Dynamic programming: Gobet and Munos (2005); Han and E (2016); Bachouch et al. (2020). The resolution of SOCs by neural networks scales to the high dimension, contrary to Hamilton-Jacobi-Bellman (HJB) optimality conditions, stochastic dynamic programming.

In the literature:

\[ \theta \]

Euler-Maruyama

The corresponding random noise that affects the evolution of the system.

\[ X_t \]

to optimize a functional of a trajectory of a SDE

Stochastic Differential Equation

We consider the following that Langevin and Layer-Langevin algorithms improve the training on various stochastic 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):

\[
\min_{u} J(u) = \mathbb{E} \left[ \int_{0}^{T} G(X_t) dt + F(X_T) \right].
\]

(1)

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

(2)

where \( X_t \) is the trajectory vector, \( u_t \) is the control vector, \( b(X_t, u_t) \) is the controlled drift vector, \( \sigma(X_t) \) 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:

\[
\min_{\theta} J(\theta) = \mathbb{E} \left[ \sum_{k=0}^{N} r_k G(X_{t_k}) + F(X_T) \right] + \frac{1}{2} \sum_{k=0}^{N} \sigma_k^2 \mathbb{E} \left[ \left( \frac{dX_{t_k}}{dt} \right)^2 \right].
\]

(3)

\[
X_{t_k} = X_{t_{k-1}} + (t_k - t_{k-1})b(X_{t_{k-1}}, u_{t_{k-1}}) + \sigma(X_{t_{k-1}}, u_{t_{k-1}}) \sqrt{t_k - t_{k-1}} + \epsilon_{t_k}
\]

(4)

\[ \epsilon_t \sim \mathcal{N}(0, \sigma^2). \]

- Time discretization of \([0, T]; \quad t_k = kT/N, \; k \in \{0, \ldots, N\}; \quad T = RN\).
- Control \( u \) with parameter \( \theta \) using either one time-dependent neural network either \( N \) distinct neural networks: \( u_{t_k} = u_{\theta}(X_{t_{k-1}}) \) or \( u_{t_k} = u_{\theta}(X_{t_{k-1}}) \).
- Since the process is Markovian, we assume the control depends only on the running position \( X_t \) (instead of the whole previous trajectory \( X_{[0,t]} \)).

The parameter \( \theta \) is optimized by gradient descent:

- Simulate batches of trajectories \( \tilde{X}_t \) depending on the Brownian motion.
- Compute \( \nabla \theta J(\theta) = \sum_{i=1}^{N} \left( \frac{d}{dt} G(X_i(t)) + \frac{d}{dt} F(X_T) \right) \)

The gradient is computed by automatic differentiation as the gradient w.r.t. to \( \theta \) 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. (2020); Lauroir 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 \( F \) (Adam, RMSProp...), the preconditioned Gradient Langevin algorithm reads:

\[
a_{t_k} = a_{t_{k-1}} - \eta \nabla_{u_t} J(\theta_t) + \sigma(\tilde{u}_{t_k}) \sigma(\bar{u}_{t_k}) \mathcal{N}(0, \sigma^2).
\]

(5)

- Bras (2022): the deeper the network, 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.

**Simulations on three different SOC models**

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

\[
dX_t = X_t \left( \left( r - u_t - \kappa X_t \right) dt + \sigma dW_t \right).
\]

\[
J(\theta) = \mathbb{E} \left[ \int_{0}^{T} \left( r X_t^2 - (\kappa u_t + \sigma^2) \right) dt \right] + \frac{1}{2} \sum_{k=1}^{N} \left( u_{t_k} - u_{t_{k-1}} \right)^2.
\]

Deep financial hedging Bucher 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:

\[
J(\theta) = \mathbb{E} \left[ \left( -Z + \sum_{k=1}^{N} (u_{t_k} - S_{t_k}) + \sum_{k=1}^{N} (u_{t_k} - S_{t_k}) \right)^2 \right]
\]

(6)

where \( Z \) 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 Drilling Gouette et al. (2018); Galí 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 \( P_t \).

\[
dP_t = \mu dt + \sigma dW_t,
\]

\[
E_t = \int_{0}^{T} (q_t^s + q_t^d) dt,
\]

\[
S_t = \int_{0}^{T} (q_t^s - q_t^d) dt.
\]

(7)

where \( E(t) \) is the utility function and \( q_t = (q_t^s, q_t^d) \) is the control (extracted, stored, sold from storage).