Local Policy Optimization for Trajectory-Centric Reinforcement Learning

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

TR2020-062 June 03, 2020

Abstract

The goal of this paper is to present a method for simultaneous trajectory and local stabilizing policy optimization to generate local policies for trajectory-centric model-based reinforcement learning (MBRL). This is motivated by the fact that global policy optimization for non-linear systems could be a very challenging problem both algorithmically and numerically. However, a lot of robotic manipulation tasks are trajectorycentric, and thus do not require a global model or policy. Due to inaccuracies in the learned model estimates, an openloop trajectory optimization process mostly results in very poor performance when used on the real system. Motivated by these problems, we try to formulate the problem of trajectory optimization and local policy synthesis as a single optimization problem. It is then solved simultaneously as an instance of nonlinear programming. We provide some results for analysis as well as achieved performance of the proposed technique under some simplifying assumptions.

IEEE International Conference on Robotics and Automation (ICRA)

Local Policy Optimization for Trajectory-Centric Reinforcement Learning

Patrik Kolaric[1], Devesh K. Jha[2], Arvind U. Raghunathan[2], Frank L. Lewis[1],
Mouhacine Benosman[2], Diego Romeres[2] Daniel Nikovski[2]

Abstract— **The goal of this paper is to present a method for simultaneous trajectory and local stabilizing policy optimization to generate local policies for trajectory-centric model-based reinforcement learning (MBRL). This is motivated by the fact that global policy optimization for non-linear systems could be a very challenging problem both algorithmically and numerically. However, a lot of robotic manipulation tasks are trajectory-centric, and thus do not require a global model or policy. Due to inaccuracies in the learned model estimates, an open-loop trajectory optimization process mostly results in very poor performance when used on the real system. Motivated by these problems, we try to formulate the problem of trajectory optimization and local policy synthesis as a single optimization problem. It is then solved simultaneously as an instance of nonlinear programming. We provide some results for analysis as well as achieved performance of the proposed technique under some simplifying assumptions.**

I. 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 [1], [2]. Although RL has witnessed huge successes in recent times [3], [4], 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 [5].

Model-based reinforcement learning (MBRL) methods [6] learn a model from exploration data of the system, and then exploit the model to synthesize a trajectory-centric controller for the system [7]. These techniques are, in general, harder to train, but could achieve good data efficiency [8]. 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 [9], [10]. These uncertainty estimates could be used to 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 [11].

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. 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 present analysis of the proposed technique that allows us to analytically derive the gradient of the robustness constraint for the optimization problem.

It is noted that this paper only presents the controller synthesis part for MBRL – a more detailed analysis of the interplay between model uncertainties and controller synthesis is deferred to another publication.

II. Related Work

MBRL has raised a lot of interest recently in robotics applications, because model learning algorithms are largely task independent and data-efficient [12], [8], [6]. 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 [13] or its first-order approximation, the iterative Linear Quadratic Regulator (iLQR) [14] is very popular, as it allows the use of gradient-based optimization, and thus could be used for

[1]UTA Research Institute, University of Texas at Arlington, Fort Worth, TX,USA (Email: `patrik.kolaric@mavs.uta.edu`, `lewis@uta.edu`

[2] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA. Email: `{jha,raghunathan,benosman,romeres,nikovski}@merl.com`

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 [8], [6]. The method proposed in [8] 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 [15] to jointly optimize trajectories and learn the neural network policies. This approach has achieved impressive performance on several robotic tasks [8]. The method has been shown to scale well for systems with higher dimensions. Several different variants of the proposed method were introduced later [16], [17], [18]. 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 [11]. 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 [19].

Another set of relevant work could be found in [20], [21] where the idea is to allow constraint satisfaction for the partially-known system by appropriately bounding model errors. Furthermore, authors in [22] present a way to perform approximate dynamic programming using the ideas of invariant sets. However, these methods find the global controller for the system which could be inefficient for high-dimensional systems. lastly, some model-free trajectory-centric approaches could be found in [23], [24].

III. 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.

A. 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 $\mathcal{X} := \{x \in \mathbb{R}^{n_x} \,|\, \underline{x} \le x \le \overline{x}\} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} := \{u \in \mathbb{R}^{n_u} \,|\, \underline{u} \le u \le \overline{u}\} \subseteq \mathbb{R}^{n_u}$, *i.e.* $x_k \in \mathcal{X}, u_k \in \mathcal{U}$. We use $[K]$ to denote the index set $\{0, 1, \dots, K\}$. Further, there may exist nonlinear inequality constraints of the form

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

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*

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

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 short-hand notation, $||v||_M^2 = v^T M v$. We denote the nominal trajectory as $X \equiv x_0, x_1, x_2, x_3, \dots, x_{T-1}, x_T$, $U \equiv u_0, u_1, u_2, u_3, ..., u_{T-1}$. The actual trajectory followed by the system is denoted as $\hat{X} \equiv \hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \dots, \hat{x}_{T-1}, \hat{x}_T$. 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 $J = \sum_{k\in[T]} c(x_k, u_k)$.

B. 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 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

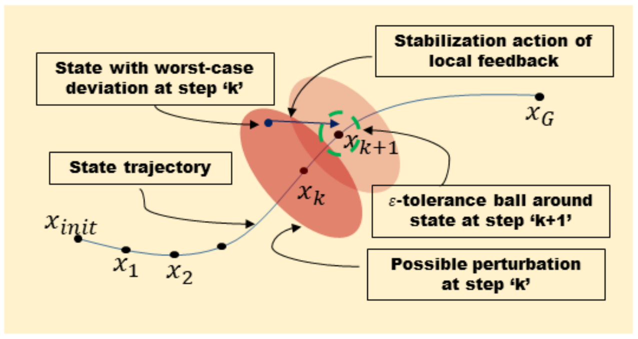
Stabilization action of local feedback
State with worst-case deviation at step 'k'
x_G
x_{k+1}
State trajectory
x_k
ϵ-tolerance ball around state at step 'k+1'
x_{init}
x_1
x_2
Possible perturbation at step 'k'

Fig. 1: A schematic representation of the *robustness constraint* introduced in the paper.

$\delta x_k \equiv x_k - \hat{x}_k$. 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:

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

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).

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

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

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

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.

IV. 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 Danskins Theorem [25], which is stated next.

Danskin'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. (a)

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 [26], Theorem 10.2.1.

Danskin'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 III-B, 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

$$M_k(x_k, u_k, W) \equiv A(x_k, u_k) + B(x_k, u_k) \cdot W \tag{9}$$

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

$$\begin{aligned} &d_{max} = \\ &\max_{\delta x_k} ||A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k)||_P^2 \\ &s.t. \quad \delta x_k^T S_k \delta x_k \leq 1 \end{aligned} \tag{10}$$

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

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*

$$\begin{aligned} &d_{max} = \\ &\lambda_{max}(S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}) = ||P^{\frac{1}{2}} M_k S_k^{-\frac{1}{2}}||_2^2 \end{aligned}$$

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

$$\begin{aligned} &\delta_{max} = \\ &\{\delta x_k : \left[S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}\right] \cdot \delta x_k = d_{max} \cdot \delta x_k\} \end{aligned} \tag{10}$$

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

$$\begin{aligned} &\max_{\delta\tilde{x}_k} \delta\tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta\tilde{x}_k \\ &s.t. \quad \delta\tilde{x}_k \delta\tilde{x}_k \leq 1 \end{aligned} \tag{11}$$

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

$$\begin{aligned} &\max_{\delta\tilde{x}_k} \delta\tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta\tilde{x}_k \\ &s.t. \quad \delta\tilde{x}_k \delta\tilde{x}_k = 1 \end{aligned} \tag{12}$$

The formulation (12) is a special case with a known analytic solution. Specifically, the maximizing deviation δ_{max} that solves (12) 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.

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: 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.

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.

V. Experimental Results

In this section, we present some results using the proposed algorithm for an under-actuated inverted pendulum, as well as on a experimental setup for a ball-and-beam system. 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 questions:

1) Can an off-the-shelf optimization solver find feasible solutions to the joint optimization problem described in the paper?
2) Can the feedback controller obtained by this optimization stabilize the open-loop trajectory in the presence of bounded uncertainties?
3) How good is the performance of the controller on a physical system with unknown system parameters ?

In the following sections, we try to answer these questions using simulations as well as experiments on real systems.

A. *Simulation Results for Underactuated Pendulum*

The objective of this subsection is twofold: first, to provide insight into the solution of the optimization problem; and second, to demonstrate the effectiveness of that solution in the stabilization of the optimal trajectory. 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 $\alpha = 10$. The uncertainty region is roughly estimated as $x_k^T \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 5.5 \end{bmatrix} x_k < 1$ along the trajectory. Detailed analysis on uncertainty estimation based on Gaussian processes is deferred to future work, due to space limits. The optimization procedure terminates in 50 iterations with the static solution $W = [-2.501840, -7.38725]$.

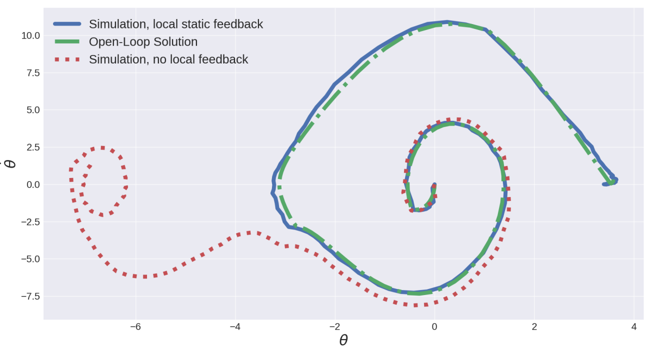

Fig. 2: State-space representation of the optimal trajectory (green), stable closed-loop system using the obtained solution (blue), and unstable open-loop trajectory without the local feedback (red). Note that the feedback is time-invariant.

The controller generated by the optimization procedure is then tested in simulation, with noise added to each state of the pendulum model at each step of simulation as $x_{k+1} = f(x_k, u_k) + \omega$ with $\omega_\theta \sim \mathcal{U}(-0.2rad, 0.2rad])$ and $\omega_{\dot{\theta}} \sim \mathcal{U}(-0.05rad/s, 0.05rad/s])$.

We tested the controller over several settings and found that the underactuated setting was the most challenging to stabilize. In Figure 2, we show the state-space trajectory for the controlled (underactuated) system with additional noise as described above. As seen in the plot, the open-loop controller becomes unstable with the additional noise, while the proposed controller can still stabilize the whole trajectory.

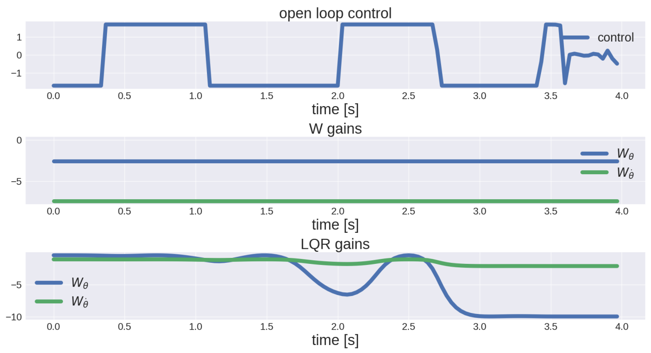

Fig. 3: a) open-loop control b) static feedback matrix obtained from optimization c) LQR feedback

In Figure 3, we show the control inputs, the time-invariant feedback gains obtained by the optimization problem. We also the time-varying LQR gains obtained along the trajectory to show provide some insight between the two solutions. As the proposed optimization problem is finding the feedback gain for the worst-case deviation from the trajectory, the solutions are different than the LQR-case. Next, in Figure 4, we plot the error statistics for the controlled system (in the underactuated setting) over 2 different uncertainty balls using each 12 sample for each ball. We observe that the steady-state error goes to zero and the closed-loop system is stable along the entire trajectory. As we are using a linear approximation of the system dynamics, the uncertainty sets are still small, however the results are indicating that incorporating the full dynamics during stabilization could allow to generate much bigger basins of attraction for the stabilizing controller.

B. *Results on Ball-and-Beam System*

Next, we implemented the proposed method on a ball-and-beam system (shown in Figure 5) [27]. The ball-and-beam system is a low-dimensional non-linear system with

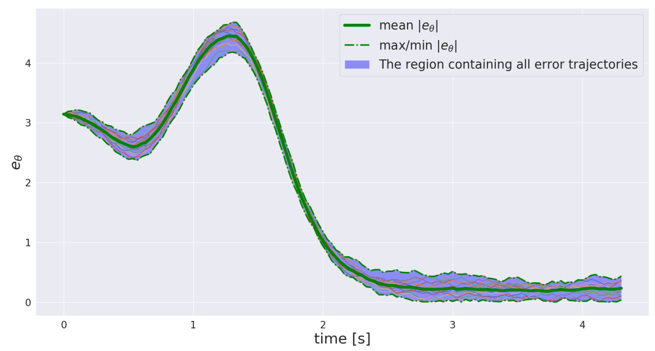

Fig. 4: Error statistics for the controlled system using the proposed method on the underactuated pendulum with noise amplitude

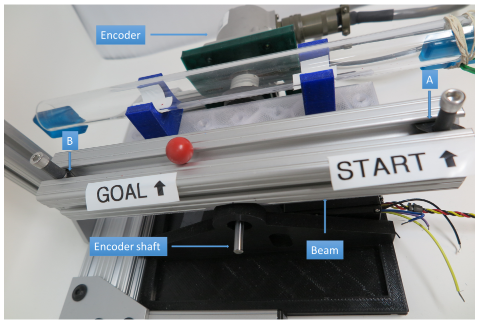

Fig. 5: The ball-and-beam system used for the experiments. There is an RGB camera above that measures the location of the ball. The encoder (seen in the figure) measures the angular position of the beam.

the non-linearity due to the dry friction and delay in the servo motors attached to the table (see Figure 5). The ball-and-beam system can be modeled with 4 state variables $[x, \dot{x}, \theta, \dot{\theta}]$, where x is the position of the ball, $\dot{x}$ is the ball's velocity, θ is the beam angle in radians, and $\dot{\theta}$ is the angular velocity of the beam. The acceleration of the ball, $\ddot{x}$, is given by

$$\ddot{x} = \frac{m_{ball}x\dot{\theta}^2 - b_1\dot{x} - b_2 m_{ball} g\cos(\theta) - m_{ball} g \sin(\theta)}{\frac{I_{ball}}{r_{ball}^2} + m_{ball}},$$

where m_{ball} is the mass of the ball, I_{ball} is the moment of inertia of the ball, r_{ball} is the radius of the ball, b_1 is the coefficient of viscous friction of the ball on the beam, b_2 is the coefficient of static (dry) friction of the ball on the beam, and g is the acceleration due to gravity. The beam is actuated by a servo motor (position controlled) and an approximate model for its motion is estimated by fitting an auto-regressive model. We use this model for the analysis where the ball's rotational inertia is ignored and we approximately estimate the dry friction. The model is inaccurate, as can be seen from the performance of the open-loop controller in Figure 6. However, the proposed controller is still able to regulate the ball position at the desired goal showing the stabilizing behavior for the system (see the performance of the closed-loop controller in Figure 6). The plot shows the mean and the

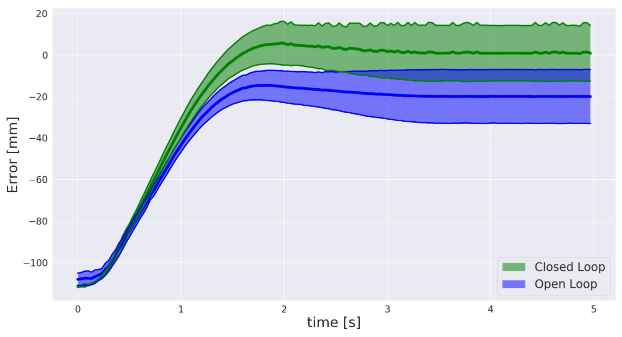

Fig. 6: Comparison of the performance of the proposed controller on a ball-and-beam system with the open-loop solution. The plot shows the error in the position of the ball from the regulated position averaged over 12 runs.

standard deviation of the error for 12 runs of the controller. It can be observed that the mean regulation error goes to zero for the closed-loop controller. We believe that the performance of the controller will improve as we improve the model accuracy. In future research, we would like to study the learning behavior for the proposed controller by learning the residual dynamics using GP [10].

VI. 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 optimization 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 as well as a physical system. Due to space limitations, we have to skip extra results regarding the behavior of the algorithm.

However, the current approach has two limitations– it makes linear approximation of dynamics for finding the worst-case deviation, and secondly, the linear parameterization of the stabilizing controller can be limiting for a lot of robotic systems. In future research, we will incorporate these two limitations using Lyapunov theory for non-linear control, for better generalization and more powerful results of the proposed approach. We also hope to extend the current formulation to more complex settings for feedback control of manipulation problems [28].

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction (2nd Edition)*. MIT press Cambridge, 2018, vol. 1.

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

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[5] K. Vamvoudakis, P. Antsaklis, W. Dixon, J. Hespanha, F. Lewis, H. Modares, and B. Kiumarsi, "Autonomy and machine intelligence in complex systems: A tutorial," in *American Control Conference (ACC), 2015*, July 2015, pp. 5062–5079.

[6] M. Deisenroth and C. 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)*, 2011, pp. 465–472.

[7] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[9] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[10] D. Romeres, D. K. Jha, A. DallaLibera, B. Yerazunis, and D. Nikovski, "Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3195–3202.

[11] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[12] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking Model-Based Reinforcement Learning," *arXiv e-prints*, p. arXiv:1907.02057, Jul 2019.

[13] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 411–440, 1968.

[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3381–3388.

[17] W. H. Montgomery and S. Levine, "Guided policy search via approximate mirror descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 4008–4016.

[18] A. Nagabandi, G. Kahn, R. S. Fearing, and S. 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)*. IEEE, 2018, pp. 7559–7566.

[19] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4054–4061.

[20] A. Chakrabarty, D. K. Jha, and Y. Wang, "Data-driven control policies for partially known systems via kernelized lipschitz learning," in *2019 American Control Conference (ACC)*, July 2019, pp. 4192–4197.

[21] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation," *arXiv preprint arXiv:1907.02151*, 2019.

[22] A. Chakrabarty, R. Quirynen, C. Danielson, and W. Gao, "Approximate dynamic programming for linear systems with state and input constraints," in *2019 18th European Control Conference (ECC)*, June 2019, pp. 524–529.

[23] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *journal of machine learning research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[24] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[25] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

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

[27] D. K. Jha, D. Nikovski, W. Yerazunis, and A. Farahmand, "Learning to regulate rolling ball motion," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov 2017, pp. 1–6.

[28] W. Han and R. Tedrake, "Local trajectory stabilization for dexterous manipulation via piecewise affine approximations," *arXiv e-prints*, 2019, https://arxiv.org/abs/1909.08045.