Learning Control Policies from Optimal Trajectories

Christoph Zelch[1], Jan Peters[2], Oskar von Stryk[1]

Abstract— **The ability to optimally control robotic systems offers significant advantages for their performance. While time-dependent optimal trajectories can numerically be computed for high dimensional nonlinear system dynamic models, constraints and objectives, finding optimal feedback control policies for such systems is hard. This is unfortunate, as without a policy, the control of real-world systems requires frequent correction or replanning to compensate for disturbances and model errors.**

In this paper, a feedback control policy is learned from a set of optimal reference trajectories using Gaussian processes. Information from existing trajectories and the current policy is used to find promising start points for the computation of further optimal trajectories. This aspect is important as it avoids exhaustive sampling of the complete state space, which is impractical due to the high dimensional state space, and to focus on the relevant region.

The presented method has been applied in simulation to a swing-up problem of an underactuated pendulum and an energy-minimal point-to-point movement of a 3-DOF industrial robot.

I. Introduction

Optimal control theory [1] enables the formulation of optimality criteria and provides a basis for numerical methods for model-based computation of optimal trajectories [2], [3]. These enable the offline computation of optimal trajectories even for large-scale nonlinear robot dynamics models, nonlinear cost functions as well as nonlinear state and control constraints. Good nonlinear robot dynamics models are often available and enable deep insights in the system's dynamic behavior which can be only fully utilized by an optimal control approach. However, for practical applications, even small model inaccuracies as well as inevitable external disturbances or small deviations from the start position lead to deviations from the precomputed trajectory. These must be dealt with in a non-optimal manner, e.g., by real-time trajectory tracking controllers. This motivates the search for a near-optimal policy, that allows to proceed in real-time even under disturbances acting on the robotic system.

The approach of machine learning to find a near-optimal policy is usually either to approximate the value function as a solution of the Hamilton-Jacobi-Bellman equation to derive the optimal control, or to use policy iteration. Both approaches lack the generalization to large scale nonlinear system dynamics and nonlinear constraints. They usually do not account for the capable model-based numerical trajectory optimization methods. However, some of these provide valuable information that can be highly beneficial to be utilized for the learning process.

We explore the potential of generalizing trajectories generated from different start points using the direct collocation method DIRCOL and its ability to estimate the optimal trajectory's co-states [4]. The policy is approximated using a Gaussian process (GP) [5] which additionally provides information about the certainty of the current model. We iteratively select start points for new optimal trajectories to exploit information from existing trajectories.

The rest of the paper is organized as follows. The next two subsections present related work as well as the formal definition of the considered problem. Section II describes the main steps of our approach. Results of the evaluation of the presented method for an under-actuated swing-up problem and a variable-stiffness arm are given in Section III. Section IV concludes this paper.

A. Relevant Prior Work

The generalization of trajectory data with a neural network is explored in [6]–[8]. Pesch et al. use analytical solutions of the considered problem, Hardt and Breitner employ direct (DIRCOL [2], [4]) and indirect (MUMUS [9]) methods for trajectory optimization. They did not investigate the selection of start points for the optimal trajectories further.

Schierman et al. [10] deal with the guidance of re-entering space shuttles with compensation of technical failures of control effectors. Their aim is repeated fast online-generation of an optimal trajectory for unknown start conditions. To generalize from a set of precomputed trajectories, [10] use polynomial neural networks that map (initial) states to coefficients of basis functions describing a trajectory.

In [11], a learning classifier system (XCSF) is trained to interpolate optimized trajectories, resulting in a parametric approximation of the optimal policy. Training data has been derived from demonstration as well as from optimal trajectories computed using direct policy search. The goals for these trajectories have been selected using random sampling.

An improved dynamic programming algorithm based on Gaussian process models that approximate the value function is presented in [12]. Here, the state space is randomly sampled to get the data points for the GP.

All work presented so far does not consider the selection of new start values in a systematic and adaptive manner. They choose all start points at once, compute in a second step all respective optimal trajectories and subsequently generalize towards optimal feedback control. The start points are mostly

[1]Christoph Zelch and Oskar von Stryk are with the Simulation, Systems, Optimization and Robotics Group, Department of Computer Science, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {zelch, stryk}@sim.tu-darmstadt.de

[2]Jan Peters is with the Intelligent Autonomous Systems Group, Department of Computer Science, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany jan.peters@tu-darmstadt.de

located on a dense grid, Deisenroth et al. [12], [13] uses randomly sampled points as input for the Gaussian process.

In contrast, Atkeson et al. [14]–[16] explore the advantages of successive selection of start points. To improve the Differential Dynamic Programming (DDP) algorithm on global optimization problems, they generalize trajectories from different start positions. The advantage of the DDP approach to get optimal local models is that a second-order approximation of the value function and the policy is computed at all points and can be used to build a parametric representation of the value function and the policy at sampled locations. Variants of random sampling [15] in combination with a number of acceptance criteria is used to select sampling states. Globalization of the local models is done by using the nearest sampled state. Their approach allowed to compensate moderate pushes of a four link balancing leg.

Da Silva et. al. [17] consider parameterized skills to solve families of control tasks, they use Bayesian optimization with a specially designed acquisition function to select new training tasks that maximize expected improvement in overall skill performance. Further, they reuse unsuccessful policies that appeared during training as training samples for a different goal [18]. Levine and Koltun [19] use optimized trajectories to create samples (used for the gradient estimation) that direct the policy search algorithm to regions of high reward. Learning parameterized motor skills, [20] found out that using the partially optimized policy as initial guess for the optimization process of new tasks speeds up policy search and also increases the number of successfully solved tasks. We use this method as one of three approaches to get initial guesses for trajectory optimization.

Special focus on the selection of new training tasks for learning of parameterized policies for families of tasks is in [21]. The authors present a detailed algorithm to actively select new training tasks to explore the task space. The self-generation of new tasks is based on a measure of interest, defined as variation of competence, which is related to the number of efficient attempts to reach a goal.

A widespread approach for model-based optimal feedback control is Model Predictive Control (MPC). For systems with linear dynamics and quadratic cost functions, the structure of the underlying optimization problem can be exploited, which allows MPC to compute a feedback control in real-time [22]. But also for Nonlinear MPC, progress has been made that brings real-time application for systems with fast dynamics in reach [23]–[26]. Nevertheless, nonlinear MPC of large systems with highly nonlinear dynamics and constraints require considerable efforts to compute the control in each control cycle in real-time. In [27], [28], a Model Predictive Control approach with slow update rate is combined with an optimized feedback controller with high input rate that complies with equality and inequality constraints.

B. Problem Statement and Notation

In the following, we consider the general continuous time nonlinear dynamic systems model for robot dynamics transformed from second to first order

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

with state vector $x \in \mathcal{X} := \{x \in \mathbb{R}^{n_x} : x \in [x_{\min}, x_{\max}]\}$, control $u \in \mathcal{U} := \{u \in \mathbb{R}^{n_u} : u \in [u_{\min}, u_{\max}]\}$, $f \in C^1(\mathcal{X} \times \mathcal{U}, \mathcal{X})$ and $t \in [0, t_f]$.

Given some cost function

$$\mathcal{J}(x, u, t_f) := \Phi(x(t_f), u(t_f), t_f) + \int_0^{t_f} L(x(\tau), u(\tau), \tau)\ d\tau$$

with terminal cost Φ and running cost L, as well as some start state x_0, the solution of an optimal control problem consists of some control function $\tilde{u}$, state function $\tilde{x}$ and co-state function $\tilde{\lambda}$ that minimize $\mathcal{J}$ satisfying equation (1). We consider only problems with finite end time $t_f \ll \infty$.

The value function

$$V(x,t) = \min_u \left\{ \Phi(x(t_f), u(t_f), t_f) + \int_t^{t_f} L(x(\tau), u(\tau), \tau)\, d\tau \right\}$$

gives the minimum cost to proceed from the current system state [1]. It should be noted that the approach presented in this paper can be extended to cover also nonlinear state and control constraints.

In this article, we aim to find some approximation of the state-dependent feedback control $\pi(x(t), t)$, such that its application produces for all feasible initial states $x_0 \in \mathcal{X}$ an optimal trajectory $\tilde{x}$ such that $u(t) = \pi(x(t), t)$ minimizes $\mathcal{J}$. We consider only problems without contact situation.

Algorithm 1 Generalization of Optimal Trajectories

Input: problem description, start points startpts
Output: policy pi
  for fixed number of iterations **do**
    optTrajs ← GETOPTIMALTRAJS(startpts, pi)
    pointList ← DISCRETIZEANDSELECT(optTrajs)
    pi ← TRAINGPS(pi, pointList)
    startpts ← GETNEWSTARTPOINTS(optTrajs)
  end for

II. Successive Optimal Trajectory Generation and Approximation of Near-Optimal Policy

To learn a policy for a given problem formulation, we use information collected from a number of optimal trajectories for different start points in the state space. These trajectories are generated using the solver DIRCOL [4] that implements a direct collocation method to solve optimal control problems.

We do not determine all start points at the beginning but use an iterative approach that allows to continually supervise the improvement of the policy, select new start points based on progress achieved until then, and add only relevant data. The optimal control policy is approximated by a Gaussian process (GP), retrained in each iteration with data points on the optimal trajectories. The main steps of the algorithm are outlined as pseudo-code in Algorithm 1. They are motivated and described in more detail in the following subsections.

A. Computation of Optimal Trajectories

DIRCOL [2], [4] is a direct collocation method that uses SNOPT [29], [30] to solve the respective nonlinear optimization problem. It requires a rough initial guess of state and control trajectories to start the numerical optimization, however, this guess needs not to be feasible subject to the dynamics and other constraints. Direct collocation methods are in general more robust to poor initial guesses than indirect methods [3], [31], nevertheless, a reasonably good user-provided initial guess may make the difference between success and failure of the iterative optimization method.
We use three different approaches to get initial guesses:

- The simplest and most unbiased initial guess is to linearly interpolate between start and (expected) end configuration, with the control constantly zero. In the first iteration, this is the only initial guess that is used.
- Select an already computed trajectory that starts or passes as close as possible to the current start configuration and use it as initial guess. While this approach leads in most cases to convergence of the solver, it produces not always optimal results, since close start points do not necessarily lead to similar globally optimal trajectories. If the distance to the closest data point is too large, we add intermediate start points and compute the corresponding trajectories before, such that each optimal control problem has at least one optimal trajectory in close proximity that can be used as initial guess.
- Use the current approximation of the optimal feedback control to generate an initial guess. To be applicable, this approach requires a sufficiently accurate approximation in the relevant region, it is thus very error-prone in the first iterations. This has also been done in [20].

For each start position, we run the solver with all three initial guesses to avoid using suboptimal solutions and keep the best valid solution (the one with lowest total cost). DIRCOL is started with 11 equidistant grid points, at most ten grid refinements are allowed. Further, DIRCOL's automatic scaling of variables and functions is enabled.

If the solution process fails, we start a costly homotopy approach where we solve a series of optimal control problems where the final time is stepwise increased from a small fraction to the original value, using each time the solution of the previous run as initial guess. An outlier detection method is required to reject suboptimal results.

B. Discretization of the Trajectories

The GP learns the control values evaluated at discrete data points, which makes it necessary to extract significant value pairs $(x(t_i), u(t_i))$ from each computed trajectory and add them to the data used to train and evaluate the policy. From each trajectory, we get a set of at most N data points $\mathcal{S} = \{(x(t_i), u(t_i))\}_{i=0,\dots,N}$ with $t_i = i \cdot t_f/(N-1)$ that are used to improve the learner. We found that $N = \min\{50, t_f/t_{\min}\}$ gives good results, this extracts 50 data points from each trajectory and reduces this number for short trajectories, as it enforces a minimum distances $t_{\min}$ (in time) between the discretization points.

Subsequently, all points that are closer than some minimum distance $d_{\min}$ to any previously added state are removed from $\mathcal{S}$. The remaining data in $\mathcal{S}$ is used as training data for the learner.

C. Start Point Generation

The selection of new start points is highly relevant for the "exploration" of unknown areas of the state space and determines how the algorithm progresses. It is important to acquire more information in regions where a deviation from the optimal trajectory is more relevant, i.e., where suboptimal actions cause a significant change in the total cost of the executed path.

For some optimal trajectory $x(t)$, the relation between the co-states $\lambda(t)$ (or adjoint variables) and the value function is described by

$$\lambda(t) = \frac{\partial V(x(t), t)}{\partial x(t)},$$

(see [1]), large (absolute) co-state values indicate where some deviation from the trajectory in state has considerable impact on the value function. This motivates the use of the trajectory's co-states $\lambda(t)$ as an indicator for sensitive regions where more neighboring trajectories are favorable. DIRCOL allows to estimate the Lagrange multiplier function of a computed approximation of an optimal trajectory.

We select points t_i on the trajectory with local maxima $\max\{\|\lambda(t_i)\|\}$, as well as points where the norm of the co-states at the grid-points are outside the 67%-quantile (with minimum distance between the grid-points enforced).

For some point $x_i := x(t_i)$ on a trajectory with large co-state value $\lambda_i := \lambda(t_i)$, we choose a new start point candidate $\tilde{x}_i$ in the direction of the co-state vector λ_i. The Euclidean distance of $\tilde{x}_i$ from x_i has to be in the interval $[b_l, b_u]$. Since large absolute co-state values indicate more sensitive regions, the distances between respective points are smaller. The exponential decrease favors start points close to the trajectory. In summary, the mapping from some trajectory point to a new start point is described by

$$\tilde{x}_i(x_i, \lambda_i) = x_i + \kappa(\lambda_i)\,\lambda_i$$

with $\kappa : \lambda \mapsto \frac{1}{\lambda}\left(b_l + (b_u - b_l)e^{-\nu\lambda}\right)$. The real-valued parameter $\nu > 0$ needs to be tuned depending on the range of occurring co-state values.

The points in state space retrieved from the trajectory are added to a list of candidate start points. If a start value for a new optimal trajectory is needed, the variance of learned policy is evaluated for all these candidates and the one with the highest variance is selected.

An initial non-empty set of start values has to be provided with the problem formulation. The solution trajectories for these values should be not too difficult to compute, because, contrary to the following iterations where already computed trajectories and the approximation of the feedback control can be used, no problem specific information to generate initial guesses is available.

In contrast to [17], in which the acquisition of new learning data is guided by Bayesian Optimization, we use

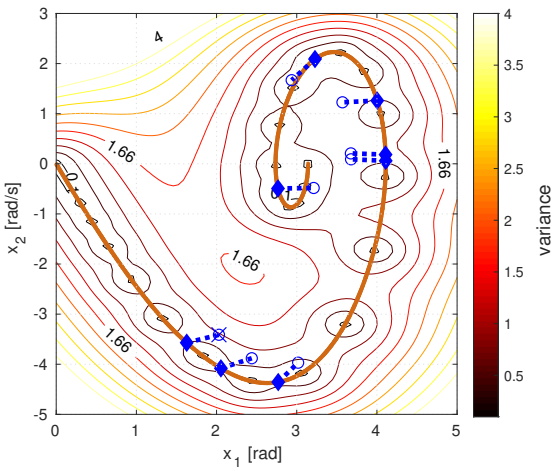

Fig. 1: This example illustrates how start points are selected based on co-state values and variance of the GP. Points of an extremum in the co-state values (filled marker) as well as the connected start point candidate are shown. The selected new start point at the state with the largest variance (given as contour) is marked with an x.

information about the sensitivity of the trajectory in the state space with respect to the total cost and combine it with a measure of the current approximation quality of the learned near-optimal control.

D. Learning Optimal Control Policies from Trajectories

Gaussian processes [5] have been successfully used in [12], [32] to approximate the control policy or the value function of optimal control problems. We use GPs as non-parametric model of the learned policy. The covariance function is set to be a multi-layer perceptron kernel. The GP is retrained each time the data points of a trajectory are added. The main advantage of using a Gaussian process function to approximate the near-optimal policy is that it provides for each point a variance which serves as measure of uncertainty.

The high order of differentiability that is inherent to many GP kernels like the squared exponential or RBF covariance function makes them unsuitable to model the control function which may exhibit many different local properties. Instead, we use a multi-layer perceptron kernel, which allows us to fit functions with both steep slopes and smooth plateaus.

To better represent discontinuities in the control function, the policy model can be further extended by a concept presented in [13]: Two separated GPs for positive and negative control signals respectively are trained along with a classifier that learns which GP to use at which point in the state space. The applicability of this technique must be examined in further work.

A large number of data points may slow down training as well as evaluation of the GP. To attenuate this, it is possible to use some sparse pseudo-input model as e.g. introduced in [33] to reduce the computational cost for both operations. However, the use of a sparse approximation degrades the quality of the learned near-optimal policy.

It must be noted that a policy implemented by a GP is likely to violate the box constraints of the control. Consequently, it is necessary to reset values outside $\mathcal{U}$ to some valid controls on the boundary of $\mathcal{U}$.

E. Additional Samples around the Goal State

As can be seen in Fig. 2 left, the sampling of the state space close to the goal state may stay sparse. All generated trajectories approach this state in a narrow tube, leaving a significant area around the goal state unexplored.

In the example shown in Fig. 2, the states are not constrained to the goal state at the end of the trajectory (the convergence to the goal state is accomplished only by the formulation of the objective function). This forces the co-states to become zero at the end point of the trajectory (for details see [1]), leading to small co-state values and hence no new start points around the goal state.

Altogether, while the co-states give valuable information about sensible regions around a large portion of the computed optimal trajectories, they are of little use to promote exploration in the direct proximity of the goal state. We thus complement our start point selection strategy based on co-state values by adding short trajectories starting at additional (e.g. randomly placed) positions on a ball with small (problem dependent) radius around the goal state. These trajectories are much shorter and accordingly add less data points to the policy learner than the ones starting from points imposed by the method described in Section II-C.

III. Evaluation and Experiments

The approach has been implemented in Matlab R2018b, the used GP implementation is the package *GPmat* by Lawrence and others [34]. The link to DIRCOL has been established using mex-files that call interface classes in C++.

A. Under-actuated Pendulum Swing-Up

In the following, we consider an under-actuated pendulum with system dynamics

$$\ddot{\theta}(t) = \frac{g}{l}\sin\left(\theta(t)\right) + \frac{u(t)}{ml^2} - \mu\frac{\dot{\theta}(t)}{ml^2},$$

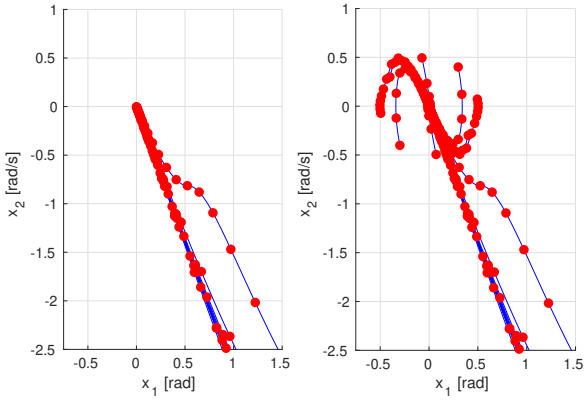

Fig. 2: Left: Generated optimal trajectories approach the goal state in a small tube. Right: Additional short trajectories sampled in the proximity of the goal state amend this deficit.

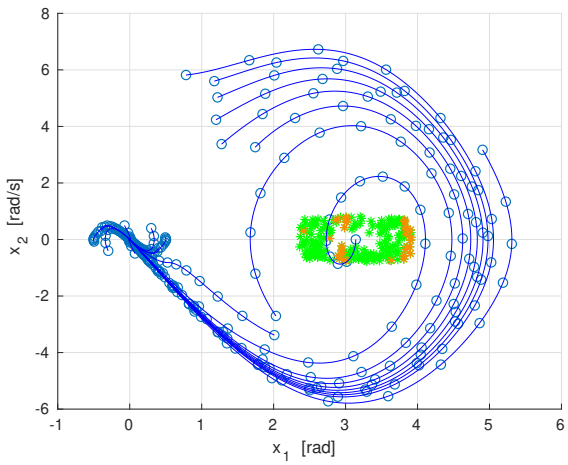

Fig. 3: The trajectories and selected data points used to learn the near-optimal policy and the test set with 200 start points. The figure shows the result for the MLP covariance function (no sparse approximation).

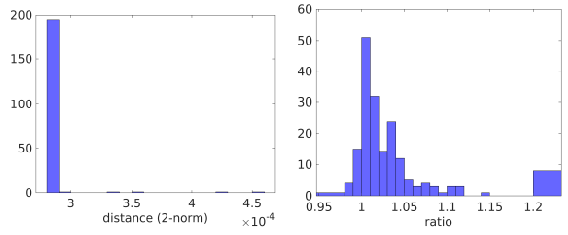

Fig. 4: Left: The Euclidean distance from the goal state at the end point. Right: Ratio "simulated cost to optimal cost" of the trajectories of the test set.

as given in [12], [35], [36] with link length $l = 1\,\mathrm{m}$, mass $m = 1\,\mathrm{kg}$, friction coefficient $\mu = 0.05\,\mathrm{kg\,m^2/s}$ and $g = 9.81\,\mathrm{m/s^2}$ and define the state $x = (\theta, \dot{\theta})^T$ in $(\mathrm{rad}, \mathrm{rad/s})^T$.

We use a quadratic cost function of the form

$$\mathcal{J}(x, t_f) = \frac{1}{2}\left(10\bar{x}_1(t_f)^2 + 10\bar{x}_2(t_f)^2\right) + \frac{1}{2}\int_0^{t_f} \bar{x}_1(t)^2 + \bar{x}_2(t)^2 + u(t)^2 + \bar{x}_1(t)u(t)\ \mathrm{d}t$$

with $\bar{x}(t) := x_f - x(t)$, similar to [13].

The applicable torque is constrained to be $u \in [-5, 5]\,\mathrm{Nm}$. This restriction makes the task non-trivial, as the maximum available torque is not sufficient to bring the pendulum from a hanging position ($\theta = \pi$) to the goal state, so the solution must contain some swing-up procedure [36].

We start with an initial start position $\theta = \pi$, $\dot{\theta} = 0$ and apply the algorithm as described in Section II. To evaluate the learned control, we simulate the swing-up problem starting from 200 different points s_i in the state space with $0.3 \leq \| (\pi, 0)^T - s_i \|_2 \leq 1.0$. We use the classical 4th order Runge-Kutta scheme with time steps of $0.001\,\mathrm{s}$. The total cost of the trajectory is integrated using the trapezoidal rule.

The results of the simulations with a policy learned in ten iterations of the presented algorithm is summarized in Fig. 4. We performed 10 iterations, resulting in 10 trajectories and 219 data points selected for training of the GP. The 8 additional trajectories around the goal state (see Section II-E) increase the number of used data points to a total of 308. To give an unbiased view on the start point selection procedure, we manually check all computed trajectories during execution and, if necessary, initiate a re-computation to make sure that the trajectories for all start points are used.

In the following, we consider the goal as reached, if the Euclidean distance of the state x to the goal state is less than 10^{-3} and we say that a near-optimal trajectory has been found, if its approximated cost is at most 110% of the trajectory cost computed by DIRCOL for the same start point. With MLP we refer to a multi-layer perceptron covariance function, based on a 2-layer feed-forward network [34] with linear activation function, six inputs, one output and five hidden units.

The result for the MLP covariance function with a non-sparse posterior variance approximation is shown in Figs. 3 and 4. Green stars represent successful test cases, orange stars indicate trajectories reaching the goal state with costs more than 110% of the optimum and red stars show test cases where the goal state has been missed. Further, for a small number of test cases (here 20) the ratio is slightly lower than 1.0 (typically not lower than 0.95), indicating a lower cost than the optimal trajectory computed with DIRCOL. The reason is a less accurate cost estimation for the simulated trajectories (linear approximation whereas DIRCOL uses cubic spline interpolation of the states). For the Matérn-$5/2$ and the MLP covariance function, the goal has been reached for all test cases. If we use a sparse approximation of the GP the value decreases using the MLP covariance function to 195 (for DTCVAR [37]) and 170 (for FITC [33]) out of 200. If the eight additional trajectories are not included, only 57 out of 200 trajectories reach the goal state.

Note that for the comparison of the different covariance functions and posterior variance approximations the same training set (created using the non-sparse MLP covariance function) is used. The effect of the GP model on the start point selection for new trajectories is therefore neglected.

B. Feedback Control of the Robot Manutec R3

In this section, we apply our approach to approximate an optimal feedback control of the point-to-point movement of an industrial Manutec r3 robot. The Manutec r3 has six joints, we use the first three which mainly determine the position of the end-effector. The dynamic model of the Manutec r3 robot can be found in [38]. It has been formulated with focus on realism and is highly non-linear, which makes finding a near-optimal control for this model significantly harder than for the underactuated pendulum.

We adopt the definition of the optimal control problem described in [39] with varying start values to compute optimal trajectories. In contrast to [39], we only consider minimum energy movements to reach some fixed goal state with cost function

$$\mathcal{J}(x, u, t_f) = t_f + \rho \int_0^{t_f} \sum_{i=1}^{3} u_i(t)^2\ \mathrm{d}t$$

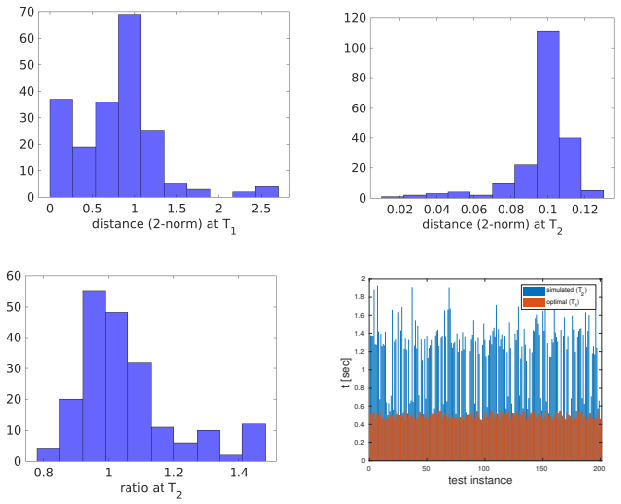

Fig. 5: Euclidean distances from the goal configuration at optimal end time (T_1) and at the closest point (T_2). Further, the ratio " simulated Lagrange-term cost to optimal Lagrange-term cost" and the time at which the closest point to the goal state has been reached.

with $\rho = 10^{-3}$. Further, the six state variables and three control variables are restricted by box constraints. The initial joint configuration q_{start} of the first trajectory is at $(q_{\text{start},1}, q_{\text{start},2}, q_{\text{start},3}, \dot{q}_{\text{start},1}, \dot{q}_{\text{start},2}, \dot{q}_{\text{start},3}) = (0, -1.5, 0, 0, 0, 0)$, the goal configuration is set to be $(1, -1.95, 1, 0, 0, 0)$. After 20 iterations we have 869 data points from 18 optimal trajectories to train the GP (two generated trajectories have been rejected because the optimization process failed). Together with 9 additional short optimal trajectories around the goal state, this results in a total of 1006 from 27 trajectories. We again use the non-sparse GP with MLP covariance function.

We simulate the movement of the robot arm caused by the learned control from 200 start configurations. The Euclidean distance of these start configurations to q_{start} in joint space is between 0.1 and 0.3. The result is presented in Fig. 5. We denote the end time of the optimal trajectory of a test instance with T_1, T_2 is the time at which the minimum distance to the goal state is reached by the test instance. It can be seen that at the end time of the respective optimal trajectory, the test instances have a large distance from the goal state. This distance is reduced to 0.1 at T_2, but with a significant time delay. Considering the Lagrange part of the objective function which is proportional to the energy cost, the approximated cost of the simulated trajectories are for three out of four test instances below 110%. However, the total cost of the simulated movements is much higher than the optimum, as it includes the final time.

C. Comparison with Naive Approach

In the following, we compare our approach with a naive start point sampling strategy. We use data points from 16 trajectories that start at random start points in the state space to train a GP with a MLP covariance function. These trajectories provide 877 data points which is comparable to the number of data points used above. We reuse the nine trajectories close to the goal state and get a total of 1013 data points. The results on our test set of 200 start configurations is given in Fig. 6. At T_1, the distance to the goal state is comparable to the distance achieved with the proposed method. At T_2 the distance is reduced to values between 0.12 and 0.15, which is slightly worse than with our approach. However, the naively learned policy needs much longer to bring the arm close to the goal state, in consequence, the overall cost at T_2 is typically also substantially higher.

IV. Conclusion

An optimal feedback control of dynamic systems to handle unknown disturbances is computationally expensive to obtain for most problems. In this paper, we iteratively improved a policy with samples from optimal trajectories. Focus has been on the placement of start positions. With a sophisticated strategy of new start point selection using information from existing trajectories, we are able to avoid full sampling of the joint space in the relevant region. We proposed an alternative to random sampling that uses information provided by the co-states of the optimal feed-forward solutions and the covariance of the learned policy. The method has been successfully applied to a 3-DOF industrial robot representing realistic highly nonlinear problems with time-invariant cost function.

The model of the Manutec r3 arm does not include models of the actuator and transmission dynamics. In principle, the modeling can be extended to also include higher order dynamics as well as bandwidth limitations introduced by the actuators. An important limitation of the presented approach is its dependency on a sufficiently accurate model, which is used in the trajectory optimization step. This model may be difficult to obtain in practice. A possible solution is the implementation of a feedback compensation of model errors using a locally optimal tracking controller, e.g. similar to the approach introduced in [27]. In [27], [28], the cost function used for Model Predictive Control has been extended to penalize high frequencies of the actuators in the trajectory. The examination of possible extensions to compensate for model inaccuracies or uncertainties and the implementation on an actual robot will be subject to further work.

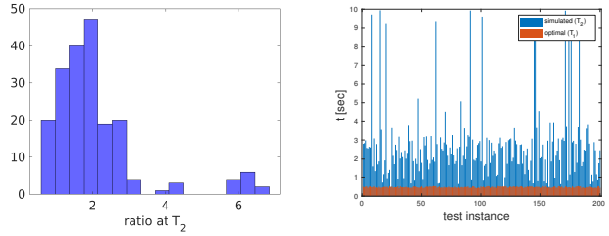

Fig. 6: Results for the naive sampling approach. The left figure shows a slightly worse cost ratio at T_2, the right figure gives the time at which the closest trajectory point to the goal state has been reached.

REFERENCES

[1] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[2] O. v. Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357–373, 1992.

[3] J. T. Betts, "A survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[4] O. von Stryk, "Numerical solution of optimal control problems by direct collocation," in *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*. Birkhäuser Basel, 1993, vol. 111, pp. 129–143.

[5] C. Rasmussen and K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[6] H. J. Pesch, I. Gabler, S. Miesbach, and M. H. Breitner, "Synthesis of optimal strategies for differential games by neural networks," in *New Trends in Dynamic Games and Applications*. Birkhäuser Boston, 1995, pp. 111–141.

[7] M. H. Breitner, "Robust optimal onboard reentry guidance of a space shuttle: Dynamic game approach and guidance synthesis via neural networks," *Journal of Optimization Theory and Applications*, vol. 107, no. 3, pp. 481–503, 2000.

[8] M. Hardt, "Multibody dynamical algorithms, numerical optimal control, with detailed studies in the control of jet engine compressors and biped walking," phdthesis, University of California, 1999.

[9] P. Hiltmann, K. Chudej, and M. Breitner, "Eine modifizierte mehrzielmethode zur loesung von mehrpunkt-randwertproblemen benutzeranleitung," Technische Univ. Muenchen (Germany). Sonderforschungsbereich 255: Transatmosphaerische Flugsysteme, Grundlagen der Aerothermodynamik, Antriebe und Flugmechanik," Report, 1993.

[10] J. D. Schierman, D. G. Ward, J. R. Hull, N. Gandhi, M. Oppenheimer, and D. B. Doman, "Integrated adaptive guidance and control for reentry vehicles with flight test results," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 6, pp. 975–988, 2004.

[11] D. Marin and O. Sigaud, "Reaching optimally over the workspace: A machine learning approach," in *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, June 2012, pp. 1128–1133.

[12] M. P. Deisenroth, C. E. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, no. 7, pp. 1508–1524, 2009.

[13] M. P. Deisenroth, J. Peters, and C. E. Rasmussen, "Approximate dynamic programming with gaussian processes," in *2008 American Control Conference*. IEEE, 2008, pp. 4480–4485.

[14] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," *Advances in Neural Information Processing Systems*, pp. 663–670, 1994.

[15] C. Atkeson and B. Stephens, "Random sampling of states in dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 924–929, 2008.

[16] C. G. Atkeson and C. Liu, *Modeling, Simulation and Optimization of Bipedal Walking*. Springer Berlin Heidelberg, 2013, ch. Trajectory-Based Dynamic Programming, pp. 1–15.

[17] B. Da Silva, G. Konidaris, and A. Barto, "Active learning of parameterized skills," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, no. 2. PMLR, 22–24 Jun 2014, pp. 1737–1745.

[18] B. da Silva, G. Baldassarre, G. Konidaris, and A. Barto, "Learning parameterized motor skills on a humanoid robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 5239–5244.

[19] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 28, no. 3. PMLR, 17–19 Jun 2013, pp. 1–9.

[20] J. F. Queißer, R. F. Reinhart, and J. J. Steil, "Incremental bootstrapping of parameterized motor skills," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 223–229.

[21] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[22] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[23] M. J. Tenny, S. J. Wright, and J. B. Rawlings, "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming," *Comput. Optim. Appl.*, vol. 28, no. 1, pp. 87–121, 2004.

[24] M. Diehl, H. Bock, and J. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[25] B. Lantos and B. Kiss, "Nonlinear model predictive control of robots, cranes and vehicles," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 397–402, 2005.

[26] A. Zanelli, G. Horn, G. Frison, and M. Diehl, "Nonlinear model predictive control of a human-sized quadrotor," in *2018 European Control Conference (ECC)*, 2018, pp. 1542–1547.

[27] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 4730–4737.

[28] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, April 2019.

[29] P. Gill, W. Murray, and M. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, vol. 47, no. 1, pp. 99–131, 2005.

[30] P. E. Gill, W. Murray, and M. A. Saunders, "Users guide for SNOPT 5.3: A FORTRAN package for large-scale nonlinear programming," p. 71, 1997.

[31] V. Putkaradze and S. Rogers, "Constraint control of nonholonomic mechanical systems," *Journal of Nonlinear Science*, vol. 28, no. 1, pp. 193–234, 2018.

[32] C. Rasmussen, M. Kuss, S. Thrun, L. Saul, and B. Schölkopf, "Gaussian processes in reinforcement learning," *Advances in Neural Information Processing Systems 16, 751-759*, 2004.

[33] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in Neural Information Processing Systems*, vol. 18, p. 8, 2005.

[34] N. L. et al. (2015) Matlab gpmat toolbox. University of Sheffield. [Online]. Available: https://github.com/SheffieldML/GPmat

[35] S. Schaal, "Learning from demonstration," *Advances in Neural Information Processing Systems*, vol. 9, pp. 1040–1046, 1997.

[36] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, pp. 219–45, 2000.

[37] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574.

[38] M. Otter and S. Türk, "The DFVLR models 1 and 2 of the manutec r3 robot," DFVLR-Mitt. 88-13, Institut für Dynamik und der Flugsysteme, Oberpfaffenhofen, Germany, Tech. Rep., 1988.

[39] O. v. Stryk and M. Schlemmer, "Optimal control of the industrial robot manutec r3," in *Computational Optimal Control, International Series of Numerical Mathematics*, vol. 115. Basel: Birkhäuser, 1994, pp. 367–382.