Machine Learning en Finance
Projet - Méthode Deep Parametric PDE (DPDE)

Michal Tanguy[1], Pécheul Ronan[1], Sanglier Nathan[1]

[1]M2MO, Université Paris Cité

Introduction

- Analyse du papier [GW22] de K. Glau et L. Wunderlich: "*The Deep Parametric PDE Method and Applications to Option Pricing*" publié en 2022.

- DPDE: méthode de Deep Learning pour résoudre une EDP parabolique paramétrique en haute dimension, i.e. $\forall \mu \in P$:

$$\begin{cases} \partial_t u(t,x;\mu) + \mathcal{L}_x^\mu u(t,x;\mu) & = f(t,x;\mu) & (t,x) \in (0,T] \times \Omega, \\ u(0,x;\mu) & = g(x;\mu) & x \in \Omega, \\ u(t,x;\mu) & = u_\Sigma(t,x;\mu) & (t,x) \in (0,T] \times \partial\Omega. \end{cases} \tag{1}$$

Extension de la méthode Deep Galerkin (DG) I

- [SS18] reformule l'EDP (1) (à μ fixé) comme un problème d'optimisation stochastique:

$$\begin{aligned}\mathbb{L}(v) =& \mathbb{E}\left[|\partial_t v(\hat{t},\hat{x};\mu) + \mathcal{L}_x^\mu v(\hat{t},\hat{x};\mu) - f(\hat{t},\hat{x};\mu)|^2\right. \\ &\left. + |v(0,\tilde{x};\mu) - g(\tilde{x};\mu)|^2 + |v(\bar{t},\bar{x};\mu) - v_\Sigma(\bar{t},\bar{x};\mu)|^2\right],\end{aligned}$$

  où $(\hat{t},\hat{x}) \sim \nu_1$, $\tilde{x} \sim \nu_2$, et $(\bar{t},\bar{x}) \sim \nu_3$ (eg. distributions uniformes).

- u solution de (1) $\Leftrightarrow \mathbb{L}(u) = \inf_v \mathbb{L}(v) = 0$.

- Problème de dimension infinie $\Rightarrow$ focus sur fonctions v_θ d'un réseau de neurones.

Extension de la méthode Deep Galerkin (DG) II

- Trouver θ^* tel que $\mathbb{L}(v_{\theta^*}) = \inf_\theta \mathbb{L}(v_\theta)$, qui comporte une espérance qu'on ne sait pas calculer.

- Solution: Stochastic Gradient Descent
  Tant qu'un critère de convergence n'est pas satisfait:
  1. Générer $s_n = \left\{(\hat{t}_n, \hat{x}_n), \tilde{x}_n, (\bar{t}_n, \bar{x}_n)\right\}$.
  2. Calculer l'erreur empirique:

$$\begin{aligned} I(\theta_n, s_n) =& |\partial_t v_{\theta_n}(\hat{t}_n, \hat{x}_n; \mu) + \mathcal{L}_x^\mu v_{\theta_n}(\hat{t}_n, \hat{x}_n; \mu) - f(\hat{t}_n, \hat{x}_n; \mu)|^2 \\ &+ |v_{\theta_n}(0, \tilde{x}_n; \mu) - g(\tilde{x}_n; \mu)|^2 + |v_{\theta_n}(\bar{t}_n, \bar{x}_n; \mu) - v_\Sigma(\bar{t}_n, \bar{x}_n; \mu)|^2. \end{aligned}$$

  3. Descente de gradient: $\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta I(\theta_n, s_n)$.

Extension de la méthode Deep Galerkin (DG) III

- DPDE est une extension de DG en randomisant aussi μ:

$$\begin{aligned} I(\theta_n, s_n) =& |\partial_t v_{\theta_n}(\hat{t}_n, \hat{x}_n; \hat{\mu}_n) + \mathcal{L}_x^{\hat{\mu}_n} v_{\theta_n}(\hat{t}_n, \hat{x}_n; \hat{\mu}_n) - f(\hat{t}_n, \hat{x}_n; \hat{\mu}_n)|^2 \\ &+ |v_{\theta_n}(0, \tilde{x}_n; \tilde{\mu}_n) - g(\tilde{x}_n; \tilde{\mu}_n)|^2 \\ &+ c_{bc} \cdot |v_{\theta_n}(\bar{t}_n, \bar{x}_n; \bar{\mu}_n) - v_\Sigma(\bar{t}_n, \bar{x}_n; \bar{\mu}_n)|^2, \end{aligned}$$

  où $s_n = \left\{(\hat{t}_n, \hat{x}_n, \hat{\mu}_n), (\tilde{x}_n, \tilde{\mu}_n), (\bar{t}_n, \bar{x}_n, \bar{\mu}_n)\right\}$.

- En pratique, Adam est utilisé plutôt que SGD et $c_{bc} = 0$.

Propriétés Essentielles de DPDE

- Méthode non-supervisée et meshfree: il suffit juste de générer des samples (t, x, μ) selon une distribution donnée.
- Réseau de neurones Higway: utilisation de portes (tanh) pour contrôler le flux d'information à travers les couches sans atténuation.
- Preuve théorique de l'existence d'un réseau de neurones qui approxime la vraie solution de l'EDP paramétrique:
  - pour n'importe quelle précision donnée,
  - pour une condition initiale non-régulière,
  - en supposant convergence globale de l'optimiser.
- 1 seul apprentissage de la fonction solution pour toute valeur de μ.

État de l'Art I

1. Approches Déterministes.
   - Différences finies: erreurs d'approximation bien quantifiées, mais "curse of dimensionality" (maillage spatial et temporel).
   - Réseaux de neurones profonds: meshfree mais performance dépend fortement du choix des mesures d'entraînement.
     - Deep Galerkin [SS18] et DPDE [GW22].
     - PINNs [RPK19]: incorporer de l'information préalable sur l'EDP via un apprentissage supervisé supplémentaire.
     - PPINNs [BdBS$^+$24]: extension pour résoudre une EDP paramétrique.

État de l'Art II

2. Approches Probabilistes: reformuler l'EDP comme une espérance conditionnelle d'un processus stochastique satisfaisant une SDE.
   - Monte-Carlo: valeur de u en un point spécifique.
   - Régression par réseaux de neurone: apprentissage supervisé.
     - Deep BSDE [HJE18]: résoudre EDP backward via simulations forward.
     - Deep BDP [HPW20]: amélioration via une approche backward et locale.
     - Deep Kolmogorov [BDG20]: samples iid. de (t, x, μ) selon loi uniforme et de la variable aléatoire cible via schéma d'Euler, puis réseau de neurones multilevel.

Black-Scholes Pricing Option Basket Européenne I

- d actifs $S_t(\mu) = (S_t^1(\mu), \cdots, S_t^d(\mu))^T$ et $\mathbb{Q}$ proba risque-neutre tq:

$$\begin{aligned} dS_t &= \text{diag}(S_t)\left(r\mathbf{1}dt + \sigma dY_t\right) \\ \Leftrightarrow dS_t &= \text{diag}(S_t)\left(r\mathbf{1}dt + \sigma L dW_t\right), \end{aligned}$$

où $\sigma = \text{diag}(\sigma_1, \cdots, \sigma_d)$ et $\rho = LL^T$ décomposition de Cholesky de la matrice de corrélation de Y vecteur de d browniens.

- $\mathbb{Q}$-prix option de payoff $g(S_T)$ en T:

$$P(\tau, s; \mu) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-\tau)} g\left(S_T(\mu)\right) \mid S_\tau(\mu) = s\right].$$

Black-Scholes Pricing Option Basket Européenne II

- Théorème de Feynman-Kac (backward, log-prix) $\Rightarrow$ EDP (1) tq:

$$\begin{cases} \mathcal{L}_x^\mu u(t,x;\mu) &= ru(t,x;\mu) - \sum_{i=1}^d \left(r - \frac{\sigma_i^2}{2}\right) \partial_{x_i} u(t,x;\mu) \\ & \quad - \sum_{i,j=1}^d \frac{\rho_{ij}\sigma_i\sigma_j}{2} \partial_{x_i x_j} u(t,x;\mu), \\ f(t,x;\mu) &= 0, \end{cases}$$

où $u(t,x;\mu) = P(T-t, e^x;\mu)$ avec $e^x = (e^{x_1}, \cdots, e^{x_d})^T$.

- ! [GW22]: conditions aux bords ?

- Comment garantir $\rho \succeq 0$ et symétrique ?
  Paramétriser ρ par corrélations pairwise $\hat{\rho}_i = \rho_{i,i+1}$, et
  $\rho_{ij} = \rho_{ji} = \prod_{k=i}^{j-1} \hat{\rho}_k$, $\forall j > i$.

$$\mu = (r, \sigma_1, \cdots, \sigma_d, \hat{\rho}_1, \cdots, \hat{\rho}_{d-1}).$$

Pricing Put Basket Européen I

- [GW22] : résultats numériques que pour un call. Pour un put:

$$g(x;\mu) = g(x) = \left(K - \frac{1}{d}\sum_{i=1}^{d} e^{x_i}\right)_+ .$$

- Information a priori via borne de non-arbitrage:

$$u(t,x;\mu) \geq \hat{u}_\infty(t,x;\mu) = \left(Ke^{-rt} - \frac{1}{d}\sum_{i=1}^{d} e^{x_i}\right)_+ .$$

- Approximation plus régulière:

$$\hat{u}_\lambda(t,x;\mu) = \frac{1}{\lambda}\ln\left(1 + e^{\lambda\left(Ke^{-rt} - \frac{1}{d}\sum_{i=1}^{d} e^{x_i}\right)}\right),$$

Pricing Put Basket Européen II

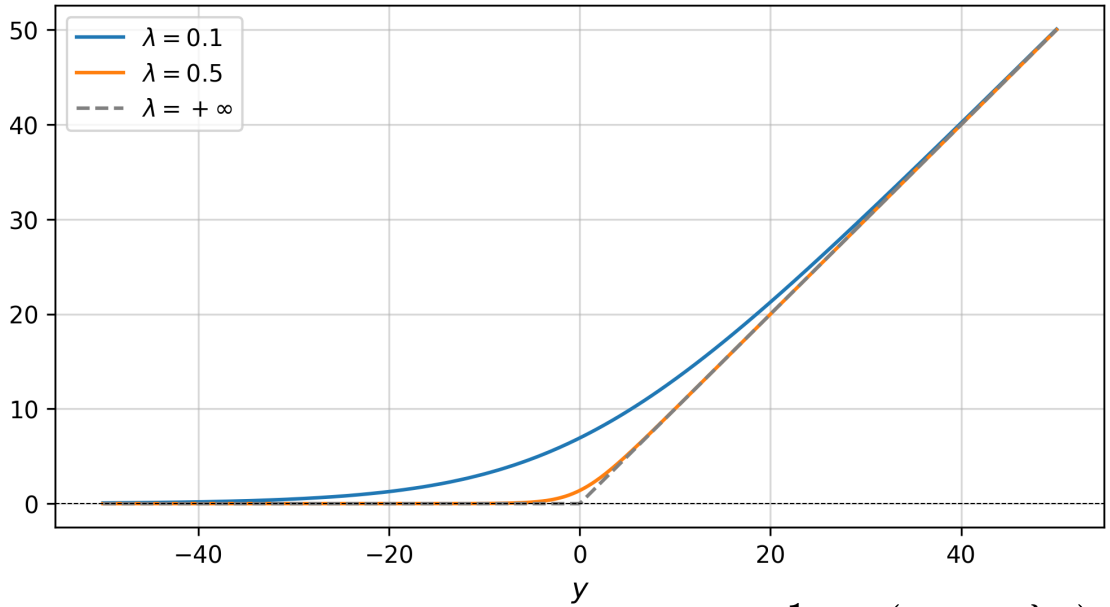

Figure 1: Fonction $h_\lambda : y \mapsto \frac{1}{\lambda} \ln\left(1 + e^{\lambda y}\right)$.

- Résoudre EDP du résidu $v = u - \hat{u}_\lambda$ via DPDE.

Résultats Numériques I

- $d = 2$ actifs, performance comparée au pricer de [BST17] et [Pö20] (donné comme groundtruth).
- Training sur un plus petit réseau de neurones (3 couches cachées, 90 neurones par couche).

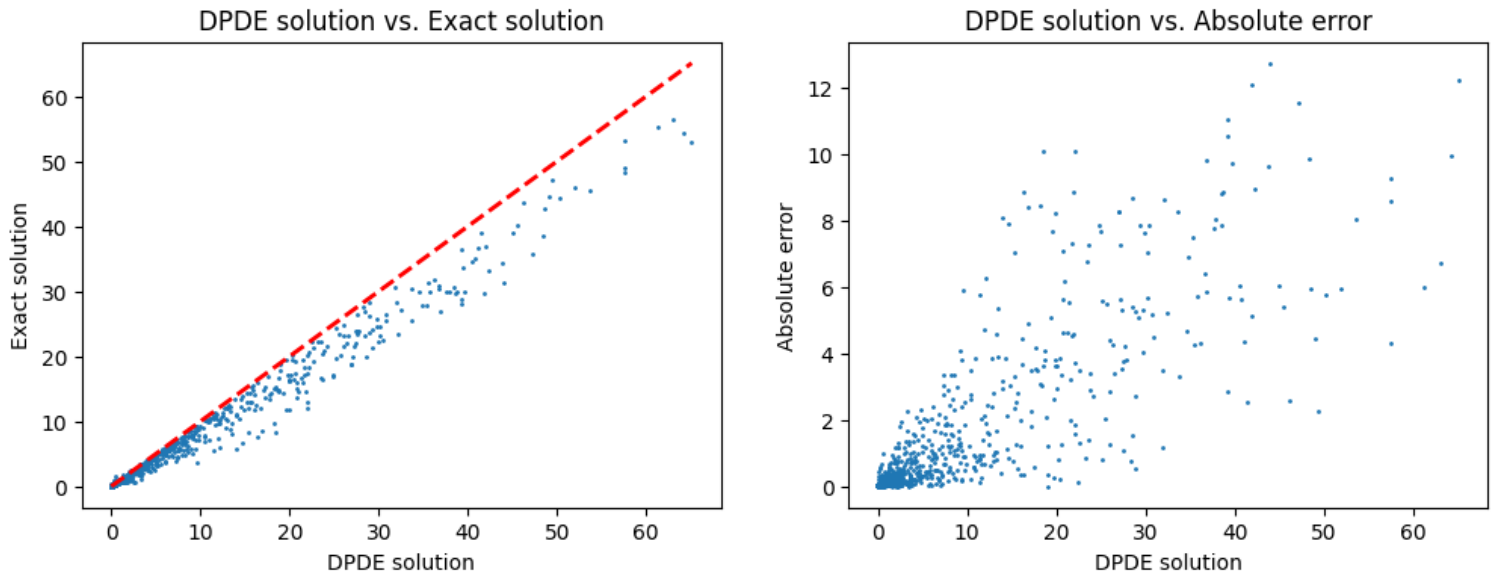

Figure 2: Option put. Prix prédits vs. prix de référence sur test set. Norm. RMSE $\approx 23\%$.

Résultats Numériques II

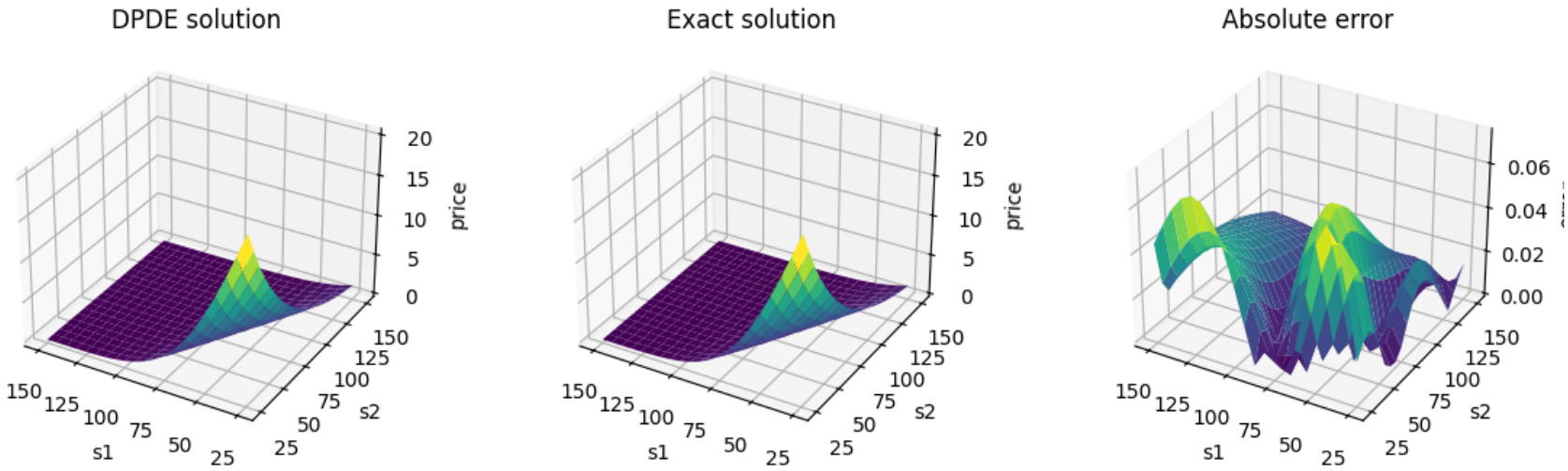

Figure 3: Option put. Surfaces évaluées à (t, μ) fixés.
Domaine de pricing $\subset$ domaine de calcul.

Résultats Numériques III

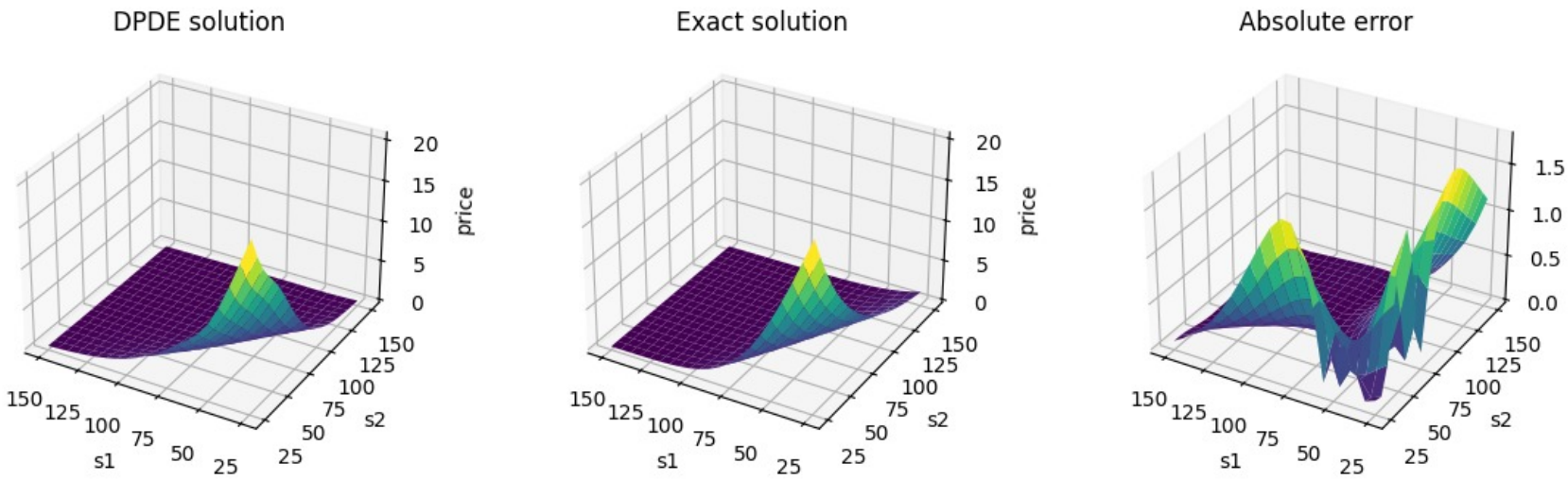

Figure 4: Option put.
Surfaces évaluées à (t, μ) fixés. Domaine de pricing = de calcul.

Limitations Possibles

- Absence de prise en compte des conditions aux bords en pratique.
- Choix de l'architecture et des hyperparamètres a un impact significatif sur la performance. Finetuning, robustesse ?
- Ne pas inclure la borne de non-arbitrage $\hat{u}_\lambda$ dégrade fortement les résultats obtenus.

Extensions Possibles

- Réintégrer la condition aux bords ($c_{bc} = 1$).
- Deep Parametric PDE permet de calculer directement les Grecques. Celles qui n'apparaissent pas dans l'EDP (1) sont en général mal calculées. Possibilité de modifier la loss pour les prendre en compte ?
- Généraliser le framework Deep Parametric PDE à d'autres EDPs, ie. d'autres produits dérivés (eg. option Américaine).

Conclusion

- Deep Parametric PDE est une extension naturelle de Deep Galerkin.
- Méthode non-supervisée et meshfree, 1 seul training pour toute valeur de μ.
- Point de vigilance sur les conditions aux bords.

Merci de votre attention!

Des questions?

Bibliography I

[BdBS+24] Lise Le Boudec, Emmanuel de Bezenac, Louis Serrano, Ramon Daniel Regueiro-Espino, Yuan Yin, and Patrick Gallinari. Learning a neural solver for parametric pde to enhance physics-informed methods, 2024.

[BDG20] Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional kolmogorov partial differential equations via deep learning, 2020.

[BST17] Christian Bayer, Markus Siebenmorgen, and Raul Tempone. Smoothing the payoff for efficient computation of basket option prices, 2017.

[GW22] Kathrin Glau and Linus Wunderlich. The deep parametric pde method and applications to option pricing. *Applied Mathematics and Computation*, 432, 2022.

Bibliography II

[HJE18] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115, 2018.

[HPW20] Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear pdes, 2020.

[Pö20] Christian Pötz. *Function approximation for option pricing and risk management Methods, theory and applications*. PhD thesis, School of Mathematical Sciences Queen Mary, University of London, 2020.

[RPK19] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 2019.

Bibliography III

[SS18] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 2018.