Machine learning schemes
for high-dimensional nonlinear PDEs

Huyên PHAM*

*Université de Paris, LPSM

Based on joint works with
Côme HURE, Université de Paris, LPSM
Xavier WARIN, EDF & FiME

August 5-9, 2019
Workshop Machine learning and FinTech
NUS, RMI
Nonlinear Partial Differential Equation

- Parabolic semi-linear PDE

\[
\begin{cases}
\frac{\partial v}{\partial t} + \frac{1}{2} \Delta_x v &= f(t, x, v, D_x v), \quad (t, x) \in [0, T) \times \mathbb{R}^d \\
v(T, x) &= g(x), 
\end{cases}
\]

where \( \Delta_x \) is the Laplacian operator

\[
\Delta_x = \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2},
\]

generator \( f: [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \mapsto \mathbb{R} \), and terminal data \( g: \mathbb{R}^d \mapsto \mathbb{R} \).

Remark: PDE (1) includes Bellman equation from stochastic control with controlled drift.

- Challenging issue: numerical resolution of (1) in high dimension \( d = 10, 50, 100! \) ↔ curse of dimensionality
Representation of feedforward Deep Neural Networks (DNN)

- **DNN**: composition of simple functions \( \neq \) additive approximation theory

  - Represented by parametrized function:
    \[
    x \in \mathbb{R}^d \mapsto \Phi(x; \theta) = A_{\text{out}} \circ \sigma \circ A^L \circ \ldots \circ \sigma \circ A_1(x) \in \mathbb{R}^{d_{\text{out}}},
    \]
    \[
    A_\ell(x) = w_\ell x + b_\ell \in \mathbb{R}^{K},
    \]
    with \( L \) hidden layers (layer \( \ell \) with \( K \) units/neurons), activation function \( \sigma \) (Sigmoid, ReLu, ELU, tanh, etc), and parameters \( \theta = (w_\ell, b_\ell)_\ell \in \mathbb{R}^{NK} \). (Composition with \( \sigma \) is componentwise). \( \mathcal{NN}_{d, d_{\text{out}}, L, K}^\sigma \): class of such NN

- Other types of NN: convolutional, recurrent, LSTM, etc
Universal approximation theorems

Aim: approximate a function $F : \mathbb{R}^d \mapsto \mathbb{R}^{d_{out}}$

- **Universal approximation theorem I** (Hornick et al. 89): the set $\mathcal{N}_d,d_{out},L = \bigcup_K \mathcal{N}_d,d_{out},L,K$ is dense in $L^2(\nu)$ for any finite measure $\nu$ on $\mathbb{R}^d$, once $\sigma$ is continuous and non-constant:

$$\inf_{\Phi \in \mathcal{N}_d,d_{out},L,K} \int |F(x) - \Phi(x)|^2 \nu(dx) \to 0,$$

as $K$ goes to infinity.

- **Universal approximation theorem II for derivatives** (Hornick et al. 91, Attali and Pagès 96): $d_{out} = 1$, $L = 1$. If the (non constant) activation function $\sigma$ is $C^k$, then $\mathcal{N}_d,1,1$ arbitrarily approximates $F$ and all its derivatives up to order $k$ on any compact set of $\mathbb{R}^d$. 
Finding optimal parameters in $\mathcal{NN}_{d,d_{\text{out}},L,m}^\sigma$

Minimize over $\theta$

$$\mathbb{E}[\ell(X, \theta)] \text{ with } \ell(x, \theta) = |F(x) - \Phi(x; \theta)|^2,$$

with $X \sim \nu(dx)$ (training distribution)

- Stochastic gradient descent (SGD, a.k.a. Robbins Monroe) methods
  - Samples $X^{(m)}, m = 1, \ldots, M$ of $\nu(dx)$ (training input data)
  - Iterations: $(\gamma_m)_m$ learning rate, typically $\gamma_m = Cm^{-p}, p \in (1/2, 1]$

$$\theta^{m+1} = \theta^m - \gamma_m D_\theta \ell(X^{(m)}, \theta^m), \quad m = 1, \ldots, M.$$

- Key feature: reverse-mode automatic differentiation (backpropagation) for computing derivatives of neural networks

- Variants:
  - Mini-batch SGD
  - Adam optimizer, etc

→ Implemented in TensorFlow, Keras, PyTorch, etc
Deep learning for PDEs: direct approximation

Deep Galerkin Method: Sirignano and Spiliopoulos


DGM Algorithm

(i) Represent the solution $v$ of PDE as a (deep) neural network DNN: $(t, x) \mapsto U(t, x; \theta)$ with parameter $\theta$.

(ii) Learn the optimal parameter $\theta^*$ by minimizing a “good” objective function via stochastic gradient-descent (SGD) method
Details on DGM

- **Training measures:** $\nu_T, \nu_d$ be measures with support on $[0, T]$ and $\mathbb{R}^d$ (random sampling of time and space points)

- **Objective function:**

$$J(\theta) = \left\| \frac{\partial U}{\partial t}(., .; \theta) + \frac{1}{2} \Delta_x U(., .; \theta) - f(., U(., .; \theta), D_x U(., .; \theta)) \right\|_{L^2([0, T] \times \mathbb{R}^d, \nu_T \times \nu_d)}$$

$$+ \left\| U(T, .; \theta) - g \right\|_{L^2(\mathbb{R}^d, \nu_d)}$$

**Comments on DGM:**

+ Simple to implement (TensorFlow)

- Performance highly relies on the choice of the training measures: trade-off between exploration and exploitation!
BSDE representation: Pardoux-Peng 92

- Forward process $X \leftrightarrow \frac{1}{2} \Delta x$: $(W_t)_t$ d-dimensional Brownian motion

$$X_t = X_0 + W_t, \quad 0 \leq t \leq T.$$  

- Backward SDE $(Y, Z)$:

$$Y_t = g(X_T) - \int_t^T f(s, X_s, Y_s, Z_s)ds - \int_t^T Z_s.dW_s, \quad 0 \leq t \leq T.$$  

Connected (by Itô’s formula) to PDE (1) via:

$$Y_t = \nu(t, X_t), \quad Z_t = D_x\nu(t, X_t).$$

- Simulation of $(X, Y, Z) \rightarrow$ computation of $\nu$ and $D_x\nu$.  

- Simulation of $(X, Y, Z) \rightarrow$ computation of $\nu$ and $D_x\nu$.  

Time discretization

- Time grid $\pi = \{t_i, i = 0, \ldots, N\}$ of $[0, T]$ with time step $\Delta t_i = \frac{T}{N}$.
  Denote $\Delta W_{t_i} = W_{t_{i+1}} - W_{t_i}$.

- Euler scheme for backward SDE $(Y, Z)$: starting from $Y_{t_N} = g(X_{t_N})$
  
  $$Y_{t_i} \sim Y_{t_{i+1}} - f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}) \Delta t_i - Z_{t_i} \cdot \Delta W_{t_i}$$
Time discretization

- Time grid \( \pi = \{ t_i, i = 0, \ldots, N \} \) of \([0, T]\) with time step \( \Delta t_i = \frac{T}{N} \). Denote \( \Delta W_{t_i} = W_{t_{i+1}} - W_{t_i} \).

- Euler scheme for backward SDE \((Y, Z)\) starting from \( Y_{t_N} = g(X_{t_N}) \)

\[
Y_{t_i} \approx Y_{t_{i+1}} - f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}) \Delta t_i - Z_{t_i} \cdot \Delta W_{t_i}
\]

\( \rightarrow \) Estimation of \( Z \) and \( Y \) in two steps:

\[
\begin{align*}
Z_{t_i}^{\pi} & = \mathbb{E}_i \left[ Y_{t_{i+1}}^{\pi} \frac{\Delta W_{t_i}}{\Delta t_i} \right] \\
Y_{t_i}^{\pi} & = \mathbb{E}_i \left[ Y_{t_{i+1}}^{\pi} \right] - f(t_i, X_{t_i}, Y_{t_i}^{\pi}, Z_{t_i}^{\pi}) \Delta t_i, \quad i = N - 1, \ldots, 0.
\end{align*}
\]

- Simulation of \((Y_{t_i}^{\pi}, Z_{t_i}^{\pi}) = (v_i(X_{t_i}), z_i(X_{t_i}))\): conditional expectation \( \mathbb{E}_i \) approximation by regression methods, e.g. on basis functions, see Bouchard, Touzi (04), Gobet, Lemor, Warin (05), Bender, Steiner (13), etc.
Han, E., Jentzen (17,18) approach: deep BSDE

- Write the discrete-time BSDE in forward Euler form

\[
Y_{t_{i+1}} \simeq Y_{t_i} + f(t, X_{t_i}, Y_{t_i}, Z_{t_i}) \Delta t_i + Z_{t_i} \Delta W_{t_i}, \quad i = 0, \ldots, N - 1.
\]

\[=: F(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}, \Delta t_i, \Delta W_{t_i})\]

- Approximate \(Y_0\) by a DNN at time 0: \(Y_0 \simeq U(X_0; \theta_0) := Y_0^\theta\)

- Approximate \(Z_{t_i}\) by a DNN at time \(t_i\): \(Z_{t_i} \simeq Z(X_{t_i}; \theta_i)\)

- Simulate \(Y_{t_i}^\theta\) in forward induction by

\[
Y_{t_{i+1}}^\theta := F(t_i, X_{t_i}, Y_{t_i}^\theta, Z(\pi_{t_i}; \theta_i), \Delta t_i, \Delta W_{t_i})
\]

\[\rightarrow \text{Output}: Y_{t_N}^\theta \text{ aiming to match } g(X_{t_N})\]

\[\text{Minimize over } \theta = (\theta_0, \ldots, \theta_{N-1})\]

\[
\mathbb{E}\left| Y_{t_N}^\theta - g(X_{t_N}) \right|^2
\]
Some remarks on deep BSDE

+ Simple to implement
  - Huge NN composed of DNN at each period with parameters \( \theta = (\theta_0, \ldots, \theta_{N-1}) \)
  - Unstable and may diverge:
    - Global learning of the optimal parameters can be a daunting task when too many time steps \( N \)
    - SGD over the global DNN may be trapped in local minimizer
Our approach: deep BDP

- **Deep Backward Dynamic Programming (DBDP):**
  - Simultaneous estimation of $Y$ and $Z$ by DNN (two possible algorithms)
  - Backward scheme: learn the optimal parameters backward in time via the training of the forward process $X$

- Extension to free boundary problems (VI) $\leftrightarrow$ reflected BSDE $\leftrightarrow$ Optimal stopping problems

- Convergence analysis

- Numerical and comparison tests
Algo DBDP1: Representation of $Y$ and $Z$ by DNN

DBDP1 Algorithm

(i) Represent the solution $(Y_{t_i}, Z_{t_i}) = (v(t_i, X_{t_i}), D_x v(t_i, X_{t_i}))$ of BSDE as a pair of DNN $(U_i(\cdot; \theta), Z_i(\cdot; \theta))$ with parameter $\theta$

(ii) Learn the optimal parameters step by step in a Backward way, starting from the known terminal condition $g$
Algorithm 1 DBDP1 Algorithm

Input:

1: Initialize $\hat{U}^{(1)}_N = g$
2: for $i = N - 1, \ldots, 0$ do
3: Compute by SGD based on training data of $(X_{t_i}, X_{t_{i+1}}, \Delta W_{t_i})$

$$\theta_i^* \in \arg\min_{\theta} \mathbb{E} \left| \hat{U}^{(1)}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, U_i(X_{t_i}; \theta), Z_i(X_{t_i}; \theta), \Delta t_i, \Delta W_{t_i}) \right|^2;$$

4: Assign $\hat{U}^{(1)}_i = U_i(\cdot; \theta_i^*);$
5: Assign $\hat{Z}^{(1)}_i = Z_i(\cdot; \theta_i^*);$
6: end for

Output:

- $(\hat{U}^{(1)}_i)_{i=0}^N$: Estimates of $v(t_i, \cdot)$ for $i = 0, \ldots, N;$
- $(\hat{Z}^{(1)}_i)_{i=0}^N$: Estimates of $D_x v(t_i, \cdot)$ for $i = 0, \ldots, N;$
Algo DBDP2: Representation of $Y$ only by DNN

DBDP2 Algorithm

(i) Represent the solution $Y_{t_i} = \nu(t_i, X_{t_i})$ of the BSDE as a DNN $U_i(.; \theta)$ with parameter $\theta$.

(ii) Represent $Z_{t_i} \simeq D_x U_i(X_{t_i}; \theta)$.

(iii) Learn the optimal parameters in a Backward way, starting from the known terminal condition $g$.
Algorithm 2 DBDP2 Algorithm

Input:
1: Initialize $\hat{U}^{(2)}_N = g$
2: for $i = N - 1, \ldots, 0$ do
3: Compute by SGD

$$\theta_i^* \in \arg\min_{\theta} \mathbb{E} \left| \hat{U}^{(2)}_{i+1}(X_{t_{i+1}}) - F(t_i, X_t, U_i(X_t; \theta), D_x U_i(X_t; \theta), \Delta t_i, \Delta W_t) \right|^2;$$

4: Assign $\hat{U}^{(2)}_i = U_i(\cdot; \theta^*_i)$;
5: end for

Output:
- $(\hat{U}^{(2)}_i)_{i=0}^N$: Estimates of $v(t_i, \cdot)$ for $i = 0, \ldots, N$;
- $(\hat{Z}^{(2)}_i)_{i=0}^N$: Estimates of $D_x v(t_i, \cdot)$ for $i = 0, \ldots, N$;
Extensions to variational inequalities

\[
\begin{aligned}
\min \left[ -\frac{\partial v}{\partial t} - \frac{1}{2} \Delta_x v + f(., v, D_x v), v - g \right] &= 0, \quad \text{on } [0, T) \times \mathbb{R}^d \\
v(T, .) &= g, \quad \text{on } \mathbb{R}^d,
\end{aligned}
\]

\[\leftrightarrow\text{ Reflected BSDE: } (Y, Z, K) \text{ solution to }\]

\[
\begin{aligned}
Y_t &= g(X_T) - \int_t^T f(s, X_s, Y_s, Z_s) ds - \int_t^T Z_s.dW_s + K_T - K_t, \\
Y_t &\geq g(X_t), \quad 0 \leq t \leq T, \\
\int_0^T (Y_t - g(X_t)) dK_t &= 0.
\end{aligned}
\]

\[\leftrightarrow\text{ Optimal stopping problems}\]
Algorithm 3 Deep RBDP Algorithm

Input:
1: Initialize $\hat{U}^{(R)}_N = g$
2: for $i = N - 1, \ldots, 0$ do
3:      Compute:

$$\theta_i^* \in \arg\min_{\theta \in \mathbb{R}^{Nm}} \mathbb{E} \left| \hat{U}^{(R)}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, U_i(X_{t_i}; \theta), Z_i(X_{t_i}; \theta), \Delta t_i, \Delta W_{t_i}) \right|^2,$$

(3)

4:      Assign

$$\hat{U}^{(R)}_i = \max \left[ U_i(.; \theta_i^*), g \right];$$

5: end for

Output:
- $(\hat{U}^{(R)}_i)_{i=0}^{N}$: Estimates of $v(t_i, .)$ for $i = 0, \ldots, N$;
- $(\hat{Z}^{(R)}_i)_{i=0}^{N}$: Estimates of $D_x v(t_i, .)$ for $i = 0, \ldots, N$;
Some remarks on Deep BDP

- Compared to regression-based approach, we estimate/learn \((Y_{t_i}, Z_{t_i})\) simultaneously by a pair of NN.

- Compared to deep BSDE approach, we solve local (one-period) problems that are less prone to be trapped in local minimizer.

- **Practical implementation**: at each time \(t_i\), initialize the parameters of the NN to the “optimal” parameters of the time \(t_{i+1}\) previously treated.
  - **Reduce significantly** the number of iterations in SGD after the first step at time \(t_{N-1}\).
Define a measure of the (squared) error for the Deep BDP scheme by

\[
E[(\hat{U}, \hat{Z}), (Y, Z)] := \max_{i=0, \ldots, N-1} \mathbb{E}|Y_{t_i} - \hat{U}_i(X_{t_i})|^2 + \mathbb{E}\left[\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} |Z_t - \hat{Z}_i(X_{t_i})|^2 dt\right].
\]

**Theorem**

\[
E[(\hat{U}, \hat{Z}), (Y, Z)] \leq E^{time} + E^{NN}
\]

with

- \(E^{time} \to 0\) when \(N\) goes to infinity, \(= 0(1/N)\) when \(f, g\) are Lipschitz
- \(E^{NN} \leq C \sum_{i=0}^{N-1} \left[N \inf_{\theta} \mathbb{E}|\hat{v}_i(X_{t_i}) - U_i(X_{t_i}; \theta)|^2 + \inf_{\theta} \mathbb{E}|\hat{z}_i(X_{t_i}) - Z_i(X_{t_i}; \theta)|^2\right].\)
Test 1: a semi-linear PDE with quadratic gradient term

\[ f(t, x, y, z) = |z|^2: \]

\[
\begin{cases}
\frac{\partial v}{\partial t} + \frac{1}{2} \Delta_x v = |D_x v|^2, & (t, x) \in [0, T) \times \mathbb{R}^d \\
v(T, x) = g(x)
\end{cases}
\]

→ Explicit solution (via Hopf-Cole transformation):

\[ v(0, x_0) = -\ln \left( \mathbb{E} \left[ \exp \left( -g(x_0 + W_T) \right) \right] \right). \]
Numerical results: $d = 20$, $L = 2$ hidden layers, and varying the number $K$ of neurons.
Further numerical tests

- $L = 2$ hidden layers with $K = d + 10$ neurons
- $\sigma = \tanh$ as activation function
- Adam optimizer in TensorFlow and mini-batch with 1000 trajectories for SGD
Test 2a: An example of PDE with simple structure, \( T = 1, N = 120 \)

\[
\begin{align*}
  f(t, x, y, z) &= -\left( \cos(\bar{x})(e^{\frac{T-t}{2}} + \frac{1}{2}) - 0.2 \sin(\bar{x}) \right)e^{\frac{T-t}{2}} \\
  &\quad + \frac{1}{2} \left( \sin(\bar{x}) \cos(\bar{x})e^{T-t} \right)^2 + \frac{1}{2^d} (yz.1_d)^2 \\
  g(x) &= \cos(x)
\end{align*}
\]

with \( \bar{x} = \sum_{k=1}^{d} x_k \). → explicit analytic solution \( v(t, x) = e^{\frac{T-t}{2}} \cos(\bar{x}) \).

► Mean and standard deviation over 10 independent runs.

<table>
<thead>
<tr>
<th></th>
<th>Dimension ( d )</th>
<th>Mean value</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact solution</strong></td>
<td>5</td>
<td>0.46768</td>
<td></td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>5</td>
<td>0.4637038</td>
<td>0.004253</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>5</td>
<td>0.46335</td>
<td>0.00137</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>5</td>
<td>0.46562</td>
<td>0.0035</td>
</tr>
<tr>
<td><strong>Exact solution</strong></td>
<td>20</td>
<td>0.6728</td>
<td></td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>20</td>
<td>0.6760</td>
<td>0.00274</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>20</td>
<td>0.67102</td>
<td>0.00559</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>20</td>
<td>0.68686</td>
<td>0.002402</td>
</tr>
<tr>
<td><strong>Exact solution</strong></td>
<td>50</td>
<td>1.5909</td>
<td></td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>50</td>
<td>1.5903</td>
<td>0.006276</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>50</td>
<td>1.58762</td>
<td>0.00679</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>50</td>
<td>1.583023</td>
<td>0.00361</td>
</tr>
</tbody>
</table>

**Table:** Estimate of \( v(0, x_0) \) with \( x_0 = 1_d \).
Test 2b: An example of PDE with simple structure, $T = 2$, $N = 240$

- $d = 1$, Estimate of $v(0, x_0)$ with $x_0 = 1$. Mean and Std observed over 10 independent runs.

<table>
<thead>
<tr>
<th></th>
<th>Averaged value</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact solution</strong></td>
<td><strong>1.4686938</strong></td>
<td></td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>1.46332</td>
<td>0.01434</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>1.4387982</td>
<td>0.01354</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>NC</td>
<td>NC</td>
</tr>
</tbody>
</table>
Test 2b: Plots of $x \mapsto v(t,.)$ and $D_x v(t, x)$ by Deep BDP1
Test 3: An example of PDE with a more complex structure, $T = 1$, $N = 120$

\[
f(x, y, z) = -\ell(x) + \frac{1}{2\sqrt{d}}y(1_d \cdot z) + \frac{y^2}{2},
\]

and $\ell$ chosen s.t. the solution to the PDE is

\[
v(t, x) = \frac{T - t}{d} \sum_{k=1}^{d} (\sin(x_k)1_{x_k < 0} + x_k1_{x_k \geq 0}) + \cos \left( \sum_{k=1}^{d} kx_k \right).
\]

<table>
<thead>
<tr>
<th>Dimension $d$</th>
<th>Averaged value</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact solution</strong></td>
<td>2</td>
<td>0.570737</td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>2</td>
<td>0.5715359</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>2</td>
<td>0.5707974</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>2</td>
<td>0.57145</td>
</tr>
<tr>
<td><strong>Exact solution</strong></td>
<td>5</td>
<td>0.87715</td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>5</td>
<td>0.8666</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>5</td>
<td>0.83646</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>5</td>
<td>NC</td>
</tr>
<tr>
<td><strong>Exact solution</strong></td>
<td>8</td>
<td>1.16031</td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>8</td>
<td>1.169441</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>8</td>
<td>1.0758344</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>8</td>
<td>NC</td>
</tr>
<tr>
<td><strong>Exact solution</strong></td>
<td>10</td>
<td>-0.2148861</td>
</tr>
<tr>
<td>Deep BDP1</td>
<td>10</td>
<td>-0.3105</td>
</tr>
<tr>
<td>Deep BDP2</td>
<td>10</td>
<td>-0.3961</td>
</tr>
<tr>
<td>Deep BSDE</td>
<td>10</td>
<td>NC</td>
</tr>
</tbody>
</table>
Numerical tests for American option pricing

\( d \) assets of price \((X^1, \ldots, X^d)\) \(\sim\) Multi-dimensional BS model

- Parameters: \( r = 0.05, \sigma_i = 0.2, K = 1, X_0^i = 1, i = 1, \ldots, d \)

Geometric Put option: \( g(X_T) = (K - \prod_{i=1}^d X_T^i)_+, T = 1 \)

- Numerical results for deep RBDP:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>nb steps</th>
<th>( N )</th>
<th>value</th>
<th>std</th>
<th>bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td></td>
<td>0.061122</td>
<td>0.00015</td>
<td>0.060903</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td></td>
<td>0.10725</td>
<td>0.00012</td>
<td>0.10738</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td></td>
<td>0.129923</td>
<td>0.00016</td>
<td>0.12996</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td></td>
<td>0.15050</td>
<td>0.00010</td>
<td>0.1510</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td></td>
<td>0.16758</td>
<td>0.00016</td>
<td>0.1680</td>
</tr>
</tbody>
</table>

Table: Estimate of the American option price using Deep RBDP. Results from 40 independent runs of simulations.
When the solution has a “simple” structure, Deep BDP works very well in quite high dimension (at least 50)

When the solution has a “complex” structure, Deep BDP works well in medium dimension, but not so accurate in very high dimension

Extensions and perspectives:
- Fully nonlinear PDEs:
  - Combination of deep BDP and splitting method (Beck et al. 19)
- PDE in Wasserstein space of probability measures ↔ mean-field control/game problems: w.i.p.
References


Codes available at: https://github.com/comeh/

Thank you for your attention