An Idiosyncrasy of Time-discretization in Reinforcement Learning

Kris De Asis
kldeasis@ualberta.ca
Department of Computing Science
University of Alberta

Richard S. Sutton
rsutton@ualberta.ca
Department of Computing Science
University of Alberta

Abstract

Many reinforcement learning algorithms are built on an assumption that an agent interacts with an environment over fixed-duration, discrete time steps. However, physical systems are continuous in time, requiring a choice of time-discretization granularity when digitally controlling them. Furthermore, such systems do not wait for decisions to be made before advancing the environment state, necessitating the study of how the choice of discretization may affect a reinforcement learning algorithm. In this work, we consider the relationship between the definitions of the continuous-time and discrete-time returns. Specifically, we acknowledge an idiosyncrasy with naively applying a discrete-time algorithm to a discretized continuous-time environment, and note how a simple modification can better align the return definitions. This observation is of practical consideration when dealing with environments where time-discretization granularity is a choice, or situations where such granularity is inherently stochastic.

1 Introduction

Reinforcement learning provides a framework for solving sequential decision making problems based on evaluative feedback (Sutton and Barto, 2018). It remains a promising approach for robot learning as it can allow for real-time adaptation of behavior. Many reinforcement learning algorithms assume that the agent-environment interaction occurs at synchronous, discrete time steps, where the environment waits for an action before advancing. In contrast, real-world physical systems are continuous in time, and do not wait for an agent's input. As such, time-discretization becomes a necessary and important consideration, as evidenced by Mahmood et al. (2018a).

Prior work suggests that current reinforcement learning algorithms are sensitive to the choice of discretization. Tallec et al. (2019) emphasize that action-values converge to state-values as the discretization interval approaches zero, creating degenerate cases for algorithms like Q-learning. Similarly, Munos (2006) showed that the variance of policy gradients can be infinite under the same limit. Zhang et al. (2023) characterize a fundamental bias-variance trade-off with the degree of discretization while Mahmood et al. (2018a) detail another trade-off between having fine-grained control and being able to discern the changes between subsequent states. Finally, Farrahi and Mahmood (2023) provide guidelines for time-discretization-aware parameter selection by acknowledging how changes in discrete-time parameters influence the underlying continuous-time objective.

In this work, we explicitly view the discrete-time objective as a discrete approximation of the continuous-time objective. By considering *when* rewards occur, particularly in existing continuous-control environment setups, we identify an idiosyncratic dependence on the choice of discretization beyond those listed by Tallec et al. (2019) and Farrahi and Mahmood (2023). Specifically, the discrete-time return can be viewed as mixing two Riemann sums. We characterize and demonstrate that this is a relatively poor integral approximation in comparison with a conventional Riemann sum and provide a simple modification to the definition of the return to better align the objectives.

The contributions of this work are as follows:

- We acknowledge and characterize an issue with naively applying a discrete-time reinforcement learning algorithm to a *discretized* continuous-time environment in terms of a discrepancy between the discrete-time and continuous-time definitions of the return.
- Based on an integral approximation perspective, we propose a simple modification to the definition of the return to alleviate this idiosyncratic dependence on time-discretization.
- We characterize when the modification will have a modest impact and support our claim with empirical evaluation in both continuous-time prediction and control.

2 Definitions of the Return

In discrete-time reinforcement learning, the discounted return from time step t onward is defined as:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} R_{k+1}, \tag{1}$$

where T is the final time step of an episodic task, or ∞ in an infinite-horizon setting. In continuous-time reinforcement learning (e.g., Doya, 2000; Mehta and Meyn, 2009; Frémaux et al., 2013; Lee and Sutton, 2021; Tallec et al., 2019), we instead define the *integral return* from time step t onward:

$$\tilde{G}_t = \int_t^T \gamma^{\tau-t} R_\tau d\tau. \tag{2}$$

This formulation is pertinent to applications with real-time interaction (e.g., robotics). Despite being continuous in time, robots are often digitally controlled, necessitating understanding the impact of the choice of time-discretization and how it relates these two objectives.

3 When Rewards Occur

There are notational differences in the literature with respect to time indices in the discrete time return (Equation 1). Some define it to start from R_{t+1} (e.g., Sutton, 1988; Precup et al., 2000; van Seijen et al., 2009; Barreto et al., 2017), as presented in this document, while some would start from R_t (e.g., Watkins, 1989; van Hasselt, 2010; Mnih et al., 2015; Wang et al., 2016). This inconsistency is inconsequential when solely considering the discrete-time setting as the rewards occur at the same locations in an agent's stream of experience. However, it has implications when viewed as a discrete approximation to an underlying integral return. Thus, it is worth considering *when* rewards occur.

We emphasize the focus on a setting where there is an underlying continuous-time objective of which a digital learning agent samples at an arbitrary (and potentially variable) frequency. Despite the discrete-time notational differences, it is often agreed upon that from the agent's perspective, the reward and next state are jointly observed. This is reflected in environment step calls in relatively standard reinforcement learning APIs (e.g., Brockman et al., 2016), agent-environment interaction diagrams (e.g., Sutton and Barto, 2018), or explicit acknowledgement that reward can be a function of state, action, and *next state* (e.g., Puterman, 1994). In real-time settings that do not wait for an agent's input, meaningful evaluative feedback must come *after* time t as actions take time to execute and to have a causal influence. Hardware limitations on sampling rates further delay when a system can receive feedback for an action. In many existing robotics environments, where the considered setting is especially pertinent, rewards are often explicitly computed based on the next time step's state information. For example, rewards based on distance traveled in some direction between two time steps, or distance between an end-effector and a desired setpoint at the subsequent time step, as done by Todorov et al. (2012), Brockman et al. (2016), and Mahmood et al. (2018b).

Of note, semi-MDPs and options (Sutton et al., 1999; Precup, 2000) address the problem of when rewards occur, but under the assumption that one has access to higher-frequency interaction with the

environment to integrate the discounted sum of rewards within the discretization interval. It is akin to the agent being aware of and able to time when each component of a temporally-extended reward occurs. Here, we consider when one *does not* have access to these higher-frequency samples but is aware of how much time has elapsed between discrete decision points. Acquiring such information may not be possible due to hardware limitations, and highlights a nuance that arises when naively applying a discrete-time algorithm to a discretized continuous-time environment.

4 Implications for Time Discretization

If we consider rewards jointly arriving with the next state, at least from the agent's perspective, then there is an idiosyncrasy with respect to approximating an underlying integral return. While definitions of the discrete-time return may differ in their use of reward time indices, they are consistent on when discounting begins: the first reward is given weight $\gamma^0 = 1$, with subsequent rewards weighted by increasing powers of γ. We can view the integral return in Equation 2 to be of the form:

$$\int_t^T f(\tau)g(\tau)d\tau, \tag{3}$$

where $f(\tau)$ is the discounting term and $g(\tau)$ is the reward signal. A right-point Riemann sum approximation to this would yield:

$$\sum_{i=0}^{n-1} f(\tau_i)g(\tau_i)\Delta, \tag{4}$$

where $\Delta = \frac{T-t}{n}$ and $\tau = \{t+\Delta, t+2\Delta, ..., T\}$. The right-point Riemann sum beginning with $t+\Delta$ aligns with an agent jointly receiving a reward with the observation of the next state. However, this sum would weight the first reward by $\gamma^\Delta \neq \gamma^0$. This highlights that if one naively applies a discrete-time reinforcement learning algorithm to a discretized continuous-time environment, it is akin to a left-point Riemann sum for discounting and a right-point Riemann sum for rewards:

$$\sum_{i=0}^{n-1} f(\tau_i)g(\tau_{i+1})\Delta, \tag{5}$$

where $\tau \in \{t, t+\Delta, t+2\Delta, ..., T\}$. See Figure 1 for a visualization of this Riemann sum. This sum still converges to the correct integral as $n \to \infty$ as Bliss's Theorem (1914) shows that each function may be evaluated *anywhere* in the interval. However, for the specific case where a left-point Riemann sum is used for discounting, we expect this to perform *worse* than committing to a right-point Riemann sum. If one draws a rectangle with opposite corners at any two points of an exponential decay, the area above and below the curve represents the approximation errors of left- and right-point Riemann sums, respectively. There will always be more area above the curve than below due to the curvature of exponential decay, implying that an underestimate (right-point) has strictly lower error than an overestimate (left-point). This is visualized in Figure 2.

To rectify this discrepancy and commit to a right-point Riemann sum approximation, we simply multiply the discrete-time return by a factor of γ. For example, with $\Delta = 1$:

$$\gamma G_t = \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots . \tag{6}$$

For a *fixed, pre-specified* action cycle-time Δ, there is no loss of generality, as the discrete-time return is proportional by a factor of $\gamma^\Delta \Delta$. However, this is not the case when Δ may vary over time, for example, due to an adaptive algorithm (e.g., Karimi et al., 2023) or inherent stochasticity. These concerns similarly apply to a variable γ and may extend toward tuning fixed-Δ or γ in practice in terms of an unintuitive dependence on discretization. To emphasize the dependence on Δ, we note

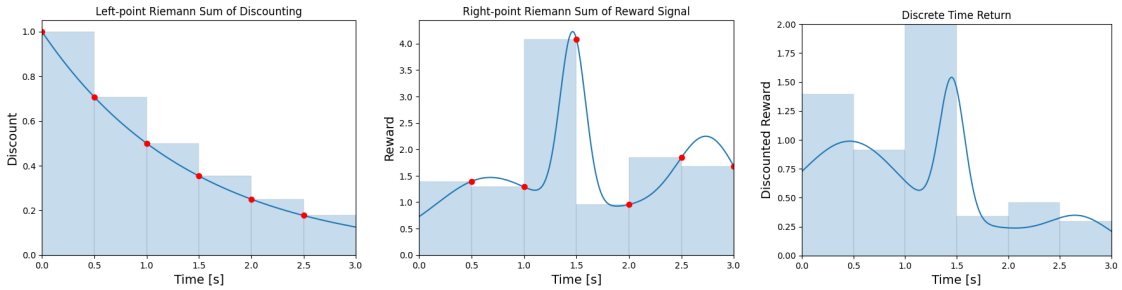


Figure 1: The resulting sum when applying a discrete-time algorithm to a discretized continuous-time domain. Note how rectangle heights may fall out of the function's range within an interval.

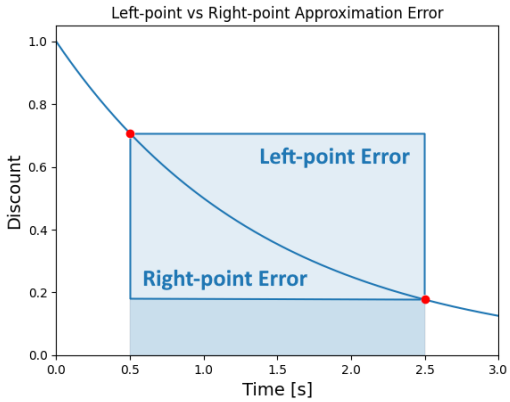


Figure 2: A visualization of the left-point and right-point Riemann sum approximation errors for an exponential decay. Due to curvature, a right-point Riemann sum will always have lower error.

the more explicit definition of the right-point Riemann sum return:

$$G_t^{RP} \stackrel{\text{def}}{=} \sum_{k=t}^{T-1} \gamma^{\sum_{i=t}^{k} \Delta_{i+1}} R_{k+1} \Delta_{k+1} \tag{7}$$
$$= \gamma^{\Delta_{t+1}} R_{t+1} \Delta_{t+1} + \gamma^{\Delta_{t+1}+\Delta_{t+2}} R_{t+2} \Delta_{t+2} + \cdots .$$

Tallec et al. (2019) and Farrahi and Mahmood (2023) have acknowledged the modifications of scaling rewards by Δ and exponentiating γ by Δ in terms of improving robustness to time-discretization. The key difference and contribution in Equation 7 being the earlier discounting.

5 Comparison with Standard Riemann Sums

To see how the discrete-time return (DTR) in Equation 5 compares with a right-point Riemann sum, we use them to numerically integrate random continuous-time signals. Inspired by robotics, we consider periodic signals and Gaussian mixtures. Periodic signals are comparable to signals pertaining to robot locomotion, while Gaussian mixtures instead resemble both sparse and distance-based rewards depending on the spread of each Gaussian. We fix the signal length to 3 seconds, with no loss of generality due to being continuous in time. Each signal generator is detailed below:

Random Periodic Signals - This signal sums 6 sinusoids $\sum_{i=0}^{5} A_i \sin(\omega_i t + \phi_i)$ with angular frequencies $\omega \in \{\frac{2\pi}{4}, \frac{2\pi}{2}, 2\pi, 4\pi, 8\pi, 16\pi\}$, amplitudes $A_i \sim \mathcal{N}(0, 1)$, and phase shifts $\phi_i \sim \mathcal{U}(0, 2\pi)$.

Random Gaussian Mixtures - This signal sums 6 Gaussians $\sum_{i=0}^{5} \mathcal{N}(\mu_i, \sigma_i)$ with means $\mu_i \sim \mathcal{U}(0, 3)$ and standard deviations $\sigma_i \sim \mathcal{U}(0, \frac{3}{2})$.

For each method, we varied the number of intervals $n \in \{5, 10, 25, 50, 100\}$, the discount factor $\gamma \in \{0.5, 0.75, 0.875\}$, and measured the absolute error of the integral approximation relative to a fine-grained mid-point Riemann sum with 10^4 intervals. The values of γ used may appear small and unrepresentative of typical values. We however note that the discount is *per second* and that for a robot sampling every 30 ms, $\gamma = 0.5$ is effectively $\gamma^{\Delta} = 0.5^{0.03} \approx 0.98$ per discrete time step. Averaged across 10^6 randomly generated signals of each type, the results can be seen in Figure 3.

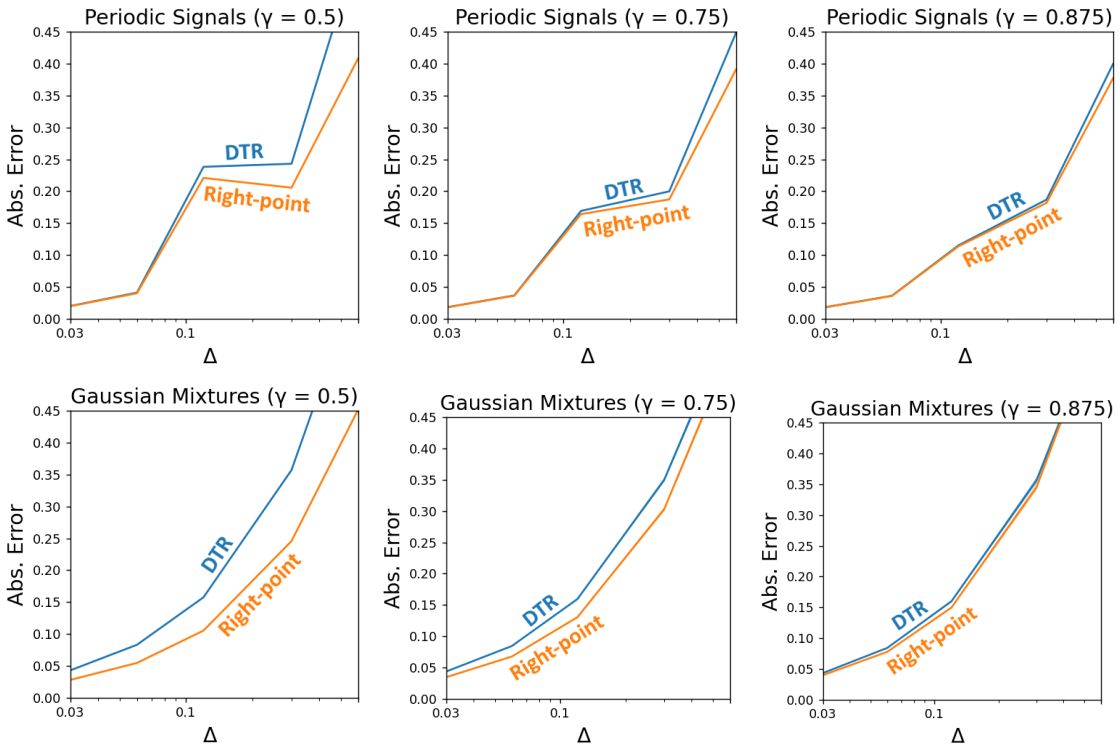


Figure 3: Numerical integration approximation error on *discounted* random signals. Results are averaged over 10^6 signals and shaded regions represent one standard error.

As expected, the errors generally increase as $\Delta \propto \frac{1}{n}$ increases. There is a consistent dip in error with the periodic signals which is likely due to the intervals coincidentally aligning with the pre-specified frequencies. Across all settings, DTR had larger absolute error and is consistent with our hypothesis that DTR would perform worse than right-point when integrating discounted signals. The gap closes as $\gamma \to 1$ as the sums are equivalent at this extreme.

We then considered stochastic intervals to simulate variable time-discretization. This was implemented by sampling, sorting, and re-scaling a set of $n + 1$ uniform random points to represent interval endpoints. This is particularly pertinent as DTR is no longer proportional to right-point and reflects the variability in applications on real-time systems. Fixing $\gamma = 0.75$, Figure 4 shows results averaged across 10^6 randomly generated signals of each type plotted against *average* Δ. Errors generally increased, with DTR maintaining larger approximation error across every setting.

Lastly, to see whether results hold beyond exponential discounting, we considered the product of each pair of the signal generators. This evaluates each sum in a more general numerical integration setting, while resembling transition-dependent γ as White (2016) has advocated for in reinforcement learning. Averaged across 10^6 randomly generated signal pairs, the results can be seen in Figure 5. Perhaps surprisingly, the gap between DTR and the right-point Riemann sum widens dramatically. This suggests that beyond the structure of discounting, DTR is a generally worse integral approximation.

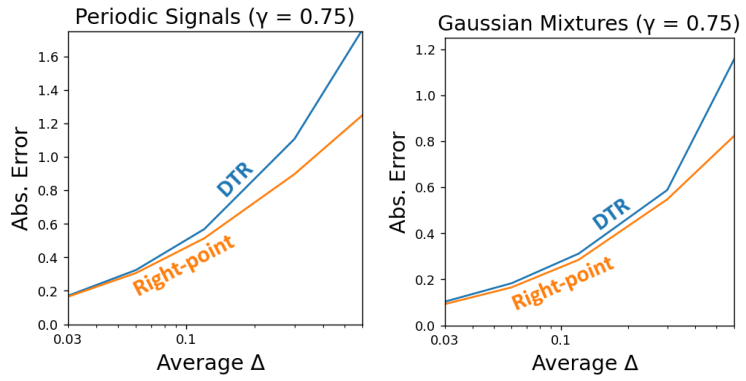


Figure 4: Numerical integration approximation error on *discounted* random signals, with *stochastic discretization intervals*. Results are averaged over 10^6 signals and shaded regions represent one standard error.

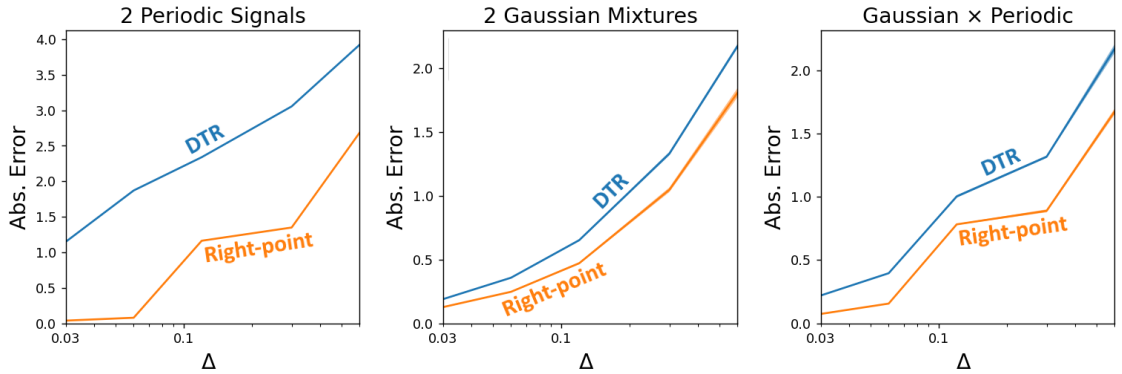


Figure 5: Numerical integration approximation error on *undiscounted* products of random signals. Results are averaged over 10^6 signals and shaded regions represent one standard error.

A key takeaway from these results is that shifting the discount factor in the discrete-time return yields a better prediction target (e.g., in value-based methods) in terms of error between the integral return. To reiterate, in the *fixed* Δ case, the sums are proportional despite the gaps in approximation error. This suggests that the improvement is inconsequential for control. However, in the *variable* Δ setting, we expect that learning from estimates which better approximate the underlying integral return should improve the capability to maximize it. We explore this further in the next section.

6 Discretized Continuous-time Control

To evaluate the right-point Riemann sum in a continuous-time control setting, we build off of the REINFORCE algorithm (Williams, 1992). Such a choice is due to the algorithm's simplicity, allowing for more confidence in attributing differences in performance. We specifically use *online* REINFORCE with eligiblity traces (Kimura et al., 1995) and dropped the γ^t term:

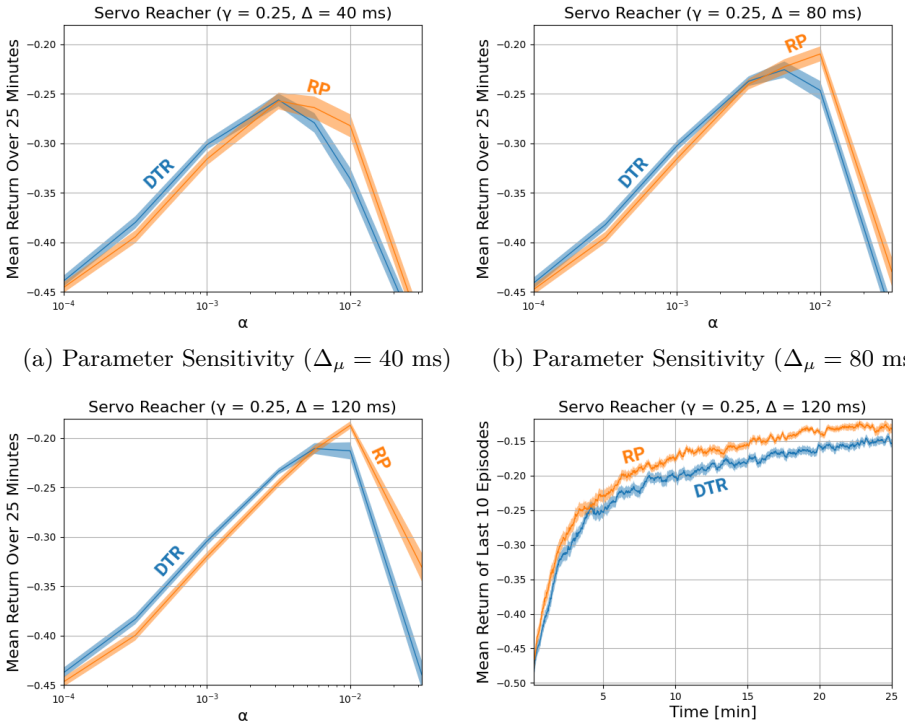
$$\mathbf{z} \leftarrow \mathbf{z} + \nabla_\theta \log \pi(A_t|S_t)$$
$$\theta \leftarrow \theta + \alpha R_{eff} \mathbf{z}$$
$$\mathbf{z} \leftarrow \gamma^{\Delta_{t+1}} \mathbf{z},$$

where Δ_{t+1} is the elapsed time between time steps t and $t+1$, $R_{eff} = R_{t+1}\Delta_{t+1}$ for the discrete-time return, and $R_{eff} = \gamma^{\Delta_{t+1}} R_{t+1}\Delta_{t+1}$ for the right-point Riemann sum. The above algorithm

employs the recommendations of Farrahi and Mahmood (2023) for making algorithms more robust to time-discretization, emphasizing that the proposed right-point modification is complimentary.

We designed a simulated *Servo Reacher* environment based on the setup by Mahmood et al. (2018b), with physical parameters sourced from a Dynamixel MX-28AT data sheet. This custom environment allows for fine-grained computation of the integral return, and flexibility in the discretization intervals an agent can sample at. Full environment specification can be found in Appendix A. To simulate the inherent stochasticity of a real robot, Gaussian noise was added to the target discretization interval, $\Delta_t \sim \mathcal{N}(\Delta_\mu, 10 \text{ ms})$, with a hard minimum interval of 1 ms. We additionally included a 1% chance to sample the interval from $\mathcal{N}(1000 \text{ ms}, 10 \text{ ms})$ to simulate "catastrophic" events akin to communication errors. Of note, in less-exhaustive experiments not presented, such catastrophic events did not strongly impact or change the conclusions of the results.

Each agent's policy used a two-hidden-layer fully-connected network with *tanh* activations, with its output being treated as the mean of a Gaussian with an initial (bias unit) standard deviation of 1. We fixed $\gamma = 0.25$, which when using an interval of 40 ms, corresponds with $\gamma^{0.04} \approx 0.95$ per discrete time step. We considered target discretization intervals $\Delta_\mu \in \{40, 80, 120\}$ ms with a 4 second time limit and measured the episodic integral return. Averaged over 100 runs of 25 (simulation) minutes, Figure 6 shows parameter sensitivity curves and the best parameters' learning curves.



(a) Parameter Sensitivity ($\Delta_\mu = 40$ ms) (b) Parameter Sensitivity ($\Delta_\mu = 80$ ms)

(c) Parameter Sensitivity ($\Delta_\mu = 120$ ms) (d) Learning Curves ($\Delta_\mu = 120$ ms)

Figure 6: Servo Reacher results for REINFORCE using the discrete-time return (DTR) and right-point Riemann sum (RP), averaged over 100 runs. Shaded regions represent one standard error.

An initial observation is a systematic lag between the sensitivity curves of the two algorithms at low α. This is due to the return magnitudes being roughly proportional by a factor of $\mathbb{E}[\gamma^{\Delta_t}]$. If one absorbs this factor into the step-size, the right-point Riemann sum can be viewed as using a smaller *effective* α in the policy gradient update. Scaling the figure to use this effective α can be found to align the curves at low α. Nevertheless, we find that after accounting for this shift, REINFORCE with the right-point Riemann sum never performed worse and can significantly outperform the discrete-time return with both algorithms properly tuned. The right-point Riemann sum is seen to

improve with *increasing* Δ_μ, in line with the approximation error results in Section 5. Acknowledging that the two returns are roughly proportional by $\mathbb{E}[\gamma^{\Delta_t}]$, the results support that improvements are expected as this term deviates from 1 (i.e., decreasing γ or increasing Δ_μ).

7 Conclusions and Future Work

In this work, we identified and characterized an idiosyncrasy of time-discretization in reinforcement learning. Specifically, a nuance between the definitions of the discrete-time and continuous-time returns when viewing one as a discretization of the other. Our results suggest that when one does not have access to evaluating the integral return via options, one can better align the objectives by shifting the discount factor to begin discounting sooner. This provides *unification* in that the discrete-time return becomes a relatively straight-forward discretization of the integral return. We strongly emphasize the *simplicity* of the modification and how apart from the $\gamma = 0$ extreme, such a modification has no loss of generality in discrete-time or with fixed discretization intervals due to proportionality with the conventional discrete-time return. The returns are equivalent as $\gamma^\Delta \to 1$, but as γ^Δ deviates from 1, the right-point return is a better prediction target in terms of integral approximation error and improves control performance with *variable* time-discretization. Beyond integral approximation, the modification has intuitive appeal in that results from catastrophically long delays are attenuated in the return, rather than fully crediting an action for that outcome.

This work assumed that rewards better align with the subsequent time-step, as is often the case in the setups of existing continuous-time environments. However, should there be domain knowledge suggesting that an environment's rewards align with some other point in an interval, the ideas generalize in that discounting should be properly exponentiated to reflect this information.

Regarding avenues for future work, the integral approximation perspective suggests opportunity to explore return modifications corresponding to other integral approximation techniques. If one were to additionally track predecessor rewards, it opens up the possibility of interpolation-based approximations like the trapezoidal rule. Notably, Ayoub et al. (2024) concurrently considered trapezoidal approximations of the Monte Carlo return while exploring *when* to discretize. For the case of exponential discounting, however, we could further leverage that term's closed-form integral.

Acknowledgements

This research was generously supported by Amii, NSERC, Google Deepmind, and the Pan-Canadian AI Strategy managed by CIFAR. We would like to thank Forte Shinko, Alan Chan, and Sungsu Lim for insights and discussions contributing to the results in this paper, and the reviewers for valuable feedback during the review process.

References

Ayoub, A., Szepesvari, D., Zanini, F., Chan, B., Gupta, D., da Silva, B. C., and Schuurmans, D. (2024). Mitigating the curse of horizon in Monte-Carlo returns. *Reinforcement Learning Journal*.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Silver, D., and van Hasselt, H. (2017). Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*.

Bliss, G. A. (1914). A substitute for Duhamel's theorem. *Annals of Mathematics*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym. ArXiv:1606.01540.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*.

Farrahi, H. and Mahmood, A. R. (2023). Reducing the cost of cycle-time tuning for real-world policy optimization. In *Proceedings of the International Joint Conference on Neural Networks*.

Frémaux, N., Sprekeler, H., and Gerstner, W. (2013). Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLOS Computational Biology*.

Karimi, A., Jin, J., Luo, J., Mahmood, A. R., Jagersand, M., and Tosatto, S. (2023). Dynamic decision frequency with continuous options. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Kimura, H., Yamamura, M., and Kobayashi, S. (1995). Reinforcement learning by stochastic hill climbing on discounted reward. In *Proceedings of the International Conference on Machine Learning*.

Lee, J. and Sutton, R. S. (2021). Policy iterations for reinforcement learning problems in continuous time and space — fundamental theory and methods. *Automatica*.

Mahmood, A. R., Korenkevych, D., Komer, B. J., and Bergstra, J. (2018a). Setting up a reinforcement learning task with a real-world robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., and Bergstra, J. (2018b). Benchmarking reinforcement learning algorithms on real-world robots. In *Proceedings of the Conference on Robot Learning*.

Mehta, P. G. and Meyn, S. P. (2009). Q-learning and Pontryagin's minimum principle. In *Proceedings of the IEEE Conference on Decision and Control*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*.

Munos, R. (2006). Policy gradient in continuous time. *Journal of Machine Learning Research*.

Precup, D. (2000). *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst.

Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the International Conference on Machine Learning*.

Puterman, M. L. (1994). *Markov decision processes*. Wiley.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: an introduction*. MIT Press, 2nd edition.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*.

Tallec, C., Blier, L., and Ollivier, Y. (2019). Making deep Q-learning methods robust to time discretization. In *Proceedings of the International Conference on Machine Learning*.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

van Hasselt, H. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems*.

van Seijen, H., van Hasselt, H., Whiteson, S., and Wiering, M. A. (2009). A theoretical and empirical analysis of expected sarsa. In *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College.

White, M. (2016). Unifying task specification in reinforcement learning. ArXiv:1609.01995.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Zhang, Z., Kirschner, J., Zhang, J., Zanini, F., Ayoub, A., Dehghan, M., and Schuurmans, D. (2023). Managing temporal resolution in continuous value estimation: a fundamental trade-off. In *Advances in Neural Information Processing Systems*.

A Servo Reacher Environment Details

The environment state $\mathbf{x}$ is a column vector containing the DC motor's angular velocity [rad/s], the DC motor's current [A], the output shaft's angle [rad], the output shaft's angular velocity [rad/s], and the output shaft's target angle [rad], respectively. The state vector is updated as follows:

$$\dot{\mathbf{x}}_t \leftarrow \begin{bmatrix} -\frac{b_m}{J_m} & \frac{K_t}{J_m} & 0 & 0 & 0 \\ -\frac{K_t}{L_a} & -\frac{R_a}{L_a} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{b_m}{J_m N \eta} & \frac{K_t}{J_m N \eta} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} 0 \\ \frac{1}{L_a} \\ 0 \\ 0 \\ 0 \end{bmatrix} A_t$$

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \dot{\mathbf{x}}_t \Delta_s$$

where $\Delta_s = 10^{-4}$ [s] is the simulation discretization granularity, and A_t is an input voltage with built-in saturation limits of $\in [-12, 12]$ [V]. The output shaft angle is clamped $\in [-1.306, 1.306]$ [rad] in accordance with Mahmood et al. (2018b). The physical parameters used are detailed below:

L_a	Armature Inductance	2.05×10^{-3} [H]
R_a	Armature Resistance	8.29 [Ohm]
J_m	Rotor Inertia	8.67×10^{-8} [$\mathrm{kg} \cdot \mathrm{m}^2$]
b_m	Rotor Friction	8.87×10^{-8} [$\mathrm{N} \cdot \mathrm{m} \cdot \mathrm{s}$]
K_t	Torque Constant	0.0107 [$\frac{\mathrm{N \cdot m}}{\mathrm{A}}$]
N	Gear Ratio	200
η	Gear Efficiency	0.836

Given a target discretization interval $> 10^{-4}$ [s], the above updates are repeated until the target elapsed time is reached, keeping track of any overshoot and compensating accordingly in the next time interval. As a reinforcement learning environment, an agent observes the output shaft's angle, angular velocity, and target angle. The initial output shaft angle, θ_0, and target angle, θ_{target}, are uniformly sampled $\in [-1.306, 1.306]$ at the start of each episode, and an episode terminates when $|\theta_{t+1} - \theta_{target}| < 0.1$ [rad] with angular velocity $\dot{\theta}_{t+1} < 0.1$ [rad/s]. An agent provides a continuous-valued action as a voltage, and receives a reward $|\theta_{t+1} - \theta_{target}|$, computed and received jointly with the next observation.