, allowing it to improve the performance of subsequent actions Sutton and Barto [2018], Vrabie et al. [2013]. Although RL has witnessed huge successes in recent times Silver et al. [2016, 2017], there are several unsolved challenges, which restrict the use of these algorithms for industrial systems. In most practical applications, control policies must be designed to satisfy operational constraints, and a satisfactory policy should be learnt in a data-efficient fashion Vamvoudakis et al. [2015].

Model-based reinforcement learning (MBRL) methods Deisenroth and Rasmussen [2011] learn a model from exploration data of the system, and then exploit the model to synthesize a trajectory-centric controller for the system Levine and Koltun [2013]. These techniques are, in general, harder to train, but could achieve good data efficiency Levine et al. [2016]. Learning reliable models is very challenging for non-linear systems and thus, the subsequent trajectory optimization could fail when using inaccurate models. However, modern machine learning methods such as Gaussian processes (GP), stochastic neural networks (SNN), etc. can generate uncertainty estimates associated with predictions Rasmussen [2003], Romeres et al. [2019]. These uncertainty estimates could be used to

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

estimate the confidence set of system states at any step along a given controlled trajectory for the system. The idea presented in this paper considers the stabilization of the trajectory using a local feedback policy that acts as an attractor for the system in the known region of uncertainty along the trajectory Tedrake et al. [2010].

We present a method for simultaneous trajectory optimization and local policy optimization, where the policy optimization is performed in a neighborhood (local sets) of the system states along the trajectory. These local sets could be obtained by a stochastic function approximator (e.g., GP, SNN, etc.) that used to learn the forward model of the dynamical system Romeres et al. [2019], Romeres et al. [2019]. The local policy is obtained by considering the worst-case deviation of the system from the nominal trajectory at every step along the trajectory. Performing simultaneous trajectory and policy optimization could allow us to exploit the modeling uncertainty as it drives the optimization to regions of low uncertainty, where it might be easier to stabilize the trajectory. This allows us to constrain the trajectory optimization procedure to generate robust, high-performance controllers. The proposed method automatically incorporates state and input constraints on the dynamical system.

Contributions. The main contributions of the current paper are:

1. We present a novel formulation of simultaneous trajectory optimization and time-invariant local policy synthesis for stabilization.
2. We demonstrate the proposed method on a non-linear pendulum system where Gaussian process is used to generate the uncertainty estimates in the learned model.

2 Related Work

MBRL has raised a lot of interest recently in robotics applications, because model learning algorithms are largely task independent and data-efficient Wang et al. [2019], Levine et al. [2016], Deisenroth and Rasmussen [2011]. However, MBRL techniques are generally considered to be hard to train and likely to result in poor performance of the resulting policies/controllers, because the inaccuracies in the learned model could guide the policy optimization process to low-confidence regions of the state space. For non-linear control, the use of trajectory optimization techniques such as differential dynamic programming Jacobson [1968] or its first-order approximation, the iterative Linear Quadratic Regulator (iLQR) Tassa et al. [2012] is very popular, as it allows the use of gradient-based optimization, and thus could be used for high-dimensional systems. As the iLQR algorithm solves the local LQR problem at every point along the trajectory, it also computes a sequence of feedback gain matrices to use along the trajectory. However, the LQR problem is not solved for ensuring robustness, and furthermore the controller ends up being time-varying, which makes its use somewhat inconvenient for robotic systems. Thus, we believe that the controllers we propose might have better stabilization properties, while also being time-invariant.

Most model-based methods use a function approximator to first learn an approximate model of the system dynamics, and then use stochastic control techniques to synthesize a policy. Some of the seminal work in this direction could be found in Levine et al. [2016], Deisenroth and Rasmussen [2011]. The method proposed in Levine et al. [2016] has been shown to be very effective in learning trajectory-based local policies by sampling several initial conditions (states) and then fitting a neural network which can imitate the trajectories by supervised learning. This can be done by using ADMM Boyd et al. [2011] to jointly optimize trajectories and learn the neural network policies. This approach has achieved impressive performance on several robotic tasks Levine et al. [2016]. The method has been shown to scale well for systems with higher dimensions. Several different variants of the proposed method were introduced later Chebotar et al. [2017], Montgomery and Levine [2016], Nagabandi et al. [2018]. However, no theoretical analysis could be provided for the performance of the learned policies.

Another set of seminal work related to the proposed work is on the use of sum-of-square (SOS) programming methods for generating stabilizing controller for non-linear systems Tedrake et al. [2010]. In these techniques, a stabilizing controller, expressed as a polynomial function of states, for a non-linear system is generated along a trajectory by solving an optimization problem to maximize its region of attraction Majumdar et al. [2013].

Some other approaches to trajectory-centric policy optimization could be found in Theodorou et al. [2010]. These techniques use path integral optimal control with parameterized policy representa-

tions such as dynamic movement primitives (DMPs) Ijspeert et al. [2013] to learn efficient local policies Williams et al. [2017]. However, these techniques do not explicitly consider the local sets where the controller robustness guarantees could be provided, either. Consequently, they cannot exploit the structure in the model uncertainty.

3 Problem Formulation

In this section, we describe the problem studied in the rest of the paper. To perform trajectory-centric control, we propose a novel formulation for simultaneous design of open-loop trajectory and a time-invariant, locally stabilizing controller that is robust to bounded model uncertainties and/or system measurement noise. As we will present in this section, the proposed formulation is different from that considered in the literature in the sense it allows us to exploit sets of possible deviation of a system to design stabilizing controller.

3.1 Trajectory Optimization as Non-linear Program

Consider the discrete-time dynamical system

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$ are the differential states and controls, respectively. The function $f : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ governs the evolution of the differential states. Note that the discrete-time formulation (1) can be obtained from a continuous time system $\dot{x} = \hat{f}(x, u)$ by using the *explicit Euler* integration scheme $(x_{k+1} - x_k) = \Delta t \hat{f}(x_k, u_k)$ where Δt is the time-step for integration.

For clarity of exposition we have limited our focus to discrete-time dynamical systems of the form in (1) although the techniques we describe can be easily extended to implicit discretization schemes.

In typical applications the states and controls are restricted to lie in sets $\mathcal{X} := \{x \in \mathbb{R}^{n_x} \,|\, \underline{x} \leq x \leq \overline{x}\} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} := \{u \in \mathbb{R}^{n_u} \,|\, \underline{u} \leq u \leq \overline{u}\} \subseteq \mathbb{R}^{n_u}$, *i.e.* $x_k \in \mathcal{X}, u_k \in \mathcal{U}$. We use $[K]$ to denote the index set $\{0, 1, \ldots, K\}$. Further, there may exist nonlinear inequality constraints of the form

$$g(x_k) \geq 0 \tag{2}$$

with $g : \mathbb{R}^{n_x} \to \mathbb{R}^m$. The inequalities in (2) are termed as *path constraints*. The trajectory optimization problem is to manipulate the controls u_k over a certain number of time steps $[T-1]$ so that the resulting trajectory $\{x_k\}_{k\in[T]}$ minimizes a cost function $c(x_k, u_k)$. Formally, we aim to solve the *trajectory optimization problem*

$$\begin{aligned} \min_{x_k, u_k} \; & \sum_{k\in[T]} c(x_k, u_k) \\ \text{s.t. Eq. (1)} - \text{(2) for } k \in [T] & \\ x_0 = \tilde{x}_0 & \\ x_k \in \mathcal{X} \text{ for } k \in [T] & \\ u_k \in \mathcal{U} \text{ for } k \in [T-1] & \end{aligned} \tag{TrajOpt}$$

where $\tilde{x}_0$ is the differential state at initial time $k = 0$. Before introducing the main problem of interest, we would like to introduce some notations.

In the following text, we use the following shorthand notation, $||v||_M^2 = v^T M v$. We denote the nominal trajectory as $X \equiv x_0, x_1, x_2, x_3, \ldots, x_{T-1}, x_T$, $U \equiv u_0, u_1, u_2, u_3, ..., u_{T-1}$. The actual trajectory followed by the system is denoted as $\hat{X} \equiv \hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_{T-1}, \hat{x}_T$. We denote a local policy as π_W, where π is the policy and W denotes the parameters of the policy. The trajectory cost is also sometimes denoted as $J = \sum_{k\in[T]} c(x_k, u_k)$.

3.2 Trajectory Optimization with Local Stabilization

This subsection introduces the main problem of interest in this paper. A schematic of the problem studied in the paper is also shown in Figure 1. In the rest of this section, we will describe how we can

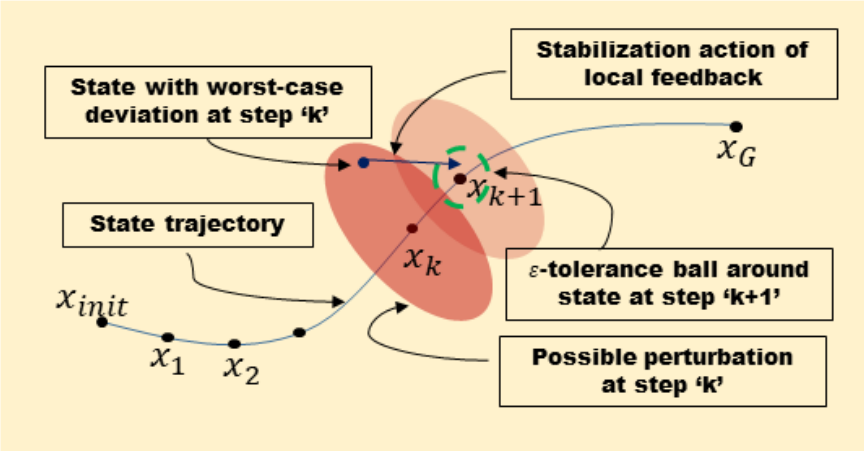


Figure 1: A schematic representation of the time-invariant local control introduced in this paper.
simplify the trajectory optimization and local stabilization problem and turn it into an optimization problem that can be solved by standard non-linear optimization solvers.

Consider the case where the system dynamics, f is only partially known, and the known component of f is used to design the controller. Consider the deviation of the system at any step 'k' from the state trajectory X and denote it as $\delta x_k \equiv x_k - \hat{x}_k$. We introduce a local (time-invariant) policy π_W that regulates the local trajectory deviation δx_k and thus, the final controller is denoted as $\hat{u}_k = u_k + \pi_W(\delta x_k)$. The closed-loop dynamics for the system under this control is then given by the following:

$$\hat{x}_{k+1} = f(\hat{x}_k, \hat{u}_k) = f(x_k + \delta x_k, u_k + \pi_W(\delta x_k)) \tag{3}$$

The main objective of the paper is to find the time-invariant feedback policy π_W that can stabilize the open-loop trajectory X locally within $\mathbb{R}_k \subset \mathbb{R}^{n_x}$ where $\mathbb{R}_k$ defines the set of uncertainty for the deviation δx_k. The uncertainty region $\mathbb{R}_k$ can be approximated by fitting an ellipsoid to the uncertainty estimate using a diagonal positive definite matrix S_k such that $\mathbb{R}_k = \{\delta x_k : \delta x_k^T S_k \delta x_k \leq 1\}$. The general optimization problem that achieves that is proposed as:

$$\begin{aligned} J^* = \underset{U,X,W}{min} \; & \underset{\delta x_k \in \mathbb{R}_k}{\mathbb{E}} [J(X + \delta X, U + \pi_W(\delta x_k)] \\ & x_{k+1} = \hat{f}(x_k, u_k) \end{aligned} \tag{4}$$

where $\hat{f}(\cdot, \cdot)$ denotes the known part of the model. Note that in the above equation, we have introduced additional optimization parameters corresponding to the policy π_W when compared to TrajOpt in the previous section. However, to solve the above, one needs to resort to sampling in order to estimate the expected cost. Instead we introduce a constraint that solves for the worst-case cost for the above problem.

Robustness Certificate. The robust trajectory optimization problem is to minimize the trajectory cost while at the same time satisfying a *robust constraint* at every step along the trajectory. This is also explained in Figure 1, where the purpose of the local stabilizing controller is to push the max-deviation state at every step along the trajectory to ϵ-tolerance balls around the trajectory. Mathematically, we express the problem as following:

$$\begin{aligned} \min_{x_k, u_k, W} \; & \sum_{k \in [T]} c(x_k, u_k) \\ \text{s.t. Eq. (1)} - \text{(2) for } & k \in [T] \\ & x_0 = \tilde{x}_0 \\ & x_k \in \mathcal{X} \text{ for } k \in [T] \\ & u_k \in \mathcal{U} \text{ for } k \in [T-1] \\ & \max_{\delta x_k \in \mathbb{R}_k} ||x_{k+1} - f(x_k + \delta x_k, u_k + \pi_W(\delta x_k))||_2 \leq \epsilon_k \end{aligned} \tag{RobustTrajOpt}$$

The additional constraint introduced in RobustTrajOpt allows us to ensure stabilization of the trajectory by estimating parameters of the stabilizing policy π_W. It is easy to see that RobustTrajOpt

solves the worst-case problem for the optimization considered in (4). However, RobustTrajOpt introduces another hyperparameter to the optimization problem, ϵ_k. In the rest of the paper, we refer to the following constraint as the *robust constraint*:

$$\max_{\delta x_k^T S_k \delta x_k \leq 1} ||x_{k+1} - f(x_k + \delta x_k, u_k + \pi_W(\delta x_k))||_2 \leq \epsilon_k \tag{5}$$

Solution of the *robust constraint* for generic non-linear system is out of scope of this paper. Instead, we linearize the trajectory deviation dynamics as shown in the following Lemma.

Lemma 1. *The trajectory deviation dynamics $\delta x_{k+1} = x_{k+1} - \hat{x}_{k+1}$ approximated locally around the optimal trajectory (X, U) are given by*

$$\begin{aligned} \delta x_{k+1} &= A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k) \\ A(x_k, u_k) &\equiv \nabla_{x_k} \hat{f}(x_k, u_k) \\ B(x_k, u_k) &\equiv \nabla_{u_k} \hat{f}(x_k, u_k) \end{aligned} \tag{6}$$

Proof. Use Taylor's series expansion to obtain the desired expression.

To ensure feasibility of the RobustTrajOpt problem and avoid tuning the hyperparameter ϵ_k, we make another relaxation by removing the *robust constraint* from the set of constraints and move it to the objective function. Thus, the simplified robust trajectory optimization problem that we solve in this paper can be expressed as following (we skip the state constraints to save space).

$$\begin{aligned} &\min_{x_k, u_k, W} \Big(\sum_{k \in [T]} c(x_k, u_k) + \alpha \sum_{k \in [T]} d_{max,k} \Big) \\ &\text{s.t. Eq. (1) } - \text{ (2) for } k \in [T] \end{aligned} \tag{RelaxedRobustTrajOpt}$$

where the term $d_{max,k}$ is defined as following after linearization.

$$d_{max,k} \equiv \max_{\delta x_k^T S_k \delta x_k \leq 1} ||A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k)||_P^2 \tag{7}$$

Note that the matrix P allows to weigh states differently. In the next section, we present the solution approach to compute the gradient for the RelaxedRobustTrajOpt which is then used to solve the optimization problem. Note that this results in simultaneous solution to open-loop and the stabilizing policy π_W.

4 Solution Approach

This section introduces the main contribution of the paper, which is a local feedback design that regulates the deviation of an executed trajectory from the optimal trajectory generated by the optimization procedure.

To solve the optimization problem presented in the last section, we first need to obtain the gradient information of the robustness heuristic that we introduced. However, calculating the gradient of the robust constraint is not straightforward, because the max function is non-differentiable. The gradient of the robustness constraint is computed by the application of Dankins Theorem Bertsekas [1997], which is stated next.

Dankin's Theorem: Let $K \subseteq \mathbb{R}^m$ be a nonempty, closed set and let $\Omega \subseteq \mathbb{R}^n$ be a nonempty, open set. Assume that the function $f : \Omega \times K \to \mathbb{R}$ is continuous on $\Omega \times K$ and that $\nabla_x f(x, y)$ exists and is continuous on $\Omega \times K$. Define the function $g : \Omega \to \mathbb{R} \cup \{\infty\}$ by

$$g(x) \equiv \sup_{y \in K} f(x, y), x \in \Omega$$

and

$$M(x) \equiv \{y \in K \,|\, g(x) = f(x, y)\}.$$

Let $x \in \Omega$ be a given vector. Suppose that a neighborhood $\mathcal{N}(x) \subseteq \Omega$ of x exists such that $M(x')$ is nonempty for all $x' \in \mathcal{N}(x)$ and the set $\cup_{x' \in \mathcal{N}(x)} M(x')$ is bounded. The following two statements (a) and (b) are valid.

1. The function g is directionally differentiable at x and
$$g'(x;d) = \sup_{y \in M(x)} \nabla_x f(x,y)^T d.$$
2. If $M(x)$ reduces to a singleton, say $M(x) = \{y(x)\}$, then g is Gâeaux differentiable at x and
$$\nabla g(x) = \nabla_x f(x, y(x)).$$

Proof See Facchinei and Pang [2003], Theorem 10.2.1.
Dankin's theorem allows us to find the gradient of the robustness constraint by first computing the argument of the maximum function and then evaluating the gradient of the maximum function at the point. Thus, in order to find the gradient of the robust constraint (5), it is necessary to interpret it as an optimization problem in δx_k, which is presented next. In Section 3.2, we presented a general formulation for the stabilization controller π_W, where W are the parameters that are obtained during optimization. However, solution of the general problem is beyond the scope of the current paper. Rest of this section considers a linear π_W for analysis.

Lemma 2. *Assume the linear feedback $\pi_W(\delta x_k) = W\delta x_k$. Then, the constraint (7) is quadratic in δx_k,*
$$\begin{aligned} &\max_{\delta x_k} ||M_k \delta x_k||_P^2 = \max_{\delta x_k} \delta x_k^T M_k^T \cdot P \cdot M_k \delta x_k \\ &s.t. \quad \delta x_k^T S_k \delta x_k \leq 1 \end{aligned} \tag{8}$$
where $M_k$is shorthand notation for
$$M_k(x_k, u_k, W) \equiv A(x_k, u_k) + B(x_k, u_k) \cdot W$$

Proof. Please see Appendix.

The next lemma is one of the main results in the paper. It connects the robust trajectory tracking formulation RelaxedRobustTrajOpt with the optimization problem that is well known in the literature.

Lemma 3. *The worst-case measure of deviation d_{max} is*
$$d_{max} = \lambda_{max}(S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}) = ||P^{\frac{1}{2}} M_k S_k^{-\frac{1}{2}}||_2^2$$
where $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue of a matrix and $||\cdot||_2$ denotes the spectral norm of a matrix. Moreover, the worst-case deviation δ_{max} is the corresponding maximum eigenvector
$$\delta_{max} = \{\delta x_k : \left[S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}\right] \cdot \delta x_k = d_{max} \cdot \delta x_k\}$$

Proof. Please see Appendix.

This provides us with the maximum deviation along the trajectory at any step 'k', and now we can use Danskin's theorem to compute the gradient which is presented next.

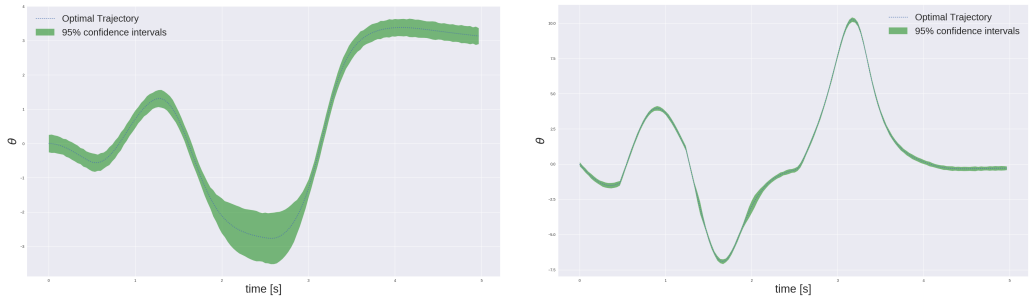
Theorem 1. *Introduce the following notation, $\mathcal{M}(z) = S_k^{-\frac{1}{2}} M_k^T(z) \cdot P \cdot M_k(z) S_k^{-\frac{1}{2}}$. The gradient of the robust inequality constraint d_{max} with respect to an arbitrary vector z is*
$$\nabla_z d_{max} = \nabla_z \delta_{max}^T \mathcal{M}(z) \delta_{max}$$
Where δ_{max} is maximum trajectory deviation introduced in Lemma 3.

Proof. Please see Appendix.

The gradient computed from Theorem 1 is used in solution of the RelaxedRobustTrajOpt– however, this is solved only for a linear controller. The next section shows some results in simulation and on a real physical system.

5 Experimental Results

In this section, we present some results using the proposed algorithm for an under-actuated inverted pendulum. We use a Python wrapper for the standard interior point solver IPOPT to solve the optimization problem discussed in previous sections. We perform experiments to evaluate the following



(a) Optimal position trajectory with the uncertainty band provided by learned GP model

(b) Optimal velocity trajectory with the uncertainty band provided by learned GP model

Figure 2: Optimal trajectory of the pendulum along with the uncertainty in the states along the trajectory predicted by the learned GP model.

question: Can we use an off-the-shelf model-learning algorithm to generate the uncertainty sets which can be then used in the optimization to generate a stable feedback solution? In the rest of this section, we try to answer these questions using simulation on an inverted pendulum system. We tested the controller over several settings and found that the underactuated setting was the most challenging to stabilize.

For clarity of presentation, we use an underactuated pendulum system, where trajectories can be visualized in state space. The dynamics of the pendulum is modeled as $I\ddot{\theta} + b\dot{\theta} + mgl \cdot sin(\theta) = u$. The continuous-time model is discretized as $(\theta_{k+1}, \dot{\theta}_{k+1}) = f((\theta_k, \dot{\theta}_k), u_k)$. The goal state is $x_g = [\pi, 0]$, and the initial state is $x_0 = [0, 0]$ and the control limit is $u \in [-1.7, 1.7]$. The cost is quadratic in both the states and input. The initial solution provided to the controller is trivial (all states and control are 0). The number of discretization points along the trajectory is $N = 120$, and the discretization time step is $\Delta t = 1/30$. The cost weight on robust slack variables is selected to be $\alpha = 10$. We assume that the model parameters of the pendulum are not perfectly known. More specifically, we assume that there is 5% uncertainty in the friction coefficient b. Additionally, we add the state-dependent noise during data collection to certain parts of the state space. The aim here is to test if the static feedback solution can handle local state-dependent noise modeled by GP. The noise of magnitude $0.5rad$ is injected close to unstable equilibrium state $\theta = \pi$ on training data as well as later during simulation. That level of noise is extreme compared to the magnitude of $0.1rad$ used in the rest of the state space. A GP model for the pendulum is used to learn the residual dynamics learned due to the parametric uncertainty and the additional measurement noise. The GP model is learned using the standard RBF kernel. The residual model allows us to generate the uncertainty sets along any given sequence of state and action. In Figure 2, we show an optimal open-loop trajectory found during optimization along with the uncertainty sets for 95% confidence interval of the GP model. The optimization uses these local sets along the trajectory to find the time-invariant stabilizing controller.

In Figure 3, we show the control inputs, the time-invariant feedback gains obtained by the optimization problem. In Figure‘4, we show several state-space trajectories for the learned system in open-loop (without the feedback matrix) and with the closed loop controller. As seen in the Figure, the open-loop system is unstable due to the uncertainty in the learned model– however, the closed-loop dynamics for the system is stable. The feedback system is computed using the RelaxedRobustTrajOpt optimization. This demonstrates the performance of the proposed optimization algorithm and demonstrates how the proposed optimization problem allows us to incorporate the uncertainty from the learned models to design the stabilizng controller simulataneously with the trajectory.

6 Conclusion and Future Work

This paper presents a method for simultaneously computing an optimal trajectory along with a local, time-invariant stabilizing controller for a dynamical system with known uncertainty bounds. The time-invariant controller was computed by adding a robustness constraint to the trajectory optimiza-

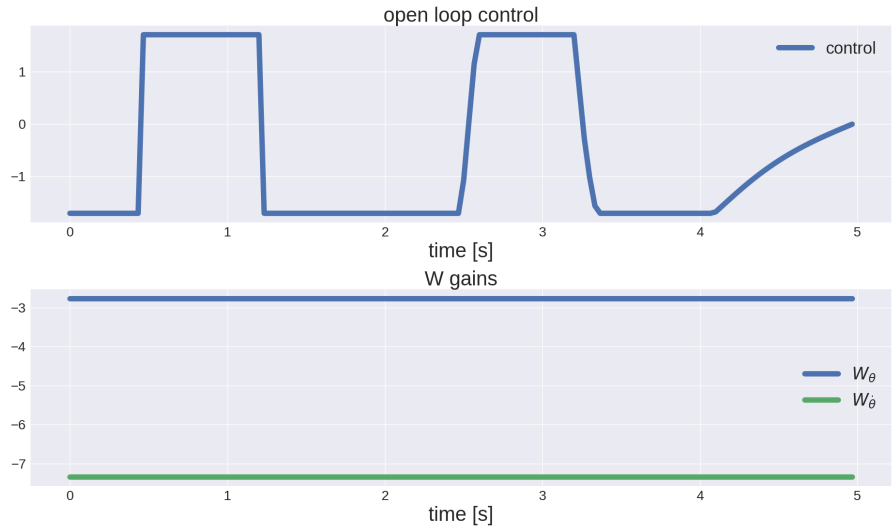


Figure 3: The optimal control signal and the time-invariant local policy obtained by solving the RelaxedRobustTrajOpt problem in the paper.

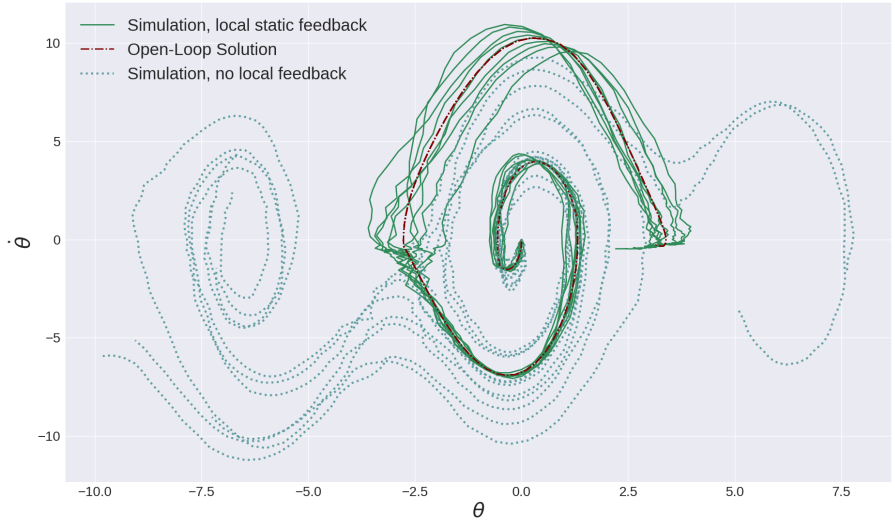


Figure 4: The state-space trajectory for the open-loop system as well as the closed-loop system. As seen in the figure, the open-loop system is not stable due to the inherent stochasticity in the learned residual dynamics.

tion problem. We prove that under certain simplifying assumptions, we can compute the gradient of the robustness constraint so that a gradient-based optimization solver could be used to find a solution for the optimization problem. We tested the proposed approach that shows that it is possible to solve the proposed problem simultaneously. We showed that even a linear parameterization of the stabilizing controller with a linear approximation of the error dynamics allows us to successfully control non-linear systems locally. We tested the proposed method in simulation using a non-linear pendulum with parametric uncertainty where a GP model is used to learn the residual dynamics and compute the local sets of uncertainty along any trajectory. In the future, we would investigate the performance of the proposed method on more complex non-linear systems Romeres et al. [2019], v. Baar et al. [2019]. We would also like to investigate the solution for non-linear parameterization of the stabilizing controller for better performance Tedrake et al. [2010].

References

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems, Volume II*. Springer, New York, NY, 2003.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–373, 2013.

David H Jacobson. New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach. *Journal of Optimization Theory and Applications*, 2(6):411–440, 1968.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.

William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pages 4008–4016, 2016.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

D. Romeres, D. K. Jha, W. Yerazunis, D. Nikovski, and H. A. Dau. Anomaly detection for insertion tasks in robotic assembly using Gaussian process models. In *2019 18th European Control Conference (ECC)*, pages 1017–1022, June 2019. doi: 10.23919/ECC.2019.8795698.

Diego Romeres, Devesh K Jha, Alberto DallaLibera, Bill Yerazunis, and Daniel Nikovski. Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3195–3202. IEEE, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction (2nd Edition)*, volume 1. MIT press Cambridge, 2018.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *journal of machine learning research*, 11(Nov):3137–3181, 2010.

J. v. Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski. Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6001–6007, May 2019. doi: 10.1109/ICRA.2019.8793561.

K.G. Vamvoudakis, P.J. Antsaklis, W.E. Dixon, J.P. Hespanha, F.L. Lewis, H. Modares, and B. Kiumarsi. Autonomy and machine intelligence in complex systems: A tutorial. In *American Control Conference (ACC), 2015*, pages 5062–5079, July 2015.

D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis. *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. IET control engineering series. Institution of Engineering and Technology, 2013. ISBN 9781849194891.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking Model-Based Reinforcement Learning. *arXiv e-prints*, art. arXiv:1907.02057, Jul 2019.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

7 Appendix

In this section, we present proofs of the Lemmas and Theorems presented earlier in Section 4.
Proof of Lemma 2.

Proof. Write d_{max} from (7) as the optimization problem

$$\begin{aligned} & d_{max} = \\ & \max_{\delta x_k} ||A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k)||_P^2 \\ & s.t. \quad \delta x_k^T S_k \delta x_k \leq 1 \end{aligned} \tag{8}$$

Introduce the linear controller and use the shorthand notation for M_k to write (8).

Proof of Lemma 3.

Proof. Apply coordinate transformation $\delta \tilde{x}_k = {S_k}^{\frac{1}{2}} \delta x_k$ in (8) and write

$$\begin{aligned} & \max_{\delta \tilde{x}_k} \delta \tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta \tilde{x}_k \\ & s.t. \quad \delta \tilde{x}_k \delta \tilde{x}_k \leq 1 \end{aligned} \tag{9}$$

Since $S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}$ is positive semi-definite, the maximum lies on the boundary of the set defined by the inequality. Therefore, the problem is equivalent to

$$\begin{aligned} &\max_{\delta\tilde{x}_k} \delta\tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta\tilde{x}_k \\ &s.t. \quad \delta\tilde{x}_k \delta\tilde{x}_k = 1 \end{aligned} \tag{10}$$

The formulation (10) is a special case with a known analytic solution. Specifically, the maximizing deviation δ_{max} that solves (10) is the maximum eigenvector of $S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}$, and the value d_{max} at the optimum is the corresponding eigenvalue.

Proof of Theorem 1.

Proof. Start from the definition of gradient of robust constraint

$$\nabla_z d_{max} = \nabla_z \max_{\delta\tilde{x}_k} \delta\tilde{x}_k \mathcal{M}(z) \delta\tilde{x}_k$$

Use Danskin's Theorem and the result from Lemma 3 to write the gradient of robust constraint with respect to an arbitrary z,

$$\nabla_z d_{max} = \nabla_z \delta_{max}^T \mathcal{M}(z) \delta_{max}$$

which completes the proof.