Résolution d'équations aux dérivées partielles semi-linéaires en grandes dimensions par réseaux de neurones

24 Juin 2023

Table of contents

Généralités sur les PDE

On considère l'EDP du second ordre non-linéaire du type :

$$-\partial_t u(t,x) - H(t,x,u(t,x),Du(t,x),D^2u(t,x)) = 0, x \in \Omega, t \in [0,T[$$

avec la condition terminale $u(T,x) = g(x)$.
Ici Ω est un ouvert de $\mathbb{R}^n$ et $H = H(t,x,z,p,r)$ est appelé l'hamiltonien (définit par un problème de contrôle optimal avec un processus contrôlé) .

- L'EDP est dite parabolique si pour tout t, x, z, p, r_1, r_2,

$$r_2 \geq r_1 \implies H(t,x,z,p,r_2) \geq H(t,x,z,p,r_1).$$

- La condition de parabolicité joue un rôle important pour caracteriser le principe de comparaison (équivalent à la propriété d'absence d'opportunité d'arbitrage).

Généralités sur les PDE

- Dans le cas où le terme de diffusion ne dépend pas du contrôle, le Hamiltonien se simplifie et on obtient l'équation semi-linéaire du type:

$$\partial_t u(t,x) + Lu(t,x) + f(t,x,u(t,x),Du(t,x)) = 0.$$

- Exemples d'équations bien connues
  1. L'équation de Schrodinger :

$$i\hbar\partial_t \Psi(t,x) = (-\frac{1}{2}\Delta + V)\Psi(t,x).$$

  2. L'équation de Black - Scholes pour le pricing d'options européenes :

$$\partial_t u + \frac{1}{2} Tr(\sigma\sigma^T D^2 u) + rDu.x - ru = 0.$$

Lien PDE semi-linéaire et BSDE

Considérons un processus d'Ito d-dimensionnel (X_t) définit par la SDE :

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t, X_0 = x \in \mathbb{R}^d.$$

(W_t) un mouvement Brownien d dimensionnel.

Definition

Une solution (markovienne) à une BSDE est un couple (Y_t, Z_t) de processus d'Itô (F_t)-adaptés vérifiant :

$$dY_t = -f(t, X_t, Y_t, Z_t)dt + Z_t dW_t$$

avec une condition terminale $Y_T = g(X_T)$.

Lien PDE semi-linéaire et BSDE

Remarque : Quand $f = 0$, $g(X_T) = Y_t + \int_t^T Z_s dW_s$.
Cela signifie que Y_t est le prix à t d'une option de pay-off $g(X_T)$ et le processus adapté Z correspond à la stratégie Delta .

Theorem

(Pardoux-Peng)
Supposons que f soit une fonction déterministe Lipschitz et que g soit de carré intégrable.
Alors il existe une unique solution (Y, Z) à la BSDE précédente .

Lien PDE semi-linéaire et BSDE

Theorem

(Formule de Feynman - Kac généralisée)
Soit u solution $C^{1,2}$ *de l'équation*
$\partial_t u(t,x) + Lu(t,x) + f(t,x,u(t,x),Du(t,x)) = 0$,$(t,x) \in [0,T] \times \mathbb{R}^d$.
Alors $Y_t = u(t,X_t)$ *et* $Z_t = \sigma(t,X_t)^T Du(t,X_t)$.

Preuve : Application de la formule d'Itô à $u(t,X_t)$.
Dans le cas où $f = 0$, Y est une martingale locale (en fait c'est une vraie martingale car Z est de carré intégrale) et donc
$Y_t = u(t,X_t) = E[g(X_T)|X_t]$ (Formule de Feynman - Kac).

Méthodes de numériques classiques

- Méthodes des éléments finis(méthode déterministe): Elles sont utilisées quand la dimension du problème $d \leq 3$. En général la complexité est une fonction exponentielle de la dimension (phénomène connu sous le nom de la malédiction de la dimension).
- Schémas d'Euler(méthode backward) : Utilisation des BSDE et calculs d'espérances conditionnelles (dont l'estimation est complexe dans le cas de plusieurs sous-jacents).
- Utilisation des Processus de Branchement(méthode forward) : Efficace quand la non-linéarité est polynomiale en la solution et ses dérivées .

Approximation par réseaux de neurones

- Succès remarquable historique dans la résolution des problèmes en grandes dimensions(Computer Vision , traitement naturel du langage ...) .
- Un réseau de neurones est juste une composition successive de fonctions non-linéaires et de fonctions linéaires.

Theorem

(Théorème d'approximation universelle)
Toute fonction raisonnable est approchable avec une précision arbitraire par un réseau de neurones avec un certain nombre de neurones cachés et pour une certaine fonction d'activation sur un compact .

Deep BSDE solver

- Résoudre notre PDE semi-linéaire est "équivalent" à la résolution de sa BSDE associée .
- Inconnues de la BSDE : Y_0, $(Z_t)_{0 \leq t \leq T}$
- on cherche à résoudre $\min_{Y_0,(z_t)} E(Y_T - g(X_T))^2$

Algorithme

Algorithme du deep solver(E, Han, and Jentzen):

- On discrétise $[0, T]$ en $0 = t_0 < t_1 < ... < t_n = T$ et on simule M trajectoire de MC de $(X_{t_1}^m, ..., X_{t_n}^m)$ via un schéma d'Euler
$$X_{i+1} = X_i + b(t_i, X_i)(t_{i+1} - t_i) + \sigma(t_i, X_i).\Delta W_i, \quad \text{pour } i \in [1, n-1]$$
$$X_0 = x,$$
- Parametriser $(z_i)_{1 \leq i \leq n-1}$ par $n-1$ réseaux de neurones de poids $\theta_i = (W^i, b^i)$ par $z_i = NN_{\theta_i}(X_{t_i})$ pour $i \in [1, n-1]$.
- Prendre z_0 et Y_0 comme des paramètres du réseau de neurones .
- Minimiser sur $(\theta_i)_{1 \leq i \leq n-1}, Y_0, z_0$ la fonction de loss empirique

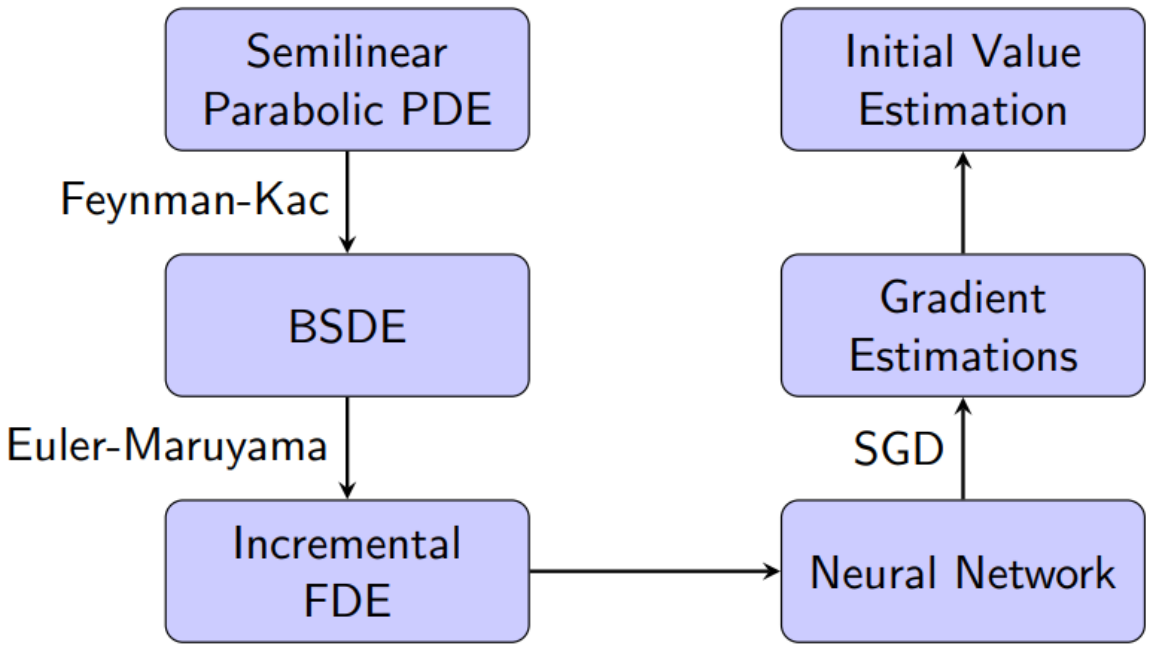
$$\frac{1}{2M} \sum_{m=1}^{M} (Y_T^m - g(X_T^m))^2$$

par une descente de gradient avec la discrétisation de la bsde de (Y_{t_i}) donnée par un schéma d'Euler :

$$Y_{t_{i+1}}^m - Y_{t_i}^m = -f(t_i, X_{t_i}, Y_{t_i}, NN_{\theta_i}(X_{t_i}))\Delta T + NN_{\theta_i}(X_{t_i}).\Delta W_i^m,$$
$$Y_0^m = Y_0.$$

Deep BSDE solver summary

BSDE Method Flowchart



Implémentation numérique

- Chaque sous-réseau a 4 couches avec une couche d'entrée de dimension d, 2 couches cachées de dimension ($d + 10$) et une couche en sortie de dimension d .
- On choisit comme fonction d'activation la fonction ReLU et on optimise avec la méthode Adam .
- L'implémentation se fait en Tensorflow .

Pricing d'un call dans le modèle de Black- Scholes

- Dans le modèle de BS l'équation de pricing d'un call est donné par :

$$\partial_t u + rS\partial_S u + \frac{1}{2}\sigma^2 S^2 \partial_{SS} u = ru$$

avec $g(x) = (x - K)_+$, K le strike.

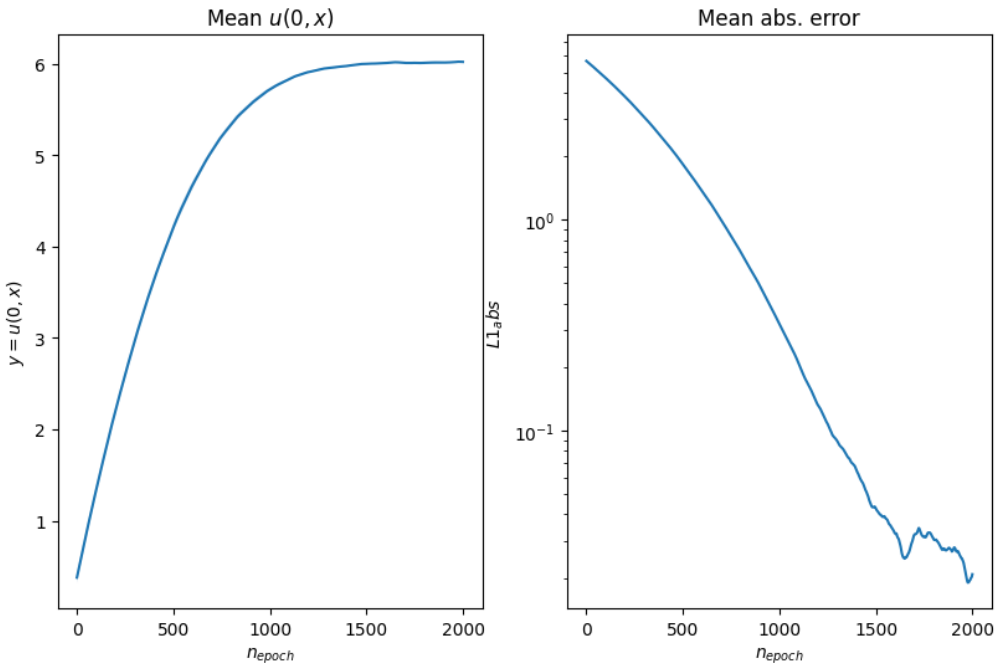


Figure : Evolution du prix d'un call de maturité $T = 1$, $K = 110$, $r = 0.05$, en fonction du nombre d'epochs.

Problème Gaussien quadratique

- Problème de contrôle :

$$u(0,x) = \min_{m_t} \mathbb{E}\left[\int_0^T \|m_t\|^2 \mathrm{d}t + g(X_T) \mid X_0 = x\right]$$

avec $g(x) = \ln(\frac{1}{2}[1 + ||x||^2])$, $x \in \mathbb{R}^d$ et

$$dX_t = 2m_t dt + \sqrt{2} dW_t$$

- Equation de HJB :

$$\partial_t u(t,x) + \Delta u(t,x) = ||\nabla u||^2(t,x)$$

.

- La solution est donnée par $u(t,x) = ln(E(e^{g(x+W_{T-t}\sqrt{2})}))$.

Problème Gaussien quadratique

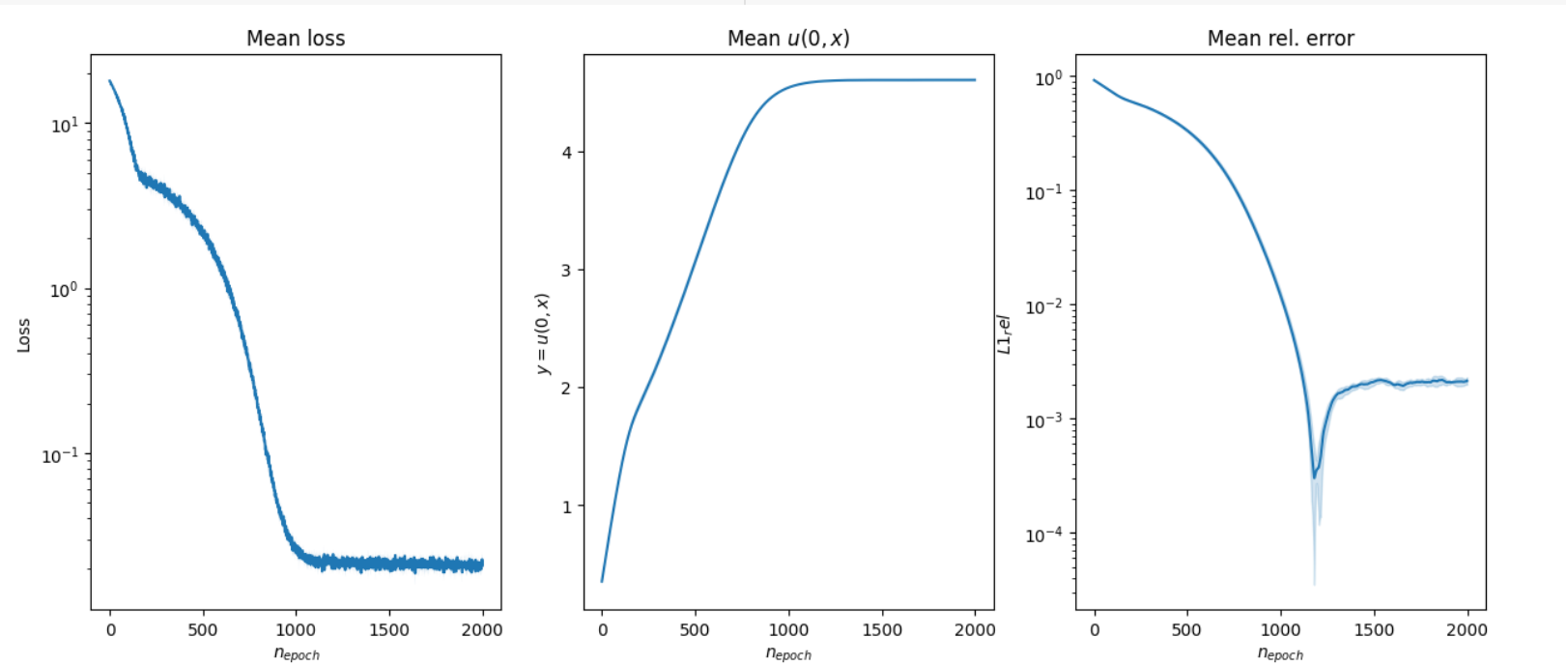


Figure : Evolution de la fonction de la loss, solution et de l'erreur relative pour $T = 1$ et $d = 100$ en fonction du nombre d'epochs.

Conclusion

- On reforme les EDP non-linéaires en grandes dimensions sous forme de BSDE dont le contrôle optimal est approché par des réseaux de neurones.
- On estime uniquement la solution et la couverture à l'instant $t = 0$.
- Le Deep BSDE Solver a une complexité linéaire en le nombre de pas de temps.

Reference

- Han, Jentzen, and E, Solving high-dimensional partial differential equations using deep learning, arXiv:1707.02568
- E, Han, and Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Communications in Mathematics and Statistics (2017)
- Henry-Labordère 2017: deep primal-dual algorithm for BSDEs