Langevin Algorithms for Markovian Neural Networks and Deep Stochastic Control

Pierre BRAS and Gilles PAGÈS, presented by Pierre BRAS

Laboratoire de Probabilités, Statistique et Modélisation
Sorbonne Université, Paris, France

Presented at the International Joint Conference on Neural Networks 2023
Gold Coast Convention and Exhibition Centre, Queensland, Australia

Acknowledgements

We thank the IEEE Computational Intelligence Society (CIS) for supporting the participation to the conference IJCNN 2023 with the IEEE CIS Conference Travel Grant for Students.

IEEE Computational Intelligence Society

Table of Contents

We consider the following **Stochastic Optimal Control** (SOC) problem associated with a **Stochastic Differential Equation** (SDE):

$$\min_u J(u) := \mathbb{E}\left[\int_0^T G(X_t)dt + F(X_T)\right], \tag{1}$$

$$dX_t = b(X_t, u_t)dt + \sigma(X_t, u_t)dW_t, \;\; t \in [0, T] \tag{2}$$

- X_t: trajectory vector
- u_t: control vector
- $b(X_t, u_t)$: controlled drift vector
- $\sigma(X_t, u_t)$: controlled diffusion matrix
- W_t: Brownian motion (white noise process)

$\implies$ Optimize a functional of a trajectory of a SDE X_t through the control u_t, including a random noise that affects the evolution of the system.

Example: Resource Management

An oil drilling company has to balance the costs of extraction and of storage of oil in a volatile energy market:

- **Trajectory:** Volatile global oil price and quantity of stored (unsold) oil for the company
- **Control:** Quantities of instantaneously extracted, stored and sold oil

Figure: Offshore oil rig - Source: Unsplash

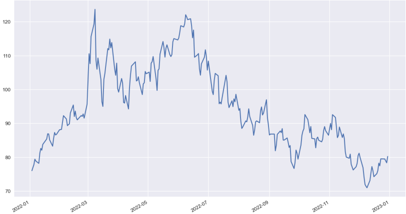


Figure: Crude oil price during the year 2022

Euler-Maruyama scheme

$$\min_{\theta} \bar{J}(\bar{u}_\theta) := \mathbb{E}\Big[\sum_{k=0}^{N-1}(t_{k+1}-t_k)G(\bar{X}^\theta_{t_{k+1}}) + F(\bar{X}^\theta_{t_N})\Big], \tag{3}$$

$$\begin{aligned}\bar{X}^\theta_{t_{k+1}} &= \bar{X}^\theta_{t_k} + (t_{k+1}-t_k)b(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k})) \\ &\quad + \sqrt{t_{k+1}-t_k}\,\sigma(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k}))\xi_{k+1},\end{aligned} \tag{4}$$

$\xi_k \sim \mathcal{N}(0, I_{d_2})$ i.i.d.

- **Time discretization** of $[0, T]$:

$$t_k := kT/N, \ k \in \{0, \dots, N\}, \quad h := T/N$$

- **Control** u with **parameter** θ using either one time-dependant neural network either N distinct neural networks: $u_{t_k} = \bar{u}_\theta(t_k, X_{t_k})$ or $u_{t_k} = \bar{u}_{\theta^k}(X_{t_k})$
- Since the process is **Markovian**, we assume the control depends only on the running position X_t (instead of the whole previous trajectory $(X_s)_{s\in[0,t]}$).

The parameter θ is optimized by **gradient descent**:

- Simulate batches of trajectories $\bar{X}$ depending on the Brownian motion.
- Compute $\nabla_\theta \bar{J} = \nabla_\theta \bar{J}(\bar{u}_{\theta_n}, (\xi_k^{i,n+1})_{1 \le k \le N})$; the gradient is computed by automatic differentiation as the gradient w.r.t. to θ is tracked all along the trajectory of the numerical scheme Giles and Glasserman (2005); Giles (2007)

In the literature:
SOCs are solved using specific techniques: Forward-Backward SDEs, Hamilton-Jacobi-Bellman (HJB) optimality conditions, stochastic dynamic programming. The resolution of SOCs by neural networks scales to the high dimension, contrary to dynamic programming Gobet and Munos (2005); Han and Weinan (2016); Bachouch et al. (2022); Laurière et al. (2023).

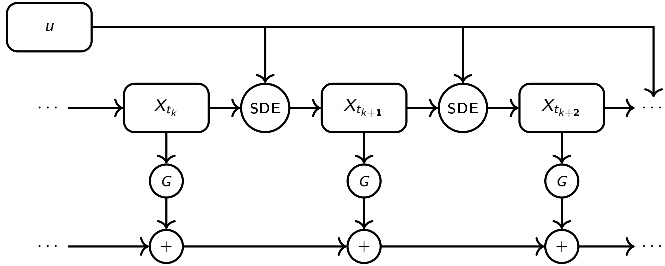


Figure: Markovian Neural Network with one control.

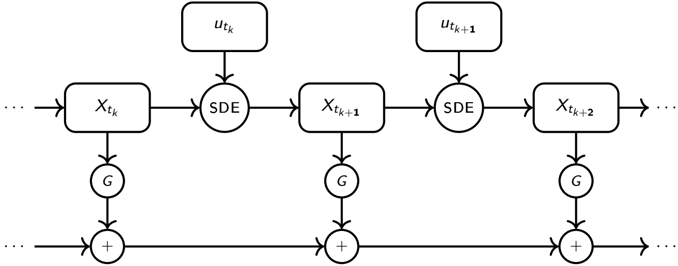


Figure: Markovian neural network with one control for every time step.

- If the control is applied at many discretization times, then the **Markovian Neural Network** becomes a **very deep** neural network, difficult to train directly.
- Adding noise during training is known to improve the learning procedure Neelakantan et al. (2015); Anirudh Bhardwaj (2019):

Gradient Langevin Algorithm

For some choice of **Preconditioner** rule P (Adam, RMSprop...), step size γ_{n+1} and and computed gradient g_{n+1}:

$$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1} + \sigma_{n+1}\sqrt{\gamma_{n+1}}\mathcal{N}(0, P_{n+1}) \tag{5}$$

$\Longrightarrow$ per-dimension adaptive noise rate.

- Bras (2022): the deeper the network is, the greater are the gains provided by Langevin algorithms; introduces the **Layer Langevin** algorithm, consisting in adding Langevin noise only to the deepest layers.

$\Longrightarrow$ Analysis was conducted especially for deep architectures in **image classification**.

Our objectives

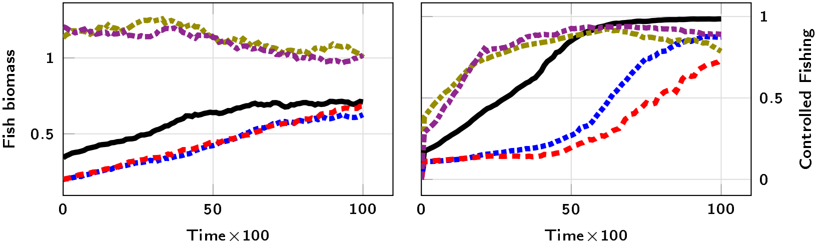
- Side-by-side comparison of non-Langevin/Langevin optimizers on different SOC problems: fishing quotas, financial hedging, energy management.
- If using multiple controls (second case), explore the benefits of Layer-Langevin.

Fish biomass $X_t \in \mathbb{R}^{d_1}$ with:

- Inter-species interaction κX_t
- Fishing following imposed quotas u_t
- Objective: keep X_t close to an ideal state $\mathcal{X}_t$.

Figure: Source: Unsplash

$$dX_t = X_t * ((r - u_t - \kappa X_t)dt + \eta dW_t)$$
$$J(u) = \mathbb{E}\left[\int_0^T (|X_t - \mathcal{X}_t|^2 - \langle \alpha, u_t \rangle)dt + \beta[u]^{0,T}\right]$$



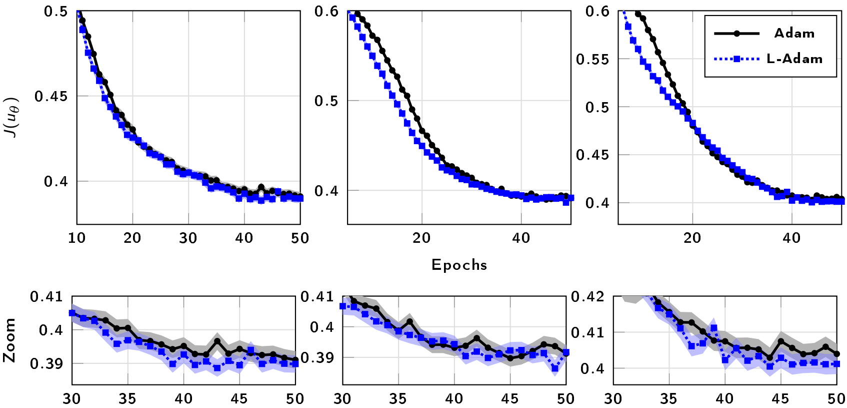


Figure: Comparison of Adam et L-Adam algorithms during the training for the fishing control problem with $N = 20, 50, 100$ respectively. J is estimated over 50×512 trajectories. A zoom on the last epochs is given.

Table: Best performance

	$N = 20$	$N = 50$	$N = 100$
Adam	0.3910	0.3912	0.4029
L-Adam	0.3886	0.3864	0.4011

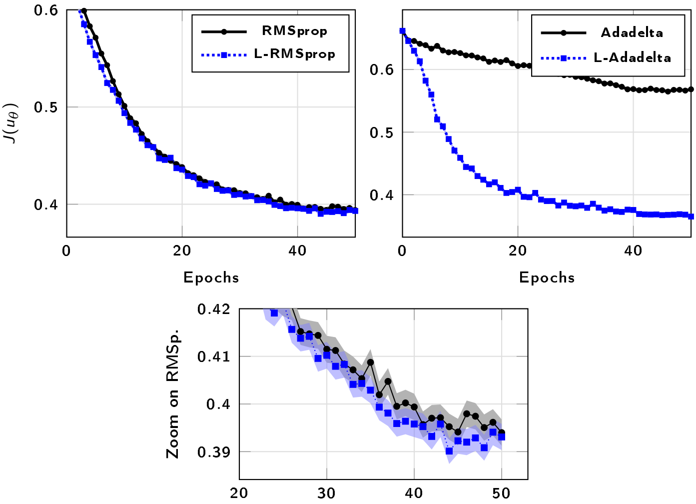


Figure: Comparison of Langevin algorithms with their non-Langevin counterparts during the training for the fishing control problem with $N = 50$.

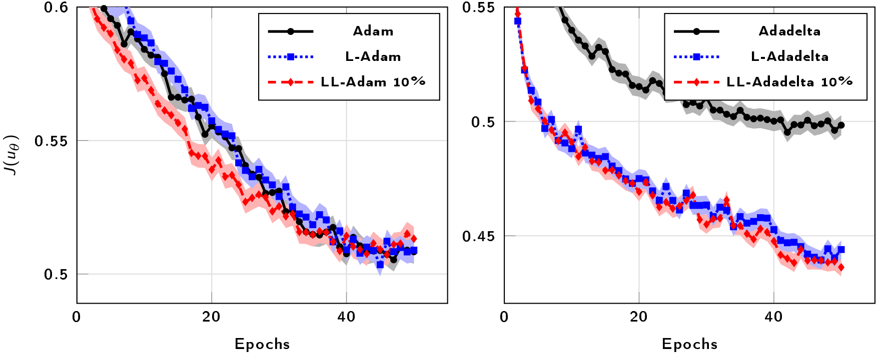


Figure: Training of the fishing problem with multiple controls with $N = 10$

We aim to replicate some payoff Z defined on some portfolio S_t by trading some of the assets with transaction costs; the control u_t is the amount of held assets. The objective is

Figure: Source: Unsplash

$$J(u) = \nu\left(-Z + \sum_{k=0}^{N-1} \langle u_{t_k}, S_{t_{k+1}} - S_{t_k} \rangle - \sum_{k=0}^{N} \langle c_{tr}, S_{t_k} * |u_{t_k} - u_{t_{k-1}}| \rangle\right) \quad (6)$$

where ν is a convex risk measure. We consider the assets S_t to be follow a Heston model and are tradable along with variance swap options.

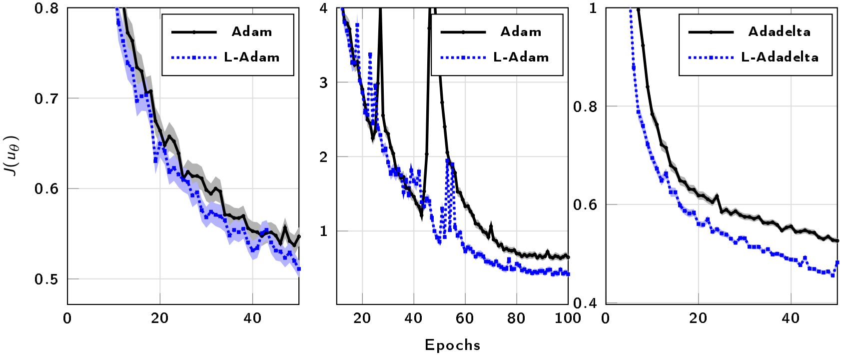


Figure: Comparison of algorithms during the training for the deep hedging control problem with $N = 30, 50, 50$ respectively

Table: Best performance

	Adam, $N = 30$	Adam, $N = 50$	Adadelta, $N = 50$
Vanilla	0.4448	0.6355	0.4671
Langevin	0.4306	0.4182	0.3773

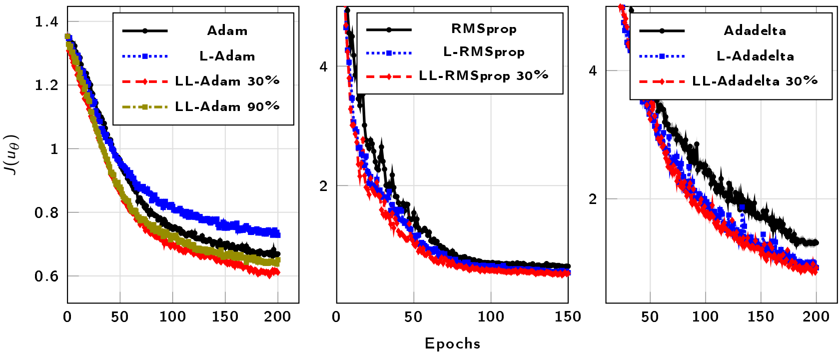


Figure: Training of the deep hedging problem with multiple controls with $N = 10$

Table: Best performance

	Adam	RMSprop	Adadelta
Vanilla	0.6626	0.5618	1.2900
Langevin	0.7278	0.4441	0.9250
Layer Langevin 30%	0.6004	0.4102	0.8554
Layer Langevin 90%	0.6377	–	–

An oil driller has to balance the costs of extraction E_t, storage S_t in a volatile energy market with oil price P_t:

$$dP_t = \mu P_t dt + \eta P_t dW_t$$

$$J(q) = -\mathbb{E}\left[\int_0^T e^{-\rho r} U\Big(q_r^v P_r + q_r^{v,s}(1-\varepsilon)P_r - (q_r^v + q_r^s)c_e(E_r) - c_s(S_r)\Big)dr\right],$$

$$E_t = \int_0^t (q_r^v + q_r^s)dr, \quad S_t = \int_0^t (q_r^s - q_r^{v,s})dr$$

where U is the utility function and $q_t = (q_t^v, q_t^s, q_t^{v,s})$ is the control (extracted, stored, sold from storage).

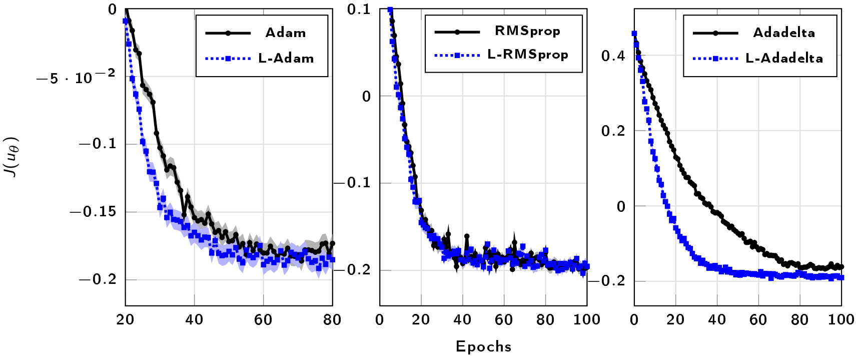


Figure: Comparison of algorithms during the training for the oil drilling control problem with $N = 50$

Table: Best performance

	Adam	RMSprop	Adadelta
Vanilla	-0.1729	-0.1985	-0.1649
Langevin	-0.1915	-0.2032	-0.1929

Conclusion

- In various problems, Langevin and Layer Langevin algorithms show improvements in comparison with their respective non-Langevin counterparts.
- Gains depend on the setting and optimizer; we observe that gains are limited or null for the RMSprop algorithm.
- For SOC with multiple controls, we proved the gains of Layer Langevin algorithms with a small number of layers ($\sim$10%-30%).

Thank you for your attention !

C. Anirudh Bhardwaj. Adaptively Preconditioned Stochastic Gradient Langevin Dynamics. *arXiv e-prints*, art. arXiv:1906.04324, June 2019.

A. Bachouch, C. Huré, N. Langrené, and H. Pham. Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodol. Comput. Appl. Probab.*, 24(1): 143–178, 2022. ISSN 1387-5841. doi: 10.1007/s11009-019-09767-9. URL https://doi.org/10.1007/s11009-019-09767-9.

P. Bras. Langevin algorithms for very deep Neural Networks with application to image classification. *arXiv e-prints*, art. arXiv:2212.14718, Dec. 2022.

H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quant. Finance*, 19(8):1271–1291, 2019. ISSN 1469-7688. doi: 10.1080/14697688.2019.1571683. URL https://doi.org/10.1080/14697688.2019.1571683.

M. Gaïgi, S. Goutte, I. Kharroubi, and T. Lim. Optimal risk management problem of natural resources: application to oil drilling. *Ann. Oper. Res.*, 297(1-2):147–166, 2021. ISSN 0254-5330. doi: 10.1007/s10479-019-03303-1. URL https://doi.org/10.1007/s10479-019-03303-1.

M. B. Giles. Monte Carlo evaluation of sensitivities in computational finance. Technical Report NA07/12, Oxford University Computing Laboratory, 2007.

M. B. Giles and P. Glasserman. Smoking adjoints: fast evaluation of Greeks in Monte Carlo calculations. Technical Report NA05/15, Oxford University Computing Laboratory, 2005.

E. Gobet and R. Munos. Sensitivity analysis using Itô-Malliavin calculus and martingales, and application to stochastic optimal control. *SIAM J. Control Optim.*, 43(5):1676–1713, 2005. ISSN 0363-0129. doi: 10.1137/S0363012902419059. URL https://doi.org/10.1137/S0363012902419059.

S. Goutte, I. Kharroubi, and T. Lim. Optimal management of an oil exploitation. *International Journal of Global Energy Issues*, 41(1/2/3/4):69–85, 2018.

J. Han and W. Weinan. Deep Learning Approximation for Stochastic Control Problems. *Deep Reinforcement Learning Workshop, NIPS (2016)*, Nov. 2016.

M. Laurière, G. Pagès, and O. Pironneau. Performance of a Markovian Neural Network versus dynamic programming on a fishing control problem. *Probability, Uncertainty and Quantitative Risk*, pages –, 2023. ISSN 2095-9672. doi: 10.3934/puqr.2023006. URL /article/id/63c741a4b5351f4889aff727.

A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding Gradient Noise Improves Learning for Very Deep Networks. *arXiv e-prints*, art. arXiv:1511.06807, Nov. 2015.