Model-Free Trajectory Optimization for Reinforcement Learning

Riad Akrour[1] AKROUR@IAS.TU-DARMSTADT.DE
Abbas Abdolmaleki[3] ABBAS.A@UA.PT
Hany Abdulsamad[2] ABDULSAMAD@IAS.TU-DARMSTADT.DE
Gerhard Neumann[1] NEUMANN@IAS.TU-DARMSTADT.DE
1: CLAS, 2: IAS, TU Darmstadt, Darmstadt, Germany
3: IEETA, University of Aveiro, Aveiro, Portugal

Abstract

Many of the recent Trajectory Optimization algorithms alternate between local approximation of the dynamics and conservative policy update. However, linearly approximating the dynamics in order to derive the new policy can bias the update and prevent convergence to the optimal policy. In this article, we propose a new model-free algorithm that backpropagates a local quadratic time-dependent Q-Function, allowing the derivation of the policy update in closed form. Our policy update ensures exact KL-constraint satisfaction without simplifying assumptions on the system dynamics demonstrating improved performance in comparison to related Trajectory Optimization algorithms linearizing the dynamics.

1. Introduction

Trajectory Optimization methods based on stochastic optimal control (Todorov, 2006; Theodorou et al., 2009; Todorov & Tassa, 2009) have been very successful in learning high dimensional controls in complex settings such as end-to-end control of physical systems (Levine & Abbeel, 2014). These methods are based on a time-dependent linearization of the dynamics model around the mean trajectory in order to obtain a closed form update of the policy as a Linear-Quadratic Regulator (LQR). This linearization is then repeated locally for the new policy at every iteration. However, the linearization of the dynamics might induce a bias and impede the algorithm from converging to the optimal policy. To circumvent this limitation, we propose in this paper a novel model-free trajectory optimization algorithm (MOTO) couched in the approximate policy iteration framework. At each iteration, a Q-Function is estimated locally around the current trajectory distribution using a time-dependent quadratic function. Afterward, the policy is updated according to a new information theoretic formulation that bounds a KL-divergence in closed form.

MOTO is well suited for high dimensional state space and continuous action control problems. The policy is represented by a time-dependent stochastic linear-feedback controller which is updated by a Q-Function propagated backward in time. We extend the work of (Abdolmaleki et al., 2015), which was proposed in the domain of stochastic search (having no notion of state space nor of sequential decisions), to that of sequential decision making and show that our policy class can be updated under a KL-constraint in closed form, when the learned Q-Function is a quadratic function of the state and action space. In order to maximize sample efficiency we rely on importance sampling to reuse transition samples from policies of all time-steps and all previous iterations in a principled way. MOTO is able to solve complex control problems despite the simplicity of the Q-Function thanks to two key properties: i) the learned Q-Function is fitted to samples of the current policy, which ensures that the function is *valid locally* and ii) the closed form update of the policy ensures that the KL-constraint is satisfied exactly irrespective of the number of samples or the non-linearity of the dynamics, which ensures that the Q-Function is *used locally*.

The experimental section demonstrates that on tasks with highly non-linear dynamics MOTO outperforms similar methods that rely on a linearization of these dynamics. Additionally, it is shown on a simulated Robot Table Tennis Task that MOTO is able to scale to high dimensional tasks while keeping the sample complexity relatively low; amenable to a direct application to a physical system.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

2. Notations

Consider an undiscounted finite-horizon Markov Decision Process (MDP) of horizon T with state space $\mathcal{S} \subset \mathbb{R}^{\mathrm{d}_s}$ and action space $\mathcal{A} \subset \mathbb{R}^{\mathrm{d}_a}$. The transition function $p(s_{t+1}|s_t, a_t)$, which gives the probability (density) of transitioning to state s_{t+1} upon the execution of action a_t in s_t, is assumed to be time-independent; while there are T time-dependent reward functions $r_t : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. A policy π is defined by a set of time-dependent density functions π_t, where $\pi_t(a|s)$ is the probability of executing action a in state s at time-step t. The goal is to find the optimal policy $\pi^* = \{\pi_1^*, \ldots, \pi_T^*\}$ maximizing the policy return $J(\pi) = \mathbb{E}_{s_1,a_1,\ldots}\left[\sum_{t=1}^{T} r_t(s_t, a_t)\right]$, where the expectation is taken w.r.t. all the random variables s_t and a_t such that $s_1 \sim \rho_1$ follows the distribution of the initial state, $a_t \sim \pi_t(.|s_t)$ and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$.

As is common in Policy Search (Deisenroth et al., 2013), MOTO operates on a restricted class of parametrized policies $\pi_\theta, \theta \in \mathbb{R}^{\mathrm{d}_\theta}$ and is an iterative algorithm comprising two main steps, *policy evaluation* and *policy update*. Throughout this paper, we will assume that each time-dependent policy is parametrized by $\theta_t = \{K_t, k_t, \Sigma_t\}$ such that π_{θ_t} is of linear-Gaussian form $\pi_{\theta_t}(a|s) = \mathcal{N}(K_t s + k_t, \Sigma_t)$, where the gain matrix K_t is a $d_a \times d_s$ matrix, the bias term k_t is a d_a dimensional column vector and the covariance matrix Σ_t which controls the exploration of the policy is of dimension $d_a \times d_a$; yielding a total number of parameters across all time-steps of $d_\theta = T(d_a d_s + \frac{1}{2} d_a(d_a + 3))$.

The policy at iteration i of the algorithm is denoted by π^i and following standard definitions, the Q-Function of π^i at time-step t is denoted by $Q_t^i(s, a) = \mathbb{E}_{s_t,a_t,\ldots}\left[\sum_{t'=t}^{T} r_{t'}(s_{t'}, a_{t'})\right]$ with $(s_t, a_t) = (s, a)$ and $a_{t'} \sim \pi_{t'}^i(.|s_{t'}), \forall t' > t$. While the V-Function is given by $V_t^i(s) = \mathbb{E}_{a\sim\pi_t(.|s)}\left[Q_t^\pi(s, a)\right]$ and the Advantage Function by $A_t^i(s, a) = Q_t^i(s, a) - V_t^i(s)$. Furthermore, the state distribution at time-step t of policy π^i is denoted by $\rho_t^i(s)$. In order to keep notations simple, the time-step (resp. the iteration number) is occasionally dropped when all the elements of an equation apply similarly to all of them.

3. Information-Theoretic Policy Update

MOTO alternates between policy evaluation and policy update. At each iteration i, the policy evaluation step generates a set of M rollouts[1] from the policy π^i in order to estimate from samples the Q-Function $\tilde{Q}^i$ (Sec. 4.1) and the state distribution $\tilde{\rho}^i$ (Sec. 4.3). Using these quantities, an information-theoretic policy update is derived at each time-step as a solution of a constrained optimization problem to compute the new policy π^{i+1}.

[1]A rollout is a Monte Carlo simulation of a trajectory according to ρ_1, π and p or the execution of π on a physical system.

3.1. Optimization Problem

The goal of the policy update is to return a new policy π^{i+1} that maximizes the Q-Function $\tilde{Q}^i$ in expectation under the state distribution $\tilde{p}_i$ of the previous policy π^i. In order to limit policy oscillation between iterations, the KL w.r.t. π^i is upper bounded. The use of the KL divergence to define the step-size of the policy update has already been successfully applied in prior work (Peters et al., 2010; Levine & Abbeel, 2014; Schulman et al., 2015). Additionally, we lower bound the entropy of π^{i+1} in order to better control the reduction of exploration. The optimization problem to obtain π^{i+1} is thus given by the following non-linear program:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \int\int \tilde{\rho}_t^i(s)\pi(a|s)\tilde{Q}_t^i(s, a)\mathrm{d}a\mathrm{d}s, && (1)\\ \text{subject to} \quad & \mathbb{E}_{s\sim\tilde{\rho}_t^i(s)}\left[\mathrm{KL}\left(\pi(.|s)|\pi_t^i(.|s)\right)\right] \leq \epsilon, && (2)\\ & \mathbb{E}_{s\sim\tilde{\rho}_t^i(s)}\left[\mathcal{H}\left(\pi(.|s)\right)\right] \geq \beta. && (3) \end{aligned}$$

The KL between two distributions p and q is given by $\mathrm{KL}(p|q) = \int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$ while the entropy $\mathcal{H}$ is given by $\mathcal{H} = -\int p(x) \log p(x) \mathrm{d}x$. The step-size ϵ is a hyper-parameter of the algorithm kept constant throughout the iterations while β is set according to the entropy of π_t^i, $\beta = \mathbb{E}_{s\sim\tilde{\rho}_t^i(s)}\left[\mathcal{H}\left(\pi_t^i(.|s)\right)\right] - \beta_0$ and β_0 is a constant entropy reduction hyper-parameter.

Eq. (1) indicates that π_t^{i+1} maximizes $\tilde{Q}_t^i$ in expectation under its own action distribution and the state distribution of π_t^i. Eq. (2) bounds the average change in the policy to the step-size ϵ while Eq. (3) controls the exploration-exploitation trade-off and ensures that the exploration in the action space (which is directly linked to the entropy of the policy) is not reduced too quickly. A similar constraint was introduced in the stochastic search domain by (Abdolmaleki et al., 2015), and was shown to avoid premature convergence. However, this constraint is even more crucial in our setting because of the inherent *non-stationarity* of the objective function being optimized at each iteration. The cause for the non-stationarity of the objective optimized at time-step t in the policy update is twofold: i) updates of policies $\pi_{t'}$ with time-step $t' > t$ will modify in the next iteration of the algorithm $\tilde{Q}_t$ as a function of s and a and hence the optimization landscape as a function of the policy parameters, ii) updates of policies with time-step $t' < t$ will induce a change in the state distribution ρ_t. If the policy had unlimited expressiveness, the optimal solution of Eq. (1) would be to choose $\arg\max_a \tilde{Q}_t$ irrespective of ρ_t. However, due to the restricted class of the policy, any change in ρ_t will likely change the optimization landscape including the position of the optimal policy parameter. Hence, Eq. (3) ensures that exploration in action space is maintained as the optimization landscape evolves and avoids premature convergence.

3.2. Closed Form Update

Using the method of Lagrange multipliers, the solution of the optimization problem in section 3.1 is given by

$$\pi'_t(a|s) \propto \pi_t(a|s)^{\eta^*/(\eta^*+\omega^*)} \exp\left(\frac{\tilde{Q}_t(s,a)}{\eta^*+\omega^*}\right), \quad (4)$$

with η^* and ω^* being the optimal Lagrange multipliers related to the KL and entropy constraints respectively. Assuming that $\tilde{Q}_t(s,a)$ is of quadratic form in a and s

$$\tilde{Q}_t(s,a) = \frac{1}{2}a^T Q_{aa} a + a^T Q_{as} s + a^T q_a + q(s), \quad (5)$$

with $q(s)$ grouping all terms of $\tilde{Q}_t(s,a)$ that do not depend[2] on a, then $\pi'_t(a|s)$ is again of linear-Gaussian form

$$\pi'_t(a|s) = \mathcal{N}(a|FLs + Ff, F(\eta^*+\omega^*)),$$

such that the gain matrix, bias and covariance matrix of π'_t are function of matrices F and L and vector f where

$$F = (\eta^*\Sigma_t^{-1} - Q_{aa})^{-1}, \quad L = \eta^*\Sigma_t^{-1}K_t + Q_{as},$$
$$f = \eta^*\Sigma_t^{-1}k_t + q_a.$$

Note that, for $\eta\Sigma_t^{-1} - Q_{aa}$ to be invertible, either $Q_t(s,.)$ needs to be concave is a (i.e. Q_{aa} is negative semidefinite), or η needs to be large enough (and for a given Q_{aa} such an η always exists). Efficient algorithms for learning model parameters with a specific semidefinite shape are available (Bhojanapalli et al., 2015). However, as it is desired for the new policy π'_t to have a small KL divergence w.r.t. π_t, the resulting η was always large enough in our experiments and F well defined, without imposing additional constraint on the shape of Q_{aa}.

3.3. Dual Minimization

The Lagrangian multipliers η and ω are obtained by minimizing the convex dual function

$$g_t(\eta,\omega) = \eta\epsilon - \omega\beta + (\eta+\omega)\int \tilde{\rho}_t(s)$$
$$\log\left(\int \pi(a|s)^{\eta/(\eta+\omega)} \exp\left(\tilde{Q}_t(s,a)/(\eta+\omega)\right)\right)\mathrm{ds}.$$

The dual function further simplifies thanks to the quadratic form of $\tilde{Q}_t(s,a)$ and by additionally assuming normality of the state distribution $\tilde{\rho}_t(\mathbf{s}) = \mathcal{N}(\mathbf{s}|\mu_\mathbf{s}, \Sigma_\mathbf{s})$ to the function $g_t(\eta,\omega) = \eta\epsilon - \omega\beta + \mu_\mathbf{s}^T M \mu_\mathbf{s} + \mathrm{tr}(\Sigma_s M) + \mu_\mathbf{s}^T m + m_0$, which can be efficiently optimized by gradient descent to obtain η^* and ω^*. The full expression of the dual function, including the definition of M, m and m_0 in addition to the partial derivatives $\frac{\partial g_t(\eta,\omega)}{\partial\eta}$ and $\frac{\partial g_t(\eta,\omega)}{\partial\omega}$ are given in the supplementary material.

[2]Constant terms and terms depending on s but not a won't appear in the policy update. As such, and albeit we only refer in this paper to $Q_t(s,a)$, the Advantage Function $A_t(s,a)$ can be used interchangeably in lieu of $Q_t(s,a)$ for updating the policy.

4. Sample Efficient Policy Evaluation

The KL constraint introduced in the policy update gives rise to a non-linear optimization problem. This problem can still be solved in closed form for the class of linear-Gaussian policies, if the learned function $\tilde{Q}_t^i$ is quadratic in s and a. The first subsection introduces the main supervised learning problem solved during the policy evaluation for learning $\tilde{Q}_t^i$ while the remaining subsections discuss how to improve its sample efficiency.

4.1. The Supervised Learning Problem

In the remainder of the section, we will be interested in finding the parameter $\mathbf{w}$ of a linear model $\tilde{Q}_t^i = \langle\mathbf{w}, \phi(s,a)\rangle$, where the feature function ϕ contains a bias and all the linear and quadratic terms of s and a, yielding $1 + (d_a+d_s)(d_a+d_s+3)/2$ parameters. $\tilde{Q}_t^i$ can subsequently be written as in Eq. (5) by extracting Q_{aa}, Q_{as} and q_a from $\mathbf{w}$.

At each iteration i, M rollouts are performed following π^i. Let us initially assume that $\tilde{Q}_t^i$ is learned only from samples $\mathcal{D}_t^i = \{s_t^{[k]}, a_t^{[k]}, s_{t+1}^{[k]}; k = 1..M\}$ gathered by the execution of the M rollouts. The parameter $\mathbf{w}$ of $\tilde{Q}_t^i$ is learned by regularized linear least square regression

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{M}\sum_{k=1}^{M}\big(\langle\mathbf{w}, \phi(s_t^{[k]}, a_t^{[k]})\rangle -$$
$$\hat{Q}_t^i(s_t^{[k]}, a_t^{[k]})\big)^2 + \lambda\mathbf{w}^T\mathbf{w}. \quad (6)$$

The target value $\hat{Q}_t^i(s^{[k]}, a^{[k]})$ is a noisy estimate of the true value $Q_t^i(s_t^{[k]}, a_t^{[k]})$. We will distinguish two cases for obtaining the estimate $\hat{Q}_t^i(s_t^{[k]}, a_t^{[k]})$.

Monte-Carlo Estimate. This estimate is obtained by summing the future rewards for each trajectory k, yielding $\hat{Q}_t^i(s_t^{[k]}, a_t^{[k]}) = \sum_{t'=t}^{T} r_{t'}(s_{t'}^{[k]}, a_{t'}^{[k]})$. This estimator is known to have no bias but high variance. The variance can be reduced by averaging over multiple rollouts, assuming we can reset to states $s_t^{[k]}$. However, such an assumption would severely limit the applicability of the algorithm on physical systems.

Dynamic Programming. In order to reduce the variance, this estimate exploits the V-Function to reduce the noise of the expected rewards of time-steps $t' > t$ through the following identity

$$\hat{Q}^i_t(s^{[k]}_t, a^{[k]}_t) = r_t(s^{[k]}_t, a^{[k]}_t) + \hat{V}^i_{t+1}(s^{[k]}_{t+1}), \quad (7)$$

that is unbiased if $\hat{V}^i_{t+1}$ is. However, we will use for $\hat{V}^i_{t+1}$ a V-Function $\tilde{V}^i_{t+1}$ learned recursively in time. This might introduce a bias which will accumulate as t goes to 1. Fortunately, $\tilde{V}$ is not restricted by our algorithm to be of a particular class as it does not appear in the policy update. Hence, the bias can be reduced by increasing the complexity of the function approximator class. Nonetheless, in this paper, a quadratic function will also be used for the V-Function which worked well in our experiments.

The V-Function is learned by first assuming that $\tilde{V}^i_{T+1}$ is the zero function[3], then recursively in time, the function $\tilde{V}^i_{t+1}$ and the transition samples in $\mathcal{D}^i_t$ are used to fit the parametric function $\tilde{V}^i_t$ by minimizing the squared loss $\sum_{k=1}^M \left(\hat{Q}^i_t(s^{[k]}_t, a^{[k]}_t) - \tilde{V}^i_t(s^{[k]}_t)\right)^2$.

In addition to reducing the variance of the estimate $\hat{Q}^i_t(s^{[k]}_t, a^{[k]}_t)$, the choice of learning a V-Function is further justified by the increased possibility of reusing sample transitions from all time-steps and previous iterations.

4.2. Sample Reuse

In order to improve the sample efficiency of our approach, we will reuse samples from different time-steps and iterations using importance sampling. Let the expected loss which $\tilde{Q}^i_t$ minimizes under the assumption of an infinite number of samples be

$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathbb{E}[\ell^i_t(s, a, s'; \mathbf{w})],$$

where the loss ℓ^i_t is the inner term within the sum in Eq. (6); the estimate $\hat{Q}^i_t(s^{[k]}_t, a^{[k]}_t)$ is taken as in Eq. (7) and the expectation is with respect to the current state $s \sim \rho^i_t$, the action $a \sim \pi^i_t(.|s)$ and the next state $s' \sim p(.|s, a)$.

Reusing samples from different time-steps. To use transition samples from all time-steps when learning $\tilde{Q}^i_t$, we rely on importance sampling, where the importance weight (IW) is given by the ratio between the state-action probability of the current time-step $z^i_t(s, a) = \rho^i_t(s)\pi^i_t(a|s)$ divided by the time-independent state-action probability of π^i given by $z^i(s, a) = \frac{1}{T}\sum_{t=1}^T z^i_t(s, a)$. The expected loss minimized by $\tilde{Q}^i_t$ becomes

$$\min_{\mathbf{w}} \mathbb{E}\left[\frac{z^i_t(s, a)}{z^i(s, a)} \ell^i_t(s, a, s'; \mathbf{w}) \mid (s, a) \sim z^i(s, a)\right]. \quad (8)$$

[3]Alternatively one could assume the presence of a final reward $R_{T+1}(s_{T+1})$, as is usually formulated in control tasks (Bertsekas, 1995), to which V^i_{T+1} could be initialized to.

Since the transition probabilities are not time-dependent they cancel out from the IW. Upon the computation of the IW, weighted least square regression is used to minimize an empirical estimate of (8) for the dataset $\mathcal{D}^i = \cup_{t=1}^T \mathcal{D}^i_t$. Note that the (nominator of the) IW needs to be recomputed at every time-step for all samples $(s, a) \in \mathcal{D}^i$. Additionally, if the rewards are time-dependent, the estimate $\hat{Q}^i_t(s^{[k]}_t, a^{[k]}_t)$ in Eq. (7) needs to be recomputed with the current time-dependent reward, assuming the reward function is known.

Reusing samples from previous iterations. Following a similar reasoning, at a given time-step t, samples from previous iterations can be reused for learning $\tilde{Q}^i_t$. In this case, we have access to the samples of the state-action distribution $z^{1:i}_t(s, a) \propto \sum_{j=1}^i z^j_t(s, a)$. The computation of $z^{1:i}_t$ requires the storage of all previous policies and state distributions. Thus, we will in practice limit ourselves to the K last iterations.

Finally, both forms of sample reuse will be combined for learning $\tilde{Q}^i_t$ under the complete dataset up to iteration i, $\mathcal{D}^{1:i} = \cup_{j=1}^i \mathcal{D}^j$ using weighted least square regression where the IW are given by $z^i_t(s, a)/z^{1:i}(s, a)$ with $z^{1:i}(s, a) \propto \sum_{t=1}^T z^{1:i}_t(s, a)$.

4.3. Estimating the State Distribution

To compute the IW, the state distribution at every time-step ρ^i_t needs to be estimated. Since M rollouts are sampled for every policy π^i only M state samples are available for the estimation of ρ^i_t, necessitating again the reuse of previous samples to cope with higher dimensional control tasks.

Forward propagation of the state distribution. The first investigated solution for the estimation of the state distribution is the propagation of the estimate $\tilde{\rho}^i_t$ forward in time. Starting from $\tilde{\rho}^i_1$ which is identical for all iterations, importance sampling is used to learn $\tilde{\rho}^i_{t+1}$ with $t > 1$ from samples $(s_t, a_t, s_{t+1}) \in \mathcal{D}^{1:i}_t$ by weighted maximum-likelihood; where each sample s_{t+1} is weighted by $z^i_t(s_t, a_t)/z^{1:i}_t(s_t, a_t)$. And the computation of this IW only depends on the previously estimated state distribution $\tilde{\rho}^i_t$. In practice however, the estimate $\tilde{\rho}^i_t$ might entail errors despite the use of all samples from past iterations, which are propagated forward leading to a degeneracy of the number of effective samples in latter time-steps.

State distribution of a mixture policy. The second considered solution for the estimation of $\tilde{\rho}^i_t$ is based on the intuition that due to to the KL constraint during the policy update, successive policies are close to each other and state samples from previous iterations can thus be reused in a simpler manner. Specifically, $\tilde{\rho}^i_t$ will be learned from samples of the mixture policy $\pi^{1:i} \propto \sum_{j=1}^i \gamma^{i-j}\pi^j$ which selects a policy from previous iterations with an exponen-

Algorithm 1 Model-Free Trajectory Optimization (MOTO)

```
**Input:** Initial policy $\pi^0$, number of trajectories per iteration M, step-size $\epsilon$ and entropy reduction rate $\beta_0$
**Output:** Policy $\pi^N$
**for** $i = 0$ **to** $N - 1$ **do**
    Sample M trajectories from $\pi^i$
    **for** $t = T$ **to** $1$ **do**
        Estimate state distribution $\tilde{\rho}_t^i$                    (Sec. 4.3)
        Compute IW for all $(s, a, s') \in \mathcal{D}^{1:i}$               (Sec. 4.2)
        Estimate the Q-Function $\tilde{Q}_t^i$                     (Sec. 4.1)
        Optimize: $(\eta^*, \omega^*) = \arg\min g_t^i(\eta, \omega)$   (Sec. 3.3)
        Update $\pi_t^{i+1}$ using $\eta^*$, $\omega^*$,$\tilde{\rho}_t^i$ and $\tilde{Q}_t^i$      (Sec. 3.2)
    **end for**
**end for**
```

tially decaying (w.r.t. to the iteration number) probability and executes it for a whole rollout. In practice, the decay factor γ is selected according to the dimensionality of the problem, the number of samples per iterations M and the KL upper bound ϵ (intuitively, a smaller ϵ yields closer policies and henceforth more reusable samples). The estimated state distribution $\tilde{\rho}_t^i$ is learned as a Gaussian distribution by weighted maximum likelihood from samples of $\mathcal{D}_t^{1:i}$ where a sample of iteration j is weighted by γ^{i-j}.

MOTO is summarized in Alg. 1. The innermost loop is split between policy evaluation (Sec. 4) and policy update (Sec. 3). For every time-step t, once the state distribution $\tilde{\rho}_t^i$ is estimated, the IWs of all the transition samples are computed and used to learn the Q-Function (and the V-Function using the same IWs, if dynamic programming is used when estimating the Q-Function), concluding the policy evaluation part. Subsequently, the components of the quadratic model $\tilde{Q}_t^i$ that depend on the action are extracted and used to find the optimal dual parameters η^* and ω^* that are respectively related to the KL and the entropy constraint, by minimizing the convex dual function g_t^i using gradient descent. The policy update then directly proceeds using the aforementioned quantities to yield a new policy π_{t+1} and the process is iterated.

In addition to the simplification of the policy update, the rationale behind the use of a local quadratic approximation for Q_t^i is twofold: i) since Q_t^i is only optimized locally (because of the KL constraint), a quadratic model would potentially contain as much information as a Hessian matrix in a second order gradient descent setting ii) If $\tilde{Q}_t$ in Eq. (4) is an arbitrarily complex model then it is common that π_t', of linear-Gaussian form, is fit by weighted maximum-likelihood (Deisenroth et al., 2013); it is clear though from Eq. (4) that however complex $\tilde{Q}_t(s, a)$ is, if both π_t and π_t' are of linear-Gaussian form then there exist a quadratic model that would result in the same policy update. Additionally, note that $\tilde{Q}_t$ is not used when learning $\tilde{Q}_{t-1}$ (sec. 4.1) and hence the bias introduced by $\tilde{Q}_t$ will not propagate back. For these reasons, we believe that choosing a more complex class for $\tilde{Q}_t$ than that of quadratic functions might not necessarily lead to an improvement of the resulting policy, for the class of linear-Gaussian policies.

5. Related Work

In the Approximate Policy Iteration scheme (Szepesvari, 2010), policy updates can potentially decrease the expected reward leading to policy oscillations (Wagner, 2011), unless the updated policy is 'close' enough to the previous one (Kakade & Langford, 2002). (Pirotta et al., 2013b) refines the lower bound proposed in (Kakade & Langford, 2002) yielding more aggressive updates, but both approaches only considered discrete action spaces. (Pirotta et al., 2013a) provides an extension to continuous domains but only for single dimensional actions.

When the action space is continuous, which is typical in e.g. robotic applications, using a stochastic policy and updating it under a KL constraint to ensure 'closeness' of successive policies has shown several empirical successes (Daniel et al., 2012; Levine & Koltun, 2014; Schulman et al., 2015). However, only an empirical sample estimate of the objective function is generally optimized (Peters et al., 2010; Schulman et al., 2015), which typically requires a high number of samples and precludes it from a direct application to physical systems. The sample complexity can be reduced when a model of the dynamics is available (Levine & Koltun, 2014) or learned (Levine & Abbeel, 2014). In the latter work, empirical evidence suggests that good policies can be learned on high dimensional continuous state-action spaces with only a few hundred episodes. The counter part being that time-dependent dynamics are assumed to be linear, which might be a simplifying assumption in practice. Learning more sophisticated models using for example Gaussian Processes was experimented by (Deisenroth & Rasmussen, 2011) and (Pan & Theodorou, 2014) in the Policy Search and Trajectory Optimization context, but it is still considered to be a challenging task, see (Deisenroth et al., 2013), chapter 3.

Most trajectory optimization methods are based on stochastic optimal control. These methods linearize the system dynamics and update the policy in closed form as an LQG. Instances of such algorithms are for example iLQG (Todorov, 2006), DDP (Jacobson & Mayne, 1970; Theodorou et al., 2010), AICO (Toussaint, 2009) and the trajectory optimization algorithm used in the GPS algorithm (Levine & Abbeel, 2014). These methods face issues in maintaining the stability of the policy update step. DDP, iLQG and AICO regularize the update by introducing a damping term in the matrix inversion step, while GPS uses a KL bound on successive trajectory distributions. These methods share the same assumptions as MOTO for ρ_t^i and π_t^i respectively considered to be of Gaussian and linear-Gaussian form. However, the additional linearization of the system dynamics boils down to assuming $p(s_t, a_t, s_{t+1})$ – and by extension the whole trajectory – to be normally distributed.

6. Experimental Validation

MOTO is experimentally validated on a set of multi-link swing-up tasks and on a robot table tennis task. The experimental section aims at analyzing the proposed algorithm from four different angles: i) the quality of the returned policy comparatively to state-of-the-art trajectory optimization algorithm, ii) the effectiveness of the proposed variance reduction and sample reuse schemes, iii) the contribution of the added entropy constraint during policy updates in finding better local optima and iv) the ability of the algorithm to scale to higher dimensional problems.

6.1. Multi-link Swing-up Tasks

A set of swing-up tasks involving a multi-link pole with respectively two and four joints is considered in this section. The set of tasks includes several variants with different torque and joint limits, introducing additional nonlinearities in the dynamics and resulting in more challenging control problems for trajectory optimization algorithms based on linearizing the dynamics. The state space consists of the joint positions and joint velocities while the control actions are the motor torques. In all the tasks, the reward function is split between an action cost and a state cost. The action cost is constant throughout the time-steps while the state cost is time-dependent and is equal to zero for all but the 20 last time-steps. During this period, a quadratic cost penalizes the state for not being the null vector, i.e. having zero velocity and reaching the upright position. Examples of successful swing-ups learned by MOTO for the double and quadruple link pole are depicted in Fig. 1.a and 2.a respectively. MOTO is compared to the trajectory optimization algorithm proposed in (Levine & Abbeel, 2014), that we will refer to as GPS[4].

GPS and MOTO both use a time-dependent linear-Gaussian policy. In order to learn the linear model of the system dynamics, GPS reuses samples from different time-steps by learning a Gaussian mixture model on all the samples and use this model to learn a joint Gaussian distribution $p(s_t, a_t, s_{t+1})$. When comparing MOTO to GPS, and in order to isolate the choice of learning a linear model of the dynamics or remain model-free, we give to both algorithm a high number of samples (200 and 400 rollouts per iteration for the double and quadruple link respectively) and bypass any kind of sample reuse.

Fig. 1.b compares GPS to two configurations of MOTO on the double link swing up task. The same initial policy and step-size ϵ are used by both algorithm. However, we found that GPS performs better with a smaller initial variance, as otherwise actions quickly hit the torque limits making the dynamics modeling harder. Fig. 1.b shows that even if the dynamics of the system are not linear, GPS still manages to improve the policy return, and eventually finds a swing-up policy. The two configurations of MOTO have an entropy reduction constant β_0 of .1 and .5. The effect of the entropy constraint is similar to the one observed in the stochastic search domain by (Abdolmaleki et al., 2015). Specifically, a smaller entropy reduction constant β_0 results in an initially slower convergence but ultimately leads to higher quality policies. In this particular task, MOTO with $\beta_0 = .1$ manages to slightly outperform GPS.

Next, GPS and MOTO are compared on the quadruple link swing-up task. We found this task to be significantly more challenging than the double link and to increase the difficulty further, soft joint limits are introduced on the three last joints in the following way: whenever a joint angle exceeds in absolute value the threshold of $\frac{2}{3}\pi$, the desired torque of the policy is ignored in favor of a linear-feedback controller that aims at pushing back the joint angle within the constrained range. As a result, Fig. 2.b shows that GPS can barely improve its average return (with the torque limits set to 25, as in the double link task.) while MOTO performs significantly better. Finally, the torque limits are reduced even further but MOTO still manages to find a swing-up policy as demonstrated by Fig. 2.a.

In the last set of comparisons, the importance of each of the components of MOTO is assessed on the double link experiment (torque limit set to 10). The number of rollouts per iteration is reduced to $M = 20$. Fig. 1.c shows that: i) the entropy constraint provides an improvement on the quality of the policy in the last iterations in exchange of a slower initial progress, ii) importance sampling greatly helps in speeding-up the convergence and iii) the Monte-Carlo estimate of $\hat{Q}_i^t$ is not adequate for such a small number of rollouts per iterations, which is further exacerbated by the fact that sample reuse of transitions from different time-steps is not possible with the Monte-Carlo estimate.

Finally, Fig. 2.c aims at finding where the balance lies between performing a small number of rollouts per iterations with small policy updates and vice versa. To do so, we start with an initial $M = 20$ and subsequently divide this number by two until $M = 5$. In each case, the entropy reduction constant is set such that, for a similar number of rollouts, the entropy is reduced by the same amount, while we choose $\gamma' = \gamma^{M/M'}$ to yield again a similar sample de-

[4]This is a slight abuse of notation as the GPS algorithm of (Levine & Abbeel, 2014) additionally feeds the optimized trajectory to an upper level policy. However, in this paper, we are only interested in the trajectory optimization part.

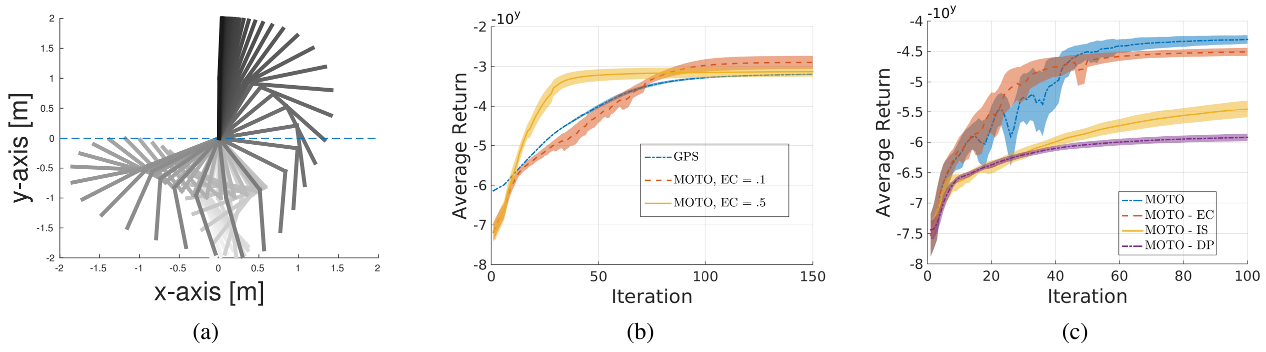

Figure 1: a) Double link swing-up policy found by MOTO. b) Comparison between GPS and MOTO on the double link swing-up task (different torque limits and state costs are applied compared to 1.c) and 2.c). c) Relevance of MOTO components; minus sign in the legend indicates the abscence of either the entropy constraint (EC), importance sampling (IS) or dynamic programming (DP).

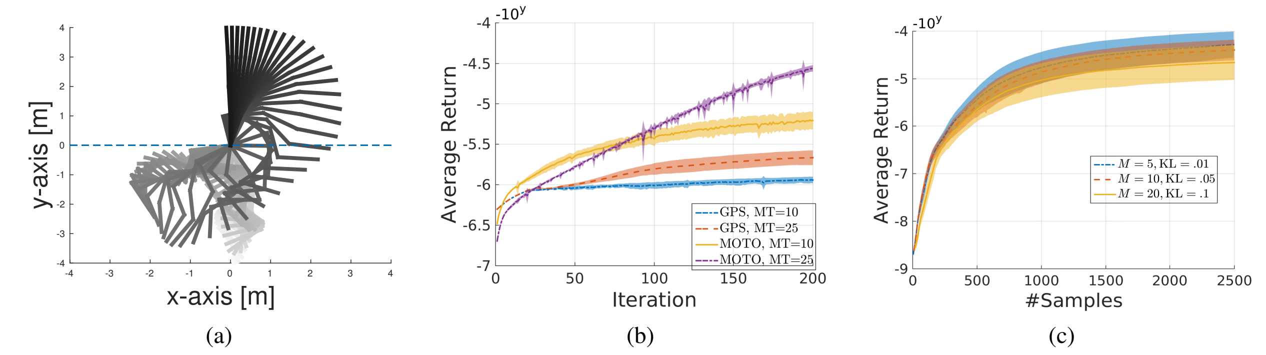

Figure 2: a) Quad link swing-up policy found by MOTO. b) Comparison between GPS and MOTO on the quad link swing-up task with restricted joint limits and two different torque limits. c) MOTO on the double link swing-up task for varying number of rollouts per episode and step-sizes. All plots of Fig. 1 and 2 are averaged over 15 runs.

cay after the same number of rollouts have been performed. Tuning ϵ, however, was more complicated and we tested several values on non-overlapping ranges for each M and selected the best one. Fig. 2.c shows that, on the double link swing-up task, a better sample efficiency is achieved with a smaller M. However, the improvement becomes negligible from $M = 10$ to $M = 5$. We also noticed a sharp decrease in the number of effective samples when M goes to 1. In this limiting case, the complexity of $z^{1:i}$ increases with the increase of the number of policies composing it, and might become a poor representation of the data-set. Fitting a state-action distribution that is more representative of the data might be the subject of future work, in order to increase the sample efficiency of the algorithm, which is crucial when applied to physical systems.

6.2. Robot Table Tennis Task

The considered robot table tennis task consists of a simulated robotic arm mounted on a floating base, having a racket attached to the end effector that has to return using

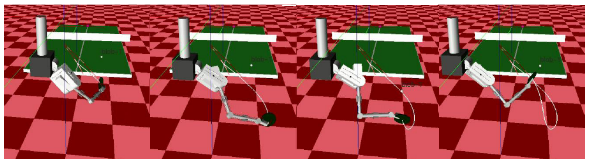

Figure 4: Robot table tennis setting and a forehand stroke learned by MOTO upon a spinning ball.

a forehand stroke incoming balls to the opposite side of the table (Fig. 4). The arm has 9 degrees of freedom comprising the six joints of the arm and the three linear joints of the base allowing (small) 3D movement. Together with the joint velocities and the 3D position of the incoming ball, the resulting state space is of dimension $d_s = 21$ and the action space is of dimension $d_a = 9$ and consists of direct torque commands.

We use the analytical player of (Mülling et al., 2011) to generate a single forehand stroke, which is subsequently used to learn from demonstration the initial policy π^1. The

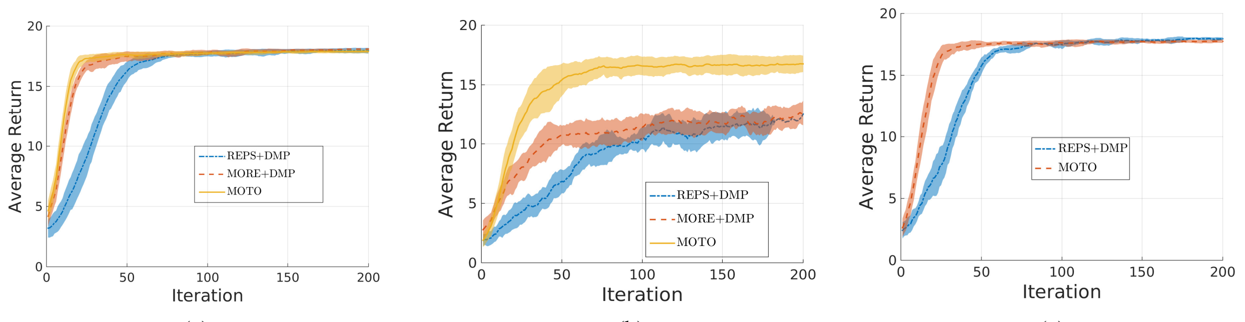

Figure 3: a) Comparison on the robot table tennis task with no noise on the initial velocity of the ball. b) Comparison on the robot table tennis task with Gaussian noise during the ball bounce on the table. c) Comparison on the robot table tennis task with initial velocity sampled uniformly in order to variate the ball position by circa 15cm at hitting time.

analytical player consists of a waiting phase (keeping the arm still), a preparation phase, a hitting phase and a return phase, which resets the arm to the waiting position of the arm. Only the preparation and hitting phase will be replaced by a learned policy. The total control time for the two learned phases comprises 300 time-steps at 500hz, although for the MOTO algorithm we subsequently divide the control frequency by a factor of 10, average the torque commands of every 10 time-steps and use these quantities as the initial bias for each of the 30 policies; while the gain matrices are initially set to zero. By doing so, the initial policy will capture the basic template of a forehand stroke but does not contain any correlation between the torque commands and the state entries such as the ball position.

Three settings of the task are considered, a noiseless case where the ball is launched with the same initial velocity, a varying context setting where the initial velocity is sampled uniformly within a fixed range and the noisy bounce setting where a Gaussian noise is added to both the x and y velocities of the ball upon bouncing on the table, to simulate the effect of a spin.

We compare MOTO to the REPS policy search algorithm (Peters et al., 2010) and the stochastic search algorithm MORE (Abdolmaleki et al., 2015) that share a related information-theoretic update. Both algorithms will optimize the parameters of a Dynamical Movement Primitive (DMP) (Ijspeert & Schaal, 2003). A DMP is a non-linear attractor system commonly used in robotics. The DMP is initialized from the same single trajectory and the two algorithms will optimize the goal joint positions and velocities of the attractor system. Note that the DMP generates a trajectory of states, which will be tracked by a linear controller using the inverse dynamics. While MOTO will directly output the torque commands and do not rely on this the inverse dynamics model.

Fig. 3.a and 3.c show that MOTO converges faster than REPS and to a smaller extent to MORE in both the noiseless and the varying context setting. This is somewhat surprising since MOTO with its time-dependent linear policy have a much higher number of parameters to optimize than the 18 parameters of the DMP's attractor. However, the resulting policy in both cases is slightly less good than that of MORE and REPS. Note that for the varying context setting, we used a contextual variant of REPS that learns a mapping from the initial ball velocity to the DMP's parameters. MORE, on the other hand couldn't be compared in this setting. Finally, Fig. 3.b shows that our policy is successfully capable of adapting to noise at ball bounce, while the other methods fail to do so since the trajectory of the DMP is not updated once generated.

7. Conclusion

We proposed in this paper a new trajectory optimization algorithm that unlike other state-of-the-art algorithms does not rely on a linearization of the dynamics. Yet, we are able to derive an efficient policy update by locally fitting a Q-Function, and outperform state-of-the-art trajectory optimization methods on the more challenging tasks. One of the main addition that would ease the transition from simulation to physical systems concerns the safety of the exploration. On a more technical side, the use of a more complex function for the estimation of the V-Function could be investigated to allow for a more refined bias-variance trade-off.

Acknowledgments

The research leading to these results was partially funded by the DFG Project *LearnRobotS* under the SPP 1527 Autonomous Learning.

References

Abdolmaleki, Abbas, Lioutikov, Rudolf, Peters, Jan R, Lau, Nuno, Pualo Reis, Luis, and Neumann, Gerhard. Model-based relative entropy stochastic search. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3523–3531. Curran Associates, Inc., 2015.

Bertsekas, Dimitri P. *Dynamic Porgramming and optimal control*. Athena Scientific, 1995.

Bhojanapalli, Srinadh, Kyrillidis, Anastasios T., and Sanghavi, Sujay. Dropping convexity for faster semi-definite optimization. *CoRR*, abs/1509.03917, 2015.

Daniel, C., Neumann, G., and Peters, J. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

Deisenroth, M. and Rasmussen, C. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *28th International Conference on Machine Learning (ICML)*, pp. 465–472, 2011.

Deisenroth, M. P., Neumann, G., and Peters, J. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, pp. 388–403, 2013.

Ijspeert, A. and Schaal, S. Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems 15*, (NIPS). MIT Press, Cambridge, MA, 2003.

Jacobson, David H. and Mayne, David Q. *Differential dynamic programming*. Modern analytic and computational methods in science and mathematics. Elsevier, New York, 1970. ISBN 0-444-00070-4.

Kakade, Sham and Langford, John. Approximately optimal approximate reinforcement learning. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pp. 267–274, 2002.

Levine, Sergey and Abbeel, Pieter. Learning neural network policies with guided policy search under unknown dynamics. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 1071–1079. Curran Associates, Inc., 2014.

Levine, Sergey and Koltun, Vladlen. Learning complex neural network policies with trajectory optimization. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 829–837, 2014.

Mülling, K., Kober, J., and Peters, J. A Biomimetic Approach to Robot Table Tennis. *Adaptive Behavior Journal*, (5), 2011.

Pan, Yunpeng and Theodorou, Evangelos. Probabilistic differential dynamic programming. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 1907–1915. Curran Associates, Inc., 2014.

Peters, J., Mülling, K., and Altun, Y. Relative Entropy Policy Search. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2010.

Pirotta, Matteo, Restelli, Marcello, and Bascetta, Luca. Adaptive step-size for policy gradient methods. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 1394–1402. Curran Associates, Inc., 2013a.

Pirotta, Matteo, Restelli, Marcello, Pecorino, Alessio, and Calandriello, Daniele. Safe policy iteration. In Dasgupta, Sanjoy and McAllester, David (eds.), *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pp. 307–315. JMLR Workshop and Conference Proceedings, May 2013b.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael I., and Moritz, Philipp. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1889–1897, 2015.

Szepesvari, Csaba. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

Theodorou, E., Tassa, Y., and Todorov, E. Stochastic Differential Dynamic Programming. In *Proceedings of the 29th American Control Conference*, (ACC 2010), Baltimore, Maryland, USA, 2010.

Theodorou, Evangelos A., Buchli, J., and Schaal, S. Path Integral Stochastic Optimal Control for Rigid Body Dynamics. In *ieee international symposium on approximate dynamic programming and reinforcement learning (adprl2009)*, 2009.

Todorov, Emanuel. Optimal control theory. *Bayesian Brain*, 2006.

Todorov, Emanuel and Tassa, Yuval. Iterative local dynamic programming. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009, Nashville, TN, USA, March 31 - April 1, 2009*, pp. 90–95, 2009.

Toussaint, M. Robot Trajectory Optimization using Approximate Inference. In *Proceedings of the 26th International Conference on Machine Learning*, (ICML), 2009.

Wagner, Paul. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. In Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2573–2581. Curran Associates, Inc., 2011.