Value Iteration in Continuous Actions, States and Time

Michael Lutter [1 2] **Shie Mannor** [1 3] **Jan Peters** [2] **Dieter Fox** [1 4] **Animesh Garg** [1 5]

Abstract

Classical value iteration approaches are not applicable to environments with continuous states and actions. For such environments, the states and actions are usually discretized, which leads to an exponential increase in computational complexity. In this paper, we propose continuous fitted value iteration (cFVI). This algorithm enables dynamic programming for continuous states and actions with a known dynamics model. Leveraging the continuous-time formulation, the optimal policy can be derived for non-linear control-affine dynamics. This closed-form solution enables the efficient extension of value iteration to continuous environments. We show in non-linear control experiments that the dynamic programming solution obtains the same quantitative performance as deep reinforcement learning methods in simulation but excels when transferred to the physical system. The policy obtained by cFVI is more robust to changes in the dynamics despite using only a deterministic model and without explicitly incorporating robustness in the optimization. Videos of the physical system are available at `https://sites.google.com/view/value-iteration`.

1. Introduction

Reinforcement learning (RL) maximizes the scalar rewards r by trying different actions $\boldsymbol{u}$ selected by the policy π. For a deterministic policy and deterministic dynamics, the discrete time optimization is described by

$$\pi^*(\boldsymbol{x}) = \arg\max_{\pi} \sum_{i=0}^{\infty} \gamma^i r(\boldsymbol{x}_i, \boldsymbol{u}_i), \qquad (1)$$

with the dynamics $\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i, \boldsymbol{u}_i)$, the state $\boldsymbol{x}$ and discounting factor $\gamma \in [0, 1)$ (Bellman, 1957; Puterman, 1994). The discrete time framework has been very successful to

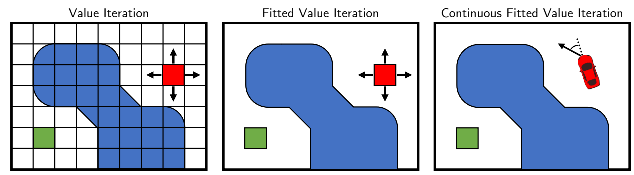

Figure 1. Value iteration (VI) can compute the optimal value function V^* and policy π^* for discrete state and action environments. Fitted value iteration extends VI to continuous states and discrete actions. Our proposed continuous fitted value iteration enables value iteration for continuous states and continuous actions, e.g., a car with steering rather than a car moving left, right, up and down.

learn policies for various applications including robotics (Silver et al., 2017; OpenAI et al., 2019; Da et al., 2020). However, the fixed pre-determined time discretization is a limitation when dealing with physical phenomena evolving in continuous time. For such environments, the time-step can be chosen to fit the environment. For example, in robotics the control frequency is only limited by the sampling frequencies of sensors and actuators, which is commonly much higher than the control frequency of deep RL agents. To avoid the fixed time steps, we use the continuous-time RL objective to frame an optimization that is agnostic to the time-discretization. Furthermore, this formulation simplifies the optimization as many robot dynamics are control affine at the continuous-time limit.

To solve the continuous-time RL problem, we use classical value iteration where the dynamics and reward function are known. For robotics, this setting is commonly a reasonable assumption. The reward is known as it is manually tuned to achieve the task without violating system constraints. The equations of motion and system parameters of robots are approximately known. The model is not perfect as the actuators are not ideal and the physical parameters vary slightly. However, the overall system behavior is approximately correct. The same assumption is also used within the vast sim2real literature to enable the sample efficient transfer to the physical world (Ramos et al., 2019; Chebotar et al., 2019; Xie et al., 2021).

In this paper we show for the continuous-time RL that

(1) classical value iteration (Bellman, 1957) can be extended to environments with continuous actions and states if the dynamics are control affine and the reward separable. Learning the value function V is sufficient as the policy can

[1]NVIDIA [2]Technical University of Darmstadt [3]Technion, Israel Institute of Technology [4]University of Washington [5]University of Toronto & Vector Institute. Correspondence to: Michael Lutter <michael@robot-learning.de>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

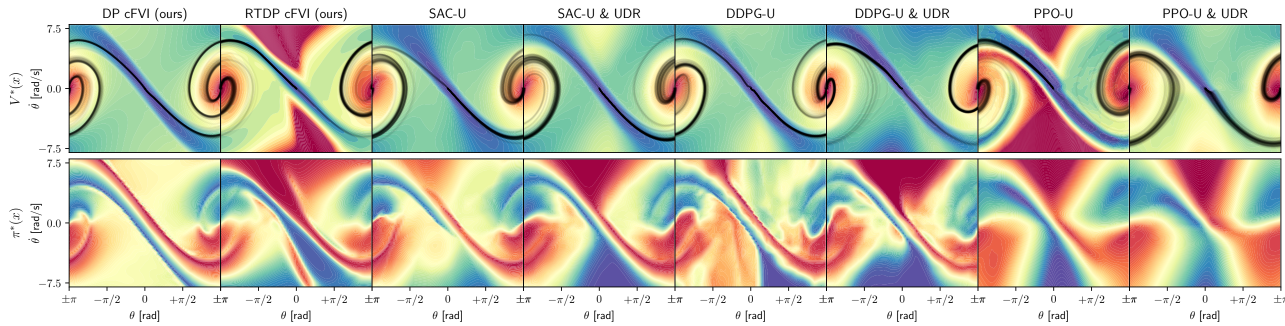

Figure 2. The optimal value function V^* and policy π^* and roll outs of the torque-limited pendulum computed by DP cFVI, RTDP cFVI and the deep RL baselines with uniform initial state distribution. cFVI learns a symmetric and smooth optimal policy that can swing-up the pendulum from both sides. Especially the DP variant has a very sharp ridge leading to the upward pointing pendulum. The baselines do not achieve a symmetric swing-up and usually prefer a one-sided swing-up.

be deduced from V in the continuous-time limit. Therefore, one does not need the policy optimization used by the predominant actor-critic approaches (Schulman et al., 2015; Lillicrap et al., 2015) or the discretization required by the classical algorithms (Bellman, 1957; Sutton et al., 1998).
(2) the proposed approach can be successfully applied to learn the optimal policy for real-world under-actuated systems using only the approximate dynamics model.
(3) We provide an in-depth quantitative and qualitative evaluation with comparisons to actor-critic algorithms. Using value iteration on the compact state domain obtains a more robust policy when transferred to the physical systems.

Summary of contributions. We extend value iteration to continuous actions as well as the extensive quantitative and qualitative evaluation on two physical systems. The latter is in contrast to prior work (Schulman et al., 2017; Haarnoja et al., 2018; Lillicrap et al., 2015; Harrison* et al., 2017; Mandlekar* et al., 2017), which mainly focuses on the quantitative comparison in simulation. Instead, we focus on the qualitative evaluation on the physical Furuta pendulum and Cartpole to understand the differences in performance.

In the following, we summarize continuous-time RL (Section 2) and introduce continuous fitted value iteration (Section 3). Section 4 describes suitable network representations for a Lyapunov value function. Finally, we analyze the performance on the physical systems in Section 5.

2. Problem Statement

We consider the deterministic, continuous-time RL problem. The infinite horizon optimization is described by

$$\pi^*(\boldsymbol{x}_0) = \arg\max_{\pi} \int_0^\infty \exp(-\rho t) \;\; r_c(\boldsymbol{x}_t, \boldsymbol{u}_t)\, dt \tag{2}$$

$$V^*(\boldsymbol{x}_0) = \max_{\boldsymbol{u}} \int_0^\infty \exp(-\rho t)\, r_c(\boldsymbol{x}_t, \boldsymbol{u}_t)\, dt \tag{3}$$

$$\text{with} \quad \boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_0^t f_c(\boldsymbol{x}_\tau, \boldsymbol{u}_\tau)\, d\tau \tag{4}$$

with the discounting constant $\rho \in (0, \infty]$, the reward r_c and the dynamics f_c (Kirk, 1970). Notably, the discrete reward and discounting can be described using the continuous-time counterparts, i.e., $r(\boldsymbol{x}, \boldsymbol{u}) = \Delta t\ r_c(\boldsymbol{x}, \boldsymbol{u})$ and $\gamma = \exp(-\rho\, \Delta t)$ with the sampling interval Δt. The continuous-time discounting ρ is, in contrast to the discrete discounting factor γ, agnostic to sampling frequencies. In the continuous-time case, the Q-function does not exist (Doya, 2000).

The deterministic continuous-time dynamics model f_c is assumed to be non-linear w.r.t. the system state $\boldsymbol{x}$ but affine w.r.t. the action $\boldsymbol{u}$. Such dynamics model is described by

$$\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}; \theta) + \boldsymbol{B}(\boldsymbol{x}; \theta)\boldsymbol{u}, \tag{5}$$

with the non-linear drift $\boldsymbol{a}$, the non-linear control matrix $\boldsymbol{B}$ and the system parameters θ. Robot dynamics models are naturally expressed in the continuous-time formulation and many are control affine. Furthermore, this special case has received ample attention in the existing literature due to its wide applicability (Doya, 2000; Kappen, 2005; Todorov, 2007). The reward is separable into a non-linear state reward q_c and the action cost g_c described by

$$r_c(\boldsymbol{x}, \boldsymbol{u}) = q_c(\boldsymbol{x}) - g_c(\boldsymbol{u}). \tag{6}$$

The action penalty g_c is non-linear, positive definite and strictly convex. This separability is common for robot control problems as rewards are composed of a state component quantifying the distance to a desired state and an action penalty. The action cost penalizes non-zero actions to avoid bang-bang control from being optimal and is convex to have an unique optimal action.

3. Continuous Fitted Value Iteration

First, we summarize the existing value iteration approaches and present the theory enabling us to extend value iteration to continuous action spaces. Afterwards, we introduce our proposed algorithm to learn the value function.

3.1. Value Iteration Preliminaries

Value iteration (VI) is an approach to compute the optimal value function for a discrete time, state and action MDP with known dynamics and reward (Bellman, 1957). This approach iteratively updates the value function of each state using the Bellman optimality principle described by

$$V^{k+1}(\boldsymbol{x}_t) = \max_{\boldsymbol{u}_{0\dots\ell}} \sum_{i=0}^{\ell-1} \gamma^i r(\boldsymbol{x}_{t+i}, \boldsymbol{u}_i) + \gamma^\ell V^k(\boldsymbol{x}_{t+\ell}),$$

with the number of look-ahead steps ℓ. VI is proven to converge to the optimal value function (Puterman, 1994) as the update is a contraction described by

$$\|V_i^{k+1} - V_j^{k+1}\| \leq \gamma^\ell \|V_i - V_j\|. \tag{7}$$

For discrete actions, the greedy action is determined by evaluating all possible actions. However, VI is impractical for larger MDPs as the computational complexity grows exponentially with the number of states and actions. This is especially problematic for continuous spaces as discretization leads to an exponential increase of state-action tuples.

VI can be applied to continuous state spaces and **discrete** actions by using a function approximator instead of tabular value function. Fitted value iteration (FVI) computes the value function target using the VI update and minimizes the ℓ_p-norm between the target and the approximation $V^k(\boldsymbol{x}; \psi)$. This approach is described by

$$V_{\text{tar}}(\boldsymbol{x}_t) = \max_{\boldsymbol{u}} \; r(\boldsymbol{x}_t, \boldsymbol{u}) + \gamma V^k(f(\boldsymbol{x}_t, \boldsymbol{u}); \; \psi_k) \tag{8}$$

$$\psi_{k+1} = \arg\min_{\psi} \sum_{\boldsymbol{x} \in \mathcal{D}} \|V_{\text{tar}}(\boldsymbol{x}) - V^k(\boldsymbol{x}; \; \psi)\|_p^p \tag{9}$$

with the parameters ψ_k at iteration k and the fixed dataset $\mathcal{D}$. The convergence proof of VI does not generalize to FVI as the fitting of the value function is not necessarily a contraction (Boyan & Moore, 1994; Baird, 1995; Tsitsiklis & Van Roy, 1996; Munos & Szepesvári, 2008). Despite these theoretical limitations, many authors proposed model-free variants using the Q-function for discrete action MDPs, e.g., fitted Q-iteration (FQI) (Ernst et al., 2005), Regularized FQI (Farahmand et al., 2009), Neural FQI (Riedmiller, 2005) and DQN (Mnih et al., 2015). The resulting algorithms were empirically successful and solved Backgammon (Tesauro, 1992) and the Atari games (Mnih et al., 2015).

For continuous actions, current RL methods use policy iteration (PI) rather than VI (Schulman et al., 2015; Lillicrap et al., 2015). PI evaluates the value function of the current policy and hence, uses the action of the policy to compute the value function target. Therefore, PI circumvents the maximization required of VI (Equation 8). To update the policy, PI uses an additional optimization to improve the policy via policy gradients. Therefore, PI requires an additional optimization compared to VI approaches.

Algorithm 1 Continuous Fitted Value Iteration (cFVI)

Input: Dynamics Model $f_c(\boldsymbol{x}, \boldsymbol{u})$ & Dataset $\mathcal{D}$
Result: Value Function $V^*(\boldsymbol{x}; \; \psi^*)$

for k = 0 . . . N **do**
  // Compute Value Target for $\boldsymbol{x} \in \mathcal{D}$:
  $V_{\text{tar}}(\boldsymbol{x}_i) = \int_0^T \beta \; \exp(-\beta t) \; R_t \; dt + \exp(-\beta T) R_T$
  $R_t = \int_0^t \exp(-\rho\tau) \; r_c(\boldsymbol{x}_\tau, \boldsymbol{u}_\tau) d\tau + \exp(-\rho t) V^k(\boldsymbol{x}_t)$
  $\boldsymbol{x}_\tau = \boldsymbol{x}_i + \int_0^\tau f_c(\boldsymbol{x}_t, \boldsymbol{u}_t) dt$
  $\boldsymbol{u}_\tau = \nabla \tilde{g} \left(\boldsymbol{B}(\boldsymbol{x}_\tau) \nabla_x V^k(\boldsymbol{x}_\tau) \right))$

  // Fit Value Function:
  $\psi_{k+1} = \arg\min_\psi \sum_{\boldsymbol{x} \in \mathcal{D}} \|V_{\text{tar}}(\boldsymbol{x}) - V(\boldsymbol{x}; \psi)\|^p$

  if RTDP cFVI **then**
    // Add samples from π^{k+1} to FIFO buffer $\mathcal{D}$
    $\mathcal{D}^{k+1} = h(\mathcal{D}^k, \{\boldsymbol{x}_0^{k+1} \; \dots \; \boldsymbol{x}_N^{k+1}\})$
  end if
end for

3.2. Deriving the Analytic Optimal Policy

One cannot directly apply FVI to continuous actions due to the maximization in Equation 8. To compute the value function target, one would need to solve an optimization in each iteration. To extend value iteration to continuous actions, we show that one can solve this maximization analytically for the considered continuous-time problem. This solution enables the efficient computation of the value function target and extends FVI to continuous actions.

Theorem 1. *If the dynamics are control affine (Equation 5), the reward is separable w.r.t. to state and action (Equation 6) and the action cost g_c is positive definite and strictly convex, the continuous-time optimal policy π^k is described by*

$$\pi^k(\boldsymbol{x}) = \nabla \tilde{g}_c \left(\boldsymbol{B}(\boldsymbol{x})^T \nabla_x V^k \right) \tag{10}$$

where $\tilde{g}$ is the convex conjugate of g and $\nabla_x V^k$ is the Jacobian of current value function V^k w.r.t. the system state.

Proof Sketch. The detailed proof is provided in the appendix. This derivation follows the work of Lutter et. al. (2019), which generalized the special cases initially described by Lyshevski (1998) and Doya (2000) to a wider class of reward functions. The value iteration target (Equation 8) can be expressed by using the Taylor series for the value function $V(\boldsymbol{x}_{t+1})$ and substituting Equation 5 and Equation 6. This reformulated value target is described by

$$V_{\text{tar}} = \max_{\boldsymbol{u}} \; \left[\gamma \nabla_x V^T \; (\boldsymbol{a} + \boldsymbol{B}\boldsymbol{u}) + \gamma \, O(.) + q_c - g_c \right] \Delta t + \gamma V$$

with the higher order terms $O(\Delta t, \boldsymbol{x}, \boldsymbol{u})$. In the continuous-time limit the higher order terms disappear and $\gamma = 1$.

Therefore, the optimal action is described by

$$\boldsymbol{u}_t^* = \arg\max_{\boldsymbol{u}} \nabla_x V^T \boldsymbol{B}(\boldsymbol{x}_t)\, \boldsymbol{u} - g_c(\boldsymbol{u}). \tag{11}$$

Equation 11 can be solved analytically as g_c is strictly convex and hence $\nabla g_c(\boldsymbol{u}) = \boldsymbol{w}$ is invertible, i.e., $\boldsymbol{u} = [\nabla g_c]^{-1}(\boldsymbol{w}) \coloneqq \nabla \tilde{g}_c(\boldsymbol{w})$ with the convex conjugate $\tilde{g}$. The solution of Equation 11 is described by

$$\boldsymbol{B}^T \nabla_x V^* - \nabla g_c(\boldsymbol{u}) \coloneqq 0 \;\; \Rightarrow \;\; \boldsymbol{u}^* = \nabla \tilde{g}_c \left(\boldsymbol{B}^T \nabla_x V^k \right).$$

□

Unrolling the closed form policy over time performs hill-climbing on the value function, where the step-size corresponds to the time-discretization. The inner part $\boldsymbol{B}(\boldsymbol{x})^T \nabla_x V^k$ determines the direction of steepest ascent and the action cost rescales this direction. For example, a zero action cost for all admissible actions makes the optimal policy take the largest possible actions, i.e., the policy is bang-bang controller. A quadratic action cost linearly rescales the gradient. A barrier shaped cost clips the gradients. A complete guide on designing the action cost to shape the optimal policy can be found in Lutter et. al. (2019). The step-size, which corresponds to the control frequency of the discrete time controller, determines the convergence to the value function maximum. If the step size is too large the system becomes unstable and does not converge to the maximum. For most real world robotic systems with natural frequencies below 5Hz and control frequencies above 100Hz, the resulting step-size is sufficiently small to achieve convergence to the value function maximum. Therefore, π^* can be used for discrete time controllers. Furthermore, the continuous-time policy can be used for intermittent control (event-based control) where interacting with the system occurs at irregular time-steps and each interaction is associated with a cost (Aström, 2008).

3.3. Learning the Optimal Value Function

Using the closed-form policy that analytically solves the maximization, one can derive a computationally efficient value iteration update for continuous states and actions. Substituting $\boldsymbol{u}^*$ (Equation 10) into the value iteration target computation (Equation 8) yields

$$V_{\text{tar}}(\boldsymbol{x}_t) = r\left(\boldsymbol{x}_t, \nabla \tilde{g}\left(\boldsymbol{B}(\boldsymbol{x}_t)^T \nabla_x V^k\right)\right) + \gamma V^k(\boldsymbol{x}_{t+1};\, \psi_k)$$
$$\text{with} \quad \boldsymbol{x}_{t+1} = f\left(\boldsymbol{x}_t, \nabla \tilde{g}(\boldsymbol{B}(\boldsymbol{x}_t)^T \nabla_x V^k)\right).$$

Combined with fitting the value function to the target value (Equation 9), these two steps constitute the value function update of cFVI. Repeatedly computing the value target and fitting the value function leads to the optimal value function and policy.

For the continuous-time limit, the computation of the *naive* value function target should be adapted as the convergence decreases with decreasing time steps. As Δt decreases, the γ increases, i.e., $\gamma = \lim_{\Delta t \to 0} \exp(-\rho \Delta t) = 1$. Therefore, the contraction coefficient of VI decreases exponentially with increasing sampling frequencies. This slower convergence is intuitive as higher sample frequencies effectively increase the number of steps to reach the goal.

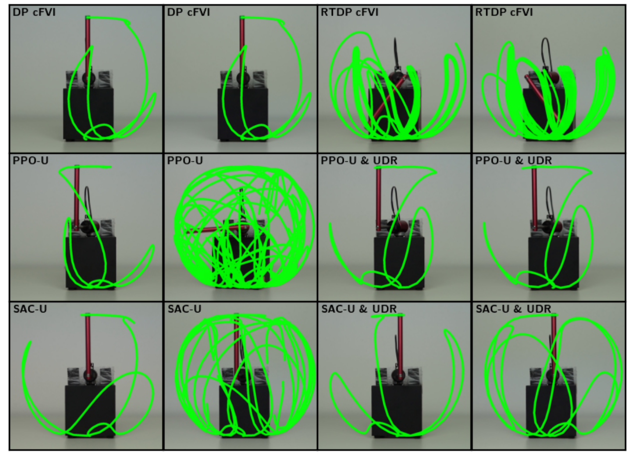

Figure 3. Tracked swing-up of the Furuta pendulum for best and worst roll out. DP cFVI can consistently swing-up the pendulum with minimal variance between roll outs. DP cFVI achieves much better performance compared to most baselines. Only PPO with domain randomization achieves comparable results. The remaining baselines are shown in the appendix.

N-Step Value Function Target To increase the computational efficiency cFVI, the exponentially weighted n-step value function target

$$V_{\text{tar}}(\boldsymbol{x}) = \int_0^T \beta\, \exp(-\beta t)\, R_t\, dt + \exp(-\beta T) R_T,$$
$$R_t = \int_0^t \exp(-\rho\tau)\, r_c(\boldsymbol{x}_\tau, \boldsymbol{u}_\tau) d\tau + \exp(-\rho t) V^k(\boldsymbol{x}_t),$$

and the exponential decay constant β, can be used. The integrals can be solved using any ordinary differential equation solver with fixed or adaptive step-size. We use the explicit Euler integrator with fixed steps to solve the integral for all samples in parallel using batched operations on the GPU. The nested integrals can be computed efficiently by recursively splitting the integral and reusing the estimate of the previous step. In practice we treat β as a hyperparameter and select T such that the weight of the R_T is $\exp(-\beta T) \coloneqq 10^{-4}$.

The convergence rate improves as this target computation corresponds to the multi-step value target. The n-step target increases the contraction rate as this rate is proportional to γ^ℓ. This approach is the continuous-time counterpart of the discrete eligibility trace of TD(λ) with $\lambda = \exp(-\beta \Delta t)$ (Sutton et al., 1998). With respect to deep RL this discounted n-step value target is similar to the generalized advantage estimation (GAE) of PPO (Schulman et al., 2015; 2017) and model-based value expansion (MVE) (Feinberg et al., 2018;

Buckman et al., 2018). GAE and MVE have shown that the n-step target increases the sample efficiency and lead to faster convergence to the optimal policy.

Offline and Online cFVI The proposed approach is off-policy as the samples in the replay memory do not need to originate from the current policy π_k. Therefore, the dataset can either consist of a fixed dataset (i.e., batch or offline RL) or be updated within each iteration. In the offline dynamic programming case, the dataset contains samples from the compact state domain $\mathcal{X}$. We refer to the offline variant as DP cFVI. In the online case, the replay memory is updated with samples generated by the current policy π_k. Every iteration the states of the previous n-rollouts are added to the data and replace the oldest samples. This online update of state distribution performs real-time dynamic programming (RTDP) (Barto et al., 1995). We refer to the online variant as RTDP cFVI. The pseudo code of DP cFVI and RTDP cFVI is summarized in Algorithm 1.

4. Value Function Hypothesis Space

The previous sections focused on learning the optimal value function independent of the value function representation. In this section, we focus on the value function representation.

Value Function Representation Most recent approaches use a fully connected network to approximate the value function. However, for the many tasks the hypothesis space of the value function can be narrowed. For control tasks, the state cost is often a negative distance measure between $\boldsymbol{x}_t$ and the desired state $\boldsymbol{x}_{\text{des}}$. Hence, q_c is negative definite, i.e., $q(\boldsymbol{x}) < 0 \;\forall\; \boldsymbol{x} \neq \boldsymbol{x}_{\text{des}}$ and $q(\boldsymbol{x}_{\text{des}}) = 0$. These properties imply that V^* is a negative Lyapunov function, as V^* is negative definite, $V^*(\boldsymbol{x}_{\text{des}}) = 0$ and $\nabla_x V^*(\boldsymbol{x}_{\text{des}}) = \mathbf{0}$ (Khalil & Grizzle, 2002). With a deep network a similar representation can be achieved by

$$V(\boldsymbol{x};\, \psi) = -\left(\boldsymbol{x} - \boldsymbol{x}_{\text{des}}\right)^T \boldsymbol{L}(\boldsymbol{x};\, \psi)\boldsymbol{L}(\boldsymbol{x};\, \psi)^T \left(\boldsymbol{x} - \boldsymbol{x}_{\text{des}}\right)$$

with $\boldsymbol{L}$ being a lower triangular matrix with positive diagonal. This positive diagonal ensures that $\boldsymbol{L}\boldsymbol{L}^T$ is positive definite. Simply applying a ReLu activation to the last layer of a deep network is not sufficient as this would also zero the actions for the positive values and $\nabla_x V^*(\boldsymbol{x}_{\text{des}}) = \mathbf{0}$ cannot be guaranteed. The local quadratic representation guarantees that the gradient and hence, the action, is zero at the desired state. However, this representation can also not guarantee that the value function has only a single extrema at $\boldsymbol{x}_{\text{des}}$ as required by the Lyapunov theory. In practice, the local regularization of the quadratic structure to avoid high curvature approximations is sufficient as the global structure is defined by the value function target. $\boldsymbol{L}$ is the mean of a deep network ensemble with N independent parameters ψ_i. The ensemble mean smoothes the initial value function and is differentiable. Similar representations have been used by prior works in the safe reinforcement learning community (Berkenkamp et al., 2017; Richards et al., 2018; Kolter & Manek, 2019; Chang et al., 2019; Bharadhwaj et al., 2021).

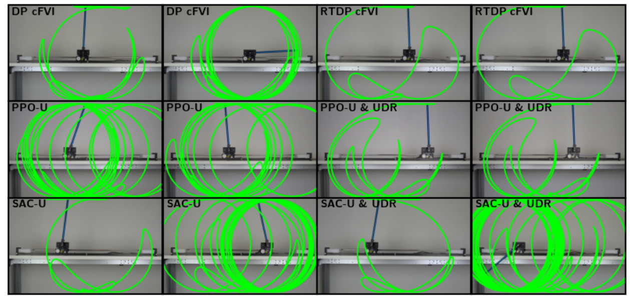

Figure 4. Tracked swing-up of the cartpole for the best and worst roll out. DP cFVI and RTDP cFVI can achieve the task. In the failure case, cFVI excels as the policy remains in the center. In contrast the deep RL baselines move the cart between the joint limits in case of failure. All baselines are shown in the appendix.

Gradient Projection of State Transformations State transformations should be explicitly incorporated in the value function and should not be contained in the environment, as for example in the openAI Gym (Brockman et al., 2016). If the transformation is implicit, $\nabla_x V$ might not be sensible. For example, the standard feature transform for a continuous revolute joint maps the joint state $\boldsymbol{x} = [\theta,\, \dot{\theta}]$ to $\boldsymbol{z} = [\sin(\theta),\, \cos(\theta),\, \dot{\theta}]$. In this case the transformation $h(\boldsymbol{x})$ must be included in the value function as the transformed state $\boldsymbol{z}$ is on the tube shaped manifold . Hence, the naive gradient of V might not be in the manifold tangent space.

If the state transform is explicitly incorporated in the value function, this problem does not occur. The transformation can be included explicitly by $V(x;\, \psi) = f(h(\boldsymbol{x});\, \psi)$ and $\nabla_x V(x;\, \psi) = \partial f(h(\boldsymbol{x});\, \psi)/\partial h \; \partial h(\boldsymbol{x})/\partial \boldsymbol{x}$. In this case, the gradient of the transformed state is projected to the tangent space of the feature transform. Therefore, the value function gradient points in a plausible direction.

5. Experiments

In the following the experimental setup and results are described. The exact experiment specification, all qualitative plots and an additional ablation study on model ensembles is provided in the appendix.

5.1. Experimental Setup

Systems The algorithms are compared using the standard non-linear control benchmark, the *swing-up* of underactuated systems. Specifically, we apply the algorithms to the torque-limited pendulum, cartpole (Figure 4) and Furuta pendulum (Figure 3). For the physical systems the dynamics model of the manufacturer is used (Quanser, 2018). The control frequency is optimized for each algorithm. The task is considered solved, if the pendulum angle α is below $\pm 5°$ degree for the last second.

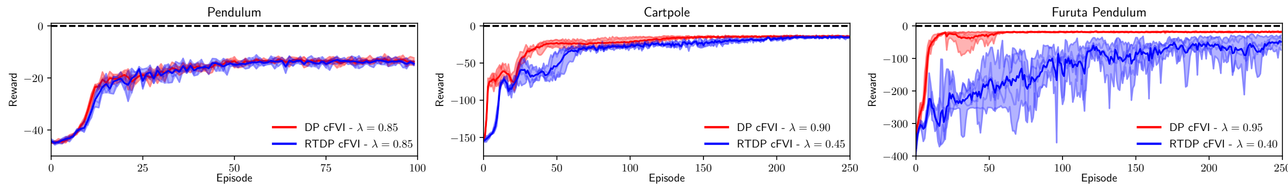

Figure 5. The learning curves for DP cFVI and RTDP cFVI averaged over 5 seeds. The shaded area displays the *min/max* range between seeds. DP cFVI achieves consistent learning of the optimal policy with very low variance between seeds. RTDP cFVI has a higher variation and learns slower compared to DP cFVI. RTDP cFVI needs to discover the steady-state state distribution first. Especially for the Furuta pendulum the range increases between seeds as minor changes in the policy lead to large changes of state-distribution.

Table 1. Average rewards on the simulated and physical systems. The ranking describes the decrease in reward compared to the best result averaged on all systems. The initial state distribution during training is noted by μ. The dynamics are either deterministic model $\theta \sim \delta(\theta)$ or sampled using uniform domain randomization $\theta \sim \mathcal{U}(\theta)$. During evaluation the roll outs start with the pendulum pointing downwards. DP cFVI obtains high rewards on all systems and is the highest ranking algorithm compared to the baselines.

Algorithm	μ	θ	**Simulated Pendulum** Success [%]	**Simulated Pendulum** Reward [$\mu \pm 1.96\sigma$]	**Simulated Cartpole** Success [%]	**Simulated Cartpole** Reward [$\mu \pm 1.96\sigma$]	**Simulated Furuta Pendulum** Success [%]	**Simulated Furuta Pendulum** Reward [$\mu \pm 1.96\sigma$]	**Physical Cartpole** Success [%]	**Physical Cartpole** Reward [$\mu \pm 1.96\sigma$]	**Physical Furuta Pendulum** Success [%]	**Physical Furuta Pendulum** Reward [$\mu \pm 1.96\sigma$]	**Average Ranking** [%]
DP cFVI (ours)	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	**−030.5 ± 000.8**	100.0	**−024.2 ± 002.1**	100.0	**−027.7 ± 001.6**	73.3	−143.7 ± 210.4	100.0	**−082.1 ± 007.6**	**−008.8**
RTDP cFVI (ours)	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	**−031.1 ± 001.4**	100.0	**−024.9 ± 001.6**	100.0	−040.1 ± 002.7	100.0	**−101.1 ± 029.0**	00.0	−1009.9 ± 004.5	−240.4
SAC	$\mathcal{N}$	$\mathcal{U}(\theta)$	100.0	**−031.1 ± 000.1**	100.0	−026.9 ± 003.2	100.0	−029.3 ± 001.5	00.0	−518.6 ± 028.1	86.7	−330.7 ± 799.0	−148.4
SAC & UDR	$\mathcal{N}$	$\delta(\theta))$	100.0	−032.9 ± 000.6	100.0	−029.7 ± 004.6	100.0	−032.0 ± 001.1	100.0	−394.8 ± 382.8	100.0	−181.4 ± 157.9	−092.3
SAC	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	**−030.6 ± 001.4**	100.0	**−024.2 ± 001.4**	100.0	**−028.1 ± 002.0**	53.3	−144.5 ± 204.0	100.0	−350.8 ± 433.3	−076.1
SAC & UDR	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	−031.4 ± 002.5	100.0	**−024.2 ± 001.3**	100.0	**−028.1 ± 001.3**	40.0	−296.4 ± 418.9	100.0	−092.3 ± 064.1	−042.4
DDPG	$\mathcal{N}$	$\mathcal{U}(\theta)$	100.0	**−031.1 ± 000.4**	98.0	−050.4 ± 285.6	100.0	−030.5 ± 003.5	06.7	−536.7 ± 262.7	46.7	−614.1 ± 597.8	−242.6
DDPG & UDR	$\mathcal{N}$	$\delta(\theta))$	100.0	−032.5 ± 000.5	100.0	−027.4 ± 002.3	100.0	−034.6 ± 009.8	00.0	−517.9 ± 117.6	86.7	−192.7 ± 404.8	−119.3
DDPG	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	−031.5 ± 000.7	100.0	−028.2 ± 005.5	100.0	−030.0 ± 001.7	06.7	−459.4 ± 248.3	100.0	−146.6 ± 218.3	−092.9
DDPG & UDR	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	−032.5 ± 003.6	100.0	−027.2 ± 001.0	100.0	−032.1 ± 001.5	00.0	−318.1 ± 063.4	100.0	−156.7 ± 246.4	−068.8
PPO	$\mathcal{N}$	$\mathcal{U}(\theta)$	100.0	−032.0 ± 000.2	100.0	−031.5 ± 007.2	100.0	−081.1 ± 018.3	00.0	−287.9 ± 068.8	33.3	−718.7 ± 456.1	−240.9
PPO & UDR	$\mathcal{N}$	$\delta(\theta))$	100.0	−032.3 ± 000.6	100.0	−084.0 ± 007.8	100.0	−040.9 ± 004.6	00.0	−435.4 ± 111.9	46.7	−935.7 ± 711.6	−338.6
PPO	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	−033.4 ± 004.7	99.0	−039.7 ± 045.7	100.0	−038.2 ± 013.1	00.0	−183.8 ± 018.0	60.0	−755.3 ± 811.0	−206.1
PPO & UDR	$\mathcal{U}$	$\mathcal{U}(\theta)$	100.0	−035.6 ± 003.1	100.0	−044.8 ± 021.4	100.0	−048.5 ± 006.2	40.0	−143.8 ± 016.1	100.0	**−080.6 ± 010.8**	−044.0

Baselines The performance is compared to the actor-critic deep RL methods: DDPG (Lillicrap et al., 2015), SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017). We compare two different initial state distributions μ. For $\mu = \mathcal{U}$, the initial pendulum angle α_0 is sampled uniformly $\alpha_0 \sim \mathcal{U}(-\pi, +\pi)$. For $\mu = \mathcal{N}$, the initial angle is sampled from a Gaussian distribution with the pendulum facing downwards $\alpha_0 \sim \mathcal{N}(\pm\pi, \sigma)$. The uniform sampling avoids the exploration problem and generates a larger state distribution of the optimal policy. In addition we augment each baseline with uniform domain randomization (Muratore et al., 2018).

5.2. Simulation Results

The average rewards of our method - Continuous Fitted Value Iteration (cFVI) and the baselines are summarized in Table 1. The learning curves are shown in Figure 5. In simulation, DP cFVI, the offline version of cFVI with fixed dataset, achieves the best rewards for all three systems and marginally outperforms SAC in terms of average reward. It's important to point out that identical mean rewards do not imply a similar qualitative performance. For example, DP cFVI and SAC-U achieve identical mean reward on the cartpole despite having very different closed-loop dynamics (Figure 11 - Appendix). Notably, RTDP cFVI, which uses the state distribution of the current policy rather than a fixed dataset, solves the tasks for all three systems. For the pendulum and the cartpole, the reward is comparable to the best performing algorithms. While for the Furuta pendulum, the reward is lower (higher) than the best (worst) performing Deep RL baselines.

The learning curves highlight that the variance between seeds is very low for DP cFVI. It is important to note that Figure 5 shows the min-max range of the different seeds rather than the confidence intervals or the standard error. Therefore, the exact weight initialization and sampling of the fixed dataset, which is different for each seed, do not have a large impact on the learning progress or obtained solution. For RTDP cFVI the variance between seed increases and the learning speed decreases compared to DP cFVI. The pendulum is an outlier for RTDP cFVI, as the state-distribution is nearly independent of the current policy if the pendulum angle is initialized uniformly. Especially for the Furuta pendulum the variance increases and learning speed decreases. For this system slightly different policies can cause vastly different state distributions due to the low condition number of the mass-matrix. Therefore, RTDP cFVI takes longer to reach a stationary distribution and increase the obtained reward.

In general it is important to point out that the exploration for RTDP cFVI is more challenging compared to other deep RL methods as RTDP cFVI uses a higher control frequency. Due to the shorter steps simple Gaussian noise averages out and does not lead to a large exploration in state space. Therefore, the performance and variance of RTDP

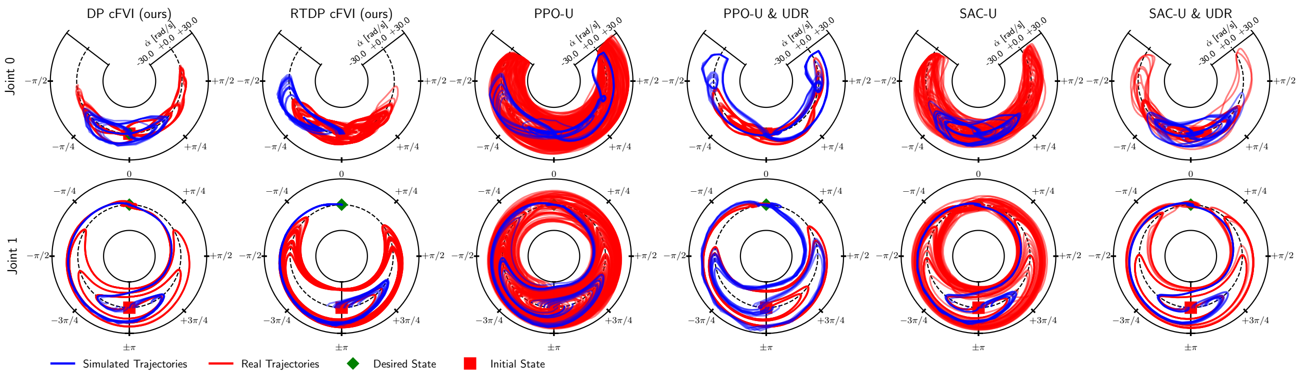

Figure 6. The simulated (blue) and real world (red) roll outs for the Furuta pendulum controlled by cFVI and the baselines. The deviation from the dashed line denotes the joint velocity. The distribution shift is clearly visible as the blue and red trajectories do not overlap, e.g., cFVI requires more pre-swings on the physical system. For PPO-U the distribution shift causes random exploration of the complete state-domain. cFVI achieves the best qualitative performance as it can swing-up the pendulum from both directions and achieves nearly identical roll-outs. In contrast the deep RL baselines have higher distribution shift due to the dynamics mismatch. The trajectories are also less consistent and vary significantly between roll outs. The plots for all baselines and the physical cartpole are shown in the appendix.

cFVI could be improved using a more coherent exploration strategy in future work.

5.3. Ablation Study - N-step Value Targets

The learning curves, averaged over 5 seeds, for different n-step value targets are shown in Figure 7. When increasing λ, which implicitly increases the n-step horizon (section 3.3), the convergence rate to the optimal value function increases. This increased learning speed is expected as Equation 7 shows that the convergence rate depends on γ^n with $\gamma < 1$. While the learning speed measured in iterations increases with λ, the computational complexity also increases. Longer horizons require to simulate n sequential steps increasing the required wall-clock time. Therefore, the computation time increases exponentially with increasing λ. For example the forward roll out in every iteration of the pendulum increases exponentially from 0.4s ($\lambda = 0.01$) to 56.4s ($\lambda = 0.99$)[1]. For the Furuta pendulum and the cartpole extremely long horizons of 100+ steps start to diverge as the value function target over fits to the untrained value function. For RTDP cFVI, the horizons must smaller, i.e., 10 - 20 steps. For longer horizons the predicted rollout overfits to the value function outside of the current state distribution, which prevents learning or leads to pre-mature convergence (see Appendix Figure 9). Therefore, DP cFVI works bests with $\lambda \in [0.85, 0.95]$ and RTDP cFVI with $\lambda \in [0.45 - 0.55]$.

5.4. Physical Experiments

The quantitative results of the physical experiments are summarized in Table 1[2], with additional plots and baselines in the appendix. DP cFVI achieves high reward on both physical systems and outperforms the baselines. Only PPO-U UDR achieves comparable performance on both physical systems. On average the performance of the deep RL variants increases with with larger initial state distribution.

The trajectories of the Furuta Pendulum are shown in Figure 6. The distribution shift between the physical and simulated trajectories is clearly visible as the trajectories largely deviate. DP cFVI achieves a highly repeatable performance on the noisy physical system. All 15 roll-outs have a similar trajectory. The baselines require domain randomization and uniform initial distribution to achieve a reward comparable to DP cFVI. All other deep RL baselines have a high variance between roll-outs. For example the simulation gap causes PPO-U to randomly explore the complete state space. RTDP cFVI does not achieve the swing-up but fails gracefully as it stabilizes the pendulum on a stable limit cycle and does not hit the joint limits. Furthermore, this limit cycle is highly repetitive and the variance in reward is low. For many robotic tasks, it is preferable to fail gracefully rather than hitting the limits and solving the task.

On the physical cartpole, DP cFVI obtains a higher reward than RTDP cFVI, if the swing-up is successful. However, RTDP cFVI has a higher average reward across all roll-outs as DP cFVI only achieves a 73% success rate. If DP cFVI fails, the resulting behavior is a deterministic limit-cycle where the cart is centered. The policy decelerates the pendulum, but due to the backslash in the linear actuator the applied force is not sufficient. Most deep RL baselines do not achieve a consistent swing-up. Only PPO-U UDR, SAC-U achieve a repeatable high reward. SAC-N UDR achieves the swing-up but always hits the joint limit and obtains a relative low reward. The failure cases of all deep RL baselines are stochastic and repeatedly hit the joint limit. If the pendulum over-swings the cart starts to oscillate between both joint limits. Notably, domain randomization does not improve the performance on the physical cartpole.

[1] Wall-clock time on an AMD 3900X and a NVIDIA RTX 3090

[2] Task Video available at: `https://sites.google.com/view/value-iteration`

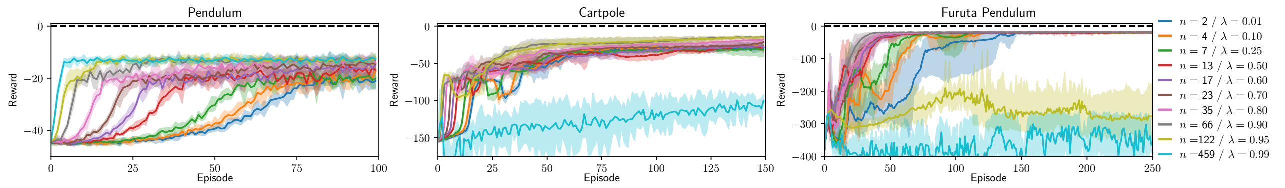

Figure 7. The learning curves averaged over 5 seeds for the n-step value function target. The shaded area displays the *min/max* range between seeds. The step count is selected such that $\lambda^n := 10^{-4}$. Increasing the horizon of the value function target increases the convergence rate to the optimal value function. For very long horizons the learning diverges as it over fits to the current value function approximation. Furthermore, the performance of the optimal policy also increases with roll out length.

For this system, the simulation gap originates from backslash and friction in the actuator rather than the uncertainty of the model parameters. As the actuation is not simulated, these parameters cannot be randomized. For the cartpole, the larger initial state distribution is required to achieve high rewards on the physical system. All baselines with a gaussian initial state distribution achieve only a low reward. See appendix for qualitative simulated and physical results.

6. Related Work

Continuous-time RL started with the seminal work of Doya (2000). Since then, various approaches have been proposed to solve the Hamilton-Jacobi-Bellman (HJB) differential equation with the machine learning toolset. These methods can be divided into trajectory and state-space based methods.

Trajectory based methods solve the stochastic HJB along a trajectory to obtain the optimal trajectory. For example path integral control uses the non-linear, control-affine dynamics with quadratic action costs to simplify the HJB to a linear partial differential equation (Kappen, 2005; Todorov, 2007; Theodorou et al., 2010; Pan et al., 2014). This differential equation can be transformed to a path-integral using the Feynman-Kac formulae. The path-integral can then be solved using Monte Carlo sampling to obtain the optimal state and action sequence. Recently, this approach has been used in combination with deep networks (Rajagopal et al., 2016; Pereira et al., 2019; 2020).

State-space based methods solve the HJB globally to obtain a optimal non-linear controller applicable on the complete state domain. Classical approaches discretize the continuous spaces into a grid and solve the HJB or the robust Hamilton-Jacobi-Isaac (HJI) using a PDE solver (Bansal et al., 2017). To overcome the curse of dimensionality of the grid based methods, machine learning methods that use function approximation and sampling have been proposed. For example, regression based approaches solved the HJB by fitting a radial-basis-function networks (Doya, 2000), deep networks (Tassa & Erez, 2007; Lutter et al., 2019; Kim et al., 2020), kernels (Hennig, 2011) or polynomial functions (Yang et al., 2014; Liu et al., 2014) to minimize the HJB residual. The naive optimization of this objective, while successful for different PDEs (Raissi et al., 2017a;b; 2020), does not work for the HJB as the exact location of the boundary condition is unknown. Therefore, authors used various optimization tricks such as annealing the noise (Tassa & Erez, 2007) or the discounting (Lutter et al., 2019) to obtain the optimal value function.

In this work we propose a DP based algorithm. Instead of trying to solve the HJB PDE directly via regression as previous learning-based methods, we leverage the continuous-time domain and solve the HJB by iteratively applying the Bellman optimality principle. The resulting algorithm has a simpler and more robust optimization compared to the previous approaches using direct regression.

7. Conclusion and Future Work

We proposed continuous fitted value iteration (cFVI). This algorithm enables dynamic programming with known model for problems with continuous states and actions without discretization. Therefore, cFVI avoids the curse of dimensionality associated with discretization and the policy optimization of actor-critic approaches. Exploiting the insights from the continuous-time formulation, the optimal action can be computed analytically. This closed-form solution permits the efficient computation of the value function target and enables the extension of value iteration. The non-linear control experiments showed that value iteration on the compact state space has the same quantitative performance as deep RL methods in simulation. On the sim2real tasks, DP cFVI excels compared deep RL algorithms that include domain randomization and use uniform initial state distribution.

In future work, we plan to extend cFVI to (1) offline/batch model-based RL, (2) stochastic maximum entropy policies and (3) explicit model robustness. cFVI uses a fixed data set and hence, can be combined with learning control-affine models to the offline RL benchmark (Gulcehre et al., 2020; Mandlekar et al., 2019). We plan to extend cFVI to maximum entropy and stochastic policies to improve the exploration of RTDP cFVI. Finally, we plan to incorporate robustness w.r.t. to changes in dynamics by using the adversarial continuous RL formulation culminating in the Hamilton-Jacobi-Isaacs differential equation rather than the HJB (Isaacs, 1999).

Acknowledgements

M. Lutter was an intern at Nvidia during this project. A. Garg was partially supported by CIFAR AI Chair. We also want to thank Fabio Muratore, Joe Watson and the ICML reviewers for their feedback. Furthermore, we want to thank the open-source projects SimuRLacra (Muratore, 2020), MushroomRL (D'Eramo et al., 2020), NumPy (Harris et al., 2020) and PyTorch (Paszke et al., 2019).

References

Aström, K. J. Event based control. In *Analysis and design of nonlinear control systems*, pp. 127–147. Springer, 2008.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Bansal, S., Chen, M., Herbert, S., and Tomlin, C. J. Hamilton-Jacobi reachability: A brief overview and recent advances. 2017.

Barto, A. G., Bradtke, S. J., and Singh, S. P. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

Bellman, R. *Dynamic Programming*. Princeton University Press, USA, 1957. ISBN 0691146683.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., and Garg, A. Conservative safety critics for exploration. In *International Conference on Learning Representations (ICLR)*, 2021.

Boyan, J. and Moore, A. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, 7:369–376, 1994.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pp. 8224–8234, 2018.

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, pp. 3245–3254, 2019.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, 2019.

Da, X., Xie, Z., Hoeller, D., Boots, B., Anandkumar, A., Zhu, Y., Babich, B., and Garg, A. Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion. In *Conference on Robot Learning (CoRL)*, 2020.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. Mushroomrl: Simplifying reinforcement learning research. `https://github.com/MushroomRL/mushroom-rl`, 2020.

Doya, K. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., and Mannor, S. Regularized fitted q-iteration for planning in continuous-space markovian decision problems. In *2009 American Control Conference*, pp. 725–730. IEEE, 2009.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 2020.

Harrison*, J., Garg*, A., Ivanovic, B., Zhu, Y., Savarese, S., Fei-Fei, L., and Pavone (* equal contribution), M. AdaPT: Zero-Shot Adaptive Policy Transfer for Stochastic Dynamical Systems. In *International Symposium on Robotics Research (ISRR)*. Springer STAR, 2017.

Hennig, P. Optimal reinforcement learning for gaussian systems. In *Advances in Neural Information Processing Systems*, pp. 325–333, 2011.

Isaacs, R. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

Kappen, H. J. Linear theory for control of nonlinear stochastic systems. *Physical review letters*, 95(20):200201, 2005.

Khalil, H. K. and Grizzle, J. W. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Kim, J., Shin, J., and Yang, I. Hamilton-jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *arXiv preprint arXiv:2010.14087*, 2020.

Kirk, D. E. *Optimal control theory: an introduction*. Courier Corporation, 1970.

Kolter, J. Z. and Manek, G. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, D., Wang, D., Wang, F.-Y., Li, H., and Yang, X. Neural-network-based online hjb solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE transactions on cybernetics*, 44 (12):2834–2847, 2014.

Lutter, M., Belousov, B., Listmann, K., Clever, D., and Peters, J. HJB optimal feedback control with deep differential value functions and action constraints. In *Conference on Robot Learning (CoRL)*, 2019.

Lyshevski, S. E. Optimal control of nonlinear continuous-time systems: design of bounded controllers via generalized nonquadratic functionals. In *Proceedings of the 1998 American Control Conference*, volume 1, pp. 205–209. IEEE, 1998.

Mandlekar*, A., Zhu*, Y., Garg*, A., Fei-Fei, L., and Savarese (* equal contribution), S. Adversarially Robust Policy Learning through Active Construction of Physically-Plausible Perturbations. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Mandlekar, A., Booher, J., Spero, M., Tung, A., Gupta, A., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9 (May):815–857, 2008.

Muratore, F. Simurlacra - a framework for reinforcement learning from randomized simulations. https://github.com/famura/SimuRLacra, 2020.

Muratore, F., Treede, F., Gienger, M., and Peters, J. Domain randomization for simulation-based policy optimization with transferability assessment. In *Conference on Robot Learning*, pp. 700–713, 2018.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation, 2019.

Pan, Y., Theodorou, E. A., and Kontitsis, M. Model-based path integral stochastic control: A bayesian nonparametric approach. *arXiv preprint arXiv:1412.3038*, 2014.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.

Pereira, M., Wang, Z., Exarchos, I., and Theodorou, E. Learning deep stochastic optimal control policies using forward-backward sdes. In *Robotics: science and systems*, 2019.

Pereira, M. A., Wang, Z., Exarchos, I., and Theodorou, E. A. Safe optimal control using stochastic barrier functions and deep forward-backward sdes. *arXiv preprint arXiv:2009.01196*, 2020.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

Quanser. Quanser courseware and resources. https://www.quanser.com/solution/control-systems/, 2018.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017a.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part ii): data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017b.

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Rajagopal, K., Balakrishnan, S. N., and Busemeyer, J. R. Neural network-based solutions for stochastic optimal control using path integrals. *IEEE transactions on neural networks and learning systems*, 28(3):534–545, 2016.

Ramos, F., Possas, R. C., and Fox, D. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators, 2019.

Richards, S. M., Berkenkamp, F., and Krause, A. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. *arXiv preprint arXiv:1808.00924*, 2018.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Tassa, Y. and Erez, T. Least squares solutions of the hjb equation with neural network value-function approximators. *IEEE transactions on neural networks*, 18(4):1031–1041, 2007.

Tesauro, G. Practical issues in temporal difference learning. *Machine learning*, 8(3):257–277, 1992.

Theodorou, E., Buchli, J., and Schaal, S. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *2010 IEEE International Conference on Robotics and Automation*, pp. 2397–2403. IEEE, 2010.

Todorov, E. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Tsitsiklis, J. N. and Van Roy, B. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94, 1996.

Xie, Z., Da, X., van de Panne, M., Babich, B., and Garg, A. Dynamics Randomization Revisited: A Case Study for Quadrupedal Locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Yang, X., Liu, D., and Wang, D. Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints. *International Journal of Control*, 87(3):553–566, 2014.