Start State Selection for Control Policy Learning from Optimal Trajectories

Christoph Zelch[1], Jan Peters[2], Oskar von Stryk[3]

Abstract— **Combination of optimal control methods and machine learning approaches allows to profit from complementary benefits of each field in control of robotic systems. Data from optimal trajectories provides valuable information that can be used to learn a near-optimal state-dependent control policy. To obtain high-quality learning data, careful selection of optimal trajectories, determined by a set of start states, is essential to achieve a good learning performance.**

In this paper, we extend previous work with new complementing strategies to generate start points. These methods complement the existing approach, as they introduce new criteria to identify relevant regions in joint space that need coverage by new trajectories. It is demonstrated that the extensions significantly improve the overall performance of the previous method in simulation on full nonlinear dynamics model of the industrial Manutec robot arm.

Further, it is demonstrated that it suffices to learn a policy that reaches the proximity of the goal state, from where a PI controller can be used for stable control reaching the final system state.

I. Introduction

With the offline computation of optimal trajectories, preexisting knowledge about the potentially large-scale and typically non-linear dynamic behavior of systems and its (non-)linear constraints on state and control variables can be utilized to produce movements that are optimal with respect to performance criteria formulated using cost functions. Applied in practice, model inaccuracies or external disturbances lead to deviations from the optimal trajectory which require additional procedures that are often sub-optimal.

In contrast, reinforcement learning approaches, which are often used in this context, are designed to adapt to unforeseen situations and learn relations previously unknown to the user that are required to achieve some defined goal. However, these methods typically need large amounts of data and many interactions between agent and system, the consideration of non-linear constraints, e.g. implicitly in the reward function, is often cumbersome. Thus, the combination of methods from optimal control theory and machine learning to benefit from the complementing advantages of both fields, seems promising.

In [1], we have proposed an approach to iteratively collect data from optimal trajectories to learn a near-optimal state-dependent control policy using a Gaussian process (GP). Special focus has been on the identification of useful start states for the optimal trajectories.

The insight that motivates this paper is that more than one strategy is required for the selection of new trajectory start states, as large co-state values are not the only reasonable criterion for a good start state. The main contribution of this work is the proposition of new approaches to determine good start positions in joint space for useful new trajectories that provide information suitable to supplement the data that is used to train a GP. Further, we use a more sophisticated method to filter linearly dependent data points that produce noise in the kernel matrix of the GP. Both improvements allow us to generate more meaningful data to the GP, which enables a more precise approximation of the near-optimal policy. Finally, we demonstrate that it is not necessary to reach the exact goal state with the learned control. Instead, it suffices to reach a ball around the goal state, from which control is passed to a traditional PI controller that guides towards and stabilizes around the goal state.

The paper is organized as follows. We start with a discussion of related work and subsequently introduce the problem considered in this paper. In Section II, we propose the extensions and improvements of [1] outlined above. The evaluation of the presented approach is described in Section III. We end with concluding remarks in Section IV.

A. Related Work

The various existing approaches to generalize a control policy from optimized trajectories use different approaches to select start states for the trajectories used. Approaches that do not rely on an iterative approach but decide before any optimization on the set of start states typically rely on a fixed grid [2]–[4] or on uniform random sampling [5], [6]. Random sampling is also used very frequently in iterative approaches, like Latin Hypercube sampling in [7] or (uniform or Gaussian) random sampling with acceptance criteria in [8].

Another approach to obtain state-feedback controllers considers parameterized skills to solve families of control tasks. Bayesian optimization with a specially designed acquisition function is used in [9] to select new training tasks that maximize expected improvement in overall skill performance. Since they solve families of control tasks instead of a policy for a single task, they can reuse unsuccessful policies that appeared during training as training samples for a different goal [10]. Levine et al. [11] focus on regions with high reward and use the results of Iterative LQR to create samples that guide their policy search algorithm into these regions.

[1,3]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

Special focus on the selection of new training tasks for learning parameterized policies for families of tasks is in [12]. The authors present a detailed algorithm to generate 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.

It should be noted, that in this paper, we aim for identifying start states of trajectories, such that these, once optimized, provide useful data to improve the learned control policy, and not for good initial guesses to warm-start trajectory optimization as done e.g. in [13], [14] or [15]. These are different types of problems and must not be interchanged.

B. *Problem Formulation*

We consider continuous time nonlinear dynamic systems

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

that model robot dynamics with state $x \in \mathbb{R}^{n_x}$ and control $u \in \mathbb{R}^{n_u}$ vectors. In general, we assume that there are constraints on the joint states and velocities $x_{\text{min}} \leq x \leq x_{\text{max}}$ and also on the control values $u_{\text{min}} \leq u \leq u_{\text{max}}$. It should be noted that the approach presented in this paper can be extended to cover also nonlinear state and control constraints.

We define the cost function

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

with terminal cost Φ and running cost L.

The control function $\tilde{u}$ and the resulting trajectory $\tilde{x}$ solve an optimal control problem if they minimize the given cost function $\mathcal{J}$ and satisfy the motion dynamics (1), the limits on state and control variables and, if existent, other nonlinear state and control constraints. The state at final and end time as well as the finite end time $t_f \in \mathbb{R}$ of the optimal control problem can be free or fixed.

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

II. Description of the Method

Extending our method presented in [1] which we briefly recapitulate in the first subsection, we present three new start state generation strategies, that focus on different aspects. We continue with the description of some improvements concerning the rejection of linearly dependent data points. In the last subsection, we describe the switch-over to a LQR controller in the neighborhood of the goal state.

A. *Existing Method*

In an iterative approach, we collect data from several optimal trajectories to learn a state-dependent control policy using a Gaussian process (GP). The start point for a new trajectory is not determined a priori but selected in each

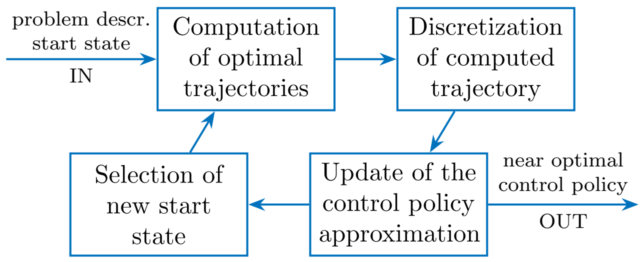

problem descr.
start state
IN
Computation of optimal trajectories
Discretization of computed trajectory
Selection of new start state
Update of the control policy approximation
near optimal control policy
OUT

Fig. 1: The four main steps of our method in [1].

iteration based on information from previously computed trajectories and the current state of the GP. A flowchart that shows the four major steps is given in Figure 1.

In a first step, we solve a given optimal control problem as defined in Subsection I-B for some initial start state using DIRCOL [16]. The resulting trajectory is discretized into a grid of data points in a second step. These data points are added to the training data of the GP, which is retrained afterwards. In a last step, we compute new candidate start states from the new trajectory and select a new start state from all candidates based on the current variance of the learned GP. With the new start state, we repeat the four steps, refining the learned control policy, until some exit condition is fulfilled.

The four steps of the algorithm can be replaced by other approaches. Instead of DIRCOL, one can use some other optimal control solver (e.g. Bocop [17]), further, the GP that acts as control policy approximator can e.g. be replaced by an artificial neural network. However, the replacement of the approximator must provide some estimate of the uncertainty of the estimation. Several methods exist to provide neural networks with this ability [18], [19], since for our method only epistemic uncertainty is relevant, one can use Monte Carlo dropout [20], [21]. An in-depth comparison of the performance of different approximators in this framework is beyond the scope of this paper and will be considered as another study.

The selection of start positions for new trajectories is crucial, as it determines the data that is added to the learning routine. For the method to be successful, there must be enough information in the relevant region. This requires the optimal trajectories that provide the data to be distributed such that the learned policy is able to deal with unexpected deviations from the original path.

In [1], we used co-state values to indicate parts of the trajectory where some deviation has considerable impact on its cost at the end point. For a given trajectory $(x(t))_{t\in[0,t_f]}$, we used local maxima of the co-states

$$\lambda(t) = \frac{\partial V(x(t), t)}{\partial x(t)} \tag{2}$$

to compute new candidate start states that accumulate accross iterations. In each iteration, we selected the one at which the GP has the highest variance.

B. New Strategies for Start State Selection

Our focus on the co-state variables that identify parts of the trajectory where deviations have a high impact on the cost, ignores other aspects that are also important for an accurate approximation of the optimal control policy. In the following, we will introduce various start point selection strategies intended to complement each other as well as the method based on co-states variables. The strategies reflect the different requirements on the trajectories that provide data to learn the near-optimal policy.

1) Sensitivity-based: While the examination of co-states indicates regions in the state space with a high impact on the resulting trajectory cost, it is also important to focus on data in regions, where small deviations in the control value may lead to large changes in the state and thus notably change the trajectory from this point.

In the following, we will need a notion of the relative perturbation error, or sensitivity of the state variables, which we define following [22]. Let $x(t_0+t) = M_f^t(x(t_0))$ denote the numerical time integration of the differential equation $f(x, u)$, representing the dynamical system, starting from state $x(t_0)$.

For $\varepsilon > 0$ small, a fixed time window $\delta t > 0$ and the j-th column vector of the identity matrix e_j, we define with

$$P_j^{M_f,\delta t,\varepsilon}(x) = \frac{M_f^{\delta t}(x+\varepsilon e_j) - M_f^{\delta t}(x)}{\varepsilon}, \tag{3}$$

the partial sensitivity approximation of f at state x. Let further be

$$p_{M_f,\delta t,\varepsilon}(x) = \left(\|P_1^{M_f,\delta t,\varepsilon}(x)\|, \cdots, \|P_n^{M_f,\delta t,\varepsilon}(x)\|\right) \tag{4}$$

the sensitivity vector that holds the normed perturbation for all partial derivatives.

Note that the absolute error of the numerical integration scheme with respect to the perturbation error ε must be sufficiently small to provide a meaningful approximation of the sensitivity. For a given trajectory $(x_{\text{ref}}(t))_{t\in[0,t_f]}$ with end time t_f, we use a coarse time grid $\Gamma = \{t_0, t_1, \ldots, t_k\}$ with $t_0 = 0$ and $t_k = t_f$.

Assume that there is a distinguished peak at $t_{\text{p}} \in \Gamma$ in the sequence $p_{M_f,\delta t,\varepsilon}(x(t_0)), \ldots, p_{M_f,\delta t,\varepsilon}(x(t_f))$, and $\Gamma \ni t_{\text{v}} < t_{\text{p}}$ such that $\|p_{M_f,\delta t,\varepsilon}(x(t_i))\|$ is increasing for $t_{\text{v}} \le t_i \le t_{\text{p}}$. Then we add the two points

$$x_{\text{ref}}(t_v) \pm \mu_{\text{sens}} \frac{p_{M_f,\delta t,\varepsilon}(x(t_{\text{v}}))}{\|p_{M_f,\delta t,\varepsilon}(x(t_{\text{v}}))\|} \tag{5}$$

to a strategy-specific set of candidate start states for new trajectories.

2) Simulation-based: For a given trajectory $x_{\text{ref}}(t)$ with control $u_{\text{ref}}(t)$ at times $t \in [0, tf]$, we simulate the trajectory that results from the currently learned control, starting from the same state as the reference trajectory $x_{\text{ref}}(0)$. We compare the resulting trajectory $(x_{\text{sim}}(t), u_{\text{GP}}(t))$ on a fine grid Γ with the reference trajectory and identify the time at which the maximum deviation occurs. In this comparison, we compensate for some time offset τ between optimal and simulated trajectory.

$$d_i = d(t_i) = x_{\text{ref}}(t_i) - x_{\text{sim}}(t_i - \tau), \quad t_i \in \Gamma$$

and $\Delta_i = \Delta(t_i) = \|d_i\|$ where τ minimizes the sum of all deviations Δ_i at the grid points in Γ.

Identify, if existent, a distinguished peak at $t_{\text{p}} \in \Gamma$ in the sequence of deviations d_i and let again $\Gamma \ni t_v < t_p$ be the time from which this peak starts growing, i.e., $\Delta(t_{i+1}) > \Delta(t_i)$ for all $t_{\text{v}} \le t_i < t_{\text{p}}$. A new candidate start state is then given as

$$x_{\text{ref}}(t_{\text{v}}) + \mu_{\text{sim}} \frac{d(t_v)}{\Delta(t_v)}, \tag{6}$$

at a user-defined distance μ_{sim} from the reference trajectory in the direction $d(t_v)$ of the deviation occurring in simulation.

3) Around the Initial State: The start positions generated from one of the two previously described methods are typically closer to the goal position than the start state of the original trajectory. To get some longer trajectories, we choose start states in the proximity of the initial, user provided start state x_0 of the first trajectory. To keep our method deterministic, we use the quasi-random Halton sequence [23] to compute start positions for new trajectories:

$$x_0 + \mu_{\text{H}} \frac{\mathcal{H}_i}{\|\mathcal{H}_i\|}. \tag{7}$$

Again, μ_{H} is some user-defined parameter that denotes the desired distance from x_0, $\mathcal{H}_i$ is the i-th element of the multi-dimensional Halton-sequence.

For this strategy, we ignore variance information from the learner and always choose the next iterate in the Halton sequence. Note that, in contrast to the other approaches, this start point generation method does not use information from already computed trajectories.

4) Combination of the Variants: All variants make sure that every new start position they provide is feasible, by either reducing the distance to the original state or adapting the direction that points from the original state. The strategies that build a set of candidate points use all optimal trajectories that have been computed to enlarge this set. For the selection of a new start state used in the next iteration, the four start point generation methods are used sequentially, one in each iteration.

C. Discretization and Training

Given some learning data $(\tilde{x}_i, \tilde{u}_i)_{i=1,\ldots,N}$, predictions of Gaussian processes require the inversion of a matrix $\left(K + \sigma^2 I\right)$, where I is the identity matrix of suitable size, σ^2 some positive noise constant and K is the Gramian matrix of the vectors $\{\tilde{x}_1, \ldots, \tilde{x}_N\}$ with respect to the selected GP kernel function [24]. The Gramian K is positive semi-definite by construction and positive definite if and only if the vectors $\tilde{x}_i$ are linearly independent. To get an invertible matrix $\left(K + \sigma^2 I\right)$ with the noise σ as small as possible, it is necessary to have only relevant and unique training data. In [1], a new data point $(\tilde{x}_{N+1}, \tilde{u}_{N+1})$ is rejected if its distance to an other data point in the existing learning data falls below some constant $c > 0$, i.e.,

$$\min_{i=1,\dots,N} \{\|\tilde{x}_{N+1} - \tilde{x}_i\|\} < \text{c}. \tag{8}$$

However, the parameter c must be chosen conservatively to avoid a reduction in quality of the GP's performance during execution of the algorithm. Instead, we now add the new training data from a trajectory, compute the resulting kernel matrix K and use a QR decomposition[1] to identify a linearly independent subset of columns $C \subseteq \{1, \dots, N\}$ in K. We restrict the set of training data to $(\tilde{x}_i, \tilde{u}_i)_{i \in C}$, which ensures that the kernel matrix K stays invertible despite the addition of arbitrary new data points[2].

The new approach more reliably filters training data that would lead to high noise, which allows us to add more training data per trajectory.

D. *LQR Control near the Goal State*

Exact convergence towards and stabilization around the goal state x_f is difficult to achieve with our learned control, as learning data becomes sparse in its close vicinity. However, there are more appropriate methods to stabilize a control system around some goal state.

We linearize the error system around x_f and use a linear quadratic regulator (LQR) approach to design a PI control law around the goal state. To deal with steady state errors (e.g. caused by model errors), we augment the system to implement an integral action control. Like this, the learned control needs not to reach the exact goal position but only some ball around the final state, from which control is passed to the computed PI controller.

This renders the use of additional short trajectories around the goal state as described in [1] unnecessary.

III. Evaluation

The improvements detailed in the previous section have been evaluated in simulation to have exact information about the behavior of the investigated system available. We use the same system as in [1], the Manutec r3 robot arm, to allow a comparison of the results.

The problem specific cost function

$$\mathcal{J}(x, u, t_f) = t_f + \rho \int_0^{t_f} \sum_{i=1}^{3} u_i(t)^2 \, \mathrm{d}t \tag{9}$$

with $\rho = 10^{-3}$ from [25] leads to contact-free energy-minimal movements with an additional penalty for the end time. We use the first three joints of the arm, which determine the position of the end-effector. The highly non-linear dynamic model of the robot arm can be found in [26].

We consider point-to-point movements from the initial joint position $x_0 = \big(0, -1.5, 0, 0, 0, 0\big)$ to the final joint position $x_f = \big(1, -1.95, 1, 0, 0, 0\big)$. The state and control variables are constrained by the box constraints $x_{\text{max}} = (2.97, 2.01, 2.86, 3.1, 1.5, 5.2)$, $x_{\text{min}} = -x_{\text{max}}$ and $u_{\text{max}} = (7.5, 7.5, 7.5)$, $u_{\text{min}} = -u_{\text{max}}$. In simulate, we tolerate small violations of the state constraints and increase the box constraints by five percent. If the learned controller exceeds the control bounds, the value is set to the boundary of the feasible region.

Our method is implemented and run in Matlab 2021b, we use the implementation of Gaussian processes provided by package *GPmat* written by Lawrence et al [27], the Fortran implementation of DIRCOL [16] is interfaced using mex-files. Simulations are performed using the `ode45` routine included in Matlab, which is based on the Dormand-Prince integration method (relative error tolerance 10^{-8}, absolute error tolerance 10^{-9}).

A. *Performance of the Start Point Generation Strategies*

We analyze the contribution of each new start point generation strategy to the resulting learned control. We compare the performance of the GP approximation that has been trained using all four proposed strategies with the learned near-optimal control that has been trained alike but with a single start point strategy missing. In the following, we will refer to these five scenarios as *full*, *noAdj*, *noSens*, *noSim* and *noHalt*. For each scenario, we stop the training after use of 30 trajectories, such that the number of trajectories used for training is approximately as large as in [1].

For each optimal trajectory $(x_{\text{ref}}(t), u_{\text{ref}}(t))_{t \in [0, t_f]}$ of the test set, we simulate the movement of the robot arm, using all five learned control policies one after another. For this evaluation, we do not use the LQR control close to the end state x_{goal} to avoid tampering the result. Instead, for a simulated trajectory $x_{\text{sim}}(t)$, we consider the time

$$t'_{\text{f,sim}} := \arg\min_t \|x_{\text{sim}}(t) - x_{\text{goal}}\| \tag{10}$$

as final time of the simulated trajectory. The distance at the final simulation state to the goal state is given by $d := \|x_{\text{sim}}(t'_{\text{f,sim}}) - x_{\text{goal}}\|$. We call a simulation successful, if $d < 0.2$.

To allow a fair comparison between simulated and optimal trajectory, we define the time at which the optimal reference trajectory has the same distance to the goal state as the simulated trajectory at its final state:

$$t'_{\text{f,ref}} : \|x_{\text{ref}}(t'_{\text{f,ref}}) - x_{\text{goal}}\| = \|x_{\text{sim}}(t'_{\text{f,sim}}) - x_{\text{goal}}\|. \tag{11}$$

For the comparison of the five learned control policies, we consider the following two performance criteria:

1) Distance of final simulation state from the goal state:

$$\|x_{\text{sim}}(t'_{\text{f,sim}}) - x_{\text{goal}}\| \tag{12}$$

2) Ratio of terminal times:

$$t'_{\text{f,sim}} / t'_{\text{f,ref}} \tag{13}$$

We use a test set of 300 optimal reference trajectories from random start positions with defined distance to the joint state x_0 to evaluate the performance of the five scenarios

[1]Matt J (2022). Extract linearly independent subset of matrix columns, MATLAB Central File Exchange

[2]This approach is inspired by a forum post by Jack Fitzsimons: `https://stats.stackexchange.com/q/189816` (version: 2016-01-08).

introduced above. The test set consists of six groups of 50 trajectories, where the start state of all trajectories in a group has the same distance from the initial start state. The Euclidean distances are 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3. The results for each scenario are summarized in Figure 2. The bar plot shows the number of successfully solved test instances. The two box plots visualize the result with respect to the final trajectory distance to the goal state (see (12)) and the ratio of simulated to optimal trajectory time (see (13)).

The scenario *noHalt* gives the highest number of successes and the second best end positions. However, the end times are very far from optimal, which justifies the use of the Halton based strategy. The sensitivity-based strategy seems to have the largest impact on the performance of the method, as the extremely low number of successfully solved trajectories in *noSens* implies. By far the worst final distances from the goal are reached without the co-state variable based strategy (*noAdj*), which underlines its usefulness.

The scenario including all start state selection strategies (*full*) gives a reasonable trade-off between the evaluated performance criteria (12) and (13). In the evaluation of the distance from the goal state, scenario *noSim* shows the best results, as in the mean the end positions are much closer to the goal state. Its ratio of end times is nearly optimal. Consequently, it turns out that the simulation based strategy has the lowest impact on the overall performance. For this reason, we use the learned control generated with the *noSim* scenario in the subsequent evaluation.

B. Comparison with previous results

To show that our new start state selection strategies improve the results achieved in [1], we evaluate the learned control created for the *noSim* scenario on the test set constructed in there. This test set consists of 200 optimal trajectories whose start states' distance from x_0 ranges from 0.1 to 0.3. Further, we use the evaluation code from our previous work. It must be noted that in [1], 9 distinct trajectories around the goal state x_f have been added to improve convergence. In this paper, these goal state trajectories are omitted in favor of a PI control close to the goal state. This PI control is left out here and evaluated in the next subsection.

We simulate the movement from the start states in the test set using the learned control and evaluate the shortest distance from the goal state that is reached by the simulated arm. Further, we consider the ratio of times for the optimal and simulated movement. The results are presented in Figure 3. The new start state selection strategies significantly improve, apart from some outliers, the costs and end times of the simulated trajectories. Some of the cost ratios are slightly below 1.0. This is not a flaw in the trajectory optimizer DIRCOL, but results from the fact that, by accepting all trajectories that reach some region around the goal state, we have relaxed the original problem formulation. This makes it possible that some simulation results "outperform" the optimal solution that actually reaches the goal state.

The closest distance to the goal state is worse than in [1], which can be attributed to the missing goal state trajectories. This underlines the importance of a PI control around the goal state, which is evaluated in the next subsection.

C. LQR Control near the Goal State

In this subsection, we evaluate the performance of the learned control policy combined with a linear feedback controller around the goal state x_{goal}. We use a linear quadratic regulator (LQR) to design PI gains that can be applied in the close proximity of x_{goal}. To compensate for steady state errors, we augment the system with additional states to integrate the state error over time. The augmented LQR problem is given by

$$\min_u \int_0^\infty \hat{x}^T Q_x \hat{x} + \hat{u}^T Q_u \hat{u} + \hat{z}^T Q_i \hat{z} \, \mathrm{d}t \tag{14}$$
$$\text{s.t. } \dot{\hat{x}} = A\hat{x} + B\hat{u},\ \dot{\hat{z}} = \hat{x}_{[\text{"states"}]},\ z(0) = 0$$
$$\hat{x} := x - x_{\text{g}},\ \hat{u} := u - u_{\text{g}}$$
$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{(x_{\text{g}},u_{\text{g}})}, \quad B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{(x_{\text{g}},u_{\text{g}})}$$

where z is the integrated error that augments the system, x_{g} and u_{g} are short notations for x_{goal} and u_{goal}. State vector x contains three joint states and three corresponding joint velocities. The weight matrices Q_x, Q_u and Q_i are diagonal matrices with values as follows:

$$\begin{aligned} Q_x &= \text{diag}\left([80, 350, 100, 0, 0, 0]\right) \\ Q_u &= \text{diag}\left([1.0, 1.0, 1.0]\right) \\ Q_i &= \text{diag}\left([10, 4500, 800, 0, 0, 0]\right) \end{aligned}$$

We use Matlab's routine `lqr` to solve the resulting algebraic Riccati equation and get the gain matrices K and K_i that determine the state space controller

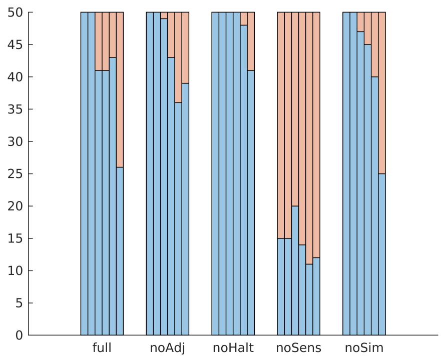
$$u(t) = u_{\text{goal}} - K(x(t) - x_{\text{goal}}) - K_i z(t). \tag{15}$$

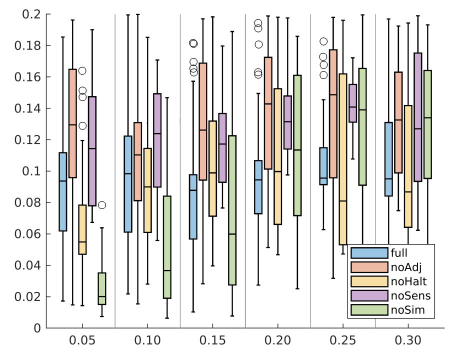
For each trajectory in the test set, we start with the trained controller evaluating the GP and switch to the PI controller (15) as soon as the normed distance of the joint states from the goal state falls below some threshold c_{in}. Once the system switched to the linear controller, we only switch back to the GP controller if the normed distance of the full system state to the goal state exceeds c_{out}.

- Start LQR control if $\|x_{1,2,3}(t) - x_{\text{goal},1,2,3}\| \le c_{\text{in}}$
- Leave LQR control if $\|x(t) - x_{\text{goal}}\| > c_{\text{out}}$

We say that the system approaches the goal state if the distance between system state and goal state falls below c_{in}, and reaches the goal state if the error is smaller than 10^{-3}. As soon as the simulated system leaves the area around the goal state that is covered by the LQR controller (15), we stop the simulation and mark this instance as failed.

We simulate the movement of the robot arm starting from the start states of the trajectories in our test set from [1] and compare the time needed to reach the goal state. The values $c_{\text{in}} = 0.15$ and $c_{\text{out}} = 0.6$ turned out to be viable. The results are given in Figure 3. The trajectories reach the goal state (distance below 10^{-3}), improving the final distance compared to the simulation without LQR control by two orders of magnitude. This comes with an increased cost and end time.


(a) Number of successful simulations (in blue).


(b) Distance at closest state

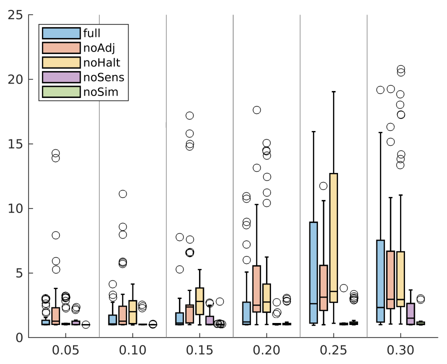

(c) Ratio of end times

Fig. 2: Comparison of the five learned control policies with all start state generation strategies and with one strategy missing. The results are computed on the test set described in Subsec. III-A. The bar plot visualizes the number of successes for six different distances of start states from the initial start state. The two box plots presents two important performance criteria. The boxes encompass the interquartile range (IQR), the maximal whisker length is $1.5 \cdot$ IQR, outliers are marked by $\circ$.

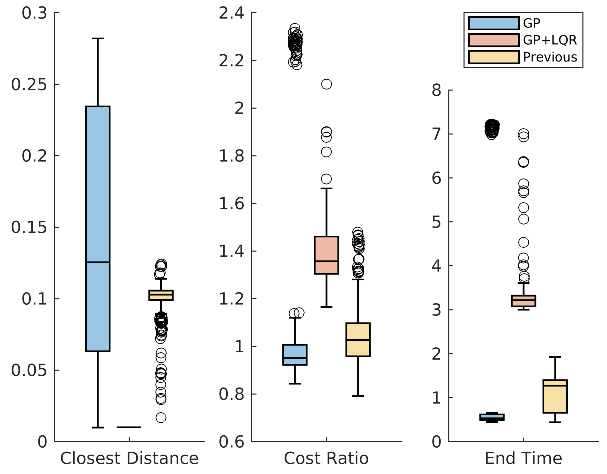


Fig. 3: Comparison of results in [1] (*Previous*) with results of this approach with and without LQR control ((*GP+LQR*), (*GP*), see Subsec. III-C and III-B). From left to right: Closest distance to the goal state, ratio of Lagrange cost term of simulated to optimal trajectory, and end time in simulation.

IV. Conclusion

Applying the changes described in this work, we are able to improve the performance of our method presented in [1]. We have developed three new strategies to identify start states for new trajectories such that they provide useful information to train the near-optimal control policy. The simulation based strategy does not show the expected results. Nevertheless, our evaluation demonstrates that the sensitivity based and the Halton based start state generators are valuable strategies improving the method developed in the previous work. This underlines the importance of a sensible selection of start states for new trajectories and justifies our effort in this subject. Further, we have examined the switch-over in proximity to the goal state from the learned near-optimal control to a stabilizing LQR controller. Although this impacts optimality, it is a crucial complement for the learned controller to make the system reach the goal state.

References

[1] C. Zelch, J. Peters, and O. von Stryk, "Learning control policies from optimal trajectories," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2529–2535.

[2] 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.

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

[4] 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.

[5] T. A. Howell, C. Fu, and Z. Manchester, "Direct policy optimization using deterministic sampling and collocation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5324–5331, 2021.

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

[7] P. Ghosh and B. A. Conway, "Near-optimal feedback strategies synthesized using a spatial statistical approach," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 4, pp. 905–919, 2013.

[8] 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.

[9] B. Da Silva, G. Konidaris, and A. Barto, "Active learning of parameterized skills," in *Proc. of the 31st Int. Conf. on Mach. Learn.*, ser. Proc. of Mach. Learn. Res., vol. 32, no. 2. PMLR, 2014, pp. 1737–1745.

[10] B. da Silva, G. Baldassarre, G. Konidaris, and A. Barto, "Learning parameterized motor skills on a humanoid robot," in *2014 IEEE Int. Conf. on Robot. and Automat. (ICRA)*, May 2014, pp. 5239–5244.

[11] S. Levine and V. Koltun, "Guided policy search," in *Proc. of the 30th Int. Conf. on Mach. Learn.*, ser. Proc. of Mach. Learn. Res., vol. 28, no. 3. PMLR, 17–19 Jun 2013, pp. 1–9.

[12] 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.

[13] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, 2020.

[14] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2986–2993.

[15] W. X. Merkt, V. Ivan, T. Dinev, I. Havoutis, and S. Vijayakumar, "Memory clustering using persistent homology for multimodality- and discontinuity-sensitive learning of optimal control warm-starts," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1649–1660, 2021.

[16] O. v. Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Ann. of Oper. Res.*, vol. 37, no. 1, pp. 357–373, 1992.

[17] J. Bonnans, Frederic, D. Giorgi, V. Grelard, B. Heymann, S. Maindrault, P. Martinon, O. Tissot, and J. Liu, "Bocop – a collection of examples," INRIA, Tech. Rep., 2017. [Online]. Available: http://www.bocop.org

[18] M. Valdenegro-Toro and D. S. Mori, "A deeper look into aleatoric and epistemic uncertainty disentanglement," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, pp. 1508–1516.

[19] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[20] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059.

[21] ——, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[22] E. Kalnay, *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

[23] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, no. 1, pp. 84–90, 1960.

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

[25] O. v. Stryk and M. Schlemmer, "Optimal control of the industrial robot manutec r3," in *Comput. Optimal Control, Int. Ser. of Numer. Math.*, vol. 115. Basel: Birkhäuser, 1994, pp. 367–382.

[26] 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.

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