Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

ANALYSIS OF MEASURE-VALUED DERIVATIVES IN A REINFORCEMENT LEARNING ACTOR-CRITIC FRAMEWORK

Kim van den Houten

Delft University of Technology
Van Mourik Broekmanweg 6
Delft, 2628XE, THE NETHERLANDS

Emile van Krieken
Bernd Heidergott

Vrije Universiteit Amsterdam
De Boelelaan 1105
Amsterdam, 1081HV, THE NETHERLANDS

ABSTRACT

Policy gradient methods are successful for a wide range of reinforcement learning tasks. Traditionally, such methods utilize the score function as stochastic gradient estimator. We investigate the effect of replacing the score function with a measure-valued derivative within an on-policy actor-critic algorithm. The hypothesis is that measure-valued derivatives reduce the need for score function variance reduction techniques that are common in policy gradient algorithms. We adapt the actor-critic to measure-valued derivatives and develop a novel algorithm. This method keeps the computational complexity of the measure-valued derivative within bounds by using a parameterized state-value function approximation. We show empirically that measure-valued derivatives have comparable performance to score functions on the environments Pendulum and MountainCar. The empirical results of this study suggest that measure-valued derivatives can serve as low-variance alternative to score functions in on-policy actor-critic and indeed reduce the need for variance reduction techniques.

1 INTRODUCTION

Reinforcement learning (RL) methods are increasingly successful in areas such as robotics (Carvalho et al. 2021), self-driving cars (Kiran et al. 2022), and energy systems (Perera and Kamalaruban 2021). After the break-trough of the REINFORCE algorithm (Williams 1992), and the policy gradient theorem (Sutton et al. 1999), a variety of successful policy gradient methods have been developed, such as A2C (Mnih et al. 2016), PPO (Schulman et al. 2017), and DDPG (Lillicrap et al. 2016). Policy gradient methods are techniques from RL that optimize a policy with respect to the expected cumulative reward by applying gradient ascent. Most policy gradient algorithms utilize the score function (SF) as gradient estimator for the stochastic objective function. Unfortunately, the policy gradient algorithms developed in the past decades suffer from the excessive variance of the SF gradient estimates (Sutton and Barto 2018). Using SF, several variance reduction techniques are necessary for convergence to the optimal policy, such as implementation of a baseline (Greensmith et al. 2002; Li 2018; Mohamed et al. 2020).

Stochastic optimization seeks gradient estimators with both low variance, and bias. This motivates the study of the measure-valued derivative (MVD), known for having significantly lower variance than SFs. The SF is not widely used in the RL community. We hypothesise that the low variance of the MVD reduces the need for variance reduction techniques. There have been few controlled studies that compare differences in performance between SF, and MVD for RL. However, the promising results from Bhatt et al. (2019), and Carvalho et al. (2021) motivate further research in using MVD for RL purposes.

In this paper, we present a novel algorithm (AC-MVD), in which the MVD is implemented in an on-policy actor-critic algorithm with parameterized state-value functions. We compare the results of this algorithm with the SF variant (AC-SF). The compared algorithms are equivalent in structure, besides the

choice of gradient estimator. The algorithms are tested on the OpenAI Gym environments *Pendulum*, and *MountainCar*, developed by Brockman et al. (2016). We conclude that the AC-MVD can achieve comparable performance to AC-SF, and that for the MVD the variance reducing baseline can be omitted.

2 BACKGROUND

2.1 Reinforcement Learning

In this section, we introduce the continuous control Markov decision process (MDP) setting. We distinguish between the agent, and the environment, where the agent is the decision-maker, and interacts with the environment. The agent-environment combination is a dynamical system that evolves in discrete time steps (Sutton and Barto 2018). We denote the state space by $\mathscr{S}$, and the action space by $\mathscr{A}$. The agent chooses their actions following a parameterized policy $\pi_\theta(a|s)$ that maps states to actions. The trajectory $\tau = (s_0, a_0, s_1, a_1 ...)$ defines the state-action history (Sutton and Barto 2018). Each state action combination (a_t, s_t) is rewarded one time step later with r_{t+1}. The goal is to optimize the cumulative reward by seeking the optimal policy π_{θ^*} (Sutton and Barto 2018):

$$E_{\pi_{\theta^*}}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right] = \max_\theta E_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right],$$

where γ is a discount term for dealing with infinite horizons.

In order to evaluate a policy, two statistics are defined: (i) The state-action value function $Q^\pi(s,a)$ that refers to the expected reward from being in a state s, and taking action a, and then following policy π; and (ii) the value function $V^\pi(s)$ that is defined as the expected reward by following policy π from state s (Sutton and Barto 2018). The two are related as follows:

$$Q(s_t,a_t) = Q^\pi(s_t,a_t) + \gamma\, V^\pi(s_{t+1}) = E_\pi\left[r_{t+1}|s_t,a_t\right] + \gamma\, V^\pi(s_{t+1}).$$

Policy gradient methods learn the parameterized policy based on an estimate of the gradient of

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right].$$

This is done iteratively by applying gradient ascent. The objective is evaluated using stochastic methods:

$$\theta_{t+1} = \theta_t + \varepsilon_t\ \widehat{\nabla_\theta J(\theta_t)}, \quad t \geq 0, \tag{1}$$

with $\varepsilon_t \to 0$, where $\widehat{\nabla_\theta J(\theta)}$ is an estimator for $\nabla J(\theta)$. One algorithm that seeks the optimal policy according to the scheme above is the on-policy actor-critic (Sutton et al. 1999). In this policy gradient method, the iterative update step described in (1) uses the state-action value function, and the SF with respect to the stochastic policy:

$$\theta_{t+1} = \theta_t + \varepsilon_t\ Q(s_t,a_t)\nabla_\theta \log\ \pi_\theta(a_t|s_t). \tag{2}$$

It is proposed to use function approximation for high dimensional, and continuous state, and action spaces, see (Sutton et al. 1999). This leads to an update step for the policy (actor) parameters, and for the parameters for the value function (critic) approximator. Following this approach, the state-action value function in the update step in (2) is replaced by its recursive. Concluding, in each actor-critic iteration, both the actor parameters θ, and the critic parameters η are updated, according to:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \varepsilon_t\left(r_{t+1} + V_\eta^\pi(s_{t+1})\right)\nabla_\theta \log\ \pi_\theta(a_t|s_t),\\ \eta_{t+1} &= \eta_t + \varepsilon_t \nabla_\eta\left(r_{t+1} + V_\eta^\pi(s_{t+1}) - V_\eta^\pi(s_t)\right).\end{aligned}$$

We summarise the important steps of the actor-critic with SF (AC-SF) in Algorithm 1.

Algorithm 1 Actor-Critic Score function (AC-SF)

Input: policy $\pi_\theta(a|s)$, state-value function $V_\eta(s)$, learning rate ε, discount γ
Initialize parameters θ, and η, environment *env*
for i **in** *epochs* **do**
  Initialize s_0 as first state of episode
  for t **in** *episode length* **do**
    Sample action a_t from π, and execute in *env*, observe next state s_{t+1}, reward r_{t+1}
    Actor update $\theta_{t+1} = \theta_t + \varepsilon_t \left(r_{t+1} + \gamma\, V_\eta^\pi(s_{t+1})\right) \nabla_\theta \log\, \pi_\theta(a_t|s_t)$
    Critic update $\eta_{t+1} = \eta_t + \varepsilon_t\, \nabla_\eta \left(r_{t+1} + \gamma\, V_\eta^\pi(s_{t+1}) - V_\eta^\pi(s_t)\right)$
  end for
  Evaluate policy
end for

2.2 Gradient Estimation

Policy gradient methods rely on the use of SF as gradient estimator, see e.g. REINFORCE (Williams 1992), and see Sutton et al. (1999) for the policy gradient theorem. In Section 2.1, we show that this is also the case for actor-critic. In practice, the standard actor-critic structure is complemented with variance reduction techniques, because the variance of SF is high (Greensmith et al. 2002; Li 2018; Mohamed et al. 2020; Cui et al. 2020). The variance of the SF is defined as:

$$V_{f_\theta}[h(X)\ S(\theta,X)] = E_{f_\theta}\left[(h(X)\frac{d}{d\theta}\ \log\ f_\theta(X))^2\right] - E_{f_\theta}\left[h(X)\frac{d}{d\theta}\ \log\ f_\theta(X)\right]^2.$$

For comparisons of the variance of the SF compared to gradient estimation methods, see (Heidergott, Vázquez-Abad, and Volk-Makarewicz 2008; Heidergott and Volk-Makarewicz 2016).

An alternative way of dealing with stochastic gradients stems from measure theory. Under general conditions it is possible to write the gradient of probability density $\frac{\delta}{\delta\theta} f_\theta(x)$ as a re-scaled difference of two densities (Pflug 1989). The mathematical notation of the gradient of probability density $f_\theta(x)$ with respect to its parameters θ is:

$$\frac{d}{d\theta} f_\theta(x) = c_\theta f_\theta^+(x) - c_\theta f_\theta^-(x). \tag{3}$$

In (3), $f_\theta^+(x)$, and $f_\theta^-(x)$ are both densities. The triple $(c_\theta, f_\theta^+(x), f_\theta^-(x))$ is the MVD of $f_\theta(x)$, and is not unique. The decomposition into a positive, and negative component can be used to define the gradient estimator of an expectation of a function $h(x)$. Therefore, we can write:

$$\begin{aligned}\frac{d}{d\theta} E_{f_\theta}[h(X)] &= \int \frac{d}{d\theta} f_\theta(x) h(x) dx \\ &= c_\theta\left(\int h(x) f_\theta^+(x) dx - \int h(x) f_\theta^-(x) dx\right) \\ &= c_\theta\left(E_{f_\theta^+}[h(X)] - E_{f_\theta^-}[h(X)]\right).\end{aligned}$$

SF, and MVD are both unbiased gradient estimators, and for that reason, we can freely interchange the SF, and the MVD representation of a gradient. The variance of the MVD is defined as:

$$V_{f_\theta}[h(X)] = V_{f_\theta^+}[h(X)] + V_{f_\theta^-}[h(X)] - 2Cov_{f_\theta^+; f_\theta^-}[h(X'), h(X)].$$

Usually this is implemented by positively correlating the positive, and the negative component of the MVD (Fisher 2012). This ensures that $Cov_{f_\theta^+; f_\theta^-}[h(X'), h(X)]$ takes a positive value, resulting in lower variance.

The unique Jordan decomposition (Fisher 2012) can reduce the variance of MVD even more, see (Heidergott and Leahu 2010).

Importantly, MVD requires two cost function evaluations for every parameter, and is therefore expensive to compute. This has caused problems in applying MVD for MDP, see e.g. (Abad and Krishnamurthy 2003). However, we can use the policy gradient theorem (Sutton et al. 1999). The policy gradient theorem states that for the gradient ascent updates, we can differentiate directly with respect to the policy instead of considering all underlying state transition probabilities of the MDP. Furthermore, an approximate value function avoids the need for computing Monte-Carlo (MC) roll-outs of trajectories (Sutton and Barto 2018). Combining the two results with an effective MVD decomposition for the policy, we can keep the computational complexity within bounds. The exact procedure is described in Section 4.

3 RELATED WORK

Bhatt et al. (2019) was the first to implement MVD within REINFORCE. In the REINFORCE-SF, every update of policy parameters requires an entire MC roll-out of a trajectory. For incorporation of MVD in this framework, not one, but two MC roll-out trajectories for each update of the policy parameters are needed (Bhatt et al. 2019). Their work nicely illustrates how MVD can work within a policy gradient method, but the REINFORCE-MVD is computationally expensive. Carvalho et al. (2021) provided an extensive analysis on MVD for policy gradients. They developed a MVD-variant of the off-policy Soft Actor-Critic (SAC) (Haarnoja et al. 2018), that in its original form uses the reparameterization trick for gradient estimation. They also compared with SAC-SF, but this method failed to solve most of their tasks because the SF needs a baseline for stabilization. Furthermore, they developed an on-policy algorithm with a non-differentiable Q-function approximator, and compared it with state-of-the art SF algorithms, being Proximal Policy Optimization (PPO) (Schulman et al. 2017), and Trust Region Policy Optimization (TRPO) (Schulman et al. 2015). They concluded that MVD can serve as alternative for policy gradient methods. Their conclusion motivates our hypothesis that the low variance of MVD can avoid the need for variance reduction techniques for the SF. We investigate this by developing an on-policy actor-critic with MVD, and compare it to its SF variant. We keep the algorithmic structure for both algorithms equivalent, besides the choice of gradient estimator, and the baseline for the SF.

4 METHODOLOGY

4.1 Policy Gradient MVD

We introduce our policy $\pi_\theta(a|s)$ parameterized by θ. However, for convenience we redefine the stochastic policy as $\pi(a|s;\omega = g(s,\theta))$. Here, we follow notation from Carvalho et al. (2021). This notation allows to derive the gradient w.r.t. θ by computation of the gradient w.r.t. ω, and utilization of the chain-rule. For example, when choosing a Gaussian policy with fixed variance σ^2, and state-dependent mean $\mu = g(s,\theta)$, we get $\pi(a|s;\omega = g(s,\theta)) = \mathcal{N}(\mu = g(s,\theta),\sigma^2)$. Adapting this notation, we can derive the policy gradient with respect to θ by implementing MVD according to:

$$\begin{aligned}\nabla_\theta \pi(a|s;\omega = g(s,\theta)) &= \nabla_\omega \pi(a|s;\omega = g(s,\theta))\nabla_\theta g(s,\theta)\\ &= c_\theta\big(\pi^+(a^+|s;\omega = g(s,\theta)) - \pi^-(a^-|s;\omega = g(s,\theta))\big)\nabla_\theta g(s,\theta).\end{aligned}$$

4.2 Actor-Critic MVD

We introduce a MVD variant for Algorithm 1, AC-MVD. The major challenge is finding a suitable procedure for implementing MVD for MDP. We aim to keep its computational complexity within bounds, by smartly defining the positive, and negative component of our policy. We define the positive trajectory starting at t as the trajectory that moves following an action a_t^+ drawn at t from the positive policy distribution π^+, and observing next state s_{t+1}^+, while the negative trajectory starts by drawing an action a_t^- at t from π^-, and observing next state s_{t+1}^-. For the remaining trajectory, we assume that actions are chosen using π.

This allows us to use the value function of the real policy for valuating the future expected rewards of the two components of the decomposition. To understand this statement, it is important to observe that:

$$\begin{aligned} Q^+(s_t, a_t^+) &= E_{\pi+}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t, a_t^+\right] \\ &= E_{\pi+}\left[r_{t+1}^+ + E_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s_{t+1}^+\right] \middle| s_t, a_t^+\right] \\ &= E_{\pi+}\left[r_{t+1}^+ + \gamma\, V^{\pi}(s_{t+1}^+) \middle| s_t, a_t^+\right]. \end{aligned}$$

The above equation holds because only for the transition at time step t, the policy follows the positive distribution, while for the remaining transitions π is used. Therefore, we can freely use the value function of the real policy for evaluating the future expected rewards of the two components of the decomposition. The same reasoning can be applied to the negative policy. This result allows us to avoid computing two MC roll-outs of trajectories per update step, such as was done by Bhatt et al. (2019). We adapt the MVD decomposition to the actor update:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \varepsilon_t\; c_\theta \Big(Q^+(s_t, a_t^+) - Q^-(s_t, a_t^-) \Big) \nabla_\theta g(s_t, \theta) \\ &= \theta_t + \varepsilon_t c_\theta \Big(\left[r_{t+1}^+ + \gamma\, V^{\pi}(s_{t+1}^+)\right] - \left[(r_{t+1}^- + \gamma\, V^{\pi}(s_{t+1}^-)\right] \Big) \nabla_\theta g(s_t, \theta). \end{aligned}$$

There is no difference in updating the critic parameters between AC-SF, and AC-MVD. The pseudocode for AC-MVD is given in Algorithm 2.

Algorithm 2 Actor-Critic MVD

Input: policy $\pi_\theta(a|s)$, policies for decomposition: $\pi_\theta^+(a|s)$, and $\pi_\theta^-(a|s)$, state-value function $V_\eta(s)$, learning rates ε, discount γ
Initialize parameters θ, and η, environment *env*
for i **in** *epochs* **do**
  Initialize s_0 (first State of episode)
  for t **in** *episode length* **do**
    Sample action a_t, observe next state s_{t+1}, reward r_{t+1}
    Sample common random number for coupling
    Compute a_t^+, and a_t^- using common random number
    Observe s_{t+1}^+, r_{t+1}^+, and s_{t+1}^+, r_{t-1}^- from applying a_t^+, and a_t^- to s_t
    Actor update
    Compute $\widehat{Q^+}(s_t, a_t^+) = r_{t+1}^+ + \gamma\, V_\eta(s_{t+1}^+)$
    Compute $\widehat{Q^-}(s_t, a_t^-) = r_{t+1}^- + \gamma\, V_\eta(s_{t+1}^-)$
    Update actor parameters $\theta_{t+1} = \theta_t + \varepsilon_t\; c_\theta \left[\widehat{Q^+}(s_t, a_t^+) - \widehat{Q^-}(s_t, a_t^-)\right] \nabla_\theta g_\theta(s_t)$
    Critic update
    Update critic parameters $\eta_{t+1} = \eta_t + \varepsilon_t\; \nabla_\eta \left[r_{t+1} + \gamma\, V_\eta^{\pi}(s_{t+1}) - V_\eta^{\pi}(s_t)\right]$
  end for
  Evaluate policy
end for

5 EXPERIMENTS

5.1 Environments

OpenAI has an RL toolkit, OpenAI Gym, that provides environments on which RL algorithms can be trained (Brockman et al. 2016). We tested on the environments *Pendulum-v0*, and *MountainCarContinous-v0*. Both its action, and state spaces are continuous. In these environments it is possible to reset the system to a specific state, which is required for our MVD implementations. In the *Pendulum* environment there is an inverted pendulum, and the task is to move the pendulum upright. The goal is to remain at zero angle, with the least rotational velocity, and the least effort. The *MountainCar* environment contains two mountains, and a car that is positioned between the two. On the right mountain, a flag is positioned, and the goal is to reach this flag. The car needs to drive back to the left mountain such that it builds up momentum, and is able to reach the right top.

5.2 Settings

We use a Gaussian policy with state-dependent mean $\mathcal{N}(\mu = g(s,\theta), \sigma^2)$, where σ^2 is defined as hyper-parameter. For deriving the MVD of π we used the MVD-triple $(c_{\theta_i}, f_\theta^+(x), f_\theta^-(x))$, where $c_\theta = \frac{1}{\sqrt{2\pi}\sigma}$, $\pi^+ = \mu + \mathcal{W}(2,(2\sigma^2)^{0.5})$, and $\pi^- = \mu - \mathcal{W}(2,(2\sigma^2)^{0.5})$. Here, $\mathcal{W}$ follows a Weibull distribution. In our experiments, we utilize linear combinations of radial basis functions (inspired by Konidaris and Osentoski (2011)), which gives:

$$g(s,\theta) = \sum_{i=1}^{m} \theta_i \phi_i(s).$$

For parameterization of the value function we also used radial basis functions such that:

$$V_\eta^\pi(s) = \sum_{i=1}^{m} \eta_i \phi_i(s).$$

For a detailed explanation on radial basis functions, see Appendix A. The hyperparameters for the actor-critic algorithm are included in Table 1. The same were used for SF, and MVD.

Table 1: Hyperparameters.

Hyperparameter	Setting
Actor optimizer	SGD
Actor learning rate	1e-4
Actor optimizer	SGD
Critic learning rate	5e-3
Max grad norm	0.5
Policy initial σ^2	0.5
Policy final σ^2	0.1
Discount	0.99
Epochs	500
Episode length	200

5.3 Evaluation

We evaluate the rolling episode reward after each epoch to measure learning performance of the algorithm. To rule out the possibility that the results are obtained randomly, we repeat for each experiment three times its 10 seeds. After collecting the 10 runs, the mean over these runs is visualized together with a 95%−CI

around the mean. The width of the CI implies the magnitude of the variance in the learning outcomes. Furthermore, we measure variance of the gradient estimators for comparing SF, and MVD. This is done by repeatedly computing the gradient estimate without updating the parameters. The system is frozen in the sense that while doing this we did not move to a next state. This procedure enables evaluation of the variance of the gradient estimators. Finally, for all experiments we measure average CPU-time. All experiments are done on a Dell Latitude E7450 with Intel Core i7 5600U. The code is available on GitHub (van den Houten 2022).

6 RESULTS

This section presents the experimental results for the different environments. Table 2 provides an overview, and shows the evaluated episode reward that is achieved after 500 epochs, and the averaged CPU-time.

Table 2: Summarising Results: * Reward$\pm$95%$-$CI refers to the average cumulative reward of the last 100 runs after training of 500 epochs. This refers therefore to the moving average evaluated at t=400. Confidence interval is for 10 runs. ** CPU-time refers to the average time of executing the algorithm for 500 epochs (evaluated over 10 independent runs).

Environment	Gradient	Reward$\pm$95%CI*	CPU-time**
Pendulum	SF	-194 ± 16	76
Pendulum	MVD	-167 ± 11	200
MountainCar	SF	88 ± 1	25
MountainCar	MVD	92 ± 0	75

6.1 Pendulum

In Figure 1, the learning curves on the environment *Pendulum* for AC-SF, and AC-MVD are presented. The achieved cumulative reward after 500 epochs with AC-MVD lies very close to the result with AC-SF. It can be seen that for both methods, already after about 250 epochs an reward of -250 is achieved. In the plots in Figure 1, we see that between epoch 250, and epoch 500 the reward fluctuates around -200. In Table 2 the average reward of the last 100 epochs (averaged over 10 runs) is given, and yields -194 for SF, and -167 for MVD. Comparing the plots in Figure 1, we observed a small difference in the magnitude of the confidence intervals which are based on the independent runs, and are visualised in light green. We observed a narrower confidence interval for the MVD. This suggests better convergence for the MVD. We compared the variances of the estimators $\frac{d}{d\mu}\widehat{J(\mu(\theta)}$ to explain the variance in the confidence intervals in Figure 2. One can observe that while the variance for the SF estimator lies around 10^5, the variance for the MVD fluctuates around 1, and decreases to 10^{-1} after 500 epochs.

6.2 MountainCar

Observing the results for the *MountainCar* environment, only small differences can be noticed, being a somewhat wider confidence interval around the mean over 10 runs for the SF that is visible in the reward plots in Figure . However, the variances of the two estimators are much further apart. This can be seen in Figure 4. The variance of the SF lies around 10^5, while the variance of the MVD lies around 10^{-2}.

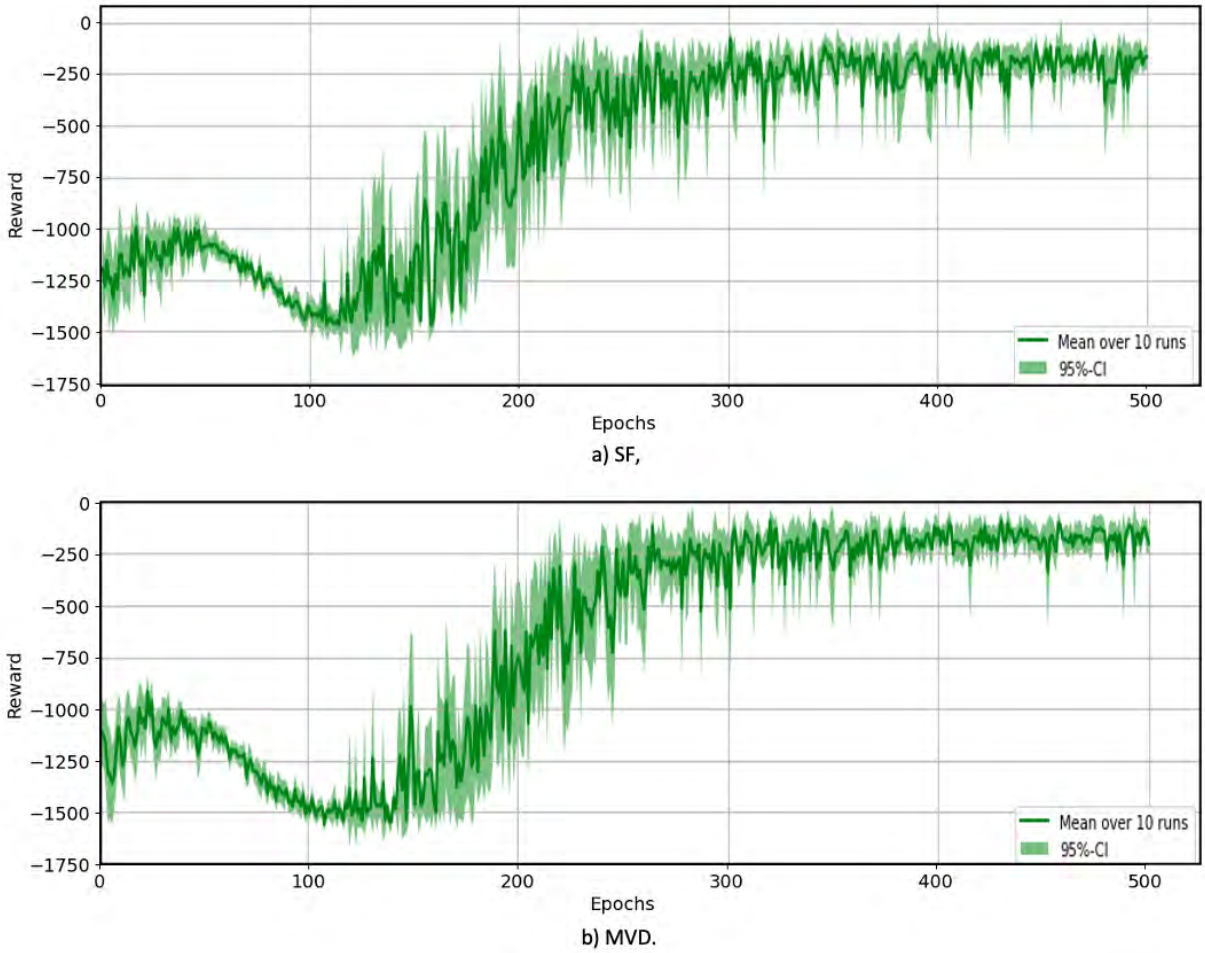

Figure 1: Epoch reward over 10 runs, tested on *Pendulum* environment

7 CONCLUSION AND DISCUSSION

In this work we investigate the effect of replacing the SF with MVD within an on-policy actor-critic algorithm. We hypothesised that variance reduction techniques can be omitted when using MVD. This study provides a novel algorithm for on-policy actor-critic MVD with value function approximation. We compared AC-SF with baseline, and AC-MVD with radial basis function as function approximator. We tested on the *Gym OpenAI* environments *Pendulum* and *MountainCar*. We observed that AC-SF with baseline and AC-MVD perform almost equally well in terms of achieved reward for both environments. The measured variance of the MVD is much lower than the variance of the SF, as expected. We observed this low variance of the MVD in the narrower confidence intervals around the average reward for different runs. Our results suggest that the variance reducing baseline can be omitted when using MVD and that the MVD indeed can serve as an alternative gradient estimator within on-policy actor-critic.

The approximating value function provides an elegant and novel procedure for implementing MVDs for continuous control MDPs. This value function captures the effect of the decomposition of the policy into a positive and negative component. This technique avoids the need of computing two entire MC trajectories for the update step, as is previously done by Bhatt et al. (2019). The additional CPU-time for AC-MVD is in line with the extra steps in the algorithm compared to AC-SF. Our method requires that the system can be reset to a certain state. For tasks as robotics this is an undesirable property, like is mentioned by

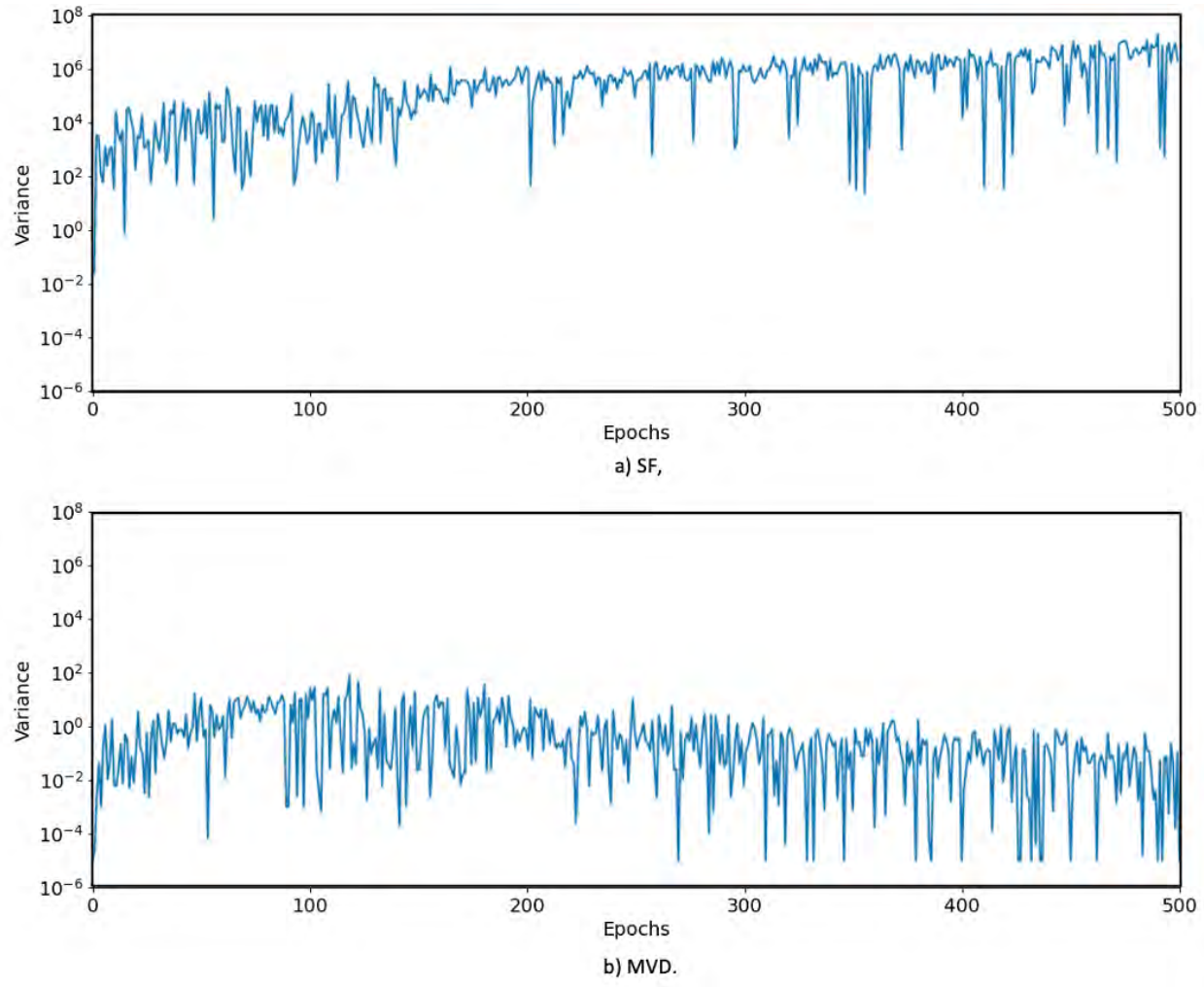

Figure 2: Variance of gradient estimator, tested on *Pendulum* environment

Carvalho et al. (2021). However, for many applications in simulation environments this is not necessarily problematic.

A direction for further research could be an extensive study into what variance reduction techniques can be omitted while using MVD. A related and interesting question is whether state-of-the-art score function policy gradient methods can be extended with MVD to get further variance reduction. A second direction for further research is to investigate whether a combination of SF and MVD within one algorithm can be of value for the RL community. One possible approach would be to start with a cheap, but high-variance SF, and then switch to MVD to stabilize training towards the end of the algorithm. These research directions can help answering the question whether MVD policy gradients can compete with state-of-the-art SF implementations.

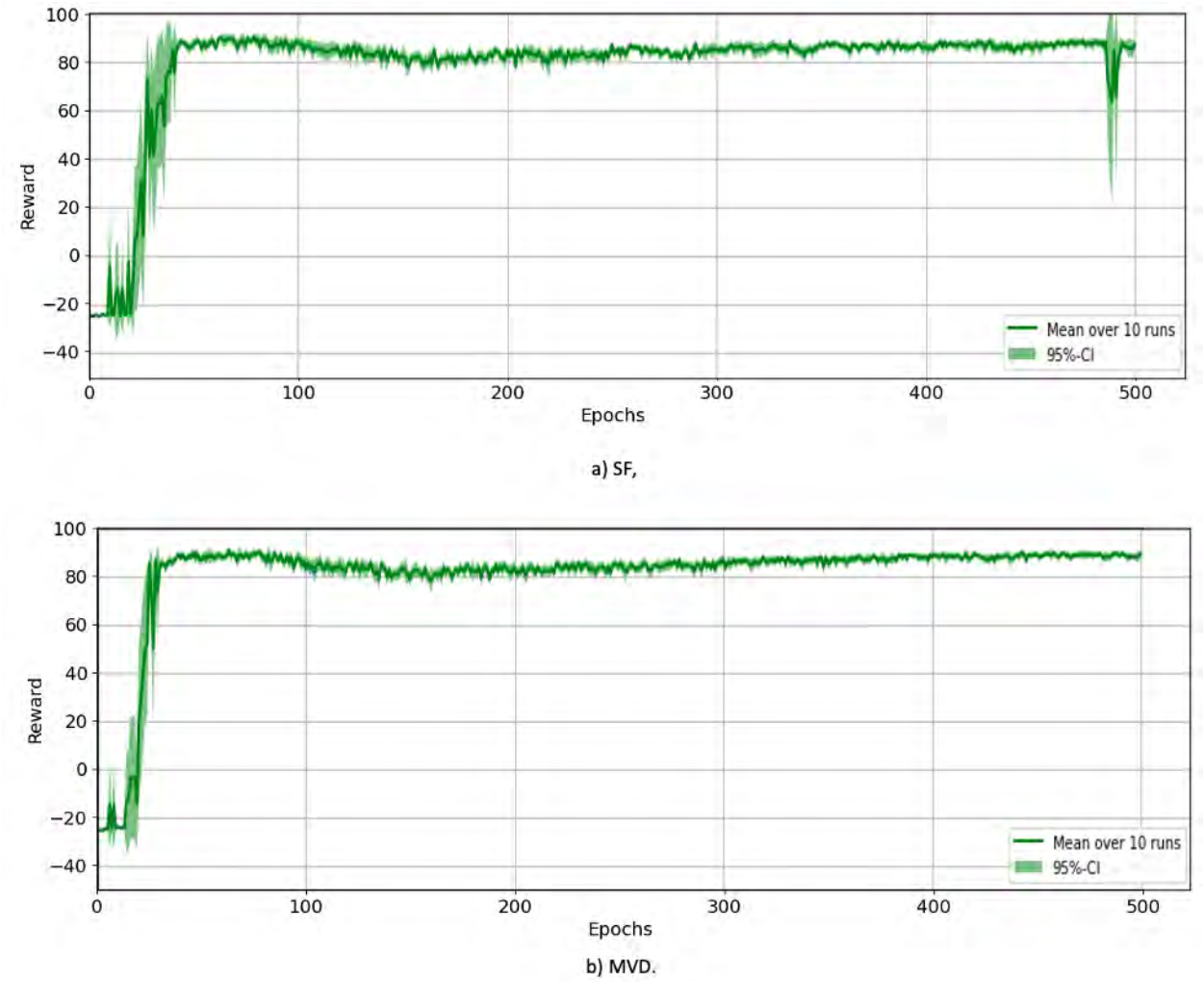

Figure 3: Epoch reward over 10 runs, tested on *MountainCar* environment.

APPENDIX

A Radial Basis Functions

In this research we used Radial Basis Functions (RBF) (Konidaris and Osentoski, 2011), to transform the original state in to a set of basis functions $\phi_1, ..., \phi_m$ and approximate the policy and value function. Suppose that the original state space has d dimensions. Each dimension can be evenly divided into n pieces. This leads to n^d centers c_i with dimension d that are used for the Gaussian basis functions. The basic functions are defined as:

$$\phi_i(s) = \frac{1}{\sqrt{2\pi\sigma_r^2}} e^{-||c_i - s||_F^2 / 2\sigma_r^2}, \ 1 \leq i \leq n^d.$$

In order to transform an original state s to RBF basis functions, the Fobrenius norm of the vector $A = s - c_i$ must be computed, which is given by:

$$||A||_F = \sqrt{\sum_{i,j} |a_{ij}|^2}.$$

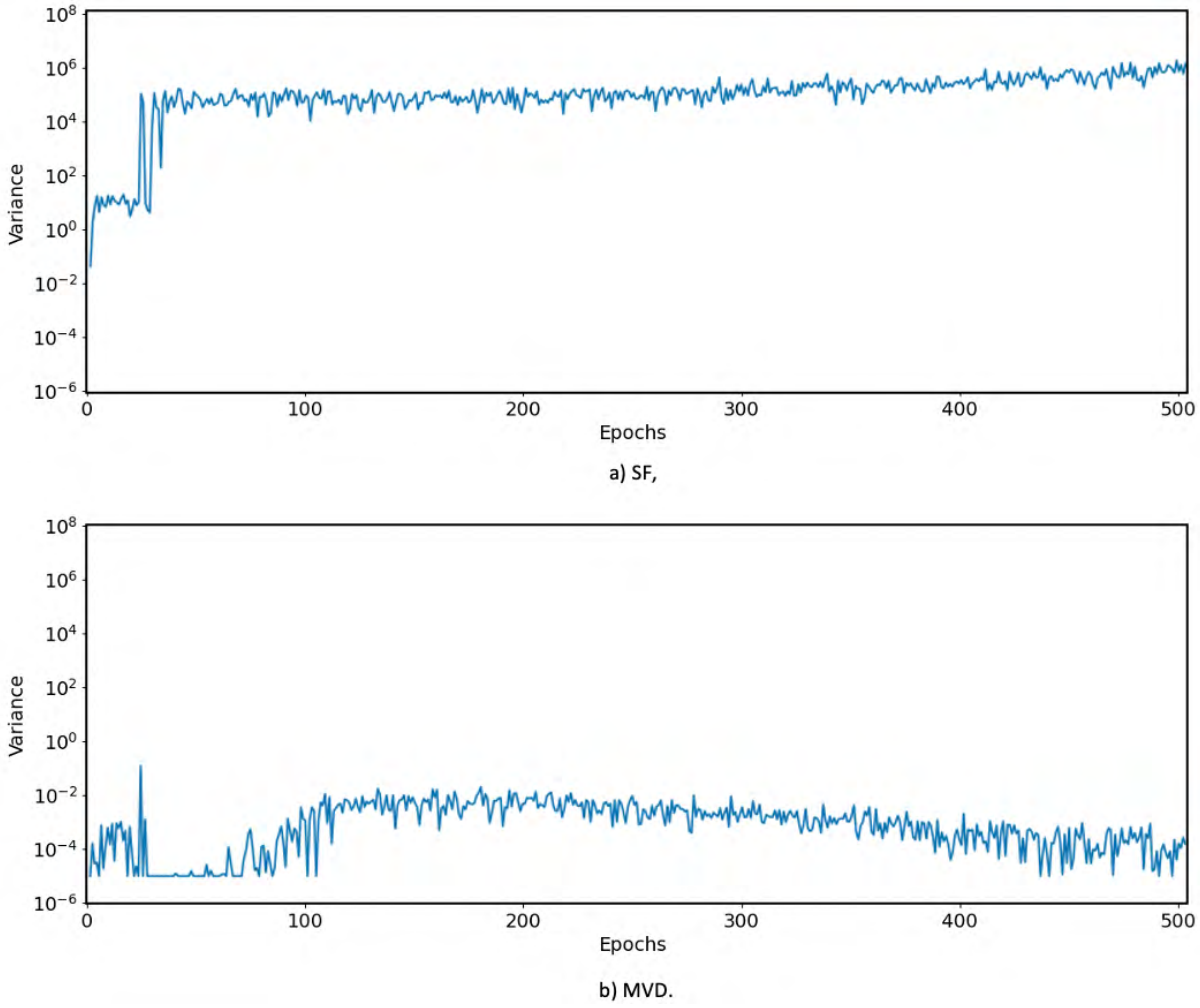

Figure 4: Variance of gradient estimator, tested on *MountainCar* environment.

For the AC-SF with radial basis functions we utilized an existing baseline implementation that is available on GitHub (Pan 2020).

REFERENCES

Abad, F., and V. Krishnamurthy. 2003. "Policy Gradient Stochastic Approximation Algorithms for Adaptive Control of Constrained Time Varying Markov Decision Processes". In *Proceedings of the 42nd IEEE International Conference on Decision and Control*. December 9th-12th, Maui, Hawaii, United States, 2823-2828.

Bhatt, S., A. Koppel, and V. Krishnamurthy. 2019. "Policy Gradient using Weak Derivatives for Reinforcement Learning". In *Proceedings of the 58th IEEE Conference on Decision and Control*. December 11th-13th, Nice, France, 5531-5537.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. 2016. "OpenAI Gym". https://github.com/openai/gym.

Carvalho, J., D. Tateo, F. Muratore, and J. Peters. 2021. "An Empirical Analysis of Measure-Valued Derivatives for Policy Gradients". In *Proceedings of International Joint Conference on Neural Networks*. July 18th-22th, Shenzen, China, 1-10.

Cui, Z., M. C. Fu, J.-Q. Hu, Y. Liu, Y. Peng, and L. Zhu. 2020. "On the Variance of Single-run Unbiased Stochastic Derivative Estimators". *INFORMS Journal on Computing* 32(2):390–407.

Fisher, T. 2012. "Existence, Uniqueness, and Minimality of the Jordan Measure Decomposition". *arXiv e-prints*. https://arxiv.org/abs/1206.5449.

Greensmith, E., P. L. Bartlett, and J. Baxter. 2002. "Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning". *Journal of Machine Learning Research* 5:1471–1530.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine. 2018. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement". In *Proceedings of the 35th International Conference on Machine Learning*. July 10th-15th, Stockholm, Sweden, 1861-1870.

Heidergott, B., and H. Leahu. 2010. "Weak Differentiability of Product Measures". *Mathematics of Operations Research* 35(1):27–51.

Heidergott, B., F. J. Vázquez-Abad, and W. Volk-Makarewicz. 2008. "Sensitivity Estimation for Gaussian Systems". *European Journal of Operational Research* 187(1):193–207.

Heidergott, B., and W. Volk-Makarewicz. 2016. "A Measure-Valued Differentiation Approach to Sensitivities of Quantiles". *Mathematics of Operations Research* 41(1):293–317.

Kiran, B., I. Sobh, V. Talpaert, P. Mannion, A. Sallab, S. Yogamani, and P. Perez. 2022. "Deep Reinforcement Learning for Autonomous Driving: A Survey". *IEEE Transactions on Intelligent Transportation Systems* 23(6):4909–4926.

Konidaris, G., and S. Osentoski. 2011. "Value Function Approximation in Reinforcement Learning Using the Fourier Basis.". In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, edited by D. Leake, R. Morris, M. Wellman, and S. Ludvik, 380–385. Menlo Park, California, United States: the Association for the Advancement of Artificial Intelligence Press.

Li, Y. 2018. "Deep Reinforcement Learning". *arXiv e-prints*. http://arxiv.org/abs/1810.06339.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2016. "Continuous Control with Deep Reinforcement learning". In *Proceedings of the 4th International Conference on Learning Representations*. May 2nd - 4th, San Juan, Puerto Rico.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu. 2016. "Asynchronous Methods for Deep Reinforcement Learning". In *Proceedings of the 33rd International Conference on Machine Learning*. June 19th-24th New York City, New York, United States, 1928–1937.

Mohamed, S., M. Rosca, M. Figurnov, and A. Mnih. 2020. "Monte Carlo Gradient Estimation in Machine Learning". *Journal of Machine Learning Research* 21(1):5183–5244.

Pan, H. 2020. *AC-SF*. https://github.com/workofart/openai-gym-baselines/blob/master/Pendulum-v0/actor_critic_baseline.py.

Perera, A., and P. Kamalaruban. 2021. "Applications of Reinforcement Learning in Energy Systems". *Renewable and Sustainable Energy Reviews* 137:110618.

Pflug, G. 1989. "Sampling Derivatives of Probabilities". *Computing* 42:315–328.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz. 2015. "Trust Region Policy Optimization". In *Proceedings of the 32nd International Conference on Machine Learning*. July 7th-9th, Lille, France, 1889-1897.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal Policy Optimization Algorithms". *arXiv e-prints*. http://arxiv.org/abs/1707.06347.

Sutton, R., and A. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, Massachusetts, United States: The MIT Press.

Sutton, R. S., D. McAllester, S. Singh, and Y. Mansour. 1999. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, edited by S. Solla, T. Leen, and K. Müller, 1057—1063. Cambridge, Massachusetts, United States: MIT Press.

van den Houten, K. 2022. *AC-MVD*. https://github.com/kimvandenhouten/AC-MVD.

Williams, R. J. 1992. "Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning". *Machine Learning* 8:229–256.

AUTHOR BIOGRAPHIES

KIM VAN DEN HOUTEN is a PhD candidate in Algorithmics at the Technische Universiteit Delft, the Netherlands. She received her master's degree in Operations Research from the Vrije Universiteit, Amsterdam. Her research interests are on the intersection of optimization and machine learning, and simulation. Her email adress is k.c.vandenhouten@tudelft.nl.

EMILE VAN KRIEKEN is a PhD candidate in Artificial Intelligence at the Vrije Universiteit Amsterdam, the Netherlands. He holds a master's degree in Artificial Intelligence from the University of Amsterdam. His research interests include neuro-symbolic AI, probabilistic deep learning and optimization. His email adress is e.van.krieken@vu.nl.

BERND HEIDERGOTT is the professor of Stochastic Optimization at the Department of Operations Analytics at the Vrije Universiteit Amsterdam, the Netherlands. He received his PhD degree from the University of Hamburg, Germany, in 1996, and held postdoc positions at various universities before joining the Vrije Universiteit. Bernd is research fellow of the Tinbergen Institute and board member of the Amsterdam Business Research Institute. His research interests are optimization and control of discrete event systems, perturbation analysis, Markov chains, max-plus algebra, and social networks. His email adress is b.f.heidergott@vu.nl