Expected Plots from Lab session of Deep Learning for PDE

Huyên Pham and Samy Mekkaoui

CMAP, Ecole Polytechnique

July 25, 2025

Main topic of the lecture

Objectives :

- Present some applications of the *Deep Galerkin* Algorithm and *Deep BSDE* Solver for solving *PDE*.
- Show how both of theses algorithms can be efficiently implemented in Python with PyTorch.

Contents

A reminder on the Deep Galerkin Algorithm
Mathematical Foundations

We are giving to present some applications of the Deep Galerkin algorithm for solving PDE in a general form :

$$\partial_v(t,x) = \mathcal{H}[v](t,x) \quad (t,x) \in [0,T) \times \mathbb{R}^d \tag{1}$$

$$v(T,x) = g(x), \quad x \in \mathbb{R}^d \tag{2}$$

where $\mathcal{H}$ is an operator which can contain multiples derivatives of v with respect to x. Given a smooth function ω on $[0,T] \times \mathbb{R}^d$, we define :

$$\mathbb{L}(\omega) = \mathbb{E}[|\omega(T,\mathcal{X}) - g(\mathcal{X})|^2] + \mathbb{E}[|\partial_t\omega(\tau,\mathcal{X}) - \mathcal{H}[\omega](\tau,\mathcal{X})|^2] \tag{3}$$

where $(\tau, \mathcal{X})$ are independant random variables $\sim v_T \otimes v_d$ supported on $[0,T] \times \mathbb{R}^d$.

Remark

From the definition of v, it is clear that v is a solution to (1) if and only if v achieves the minimum of $\mathbb{L}$. However, optimization problem (3) is infinite dimensional and not feasible numerically.

A reminder on the Deep Galerkin Algorithm
Mathematical Foundations

The idea is therefore to parametrize the space of smooth functions ω by the class of neural networks $\mathcal{U}_\theta$ on $[0, T] \times \mathbb{R}^d$ which reduces to a finite dimensional optimization problem :

$$\inf_\theta \mathbb{L}(\mathcal{U}_\theta) \tag{4}$$

Remark

- *The optimization problem (4) is done through stochastic gradient descent based on the expectation representation* $\mathbb{L}(\mathcal{U}_\theta) = \mathbb{E}\big[I(\theta, \tau, \mathcal{X})\big]$ *with*

$$I(\theta, t, x) = |\mathcal{U}_\theta(T, x) - g(x)|^2 + |\partial_t \mathcal{U}_\theta(t, x) - \mathcal{H}[\mathcal{U}_\theta](t, x)|^2$$

- *Equation (4) is finite dimensionnal because* θ *represents the values of the weights and the bias of the NN which are finite dimensionnal and optimized during the training process.*

An application in option pricing
Application in the $B - S$ model

We will do some experiments of the Deep Galerkin for the $B - S$ in dimension $d = 1$ for different type of PDE. Under option pricing theory, for an European option with price at time t denoted by $C(t, S_t)$ we know that we have the following PDE representation for the option price C defined on $[0, T] \times \mathbb{R}_*^+$ as :

$$\partial_t C + \mathcal{L}C - rC = 0, \quad (t, x) \in [0, T) \times \mathbb{R}_*^+ \tag{5}$$
$$C(T, x) = g(x), \quad x \in \mathbb{R}_*^+$$

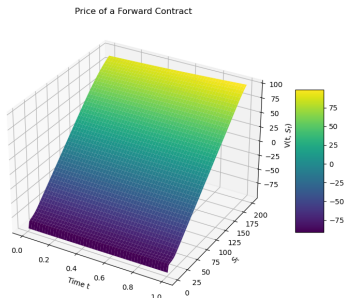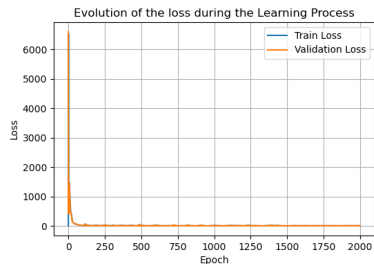where the infinitemisal generator is given by :

$$\mathcal{L}v(t, x) = rx\partial_x v(t, x) + \frac{1}{2}\sigma^2 x^2 \partial_x^2 v(t, x)$$

Remark

For the numerical experiments, we choose $r = 0.02$, $\sigma = 0.2$, $T = 1$ and we choose in this special setting τ and $\mathcal{X}$ to take values in $[0, T(\times(0, 200)$ through a meshgrid. In usual case, people assume that $\tau \sim \mathcal{U}([0, T])$ and $\mathcal{X} \sim \mathcal{U}(D)$ where D is a bounded domain of interest.

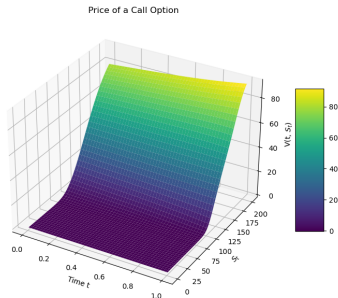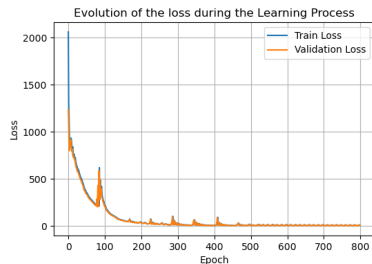# Numerical results in the $B - S$ setting

A forward contract

Figure: Evolution of the training and validation losses for the learning process of the PDE (5) related to the price of a forward contract (Case of $g(x) = x - K$ for $K = 100$)

Remark

We recover the linear relation between S_t and $C(t, S_t)$ for the case of the forward contract as we know that $C(t, S_t) = S_t - Ke^{-r(T-t)}$.

Numerical results in the $B - S$ setting

A Call Option

Figure: Evolution of the training and validation losses for the learning process of the PDE related to the price of a call option (Case of $g(x) = (x - K)^+$ for $K = 100$)

Remark

We recover the classic form for a call option noticing that $t \to C(t, S)$ is an increasing function.

Numerical results in the $B - S$ setting
PDE for the CVA of a Call option

It can be shown that the *Credit Valuation Adjustment* (CVA) in a default intensity model (such that $\mathbb{P}(\tau_C \geqslant t) = e^{-\lambda^C t}$) is solution to the following PDE :

$$\partial_t \phi(t, x) + \mathcal{L}\phi(t, x) - (r + \lambda^C)\phi(t, x) + (1 - R^c)(V_t)^+\lambda^C = 0, \quad (t, x) \in [0, T(\times \mathbb{R}_*^+ \tag{6}$$

$$\phi(T, x) = 0, \quad x \in \mathbb{R}_*^+$$

where :

- λ^C is the default intensity of the counterparty assumed to be constant.
- R^C is the recovery rate in case of default of the counterparty.
- $(V_t)^+$ is the value of the exposure of the portfolio / derivative involved between both counterparties.

Remark

In our numerical experiments, we will take the portfolio to be a standard call option with same characterics as before but with $K = 110$. Moreover, we will assume $R^C = 0$ and show the results for 2 different values of λ^C.

Numerical results in the $B - S$ setting
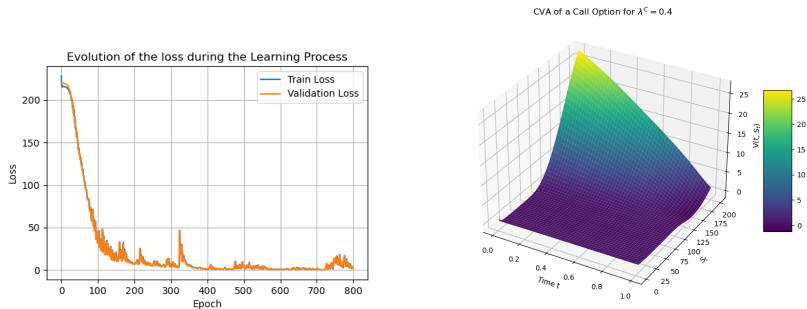
CVA on a Call Option with low λ^C
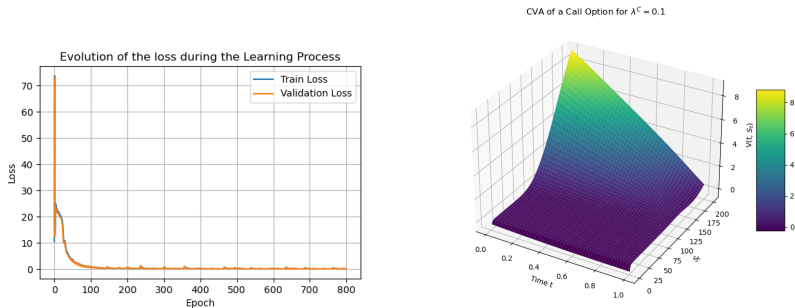
Figure: Evolution of the training and validation losses for the learning process of the PDE (6) related to the CVA price of a call option with $\lambda^C = 0.1$

Remark

We can see the terminal condition from the surface shape with $CVA(T, .) = 0$ and see that the function $S \to CVA(t, S)$ is an increasing function which is an expected behavior from the CVA definition.

Numerical results in the $B - S$ setting

CVA on a Call Option with high λ^C

Figure: Evolution of the training and validation losses for the learning process of the PDE (6) related to the CVA price of a call option with $\lambda^C = 0.4$

Remark

We can see the overall value of CVA is higher in the case of $\lambda^C = 0.4$ than in the case of $\lambda^C = 0.1$ which just explains that the counterparty is more likely to default and so the CVA to be paid by the defaultable counterparty has to be higher.

Deep Galerkin for PDE
Coupled Systems of PDE for KVA and FVA

In a toy model (See [8] if you are interested in how we can obtain such system of PDE), we can show that *KVA* and *FVA* are solution to the following coupled systems of PDE associated respectively with w and v :

$$\frac{\partial v}{\partial t} + \mathcal{L}v + \lambda(\max(\alpha f \sigma S|\frac{\partial v}{\partial S} - \Delta_{bs}|, w) + v - u_{bs})^- - rv = 0 \quad (t,x) \in]0, T[\times \mathbb{R}_*^+ \quad (7)$$

$$\frac{\partial w}{\partial t} + \mathcal{L}w + h\max(\alpha f \sigma S|\frac{\partial v}{\partial S} - \Delta_{bs}|, w) - (r + h)w = 0, \quad (t,x) \in]0, T[\times \mathbb{R}_*^+ \quad (8)$$

$$v(T, S) = w(T, S) = 0 \quad x \in \mathbb{R}_*^+$$

where h represents a dividend rate, α represents a mishedge parameter, λ is a funding rate and f is a quantile level. u_{bs} and Δ_{bs} represent the call and delta price of a single call option of same characteristics as before.

Remark

In this couple PDE systems, we parametrize two neural networks $\mathcal{U}_1(\theta_1)$ and $\mathcal{U}_2(\theta_2)$ and we solve $\inf_{\theta=(\theta_1,\theta_2)} \mathbb{L}(\mathcal{U}(\theta))$ where $\mathbb{L}$ represents the operator associated to system of PDE (7) *and* (8)*. For the numerical experiments, we took $\alpha = 0.3$, $\lambda = 0.02$, $f = 1.2$ and $h = 0.1$.*
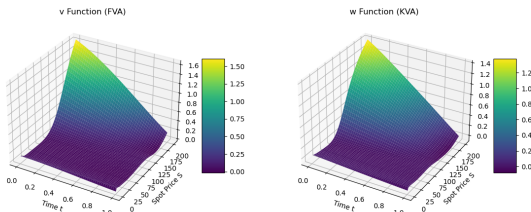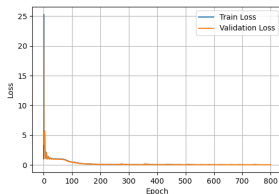
Figure: Evolution of the training and validation losses for the learning process of PDE for the coupled systems of PDE (7) and (8) for FVA and KVA and associated surfaces

Contents

A reminder on the Deep BSDE Solver
Mathematical Foundations

We are going to present some simple applications of the Deep BSDE Solver for solving PDE with the following form :

$$\partial_t v + \mathcal{L}v + f(x, v, \nabla_x v) = 0, \quad (t, x) \in [0, T) \times \mathbb{R}^d \qquad (9)$$
$$v(T, x) = g(x), \quad x \in \mathbb{R}^d$$

From this PDE, we can consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ which supports a brownian motion $W = (W_t)_{t \geqslant 0}$ with his natural filtration $\mathbb{F} = (\mathcal{F}_t)_{t \geqslant 0}$ and we can introduce the forward process $X = (X_t)_{t \geqslant 0}$ associated to the operator $\mathcal{L}$. Assuming this process known, we can consider the following pair of processes (Y, Z) solving the following BSDE :

$$-dY_t = f(t, X_t, Y_t, Z_t) - Z_t dW_t, \quad 0 \leqslant t \leqslant T \qquad (10)$$
$$Y_T = g(X_T)$$

Applying Itô Formula to the process $v(t, X_t)$ with v solving the PDE (9), we can see :

$$
\begin{aligned}
v(T, X_T) &= v(t, X_t) + \int_t^T (\partial_t v + \mathcal{L}[v])(s, X_s) ds + \int_t^T \nabla_x v(s, X_s)^\top dW_s \\
&= v(t, X_t) - \int_t^T f(s, v(s, X_s), \nabla_x v(s, X_s)) ds + \int_t^T \nabla_x v(s, X_s)^\top dW_s
\end{aligned}
$$

Differentiating this equation, we see that the process $v(t, X_t)$ solves the following BSDE :

$$
\begin{aligned}
-dv(t, X_t) &= f(t, v(t, X_t), \nabla_x v(t, X_t)) - \nabla_x v(t, X_t)^\top dW_t \\
v(T, X_T) &= g(X_T)
\end{aligned}
$$

From existence and unicity of the theory of BSDE under suitable assumptions on f and g, we have the following representation for the pair (Y, Z) :

$$
Y_t = v(t, X_t) \quad dt \otimes d\mathbb{P} \quad a.e \tag{11}
$$

$$
Z_t = \nabla_x v(t, X_t) \quad dt \otimes d\mathbb{P} \quad a.e \tag{12}
$$

A reminder on the Deep BSDE Solver
Algorithm Description

The Deep BSDE Algorithm is based on the representation of the equations (11) and (12) as it means that founding v is equivalent to founding Y. Therefore, going back to equation (10), we can look for discretization of Y and approximating $v(t_i, X_{t_i})$ as Y_{t_i}. However, note that the scheme is backward in time which would need to approximation conditional expectations at each time. Therefore, the idea of the algorithm is to treat the process Y as a forward process for an unknown y_0 and for process Z. We can then define the following loss $\mathcal{L}$ for a given $y_0 \in \mathbb{R}$ and $\mathcal{Z}$ a squared adapted integrable process as :

$$\mathcal{L}(y_0, \mathcal{Z}) = \mathbb{E}[|Y_T^{y_0, \mathcal{Z}} - g(X_T)|^2] \tag{13}$$

Therefore, as y_0 and $\mathcal{Z}$ are unknown parameters, they can be learnt through neural networks assuming that $\mathcal{Z} = \mathcal{Z}(s, X_s)$. The idea is then to learn through a neural network the mapping $(t, x) \to \mathcal{Z}(t, x)$ using a neural network $\mathcal{Z}^\theta$ and to put y_0 as a trainable parameter of this neural network which will be learnt during the training process.

A reminder on the Deep BSDE Solver
Algorithm Description

Therefore, the idea is to minimize the following loss error :

$$L(\theta) = \mathbb{E}[|Y_T^\theta - g(X_T)|^2] \tag{14}$$

where we set :

$$Y_t^\theta = y_0^\theta - \int_0^t f(X_s, Y_s^\theta, \mathcal{Z}^\theta(s, X_s))ds + \int_0^t \mathcal{Z}^\theta(s, X_s)^\top \sigma(s, X_s)dW_s \tag{15}$$

Remark

Of course, for the numerical experiments, we will discretize (15) using an Euler Scheme on a grid $0 = t_0 < t_1 < \ldots < t_N = T$ with time step Δt : starting from $Y_0^\theta = y_0^\theta$, we have :

$$Y_{t_{i+1}}^\theta = Y_{t_i}^\theta - f\big(X_{t_i}, Y_{t_i}^\theta, \mathcal{Z}_\theta(t_i, X_{t_i})^\top\big)\Delta t + \sigma(t_i, X_{t_i})\Delta W_{t_i}, \quad i = 0, \ldots, n-1$$

An application in option pricing
Application in the $B - S$ model

We now assume a $B - S$ model dynamics with the underlying dynamics $S = (S^1, \ldots, S^d)$ and $W = (W^1, \ldots, W^d)$ multidimensional brownian motion given by :

$$dS_t^i = S_t^i(rdt + \sigma^i dW_t^i), \quad S_0^i \in (\mathbb{R}_*^+), \quad i = 1, \ldots, d \tag{16}$$

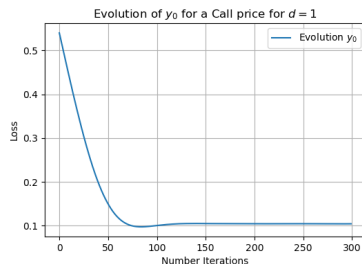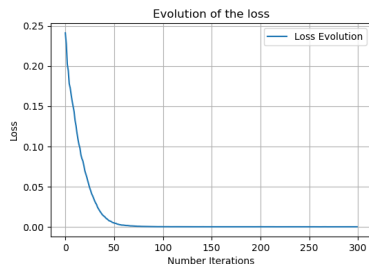Under the Option pricing theory in the $B - S$ model, we have the PDE (5) :

$$\partial_t C + \mathcal{L}C - rC = 0, \quad (t, x) \in [0, T] \times (\mathbb{R}_*^+)^d$$
$$C(T, x) = g(x), \quad x \in (\mathbb{R}_*^+)^d$$

where the infinitemisal generator is given by :

$$\mathcal{L}v(t, x) = b(t, x)^\top D_x v(t, x) + \frac{1}{2} Tr\big(\sigma(t, x)\sigma(t, x)^\top D_x^2 v(t, x)\big)$$

Numerical Results in the $B - S$ setting

A Call Option
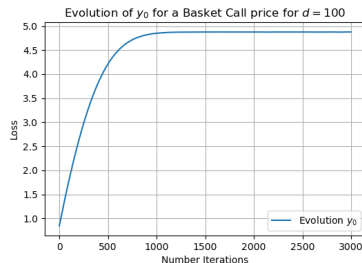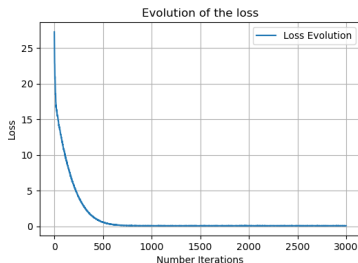

Figure: Loss and y_0 evolution for the $B - S$ PDE (5) for a Call option for $d = 1$ with payoff $g(x) = (x - K)^+$ with $r = 0.05$, $\sigma = 0.2$, $T = 1$, $x_0 = 1$ and $K = 1$.

Table: $u(0, x_0)$ Approximation for the Basket Call Option

	True Value	Estimate Value
$u(0, x_0)$	0.1045	0.1043

Numerical Results in the $B - S$ setting

A Basket Call Option

Evolution of the loss
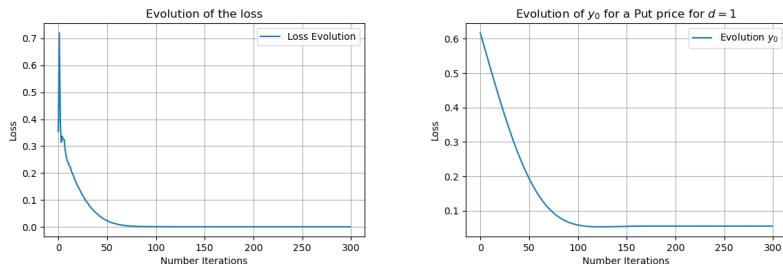
Evolution of y_0 for a Basket Call price for $d = 100$

Figure: Loss and y_0 evolution for the $B - S$ PDE (5) for a Basket Call option for $d = 100$ with payoff $g(x) = (\sum_{i=1}^{d} x_i - dK)^+$ with $r = 0.05$ and $\sigma^i = 0.2$ for $i = 1, \ldots, d$ and with uncorrelated $(W^i)_{i=1,\ldots,d}$ with $x_0 = (1, \ldots, 1) \in \mathbb{R}^d$.

Table: $u(0, x_0)$ Approximation for the Basket Call Option

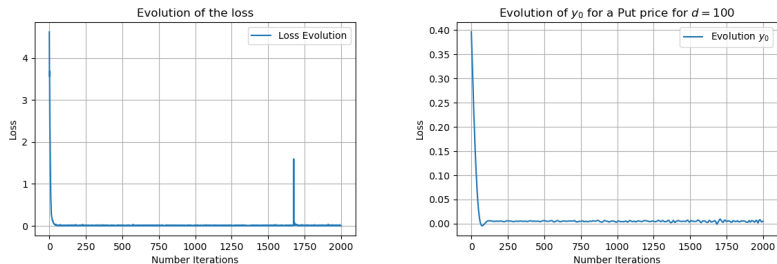|	Estimate Value
$u(0, x_0)$	4.8771

Numerical Results in the $B - S$ setting
A Put Option

Figure: Loss and y_0 evolution for the $B - S$ PDE (5) for a Put option for $d = 1$ with payoff $g(x) = (K - x)^+$ with $r = 0.05$, $\sigma = 0.2$, $T = 1$, $x_0 = 1$ and $K = 1$.

Table: $u(0, x_0)$ Approximation for the Basket Put Option

	True Value	Estimate Value
$u(0, x_0)$	0.05574	0.05568

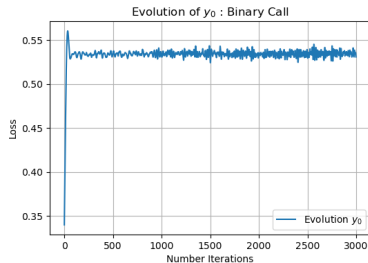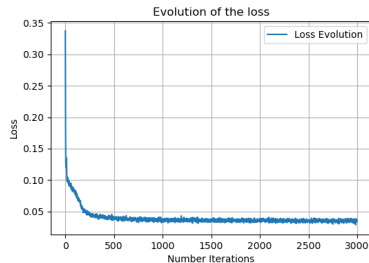# Numerical Results in the $B - S$ setting
A Basket Put Option

Figure: Loss and y_0 evolution for the $B - S$ PDE (5) for a Basket Put option for $d = 100$ with payoff $g(x) = (dK - \sum_{i=1}^{d} x_i)^+$ with $r = 0.05$ and $\sigma^i = 0.2 \quad i = 1, \ldots, d$ with uncorrelated $W = (W^1, \ldots, W^d)$ with $x_0 = (1, \ldots, 1) \in \mathbb{R}^d$.

Table: $u(0, x_0)$ Approximation for the Basket Put Option

	Estimate Value
$u(0, x_0)$	0.0051

Numerical results in the $B - S$ setting
A Binary Option

Figure: Loss and y_0 evolution for the $B - S$ PDE (5) for a Binary option for $d = 1$ with payoff $g(x) = \mathbb{1}_{x > K}$.

Table: $u(0, x_0)$ Approximation for the Binary Option

	True Value	Estimate Value
$u(0, x_0)$	0.5323	0.5307

- Impact of the discontinuity of g in the learning process.
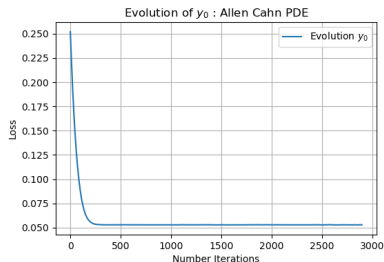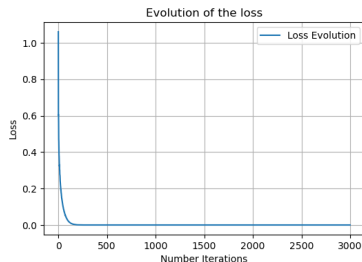
The Allen-Cahn PDE is a famous *PDE* given by the following :

$$\partial_t v + \Delta_x v + v - v^3 = 0 \quad (t,x) \in [0, T(\times \mathbb{R}^d,$$

$$v(T, x) = \frac{1}{2 + \frac{2}{5}\|x\|^2} \quad x \in \mathbb{R}^d,$$

(17)

Numerical results for other type of PDE

Allen-Cahn equation

Figure: Loss and y_0 evolution for the Allen Cahn PDE (17)

Table: $u(0, x_0)$ Approximation for the Allen Cahn Equation

	True Value	Estimate Value
$u(0, x_0)$	0.0528	0.0529

We consider the following PDE which can be shown to be the PDE arising from an HJB equation in optimal control :

$$\partial_t v + \Delta_x v - \frac{1}{2}|\nabla_x v|^2 = 0, \quad (t, x) \in [0, T) \times \mathbb{R}^d, \tag{18}$$
$$v(T, x) = g(x), \quad x \in \mathbb{R}^d,$$

Numerical results for other type of PDE
Linear Quadratic control problem
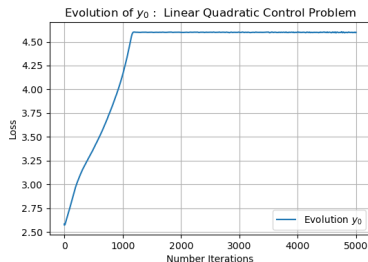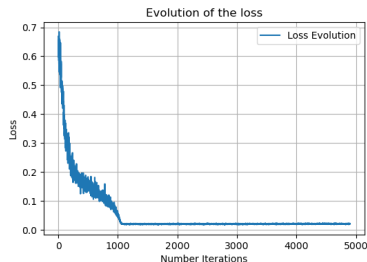
Figure: Loss and y_0 evolution for the Linear Quadratic control problem from PDE (18)

Table: $y_0 = u(0, x_0)$ Approximation for the Linear Quadratic control problem

	True Value	Estimate Value
$u(0, x_0)$	4.5901	4.5988

References

C.Huré, H.Pham, A.Bachouch, N.Langrené, 2019, *"Deep Neural Networks algorithms for stochastic control problems on finite horizon : Convergence Analysis"*

A.Bachouch, C.Huré, N.Langrené, H.Pham, 2019, *"Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications"*

K.Andersson, A.Gnoatto, M.Patacca, A.Picarelli, 2024, *"A Deep BSDE solver with jumps"*

Weinan.E, J.Han, A.Jentzen, 2017, *"Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations"*

A.Gnoatto, A.Picarelli, C.Reinsiger, 2022, *"Deep XVA solver - A neural network based counterparty credit risk management framework"*

M.Germain, H.Pham, X.Warin, 2021, *"Neural networks-based algorithms for stochastic control and PDEs in finance"*

J.Sirignano, K.Spiliopoulos, 2017 *"DGM: A deep learning algorithm for solving partial differential equations"*

Y.Armenti, S.Crépey, C.Zhou, 2018, *"The Sustainable Black-Scholes Equations"*