Langevin Algorithms for Markovian Neural Networks and Deep Stochastic Control

Pierre BRAS and Gilles PAGÈS, presented by Pierre BRAS

Sorbonne Université, Paris, France

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

Abstract

Stochastic Gradient Descent Langevin Dynamics (SGLD) algorithms, which add noise to the classic gradient descent, are known to improve the training of neural networks in some cases where the neural network is very deep. In this paper we study the possibilities of training acceleration for the numerical resolution of stochastic control problems through gradient descent, where the control is parametrized by a neural network. If the control is applied at many discretization times then solving the stochastic control problem reduces to minimizing the loss of a very deep neural network. We numerically show that Langevin and Layer-Langevin algorithms improve the training on various stochastic control problems like hedging and resource management, and for different choices of gradient descent methods.

Stochastic Optimal Control through Gradient Descent

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

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

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

where X_t is the **trajectory vector**, u_t is the **control vector**, $b(X_t, u_t)$ is the **controlled drift vector**, $\sigma(X_t, u_t)$ is the **controlled diffusion matrix** and W_t is a Brownian motion. We aim to 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.

The corresponding **Euler-Maruyama** numerical scheme is given by:

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

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

$\xi_k \sim \mathcal{N}(0, I_{d_2})$ 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 E (2016); Bachouch et al. (2022); Laurière et al. (2023).

Training very deep neural networks

- 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). For some choice of **Preconditioner** rule P (Adam, RMSprop...), the **preconditioned Gradient Langevin** algorithm reads:

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

- 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.

The analysis was conducted especially for deep architectures in **image classification**.

Objectives :

- **Side-by-side comparison of non-Langevin/Langevin optimizers** on different SOC problems: fishing quotas, financial hedging, energy management.
- If using multiple control networks, we explore the benefits of Layer-Langevin.

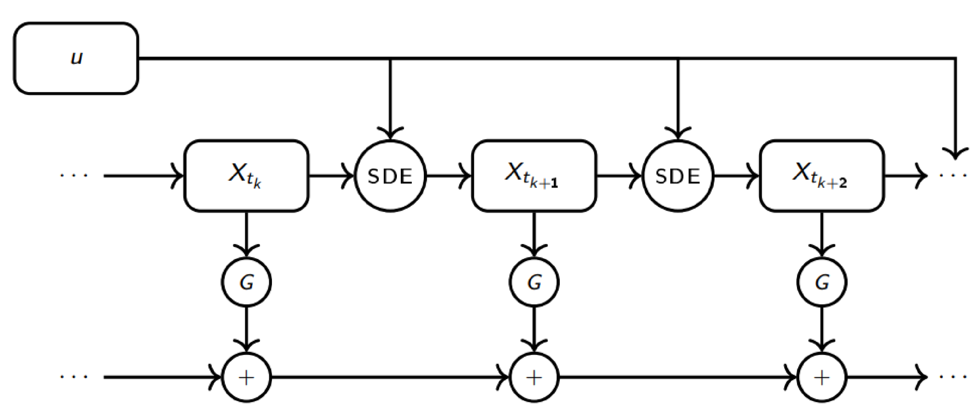

Figure 1. Markovian neural network with single control network

Simulations on three different SOC models

Fishing quotas Laurière et al. (2023): A fish biomass $X_t \in \mathbb{R}^{d_1}$ evolves with inter-species interaction κX_t and with controlled fishing u_t. The objective is to keep X_t close to some ideal state $\mathcal{X}_t$, reading:

$$dX_t = X_t * ((r - u_t - \kappa X_t)dt + \eta dW_t),$$

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

Deep financial hedging Buehler et al. (2019): We aim to replicate some payoff Z defined on a 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

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

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

Resource Management and Oil Driling Goutte et al. (2018); Gaïgi et al. (2021): 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:

$$dP_t = \mu P_t dt + \eta P_t dW_t, \quad E_t = \int_0^t (q_r^v + q_r^s)dr, \quad S_t = \int_0^t (q_r^s - q_r^{v,s})dr,$$

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

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).

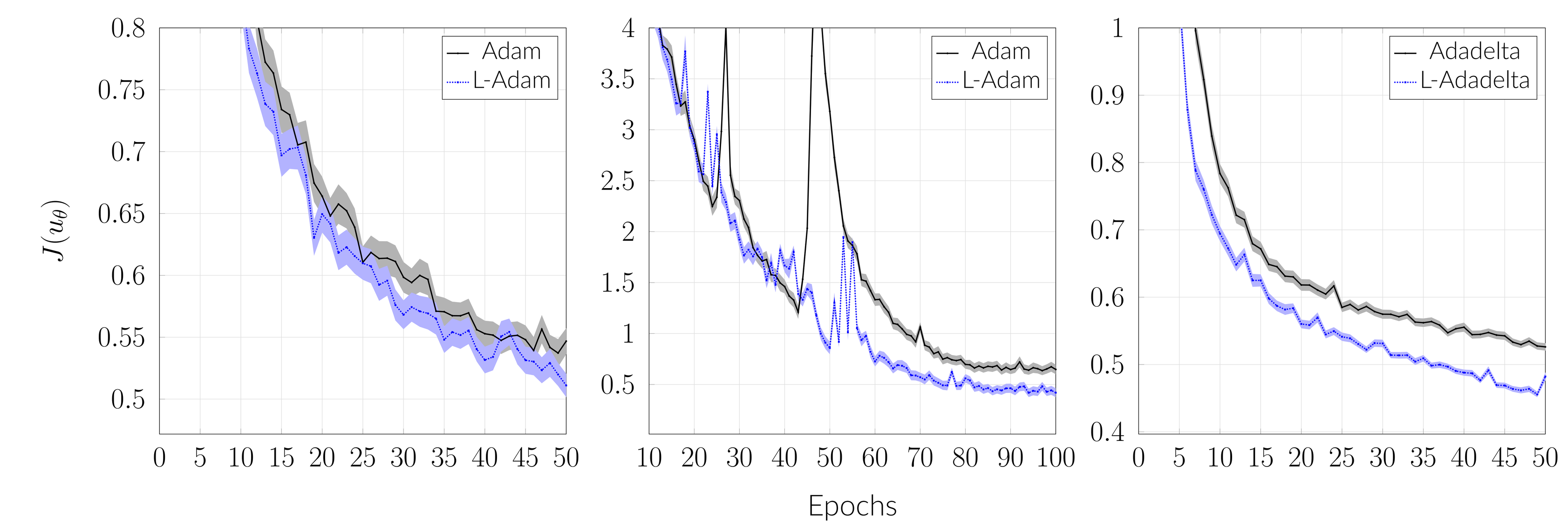

Figure 2. Comparison of algorithms during the training for the deep hedging control problem with $N = 30, 50, 50$ respectively

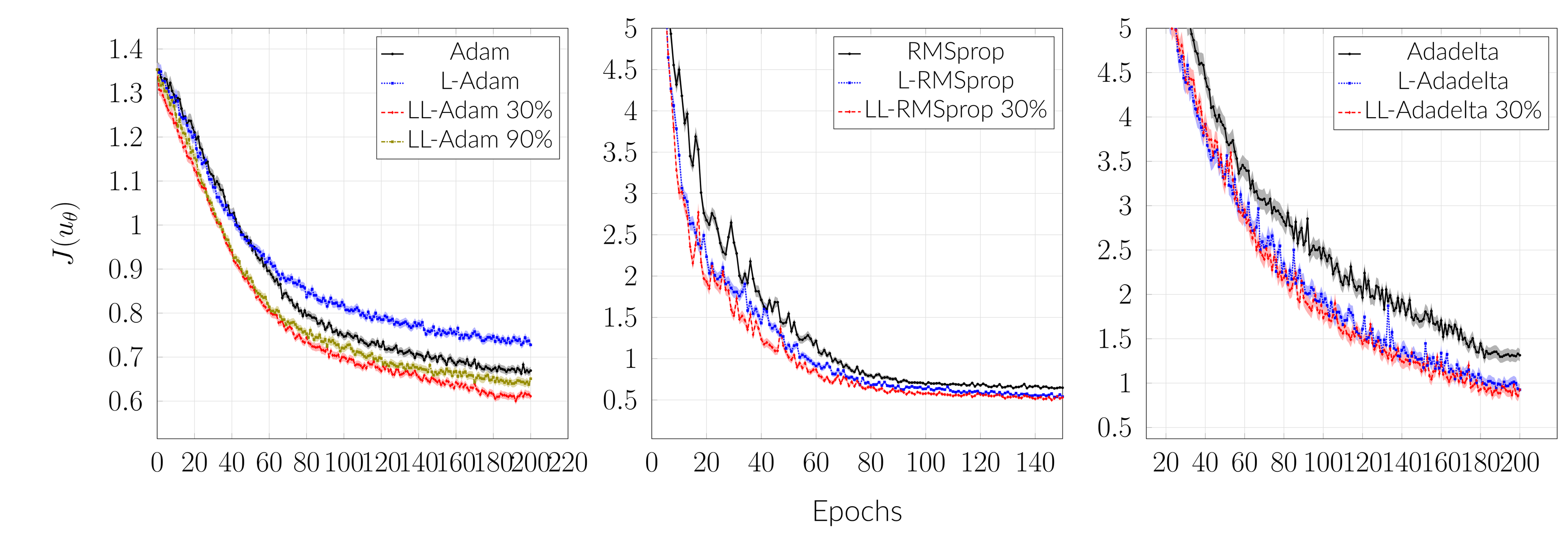

Figure 3. Training of the deep hedging problem with multiple control networks with $N = 10$

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 more limited for the RMSprop algorithm.
- For SOC with multiple control networks, we proved the benefits of Layer Langevin algorithms with a small number of layers (∼10%-30%).

pierre.bras@sorbonne-universite.fr