Learning HJB Viscosity Solutions with PINNs for Optimal Control and Continuous-Time Reinforcement Learning

***Abstract*—This internship explores Continuous Time Reinforcement Learning (CTRL) for control tasks that require high-frequency or asynchronous decision-making, such as autonomous driving and high-frequency trading. Unlike Discrete Time Reinforcement Learning (DTRL), which relies on discrete decision intervals, CTRL models system dynamics using Partial Differential Equations (PDEs) or Stochastic Differential Equations (SDEs) and solves the Hamilton-Jacobi-Bellman (HJB) equation to estimate optimal policies. The objective is to address the limitations of DTRL in continuous environments and improve control performance. However, the research also highlights key challenges, including increased computational complexity, the requirement for accurate dynamic models, and difficulties in exploration.**

***Index Terms*—Deep Learning, Reinforcement Learning, PINNs, HJB**

I. Contacts and practicalities

- **Supervisors:** Alena Shilova, Nilo Schwencke
- **Laboratory:** TAU team, LISN—Université Paris Saclay

This internship will take place at LISN Laboratory at Paris Saclay University. This internship will be paid for in accordance with current legislation.

A. Application Process

Interested candidates are encouraged to apply with a CV and a brief statement of motivation sent to supervisors with the email object **[HJB Internship application]**. Applications will be reviewed on a rolling basis with interviews until the position is filled.

II. Context

A. Optimal Control and Continuous-time Reinforcement Learning

Reinforcement Learning in the recent years has attracted a lot of attention. Deep RL managed to beat human or even expert performance in such tasks as atari games [1] and GO [2]. Even more, RL has recently demonstrated its potential in more complex tasks, such as robotics [3], controlling plasma fusion in tokamaks [4], etc. The main objective of RL is to train an agent and its behavioral policy that would suggest optimal actions based on the state of the environment the agent is in.

Before that, the similar problems of finding optimal behavioural policies for dynamical systems has been a subject of study in optimal control [5]. Optimal control has been used in many domains and applications, for example robotics, navigation, finances, etc. While RL and Optimal Control share the same goal, the settings and approaches remain mostly different. One of such things is the discrete-time vs continuous-time settings. Usually, RL assumes that the time is already discretized, which allows us to formulate the problem as a Markov Decision Problem (MDP, *cf.* [6]). In this regard, Optimal Control has an advantage of considering more general setting of continuous time, thus introducing methods agnostic of time discretization. And indeed, there are some problems for which it is necessary to take decisions at the arbitrary moments of time or at high frequency, e.g. high frequency stock trading, autonomous driving and snowboard riding. Therefore, more and more attention is drawn toward Continuous Time Reinforcement Learning (CTRL, , *cf.* [7]), which is at the intersection of RL and Optimal Control.

In the context of continuous time, the dynamics of the system are expressed as a Partial Differential Equation (PDE) for deterministic environments and Stochastic Differential Equation (SDE) for stochastic environments, instead of relying on MDPs. The value function (a useful measure to estimate the quality of a policy of actions) can be found from Hamiltonian-Jacobi-Bellman (HJB) equation that replaces Bellman equation in discrete time. The HJB equation is a PDE that formulates the problem of finding the optimal control for a dynamical system, which is studied and applicable in both Optimal Control and CTRL. Despite being an essential PDE, there does not exist many methods that can solve it in the general case, especially when considering high-dimensional problems. Even more challenging is that the HJB equation may have numerous generalized solutions (weak solutions that are not differentiable everywhere), one of which defines an optimal control, but also many others that end up in suboptimal behaviors of dynamical systems. While there exists some numerical methods (Finite Difference, or Finite Element Method, *cf.* [8]) able to distinguish and thus retrieve an optimal control solution, they typically do not scale to a dimension higher than 6. Thus, to scale further HJB-based methods in Optimal Control and CTRL, we need to consider other PDE solvers that would scale more easily to higher dimensions.

B. Solving optimal control with physics informed neural networks

The area of AI4Science or also known Scientific Machine Learning develops with extremely fast speed in the last years [9]. It promises to overpass the classical numerical methods in solving many scientific problems from physics, chemistry, biology and other fields. This new area introduces new methods

powered by neural networks that can be applied to high-dimensional problems that might be intractable by classical solvers like finite difference or finite element methods.

One example of such novel algorithm is to use physics informed neural networks (PINNs, *cf.* [10]) to solve partial differential equations (PDE). To solve a PDE, PINNs formulate the problem as an optimization problem, where the objective is to minimize the loss function that comprises a PDE equation together with boundary conditions that act in the form of regularizers. This loss function is minimized using the collocation points that are sampled inside the domain on which the PDE is defined. While, a lot of research has appeared in the recent years that prescribes how to train PINNs, a lot of challenges still remain when considering high dimensional problems. What neural architecture to use? How to sample collocation points inside the domain? How to combine different PDE and boundary equations inside a loss function? What optimizer to use? All those choices can affect a lot the training and eventually the final performance. Moreover, there is no ultimate recipe that would suit any problem.

A recent work [11] has explored a potential of using PINNs to solve HJB equations. To address the non-uniqueness of solutions, they have proposed a curriculum learning like approach to find a well-regularized solution, a viscosity solution, that can be proven to correspond to the optimal control. To be more precise, instead of solving a single PDE, they suggest solving a series of non-linear PDEs parametrized with ε, such PDEs have unique smooth solutions for $\varepsilon > 0$ and at the limit $\varepsilon \to 0$ we recover the viscosity solution and thus a solution that would return an optimal control. While the work showed empirically that PINNs can be indeed used to find an optimal control for classical control problems, such as Pendulum, Cartpole and Acrobot, current learning choices like uniform sampling of collocation points, using Adam as an optimizer and Multi-Layer Perceptron as a neural architecture may prevent it from scaling it to even more complex problems.

This internship proposes to improve the current ε -HJBPINNs algorithm by considering different learning choices. In particular, how to use Anagram [12] to improve the training error, while having a challenging setting of solving a series of non-linear parametric PDEs. What other sampling methods can improve the learning, for example a potential candidate can be an adaptation of PINNACLE adaptive sampling method [13] to Natural Neural Tangent Kernel [12]. Or considering an attention-based neural architecture [14], [15] that can better to generalize to all PDE equations considered in ε-HJBPINNs.

III. Proposal

A. Tasks and Responsibilities

- Understanding existing methods in solving HJB equations (Finite Difference, Finite Element Method [8], ε -HJBPINNs).
- Understanding and implementing different methods in PINNs: Anagram [12], PINNACLE [13].
- Improving existing ε-HJBPINNs in different directions: learning of viscosity solution (consider different optimizers, neural architectures), sampling techniques, stability of the methods.
- Testing them on high-dimensional tasks from MuJoCo [16].

B. Skills Required

- Knowledge of machine learning, deep learning, Python, and Pytorch or JAX.
- Excellent algorithmic skills.
- Autonomy and curiosity.
- *Optional (but very welcomed):* Being familiar with HJB, PINNs, PDEs.

C. Benefits of the Internship

- Getting familiar with cutting edge research topics.
- Gain hands-on experience in advanced machine learning techniques in a specialized team.
- Opportunity to publish and collaborate on research.

References

[1] V. Mnih, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] D. Silver *et al.*, "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots That Can Adapt like Animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[4] J. Degrave *et al.*, "Magnetic Control of Tokamak Plasmas through Deep Reinforcement Learning," *Nature*, vol. 602, no. 7897, pp. 414–419, Feb. 2022, doi: 10.1038/s41586-021-04301-9.

[5] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[7] K. Doya, "Reinforcement Learning in Continuous Time and Space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[8] R. Munos, "A Study of Reinforcement Learning in the Continuous Case by the Means of Viscosity Solutions," *Machine Learning*, vol. 40, pp. 265–299, 2000, doi: 10.1023/A:1007686309208.

[9] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning through Physics-Informed Neural Networks: Where We Are and What's Next," no. arXiv:2201.05624. arXiv, Jun. 2022. doi: 10.48550/arXiv.2201.05624.

[10] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.

[11] A. Shilova, T. Delliaux, P. Preux, and B. Raffin, "Learning HJB Viscosity Solutions with Pinns for Continuous-Time Reinforcement Learning," 2024.

[12] Anonymous, "ANaGRAM: A Natural Gradient Relative to Adapted Model for efficient PINNs learning," in *Submitted to The Thirteenth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=o1IiiNIoaA

[13] G. K. R. Lau, A. Hemachandra, S.-K. Ng, and B. K. H. Low, "PINNACLE: PINN Adaptive ColLocation and Experimental Points Selection," in *The Twelfth International Conference on Learning Representations*,

[14] A. Vaswani, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, 2017.

[15] E. J. Hu *et al.*, "Lora: Low-rank Adaptation of Large Language Models," *arXiv preprint arXiv:2106.09685*, 2021.

[16] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A Physics Engine for Model-Based Control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.