# Likelihood-free Inference in Reinforcement Learning: Relation to REPS and Sim-to-Real Applications

**Kai Cui** *
Department of Computer Science
TU Darmstadt
Hochschulstraße 10, 64289 Darmstadt
kai.cui.de@gmail.com

**Maximilian Hensel** *
Department of Computer Science
TU Darmstadt
Hochschulstraße 10, 64289 Darmstadt
maxi.hensel@gmx.de

## Abstract

The difficulty of transferring optimized policies from simulation to reality (Sim-to-Real gap) limits the applicability of reinforcement learning (RL) to real-world problems. Prior work addresses the Sim-to-Real gap by optimizing policy and simulation parameters alternatingly. Since the likelihood function of the simulator is intractable, prior work employed episodic relative entropy policy search (REPS) – a form of black box optimization – to implicitly find an approximate posterior over simulator parameters. The core problem of likelihood free inference arises in many other areas such as computational biology or population genetics and is usually solved by different approaches such as Approximate Bayesian Computation (ABC). In this paper, we draw connections between stochastic optimization algorithms such as REPS and approximate likelihood-free inference algorithms like ABC in order to better understand their relative strengths and weaknesses and hope to show the potential of these techniques outside their usual scope of application. Potential research directions including application of REPS to likelihood-free inference problems are sketched and discussed. Secondly, we propose a separate improvement over the state-of-the-art SimOpt algorithm by optimizing the simulator w.r.t. individual transitions rather than whole trajectories.

## 1 Introduction

The application of reinforcement learning algorithms in real world environments has recently seen a wide interest and it is yet unclear how to optimally learn policies in simulators and successfully transfer them to the real world. One key point of Sim-to-Real applications is the prohibitively high cost of obtaining training samples on the real system. While it is possible to collect large scale data on real robots, the simulator-based approach offers significantly faster rates of data acquisition. However, this approach generally exhibits the so-called reality gap phenomenon, where due to the limited simulation model precision policies learned on simulations fail on the real systems.

To relieve the reality gap issue, various recent approaches exist. For example, it is possible to perform randomization on the simulation parameters – introduced by Tobin et al. [2017] as domain randomization – such that the learned policies work for a wide range of parameters. In Mehta et al. [2019] the randomization is actively seeking to find weak points of the current policy. However, designing the randomization such that both the task remains solvable and the policy remains robust is not trivial.

In Tran et al. [2017] and Louppe et al. [2017], the randomization is instead chosen such that the simulated data is as close to a few real observations as possible. This is done by using generator

*equal contribution

Preprint. Work in progress.

and discriminator networks to generate simulated observations and distinguish between simulated and real observations respectively, and focuses on huge amounts of data which is not necessarily attainable in Sim-to-Real applications. Chebotar et al. [2019] also follows the idea of fitting the domain randomization by using Relative Entropy Policy Search to find good explanatory domain randomization parameters. They show reliable transfer of policies to two real world tasks with very low amounts of real observations.

We will focus on the latter algorithm, also known as SimOpt. In the first half of this work, we will investigate if REPS, when used as in SimOpt, performs correct posterior inference. Consequently we draw connections between Relative Entropy Policy Search (REPS), likelihood-free variational inference using discrepancies and Approximate Bayesian Computation (ABC) approaches. In the second half, we show that the sample efficiency of the original SimOpt algorithm can be drastically improved if is possible to reset the simulator to specific states and make full state observations.

## 2 Background

We will show similarities between multiple approaches to inferring parameter posteriors. Our first aim is to briefly explain the approaches and concepts used throughout this work.

### 2.1 Relative Entropy Policy Search (REPS)

We briefly review the derivation of episodic REPS from Peters et al. [2010]. In the most general setting we have a space of $n$ parameters and a reward function $R(\theta) : \mathbb{R}^n \to \mathbb{R}$ which assigns a utility value to each possible parameter $\theta \in R(\theta)$. Since this objective is hard to maximize directly without specific assumptions on $R$, REPS optimizes over a distribution $\pi(\theta)$ of parameters $\theta$, also called the policy. The objective is to find the distribution which maximizes the expected reward

$$J(\pi) = \mathop{\mathbb{E}}_{\theta\sim\pi}\left[R(\theta)\right].$$

Since this problem is still difficult to solve globally, REPS solves it locally by iteratively updating $\pi$ while constraining the KL-Divergence between the new search distribution and the previous distribution $\pi^-$ at each step:

$$\begin{aligned} \max_{\pi} \quad & J(\pi(\theta)) \\ \text{s.t.} \quad & D_{KL}(\pi(\theta) \parallel \pi^-(\theta)) < \epsilon \\ & \int \pi(\theta) d\theta = 1. \end{aligned} \tag{1}$$

Equation 1 can be solved by using Lagrangian multipliers and results in the closed form solution

$$\pi(\theta) \propto \pi^-(\theta) \exp\left(\frac{R(\theta)}{\eta}\right) \tag{2}$$

where $\eta$ is obtained from minimizing the dual function. See Deisenroth et al. [2013] for a more in-depth explanation. Since we cannot represent $\pi$ explicitly, we approximate it with a class of distributions $\pi_\omega$ parameterized by $\omega$. In a second optimization step we find the best approximation by minimizing a divergence between $\pi$ and $\pi_\omega$. For the particular choice of a Gaussian $\pi_\omega$ and $D_{KL}(\pi \parallel \pi_\omega)$ as divergence, minimization results in a closed form weighted maximum likelihood estimation with weights $\exp\left(R(\theta)/\eta\right)$.

### 2.2 Variational Inference (VI)

We present variational inference as found in Blei et al. [2017]. Assume we have observed variables $\mathbf{x} = x_{1:N}$ and latent variables $z = z_{1:M}$ with joint density $p(x, z)$. The posterior

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(x, z)}{p(x)} = \frac{p(x \mid z)p(z)}{p(x)}$$

involves marginalizing the joint distribution over $\mathbf{z}$ which is often intractable to compute. Instead, VI solves this by approximating $p(z \mid x)$ with a variational distribution $q(z)$ from a family of distributions $\Phi$ by directly minimizing the distance of this approximation to the target $p(z \mid x)$:

$$q^*(z) = \arg\min_{q(z)\in\Phi} D_{KL}(q(z) \parallel p(z \mid x)).$$

The trick of variational inference lies in the ability to decompose the KL divergence to obtain a lower bound on the evidence (the so-called Evidence Lower Bound, or ELBO). By expanding the KL divergence and the conditional $p(z \mid x)$, we obtain

$$D_{KL}(q(z) \parallel p(z \mid x)) = \mathop{\mathbb{E}}_{z\sim q(z)} [\log q(\mathbf{z}) - \log p(\mathbf{x}, \mathbf{z})] + \log p(x)$$

which is equivalent to optimizing

$$\begin{aligned} \mathrm{ELBO}(q) &= \mathop{\mathbb{E}}_{z\sim q(z)} [\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z})] \\ &= \mathop{\mathbb{E}}_{z\sim q(z)} [\log p(\mathbf{x} \mid \mathbf{z})] - D_{KL}(q(z) \parallel p(z)). \end{aligned}$$

Hence, the ELBO does not require the marginal $p(x)$ and its maximization is equivalent to the original objective. The ELBO can be used for finding an approximate posterior distribution under some approximating variational family. For a review of variational inference, see Blei et al. [2017].

However, this method is not inherently likelihood-free, as we still need to evaluate the likelihood $p(\mathbf{x} \mid \mathbf{z})$. Later, we will see that the SimOpt algorithm can be seen as approximate variational inference by using discrepancies from Approximate Bayesian Computation introduced in the following.

### 2.3 Approximate Bayesian Computation (ABC)

Approximate Bayesian Computation is another method of obtaining an approximate posterior distribution of model parameters. It is applicable even if the likelihood function $\log(\mathbf{x} \mid \mathbf{z})$ is intractable to compute. We give a brief overview of the basic principles of ABC methods as found in Karabatsos et al. [2018].

Instead of using the likelihood $p(\mathbf{x} \mid \mathbf{z})$ explicitly to evaluate posterior fitness, ABC rejection sampling uses a discrepancy function $d(\cdot, \cdot)$ between observations to obtain posterior samples. In its simplest practical form, $n$ parameters $\mathbf{z}_i$ are sampled from the prior $p(\mathbf{z})$ and one corresponding data set $\hat{\mathbf{x}}_i$ is generated. Then, the discrepancy between the real data set $\mathbf{x}$ and the generated data set $\hat{\mathbf{x}}_i$ is evaluated and any parameters that result in $d(\mathbf{x}, \hat{\mathbf{x}}_i) < \epsilon$ for some threshold $\epsilon$ are accepted. This results in exact posterior samples as $\epsilon$ goes to zero. This results in algorithm 1.

**Algorithm 1** ABC Rejection Sampling

1: $\mathbf{x} \leftarrow$ Real Observations
2: $p(\mathbf{z}) \leftarrow$ Parameter Prior
3: $\epsilon \leftarrow$ Tolerance
4: $\mathcal{M} \leftarrow \{\}$
5: **for** iteration $i \in \{0, \ldots, n\}$ **do**
6: $\quad \mathbf{z}_i \sim p(\mathbf{z})$
7: $\quad \hat{\mathbf{x}}_i \sim p(\mathbf{x} \mid \mathbf{z}_i)$
8: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathbf{z}_i$ if $d(\mathbf{x}, \hat{\mathbf{x}}_i) < \epsilon$

Note that step 7 consists of running the simulation with the parameters. As we let $n$ go to infinity, we obtain a collection of posterior samples whose distribution is the real posterior. In practice, the requirement of reaching a discrepancy of exactly zero is relaxed, since this would require infinitely many samples. Instead, we can accept any samples below some bigger threshold $\epsilon$ or within some quantile. Many more sophisticated approaches to approximate likelihoods exist and can be found in the literature. We refer the interested reader to Karabatsos et al. [2018].

## 3 Connections between REPS, VI and ABC methods

When we choose the reward function of REPS to be a discrepancy $d$ similar to the one in ABC, we can observe similarities in what both algorithms are doing.

Both REPS and ABC sample from a prior and evaluate the quality of the samples. ABC keeps a collection of samples with discrepancy $d$ under a hard threshold, resembling true posterior samples for sufficiently low thresholds. REPS weighs the samples by their respective discrepancy. To illustrate that REPS can generate real posterior samples, we focus on the ideal situation where $\log(p \mid \theta)$ is computable and therefore the ELBO can be evaluated directly. With $R(\theta) = \log p(x \mid \theta)$ and Equation (2) we obtain

$$\underbrace{p_{n+1}(\theta \mid x)}_{\text{posterior}} \propto p_n(\theta) \exp\left(\frac{R(\theta)}{\eta}\right) = \underbrace{p_n(\theta)}_{\text{prior}} \underbrace{p_n(x \mid \theta)^{\frac{1}{\eta}}}_{\text{weighted likelihood}} .$$

$\eta$ is determined by the KL-constraint of REPS and influences the weighting of prior and likelihood to limit the step size. In essence, $p_n(x \mid \theta)^{\frac{1}{\eta}}$ is concave in $p_n(x \mid \theta)$ for $\eta \geq 1$, and convex $\eta \leq 1$. The convexity skews the likelihoods in favor of likelier samples, while concavity equalizes the likelihoods of samples. If $\eta$ was fixed to 1, we obtain true posterior samples. Note that if $\eta = 1$ sequentially processing the observations is equivalent to processing all observations at once.

Similar to ABC, we can substitute the log-likelihood $\log p(\hat{\mathbf{x}} \mid \mathbf{z})$ with some discrepancy on observations $\log p(\hat{\mathbf{x}} \mid \mathbf{z}) \propto -d(\hat{\mathbf{x}}, \mathbf{x})$, and therefore $R(\theta) = -d(\hat{\mathbf{x}}, \mathbf{x})$. For real observations $\hat{\mathbf{x}}$, under the limit of an extreme discrepancy function

$$d(\hat{\mathbf{x}}, \mathbf{x}) = \begin{cases} 0 & \text{if } \hat{\mathbf{x}} = \mathbf{x} \\ \infty & \text{else} \end{cases}$$

we obtain a discrepancy of zero exactly if the corresponding parameter produced valid data and therefore belongs to the posterior under the ABC rejection sampling algorithm. Since the above discrepancy function requires infinite amounts of samples, we relax this requirement by using other discrepancy functions. The difference to ABC is that here samples are not rejected if their discrepancy is below some threshold, but instead their weight and therefore influence is low. We can view this as using REPS as an approach to likelihood-free variational inference.

Instead of approximating the posterior with a variational distribution, we can also represent the posterior distribution non-parametrically. Under the assumption that a kernel density estimate can represent the posterior and that the collection of samples $\theta_{1:N}$ from the prior $p_n$ covers the support of the posterior $p_{n+1}$ sufficiently, we can sample from $p_{n+1}$: First, we sample $\theta$ from $\theta_{1:N}$ with weights $w_i = \exp(R(\theta_i)/\eta) / \sum \exp(R(\theta_i)/\eta)$. Afterwards we sample from the kernel corresponding to $\theta$ and obtain posterior samples which can be used as prior in the next iteration.

An advantage is that any priors can be used and we only assume that the posterior can be represented by a kernel density estimate. A drawback is that for the approximation to hold, sufficiently many samples must be used, which may result in a high computational complexity.

The non-parametrical posterior approach to REPS is similar to ABC-SMC (see Karabatsos et al. [2018]). The key differences are soft rejection and limited KL-divergence between posterior and prior when generating a new population, resulting in different weighting of samples and priors.

## 4 Sim-to-Real Transfer Algorithms

We propose a new approach to Sim-to-Real transfer that does not rely on a very high simulation budget and manually specified discrepancys such as the SimOpt discrepancy from Equation 5. Instead of using discrepancies between full trajectories, our algorithm exploits the fact that the usual reinforcement learning problem can be described as a Markov decision process. This allows us to decompose real observed trajectories into independent transitions and define discrepancies between single state observations instead.

**Algorithm 2** SimOpt from Chebotar et al. [2019]

$p_{\phi_0} \leftarrow$ Initial simulation parameter distribution
$\epsilon \leftarrow$ KL-divergence step for updating $p_\phi$
**for** iteration $i \in \{0, \dots, N\}$ **do**
  env $\leftarrow$ Simulation$(p_{\phi_i})$
  $\pi_{\theta, p_{\phi_i}} \leftarrow$ RL(env)
  $\tau^{ob}_{real} \sim$ RealRollout$(\pi_{\theta, p_{\phi_i}})$
  $\xi \sim$ Sample$(p_{\phi_i})$
  $\tau^{ob}_{\xi} \sim$ SimRollout$(\pi_{\theta, p_{\phi_i}}, \xi)$
  $c(\xi) \leftarrow d(\tau^{ob}_{\xi}, \tau^{ob}_{real})$
  $p_{\phi_{i+1}} \leftarrow$ UpdateDistribution$(p_{\phi_i}, \xi, c(\xi), \epsilon)$

### 4.1 Original SimOpt

We briefly present the original SimOpt algorithm as found in Chebotar et al. [2019]. Let $\mathcal{M} = (S, A, P, R, p_0, \gamma, T)$ be a finite-horizon Markov Decision Process (MDP), where $S$ and $A$ are state and action spaces, $P : S \times A \times S \to \mathbb{R}_+$ is the state-transition probability function, $R : S \times A \to \mathbb{R}$ a reward function, $p_0 : S \to \mathbb{R}_+$ an initial state distribution, $\gamma$ a reward discount factor, and $T$ a fixed horizon. Let $\tau = (s_0, a_0, \dots, s_T, a_T)$ be a trajectory of states and actions and $R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$ be the trajectory reward.

The objective of SimOpt is to find both the parameters $\theta$ of a policy $\pi_\theta(a \mid s)$ and the parameters $\phi$ of the distribution of simulation parameters $p_\phi(\xi)$ – also known as domain randomization – such that the policy found under the domain-randomized simulation maximizes the expected discounted reward over trajectories of the real system: $J_{\xi_{real}}(\pi_\theta) = \mathbb{E}_{\pi_\theta}[R(\tau)]$ where $s_0 \sim p_0, s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$ and $a_t \sim \pi_\theta(a_t \mid s_t)$. Vuong et al. [2019] formulate this as the bi-level optimization problem

$$\begin{aligned} \arg\max_{\phi} \quad & J_{\xi_{real}}(\pi_{\theta^*(\phi)}) \\ \text{s.t.} \quad & \theta^*(\phi) = \arg\max_{\theta} \mathop{\mathbb{E}}_{\xi \sim p_\phi} \left[ J_\xi(\pi_\theta) \right]. \end{aligned} \tag{3}$$

If we assume that the real system is in the class of systems simulated by the simulator, all that remains is to find parameters of the real system such that some discrepancy function $d$ between observed real trajectories $\tau^{ob}_{real}$ and simulated trajectories $\tau^{ob}_{\xi}$ is minimized, since we then assume the simulator and the real system to be the same and the optimization of the policy under the simulator will achieve the optimal result on the real system. To do this, the optimization problem

$$\phi^* = \arg\min_{\phi} \mathop{\mathbb{E}}_{\xi \sim p_\phi} \left[ \mathop{\mathbb{E}}_{\pi_\theta} \left[ d(\tau^{ob}_{\xi}, \tau^{ob}_{real}) \right] \right] \tag{4}$$

is solved in SimOpt using episodic REPS. After that, the policy is simply optimized according to the simulator using any standard reinforcement learning algorithm such as TRPO from Schulman et al. [2015]. In their example application, Chebotar et al. [2019] used the SimOpt discrepancy

$$d(\tau^{ob}_{\xi}, \tau^{ob}_{real}) = \quad w_{\ell_1} \sum_{i=0}^{T} |W(o_{i,\xi} - o_{i,real})| + w_{\ell_2} \sum_{i=0}^{T} \|W(o_{i,\xi} - o_{i,real})\|_2^2 \tag{5}$$

consisting of weighted $\ell_1$ and $\ell_2$ norms between simulation and real world observations for their observation discrepancy function $d$.

Putting it all together, this results in algorithm 2. One problem of this approach is that the effectiveness of this discrepancy depends on the problem and usually requires a high amount of simulations. Even for more or less deterministic systems, the stochasticity of the policy will inevitably cause variance in the observed trajectories and thus, many simulated trajectory samples are required to obtain sufficiently accurate updates. Additionally, sufficiently long trajectories in stochastic systems may

become misaligned over time even if the simulation parameters match exactly, causing the trajectory distance to become useless. SimOpt counteracts this by smoothing the trajectories over multiple steps, which can only compensate small misalignments.

### 4.2 Step-based SimOpt

We adapt the SimOpt algorithm to use a discrepancy based on transitions rather than full trajectories for the optimization of the domain randomization. Assuming the underlying environment is a fully observed MDP and the likelihood function of the simulator parametrized by $\xi$ was available, we can write down the likelihood of simulating a trajectory $\tau$ as

$$\log p(\tau) = \sum_{t=0}^{T} \log p_\xi(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t).$$

Maximizing the log probability of a trajectory w.r.t $\xi$, we can drop the dependency on the policy since it is constant and just maximize the probabilities of simulating the individual transitions. From this insight we derive the transition-based discrepancy

$$R(\xi) = - \mathop{\mathbb{E}}_{(s,a,s')\sim D} [(f_\xi(s, a) - s')^2], \tag{6}$$

where $D$ is a set of observed transitions and $f_\xi : S \times A \to S$ simulates one step with given state, action and parameters $\xi$ and returns the next state. The resulting algorithm is the same as algorithm 2 except for lines 8 and 9, where we instead simulate single steps and use the step-based discrepancy respectively. This discrepancy does not suffer additional variance from accumulated errors caused by the stochasticity of the policy, model inaccuracies or wrong parameters over multiple time steps. Consequently, the transition-based discrepancy has a lower variance and the number of samples required for an accurate estimate is drastically reduced.

A drawback of the step-based method is that it requires us to observe the full state of the real system compatible with the simulator. In most practical cases such as robots and industrial plants, the systems are designed to be fully observable. However, it may sometimes be necessary to use a model of the system to construct a state observer for reconstructing the internal state of the system if it is is not directly measurable. This potentially adds to the knowledge required to apply the technique to real systems.

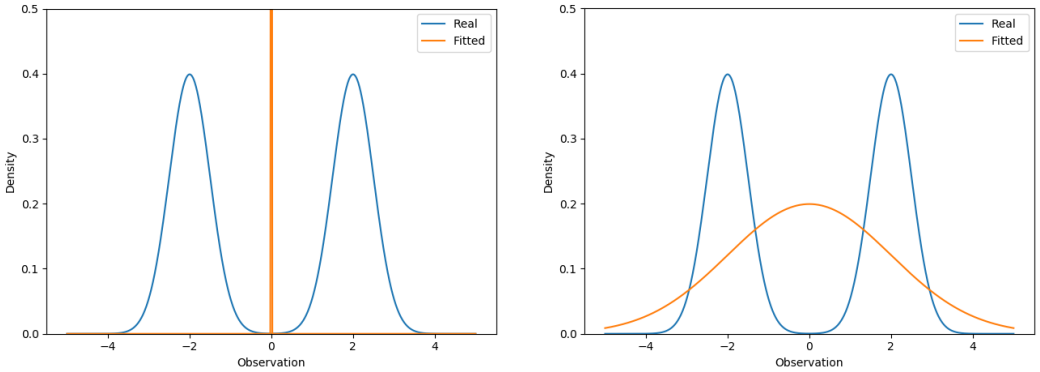
As with the original SimOpt algorithm, it remains difficult to properly represent multi-modal randomizations of the simulated environments due to two reasons: One reason is that we use a uni-modal Gaussian family due to its available closed form solutions. The other reason is that minimizing a discrepancy such as the Euclidean distance between samples becomes apparent when the real system is not deterministic and produces a distribution of observations $p(x)$. ABC methods avoid explicitly representing the posterior distribution and can thus represent multi-modal distributions since they are based on samples.

Consider Figure 1, where the observations have a bi-modal density. Approximating $p$ with some variational distribution $q$ by minimizing the Euclidean distance between samples of $p$ and $q$ may result in $q$ placing all of its probability mass centered between the two modes even if $p$ is contained in the variational class of $q$. The reason is that under the square loss it is less punishing to have a lot of small errors, rather than having a few big ones. If we choose to instead minimize a divergence between both observation distributions, we can obtain a perfect fit if $p$ is contained in the variational class of $q$. Otherwise, we still obtain a more sensible approximation (see Figure 1).

While minimizing a divergence between observation distributions is preferable, in practice this is not always possible, since ABC methods are often applied when real data is sparse and therefore no accurate distribution is available. This is counteracted by designing problem specific discrepancies which use prior expert knowledge about the problem.

## 5 Experiments

In the following, we show that if the real system lies in the class of simulated systems, the Step-based SimOpt algorithm exhibits superior performance under a very low, fixed simulation sample budget.



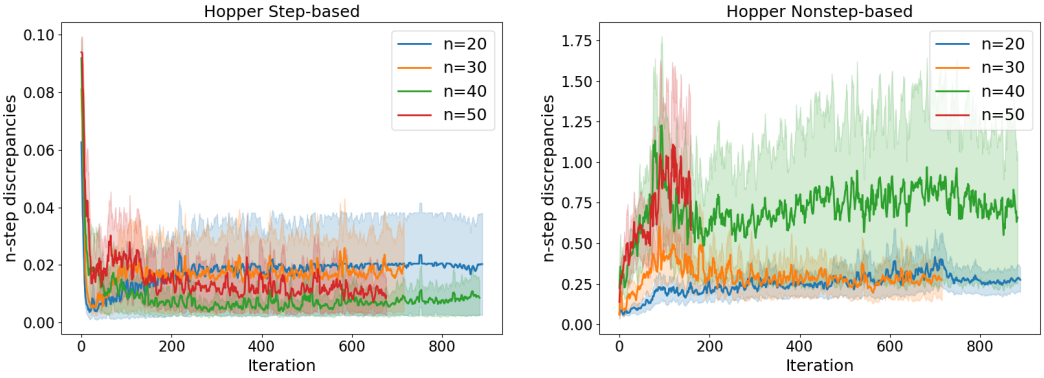

(a) Minimizing expected Euclidean distance

(b) Minimizing KL-Divergence

Figure 1: Fitting a variational Gaussian distribution to a bi-modal distribution.

All data shown is averaged over 5 runs and for each pair of step-based and non-step-based SimOpt runs, the same 'real' system variables are uniformly sampled from the parameters stated in the appendix. Detailed experiment configurations are given in the Appendix.

In Figure 2 one can see that all n-step simulation errors considered are significantly smaller than in the original SimOpt algorithm. The step-based SimOpt algorithm optimizes the optimization goal of low n-step simulation discrepancies very well, while for higher $n$ this holds less and less. The original SimOpt algorithm on the other hand was unable to optimize both low or high n-step simulation discrepancies given the very low simulation budget. One could try to progressively include training into the step-based algorithm for higher $n$ to further increase inference accuracy. We leave this open for future work.



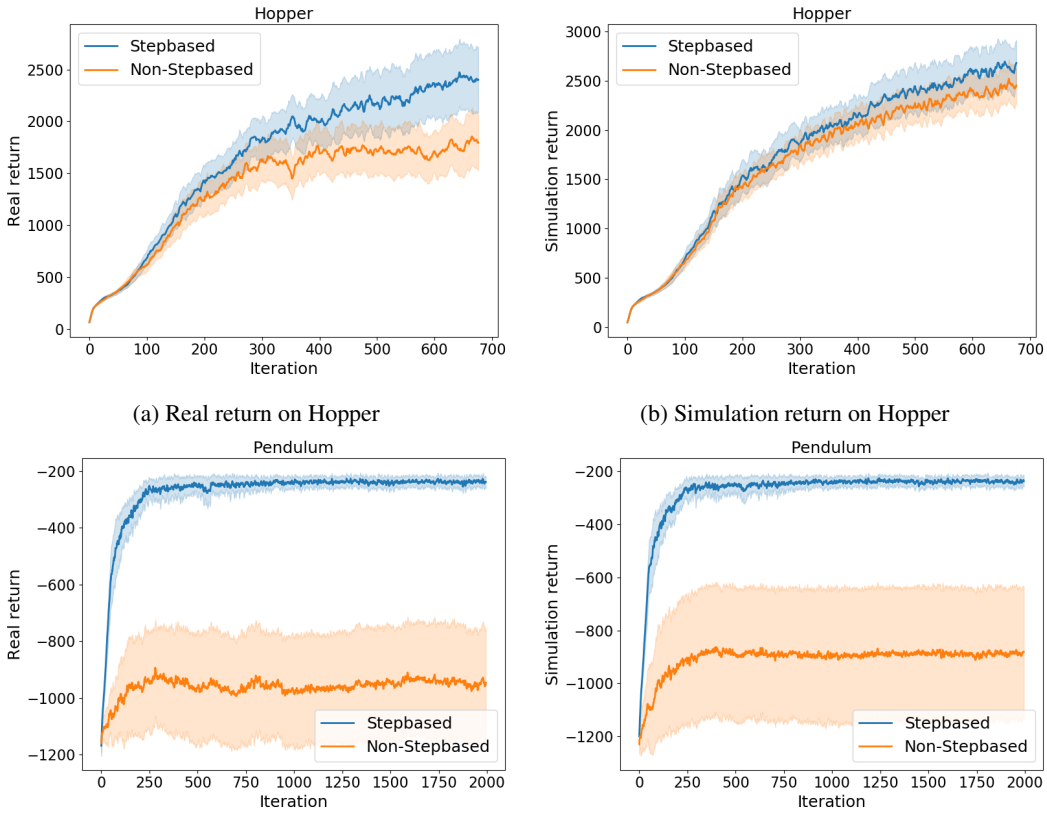

(a) Step-based SimOpt

(b) Original SimOpt

Figure 2: Average discrepancies of real and simulated states after n steps. Step-based SimOpt exhibits significantly lower discrepancies than the original SimOpt algorithm.

This confirms the intuition that the usage of transition-based optimization is more sample-efficient than trajectory-based optimization. For the Hopper environment, the algorithm finds solutions that transfer better to the real system when using the step-based variant as seen in Figure 3. This can be attributed to the step-based algorithm finding simulation parameters that explain the observations better than the original algorithm.

Additionally, the original SimOpt algorithms fails to learn useful simulator parameters under the given simulation budget in the Pendulum case, while in the Hopper case it is able to find some parameters that are not completely off. This can be attributed to the Hopper environment terminating very early when the policy has not been learned yet, resulting in a low variance trajectory of low




(a) Real return on Hopper

(b) Simulation return on Hopper

(c) Real return on Pendulum

(d) Simulation return on Pendulum

Figure 3: Comparison of SimOpt learning curves.

lengths for the episodic SimOpt algorithm, while in the Pendulum case the trajectories are always 200 steps long and induce high variance in the trajectory discrepancy.

## 6 Conclusion

In this work, we have investigated the problem of likelihood-free inference for Sim-to-Real applications. The contributions of this work have been twofold. First, We have shown connections between the episodic REPS algorithm and the ABC approach to likelihood-free inference. We found that using a discrepancy similar to discrepancies used in ABC as the REPS reward function performs an approximate version of likelihood-free variational inference.

Second, the SimOpt approach to transferring policies learned in simulations to the real world is improved by decomposing trajectories into independent transitions. This step-based SimOpt algorithm improves sample-efficiency significantly and reduces unnecessary variance in the domain randomization update steps. We have shown the effectiveness of this new approach on the Mujoco Hopper environment, where very low amounts of samples already suffice to optimize the simulation parameters.

Examining a trade-off stemming from the length of steps taken to obtain discrepancies for optimization is left open for future work as is the proper handling of multi-modal posterior approximations. For the sake of Sim-to-Real transfer, it suffices to find one good explanation of the real system, as it is not relevant for Sim-to-Real policy transfer to find all possible true parameters. Unfortunately, optimizing the expected discrepancy between states remains unsatisfactory as we have shown in a simple counterexample. Instead, one should minimize divergences between the simulated and real observed states, which is also left for future work. Finally, the reverse direction of applying ABC to Sim-to-Real transfer can also be investigated.

## References


David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

George Karabatsos, Fabrizio Leisen, et al. An approximate likelihood perspective on abc methods. *Statistics Surveys*, 12:66–104, 2018.

Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. *arXiv preprint arXiv:1707.07113*, 2017.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. *arXiv preprint arXiv:1904.04762*, 2019.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.

Quan Vuong, Sharad Vikram, Hao Su, Sicun Gao, and Henrik I Christensen. How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies? *arXiv preprint arXiv:1903.11774*, 2019.

# A Appendix

## A.1 Simulation parameters

Figure 4 shows the initial mean and diagonal values of the initial covariance matrix of the Gaussian domain randomization for the Hopper and Pendulum tasks as well as the real parameter initialization distribution for the experiments.

| | $\mu_{init}$ | diag($\Sigma_{init}$) | $p(\xi_{real})$ |
|---|---|---|---|
| **Hopper properties** | | | |
| Mass $m_1$ | 3.8 | 1.33 | $\mathcal{U}(1.0, 5.0)$ |
| Mass $m_2$ | 3.8 | 1.33 | $\mathcal{U}(1.0, 5.0)$ |
| Mass $m_3$ | 3.8 | 1.33 | $\mathcal{U}(1.0, 5.0)$ |
| Mass $m_4$ | 3.8 | 1.33 | $\mathcal{U}(1.0, 5.0)$ |
| Mass $m_5$ | 3.8 | 1.33 | $\mathcal{U}(1.0, 5.0)$ |

| | $\mu_{init}$ | diag($\Sigma_{init}$) | $p(\xi_{real})$ |
|---|---|---|---|
| **Pendulum properties** | | | |
| Gravity $g$ | 10.4 | 0.33 | $\mathcal{U}(9.0, 11.0)$ |
| Mass $m$ | 1.04 | 0.033 | $\mathcal{U}(0.9, 1.1)$ |
| Length $l$ | 1.04 | 0.033 | $\mathcal{U}(0.9, 1.1)$ |

Figure 4: Problem initialization

## A.2 SimOpt parameters

Figure 5 lists the *SimOpt* distribution update parameters for the tasks, including REPS and TRPO training parameters.

| **Simulation distribution update parameters (REPS)** | |
|---|---|
| Number of REPS updates per SimOpt iteration | 1 |
| Number of simulation parameter samples per update | 25 |
| Timesteps per simulation parameter | 200 (Pendulum)<br>up to 1000 (Hopper) |
| KL-threshold | 0.1 |
| Minimum temperature of sample weights | 0.01 |
| **Reinforcement learning training parameters (TRPO)** | |
| Timesteps per batch | 3000 |
| Hidden layer sizes | 128, 128 |
| Activation type | tanh |
| Critic l2 regularization | 1e-3 |
| Critic learning rate | 3e-4 |
| $\gamma$ | 0.995 |
| $\lambda$ | 0.97 |
| KL-threshold | 0.01 |
| Damping scale of FIM | 0.1 |

Figure 5: Algorithm parameters.