Solving non-linear Kolmogorov equations by using deep learning: a numerical comparison of discretization schemes

Raffaele Marino

Laboratoire de Théorie des Communications,
Faculté Informatique et Communications,
École Polytechnique Féderale de Lausanne,
1015, Lausanne, Switzerland

Stockholm September 8, 2020

Outline

Introduction:

- **Definition Komlogorov's Equation;**
- **Examples Komlogorov's Equation;**
- **Solution of Komlogorov's Equation as SDE;**
- **Discretization schemes;**

The algorithm:

- **Existing algorithms and issues;**
- **Description of the learning algorithm;**

Results:

- **Black-Scholes equation in d=100 + 1;**
- **Hamilton-Jacobi-Bellman equation in d=100 +1;**
- **Allan-Cahn equation in d=100 + 1;**
- **An exact solvable non-linear diffusion equation;**

(Backward) Kolmogorov Equation

$$\frac{\partial g(\vec{x},t)}{\partial t} = - A(\vec{x},t)\cdot\nabla_{\vec{x}} g(\vec{x},t) - \frac{1}{2}\mathrm{Tr}\Big(BB^T(\vec{x},t)\mathrm{Hess}_{\vec{x}} g(\vec{x},t)\Big)$$
$$- f\Big(t,\vec{x},g(\vec{x},t),B^T(\vec{x},t)\nabla_{\vec{x}} g(\vec{x},t)\Big)$$

Terminal condition $g(\vec{x},T)=\phi(\vec{x})$ where $\phi:\mathbb{R}^d\to\mathbb{R}$ is a known function.

GOAL: WE WANT TO FIND $g(\vec{x},0)$

- Describes time evolution of a probability density of diffusion processes;
- $A(\vec{x},t), BB^T(\vec{x},t)$ drift and diffusion term, respectively;
- $f\Big(t,\vec{x},g(\vec{x},t),B(\vec{x},t)^T\nabla_{\vec{x}} g(\vec{x},t)\Big)$ is known

(Forward) Kolmogorov Equation

$$\frac{\partial g(\vec{x},t')}{\partial t'} = -\nabla_{\vec{x}} \cdot [A(\vec{x},t')g(\vec{x},t')] + \frac{1}{2}\mathrm{Tr}\Big(\mathrm{Hess}_{\vec{x}}[BB^T(\vec{x},t')g(\vec{x},t')]\Big)$$
$$- f\Big(t',\vec{x},g(\vec{x},t'),B^T(\vec{x},t')\nabla_{\vec{x}}g(\vec{x},t')\Big)$$

Initial condition $g(\vec{x},0) = \phi(\vec{x})$ where $\phi : \mathbb{R}^d \to \mathbb{R}$ is a known function.

GOAL: WE WANT TO FIND $g(\vec{x},T)$

- The Forward and Backward equations are equivalent to each other;
- The Forward equation gives more directly the values of measurable quantities as a function of the observed time, *t'*, and tends to be used more commonly in physical applications.

Kolmogorov's Equations

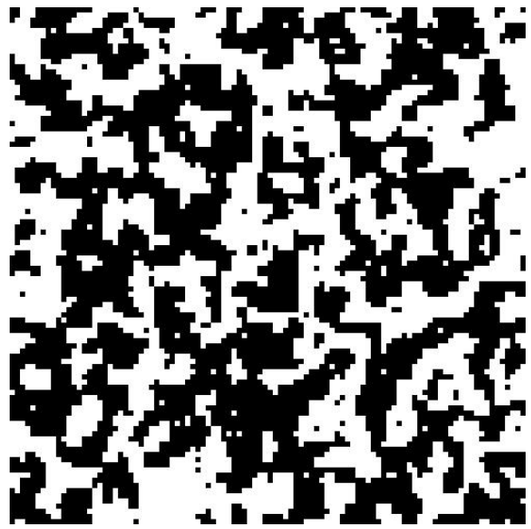
Finance:
(Backward) Black-Scholes equation

$$\frac{\partial g}{\partial t}(\vec{x},t) + \overline{\mu}\vec{x}\cdot\nabla_{\vec{x}} g(\vec{x},t) + \frac{\overline{\sigma}^2}{2}\sum_{i=1}^{d}|x_i|^2\frac{\partial^2 g}{\partial x_i^2}(\vec{x},t) +$$
$$-(1-\delta)Q(g(\vec{x},t))g(\vec{x},t) - R\,g(\vec{x},t) = 0$$

London Stock Exchange Group

Physics:
(Forward) Allan-Cahn equation

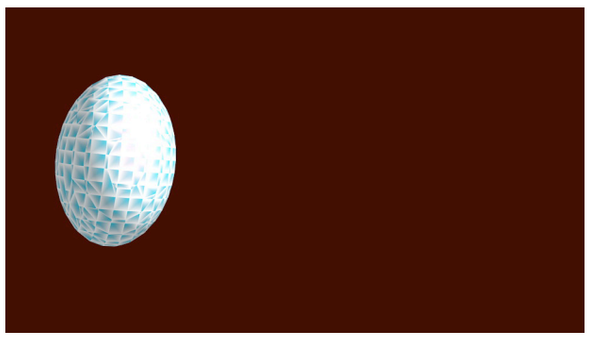
$$\frac{\partial}{\partial t'}g(\vec{x},t') = \Delta g(\vec{x},t') + \epsilon^{-2}[g(\vec{x},t') - g(\vec{x},t')^3]$$

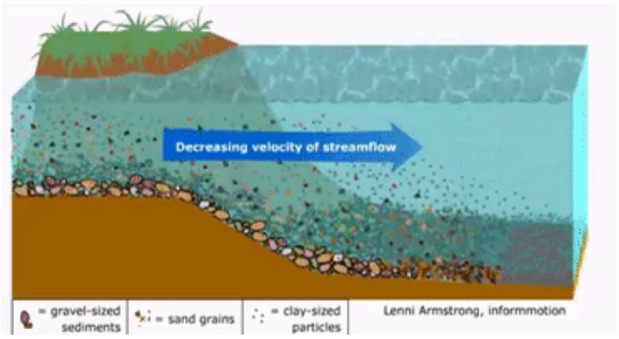


Kolmogorov's Equations

Physics:
(Backward) Fokker-Planck equation

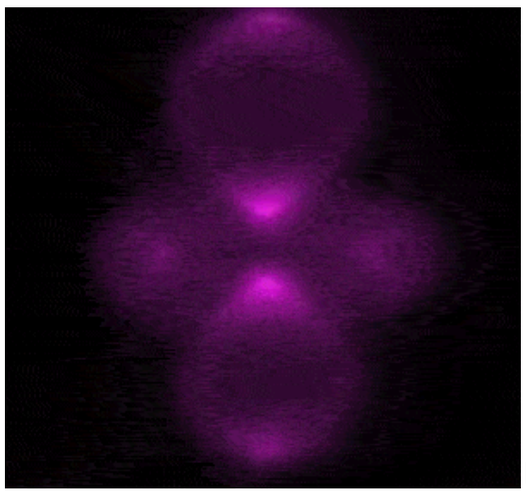
$$\frac{\partial g(\vec{x},t)}{\partial t} + \vec{\mu}(\vec{x},t)\cdot\nabla_{\vec{x}} g(\vec{x},t) + \sum_{i=1,j=1}^{d} D_{ij}(\vec{x},t)\frac{\partial^2 g(\vec{x},t)}{\partial x_j \partial x_i} = 0$$

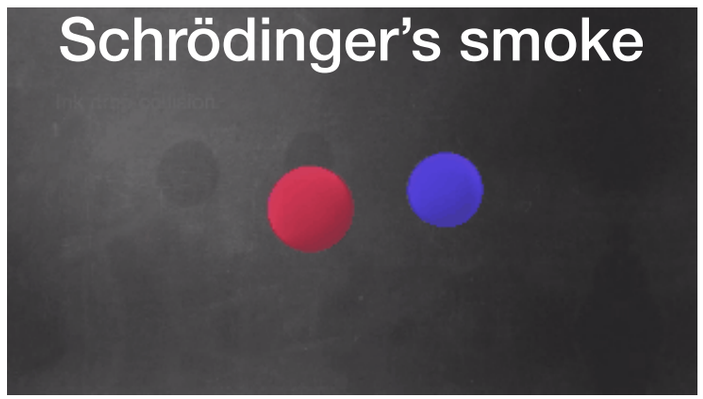




Physics:
(Forward) Schrödinger equation

$$i\hbar\frac{\partial}{\partial t'}\Psi(\vec{x},t') = \left[\frac{-\hbar}{2m}\Delta + V(\vec{x},t')\right]\Psi(\vec{x},t')$$





Purely Eulerian simulation of incompressible fluids. The fluid state is represented by a $\mathbb{C}^2$-valued wave function evolving under the Schrödinger equation subject to incompressibility constraints.

Stochastic processes

A stochastic process described by a conditional probability satisfying the Kolmogorov Equation is equivalent to the Langevin (integral) equation, where the integral is evaluated in Ito's sense.

$$\vec{X}(T) = \vec{X}(0) + \int_0^T A[\vec{X}(s), s]ds + \int_0^T B[\vec{X}(s), s]\, d\vec{W}(s)$$

It is known that a solution of Backward Kolmogorov's equation can be interpreted as a solution of BSDE :

$$g[\vec{X}(0), 0] = g[\vec{X}(T), T] - \int_0^T (\nabla_{\vec{x}} g[\vec{X}(s), s])^T B[\vec{X}(s), s] d\vec{W}(s)$$
$$+ \int_0^T f\big(s, \vec{X}(s), g[\vec{X}(s), s], B[\vec{X}(s), s]^T \nabla_{\vec{x}} g[\vec{X}(s), s]\big) ds$$

Stochastic numerical integration I

For implementing a discretisation of the SDE we divide the time interval $[0,T]$ into N sub-intervals of size $\tau = T/N$ at points $\tau_n = n\tau$, so that a function can be evaluated at times

$$\tau_0 = 0, \tau_1, \tau_2, \tau_3, \dots \ \tau_{N-1}, \tau_N = T\,.$$

Thus, the corresponding Wiener increments become $\Delta\vec{W}^n = \vec{W}(\tau_{n+1}) - \vec{W}(\tau_n)$.

By applying Ito's chain rule lemma and fixing the so-called "commutativity" condition on the diffusion tensor, i.e. $\sum_{k=1}^{d} B_{ij}[\vec{X}^n,\tau](\partial_{x_k}B_{kl}[\vec{X}^n,\tau]) = \sum_{k=1}^{d} B_{il}[\vec{X}^n,\tau](\partial_{x_k}B_{kj}[\vec{X}^n,\tau])$, one obtains:

$$X_i^{n+1} = X_i^n + A_i[\vec{X}^n,\tau_n]\tau + \sum_{j=1}^{d} B_{ij}[\vec{X}^n,\tau_n]\Delta W_j^n$$

$$+ \frac{1}{2}\sum_{j,k,l=1}^{d} B_{ij}[\vec{X}^n,\tau_n](\partial_{x_k}B_{kl}[\vec{X}^n,\tau_n])(\Delta W_j^n \Delta W_l^n - \tau\delta_{jl}) + \cdots$$

Stochastic numerical integration II

Euler-Maruyama (EM) scheme: **accuracy:** $O(\sqrt{\tau})$

$$(Y_{EM})_i^{n+1} = (Y_{EM})_i^n + A_i[\vec{Y}_{EM}^n, \tau_n]\tau + \sum_{j=1}^{d} B_{ij}[\vec{Y}_{EM}^n, \tau_n]\Delta W_j^n$$

Milstein (M) scheme: **accuracy:** $O(\tau)$

$$(Y_M)_i^{n+1} = (Y_M)_i^n + A_i[\vec{Y}_M^n, \tau_n]\tau + \sum_{j=1}^{d} B_{ij}[\vec{Y}_M^n, \tau_n]\Delta W_j^n +$$

$$+ \frac{1}{2}\sum_{j,k,l=1}^{d} B_{ij}[\vec{Y}_M^n, \tau_n](\partial_{x_k} B_{kl}[\vec{Y}_M^n, \tau_n])(\Delta W_j^n \Delta W_l^n - \delta_{jl}\tau)$$

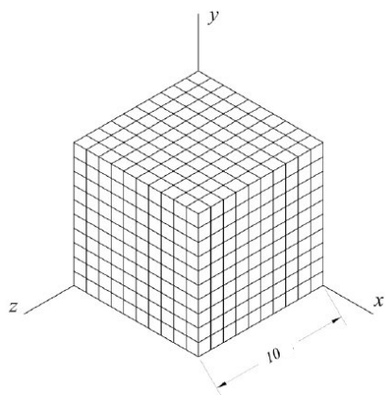
Leimkuhler-Matthews (LM) scheme: **accuracy:** $O(\tau^2)$

$$(Y_{LM})_i^{n+1} = (Y_{LM})_i^n + A_i[\vec{Y}_{LM}^n, \tau_n]\tau + \frac{1}{2}\sum_{j=1}^{d} B_{ij}[\vec{Y}_{LM}^n, \tau_n](\Delta W_j^n + \Delta W_j^{n+1}).$$

Algorithms for solving PDEs (I)

- **MCMs** for solving PDEs were developed by Fermi/Von Neumann for solving the neutron transport in the Manhattan Project.
- MC techniques are a set of algorithms used to calculate approximate numerical quantities or estimate the probabilities of an outcome of an experiment by making use of repeated random sampling.
- It is known that **Monte-Carlo sampling methods do not suffer from the curse of dimensionality if the goal of the problem is evaluate the solution of a PDE on single point.**
- However, to approximate a solution not only at a single value but, for example, on **a full hypercube $[u, v]^d$, there has been no known method not suffering from the curse of dimensionality.** In particular, there has been no known method that can provably be applied efficiently in high dimensions, say, $d \geq 100$.

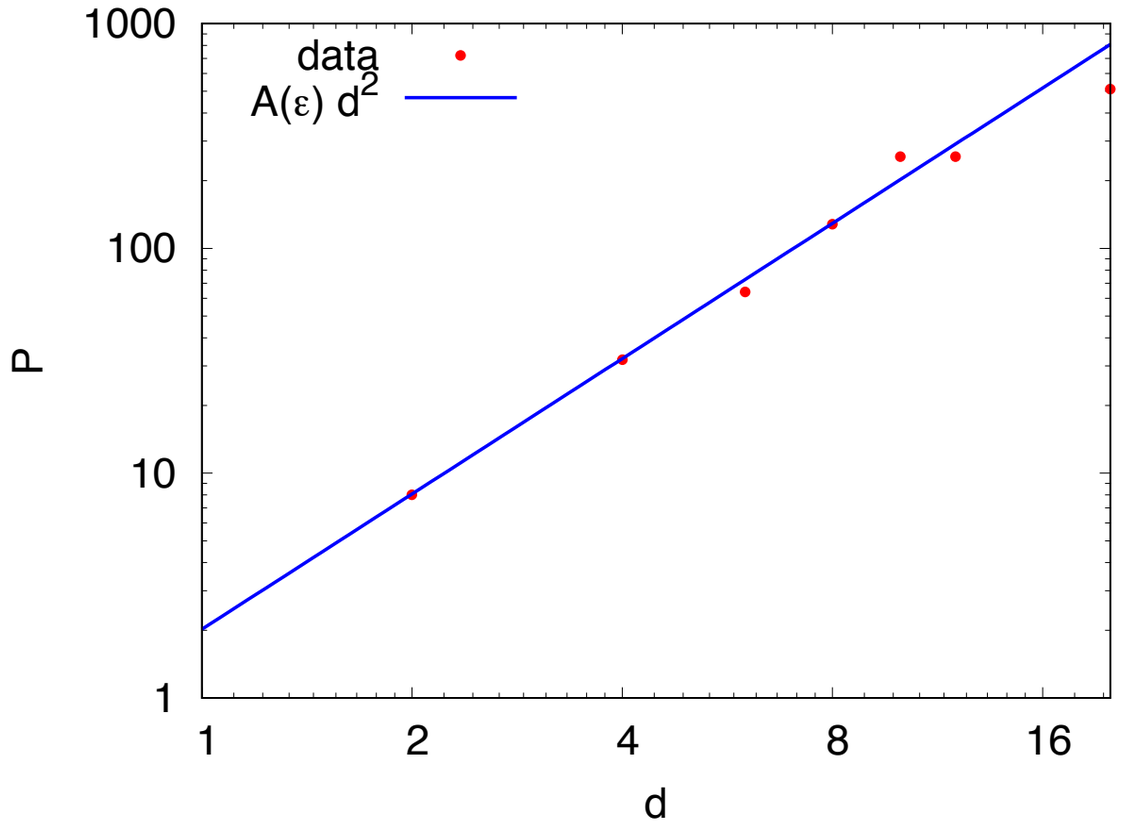
Es.:



⟶ **P~O(n^d)**

Algorithms for solving PDEs (II)

WE INTRODUCE DEEP LEARNING BASED ALGORITHMS FOR APPROXIMATING A SOLUTION OF KOLMOGOROV'S EQUATION ON A FULL HYPERCUBE $[u, v]^d$ THAT OVERCOME THE CURSE OF DIMENSIONALITY AND WE COMPARE, NUMERICALLY, DIFFERENT DISCRETISATION SCHEMES.



EPFL

(Deep) Basic Idea I

Let's discretise the formal solution of Backward Kolmogorov's equation:

$$g[\vec{Y}^{n+1}, \tau_{n+1}] = g[\vec{Y}^{n}, \tau_{n}] + \nabla g[\vec{Y}^{n}, \tau_{n}]^{T} B[\vec{Y}^{n}, \tau_{n}] \Delta \vec{W}^{n}$$
$$- f(\tau_{n}, \vec{Y}^{n}, g[\vec{Y}^{n}, \tau_{n}], B[\vec{Y}^{n}, \tau_{n}]^{T} \nabla g[\vec{Y}^{n}, \tau_{n}]) \tau$$

The only term that we do not know is $\nabla g[\vec{Y}^{n}, \tau_{n}]$

HOWEVER

We can imagine to approximate it by using a deep feed-forward neural network. (Universal Approximation Theorem)

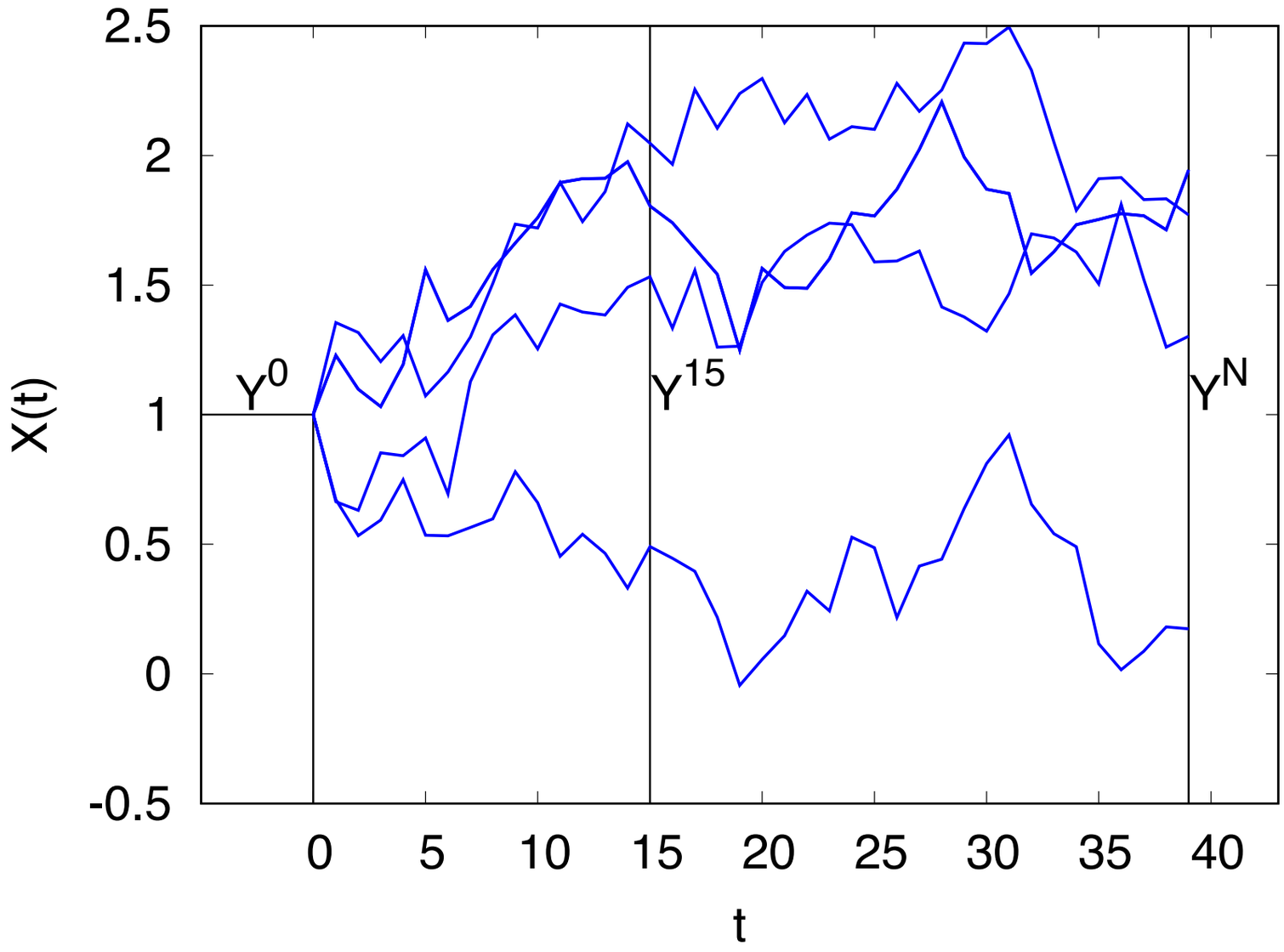
THEN

$\forall \tau_n$:

DATA INPUT (τ_n) $\longrightarrow$ $\mathcal{NN}(\vec{\theta}_n)$ $\longrightarrow$ $(\nabla g^{T} B)[\vec{Y}^{n} \mid \vec{\theta}_{n}]$

DATA INPUT (τ_n)

Ex.:



$$Y^0 : [1,1,1,1]^T$$

$$Y^{15} : [0.49,2.04,1.53,1.80]^T$$

$$Y^N : [0.17,1.77,1.30,1.94]^T$$

EPFL

(Deep) Basic Idea II

OUTPUT II

OUTPUT I

$g(Y^0,\tau_0)$ → $g(Y^1,\tau_1)$ → $g(Y^2,\tau_2)$ → ⋯ → $g(Y^{N-1},\tau_{N-1})$ → $\hat{g}(Y^N,\tau_N)$

$B^T\nabla g(Y^0\,|\,\theta_0)$ → $B^T\nabla g(Y^1\,|\,\theta_1)$ → $B^T\nabla g(Y^2\,|\,\theta_2)$ → ⋯ → $B^T\nabla g(Y^{N-1}\,|\,\theta_{N-1})$

$\mathcal{NN}_{g_0}$ $\mathcal{NN}_{\nabla g_0}$ $\mathcal{NN}_1$ $\mathcal{NN}_2$ ⋯ $\mathcal{NN}_{N-1}$

Y^0 → Y^1 → Y^2 → ⋯ → Y^{N-1} → Y^N

ΔW^0 ΔW^1 ⋯ ΔW^{N-2} ΔW^{N-1}

INPUT

$t=0$ $t=\tau_1$ $t=\tau_2$ $t=\tau_{N-1}$ $t=\tau_N$

SGD

$$\mathcal{L}(\boldsymbol{\Theta}) := \mathbb{E}\left[\left|\left|\phi(\vec{Y}^N) - \hat{g}(\{\vec{Y}^n\}_{0\le n\le N}, \{\Delta\vec{W}^n\}_{0\le n\le N}|\vec{\Theta})\right|\right|^2\right].$$

EPFL

(Deep) Basic Idea III

Consider a variational class of functions or "deep networks" $\mathcal{NN}_{g_0}(\vec{x}\,|\,\theta_0) \in \mathbb{R}$ and $\mathcal{NN}_{\nabla g_s}(\vec{x}\,|\,\theta(s)) \in \mathbb{R}^d$ where θ_0 and $\theta(s)$, $0 \leq s \leq T$ is a set of "weights" to be optimised.
We want to minimise the loss functional :

$$\mathscr{L}[\theta_0, \theta(\cdot)] = \mathbb{E}\left[\left|\Phi(\vec{X}(T)) - \left\{\mathcal{NN}_{g_0}(\vec{x}\,|\,\theta_0) - \int_0^T ds f(s, \vec{X}(s), g(\vec{X}(s), s), \mathcal{NN}_{\nabla g_s}(\vec{X}(s)\,|\,\theta(s))) + \int_0^T ds\,(\mathcal{NN}_{\nabla g_s}(\vec{x}\,|\,\theta(s)))^T d\vec{W}(s)\right\}\right|^2\right]$$

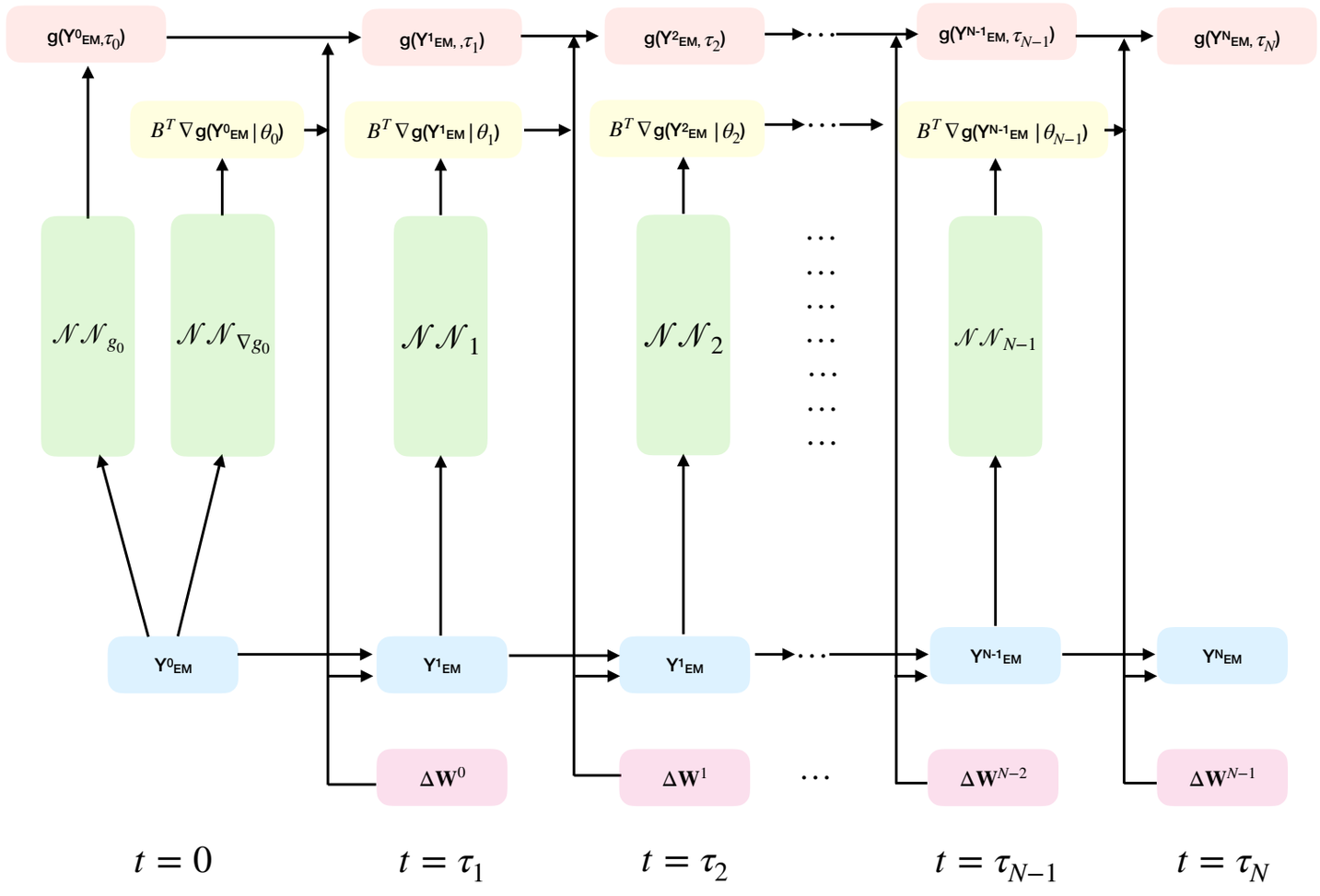
where $\vec{X}(s)$and $g(\vec{X}(s), s)$ are solutions of the FBSDE's

$$\begin{cases} d\vec{X}(s) = A(X(s), s)ds + B(X(s), s)^T d\vec{W}(s), \\ dg[\vec{X}(s), s] = -f\left(s, \vec{X}(s), g[\vec{X}(s), s], \mathcal{NN}_{\nabla g_s}(\vec{X}(s)\,|\,\theta(s))\right)ds + (\mathcal{NN}_{\nabla g_s}(\vec{X}(s)\,|\,\theta(s)))^T d\vec{W}(s) \end{cases}$$

with $g(\vec{X}(0),0) = \mathcal{NN}_{g_0}(x\,|\,\theta_0)$. The expectation in the loss function is over the Brownian trajectories $\vec{W}(s)$, $0 \leq s \leq T$ and over $\vec{X}(0) \in D$ uniformly distributed in a domain $D \subset \mathbb{R}^d$.
The non-linear Feynman-Kac Formula tells us that as long as the class of the deep networks is sufficiently expressive, there exists a minimiser $\hat{\theta}_0$, $\hat{\theta}(\cdot)$ such that the loss nearly vanishes, and $\mathcal{NN}_{g_0}(\vec{x}\,|\,\hat{\theta}_0) \approx g(\vec{x},0)$ and $\mathcal{NN}_{\nabla g_s}(\vec{x}\,|\,\hat{\theta}(s)) \approx B(\vec{x}, s)^T \nabla_{\vec{x}} g(\vec{x}, s)$.

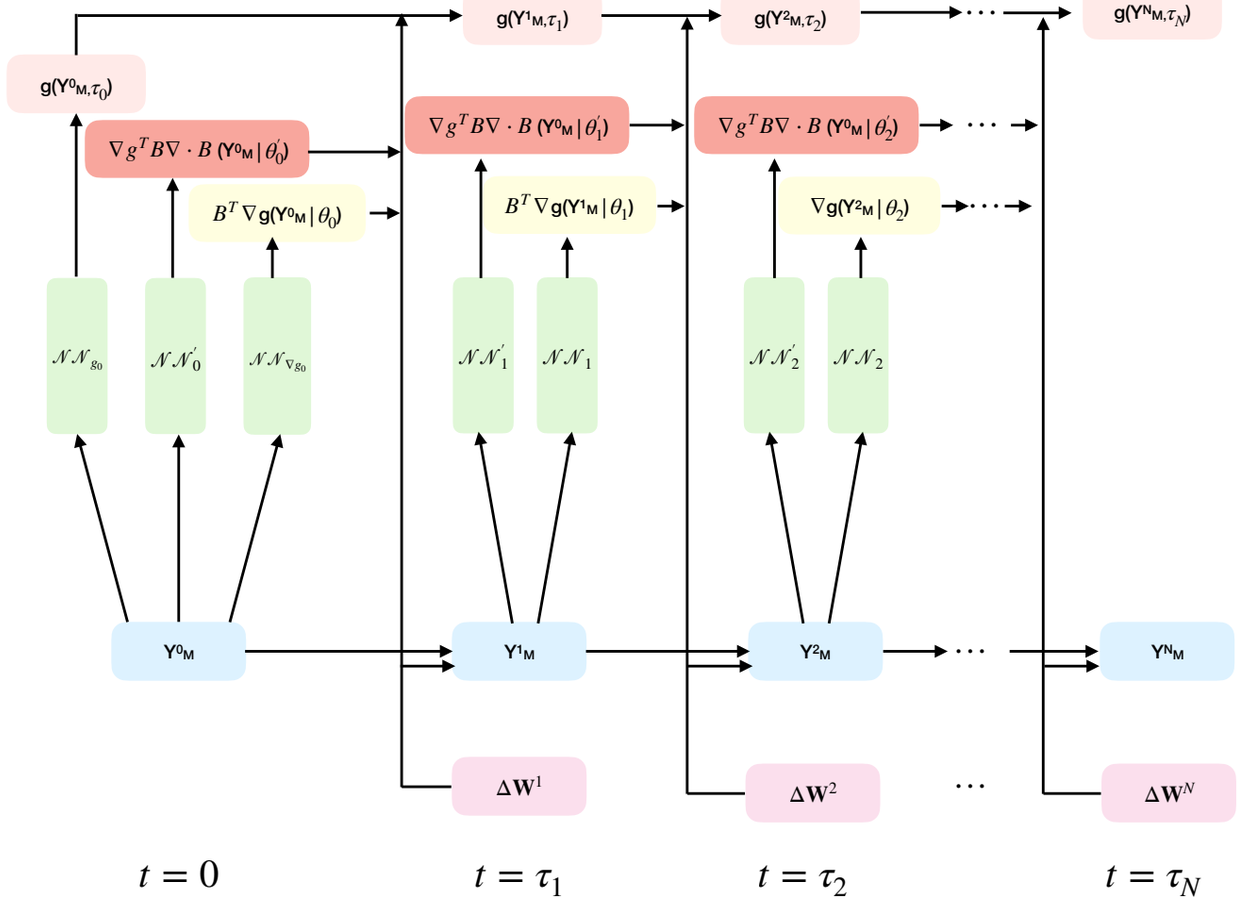
Architecture based on Euler-Maruyama (EM) scheme:

$$g[\vec{Y}^{n+1}_{EM}, \tau_{n+1}] = g[\vec{Y}^{n}_{EM}, \tau_n] + \nabla_{\vec{x}} g[\vec{Y}^{n}_{EM}, \tau_n]^T B[\vec{Y}^{n}_{EM}, \tau_n] \Delta \vec{W}^n$$
$$- f(\tau_n, \vec{Y}^{n}_{EM}, g[\vec{Y}^{n}_{EM}, \tau_n], B[\vec{Y}^{n}_{EM}, \tau_n]^T \nabla_{\vec{x}} g[\vec{Y}^{n}_{EM}, \tau_n]) \tau$$



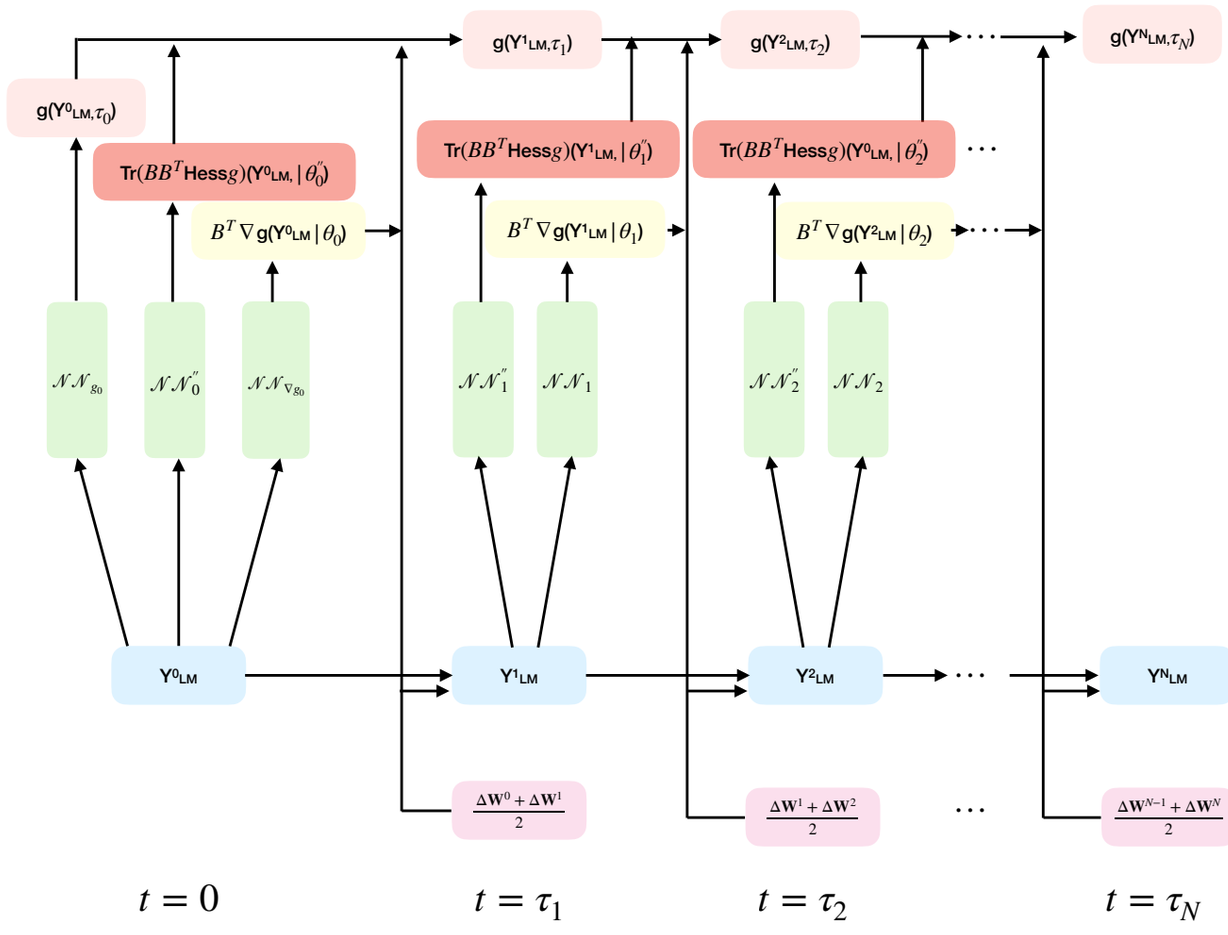
Architecture based on Milstein (M) scheme:

$$g[\vec{Y}_M^{n+1},\tau_{n+1}] = g[\vec{Y}_M^n,\tau_n] - f(\tau_n,\vec{Y}_M^n,g[\vec{Y}_M^n,\tau_n],B[\vec{Y}_M^n,\tau_n]^T\nabla_{\vec{x}}g[\vec{Y}_M^n,\tau_n])\tau$$
$$+\nabla g_{\vec{x}}[\vec{Y}_M^n,\tau_n]^T B[\vec{Y}_M^n,\tau_n]\Delta\vec{W}^n$$
$$+\frac{1}{2}\sum_{i,j=1}^{d}\Big\{\sum_{l,k=1}^{d}\partial_{x_l}g[\vec{Y}_M^n,\tau_n]B_{li}[\vec{Y}_M^n,\tau_n]\partial_{x_k}B_{kj}[\vec{Y}_M^n,\tau_n]\Big\}(\Delta W_i^n\Delta W_j^n-\tau\delta_{ij})$$



Architecture based on Leimkuhler-Matthews (LM) scheme:

$$g[\vec{Y}^n_{LM},\tau_{n+1}] = g[\vec{Y}^n_{LM},\tau_n] - f(\tau_n,\vec{Y}^n_{LM},g[\vec{Y}^n_{LM},\tau_n],B[\vec{Y}^n_{LM},\tau_n]^T\nabla_{\vec{x}}g[\vec{Y}^n_{LM},\tau_n])\tau$$
$$+\frac{1}{2}\nabla g[\vec{Y}^n_{LM},\tau_n]^T B[\vec{Y}^n_{LM},\tau_n](\Delta W^n_j + \Delta W^{n+1}_j)$$
$$-\frac{1}{4}\mathrm{Tr}(BB^T[\vec{Y}^n_{LM}(\tau_n),\tau_n]\mathrm{Hess}_{\vec{x}}g[\vec{Y}^n_{LM}(\tau_n),\tau_n])\tau$$



Results I - (backward) Black-Scholes equation in d=100 + 1

Problem:

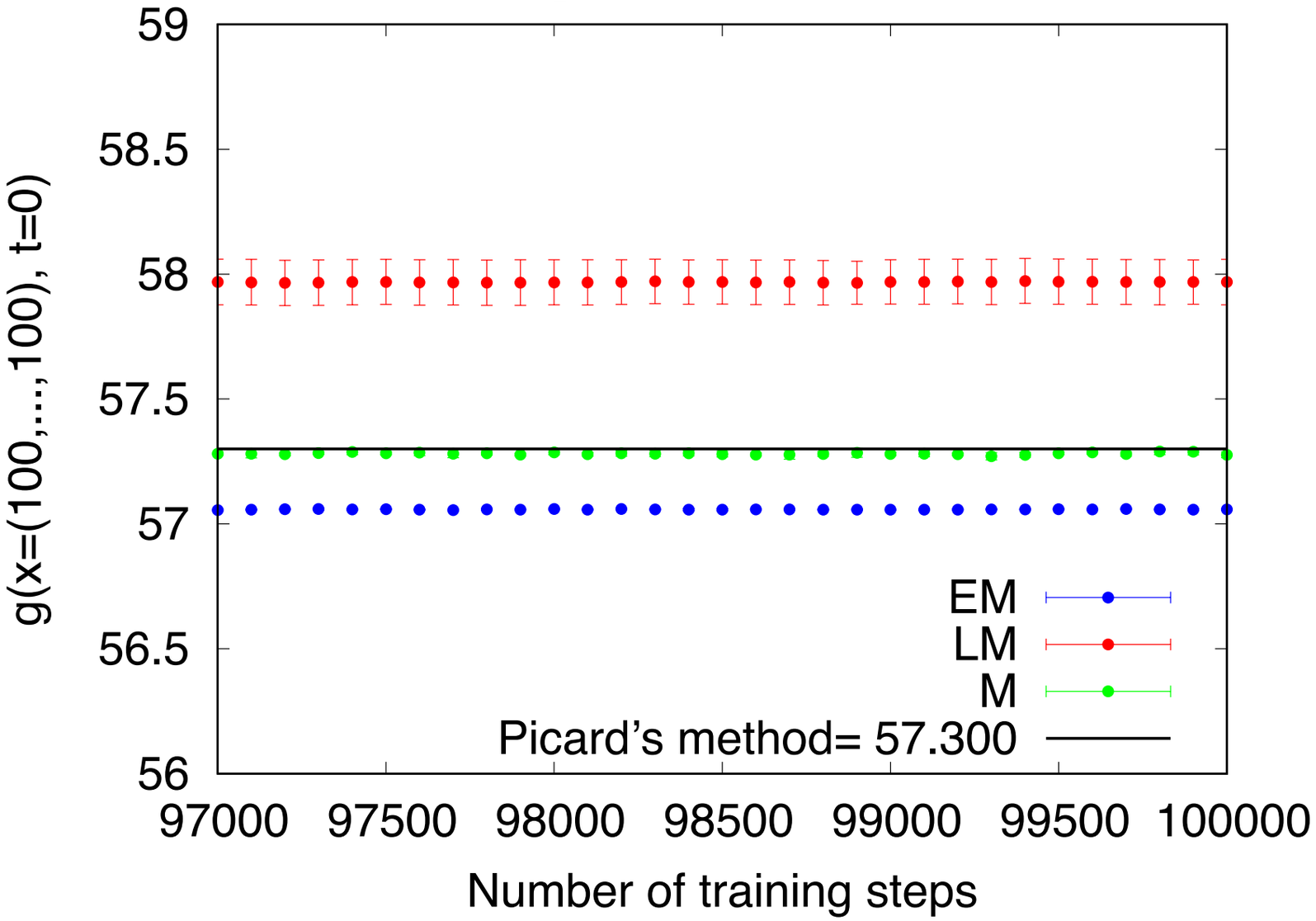
$$\frac{\partial g}{\partial t}(\overrightarrow{x},t) + \overline{\mu}\,\overrightarrow{x}\cdot\nabla_{\overrightarrow{x}} g(\overrightarrow{x},t) + \frac{\overline{\sigma}^2}{2}\sum_{i=1}^{d} |x_i|^2 \frac{\partial^2 g}{\partial x_i^2}(\overrightarrow{x},t) = (1-\delta)Q(g(\overrightarrow{x},t))g(\overrightarrow{x},t) + R\,g(\overrightarrow{x},t)$$

Where R is the interest rate of the risk free asset, and $Q(y)$ is a piece-wise linear function given by:

$$Q(y) = \textbf{ReLU}\left(\textbf{ReLU}(y-v_h)\frac{\gamma_h-\gamma_l}{v_h-v_l} + \gamma_h - \gamma_l\right) + \gamma_l\,.$$

Numerically we have chosen the values as:
$\delta = 2/3, R = 0.02, \overline{\mu} = 0.02, \overline{\sigma} = 0.2, v_h = 50, v_l = 70, \gamma_h = 0.2, \gamma_l = 0.02.$
We fix the dimension to $d = 100$, and, therefore, we impose the equation to live on the space $[0,T]\times\mathbb{R}^{100}$, with $T = 1$, and we fix the terminal condition be $g(\overrightarrow{x},T) = \phi(\overrightarrow{x}) = \min\{x_1,\ldots,x_{100}\}$.
Because the exact solution of the equation is not known, we confine ourselves on a single point $\overrightarrow{x} = (100,\ldots,100)$, where analysis performed with the multilevel Picard method indicates that the value of $g_{\text{Picard}}(\overrightarrow{x} = (100,\ldots,100),0) \approx 57.300$.

Results I - (backward) Black-Scholes equation in d=100 + 1



The figure shows the value of $g(\vec{x} = (100,\ldots,100),0)$ returned by EM (blue points), M (green points) and LM (red points) as function of the number of training steps done. Each point is the mean over five independent runs. The number of equidistant time steps was fixed at 40, i.e. $N = 40$. All the parameters were initialized randomly uniformly between [-1,1]. The total number of training steps was 10^5, while learning rate was chosen dynamically. The black horizontal line identifies the multilevel Picard method value.

Results II — (backward) Hamilton-Jacobi-Bellman equation in d=100+1

Problem:

$$\frac{\partial g}{\partial t}(\vec{x}, t) + \Delta g(\vec{x}, t) = \lambda \, || \nabla g(\vec{x}, t) ||^2$$

where λ is a positive constant representing the strength of the control.

With terminal condition $g(\vec{x}, T) = \phi(\vec{x}) = \ln((1 + || \vec{x} ||^2)/2$ with $\vec{x} \in \mathbb{R}^d$

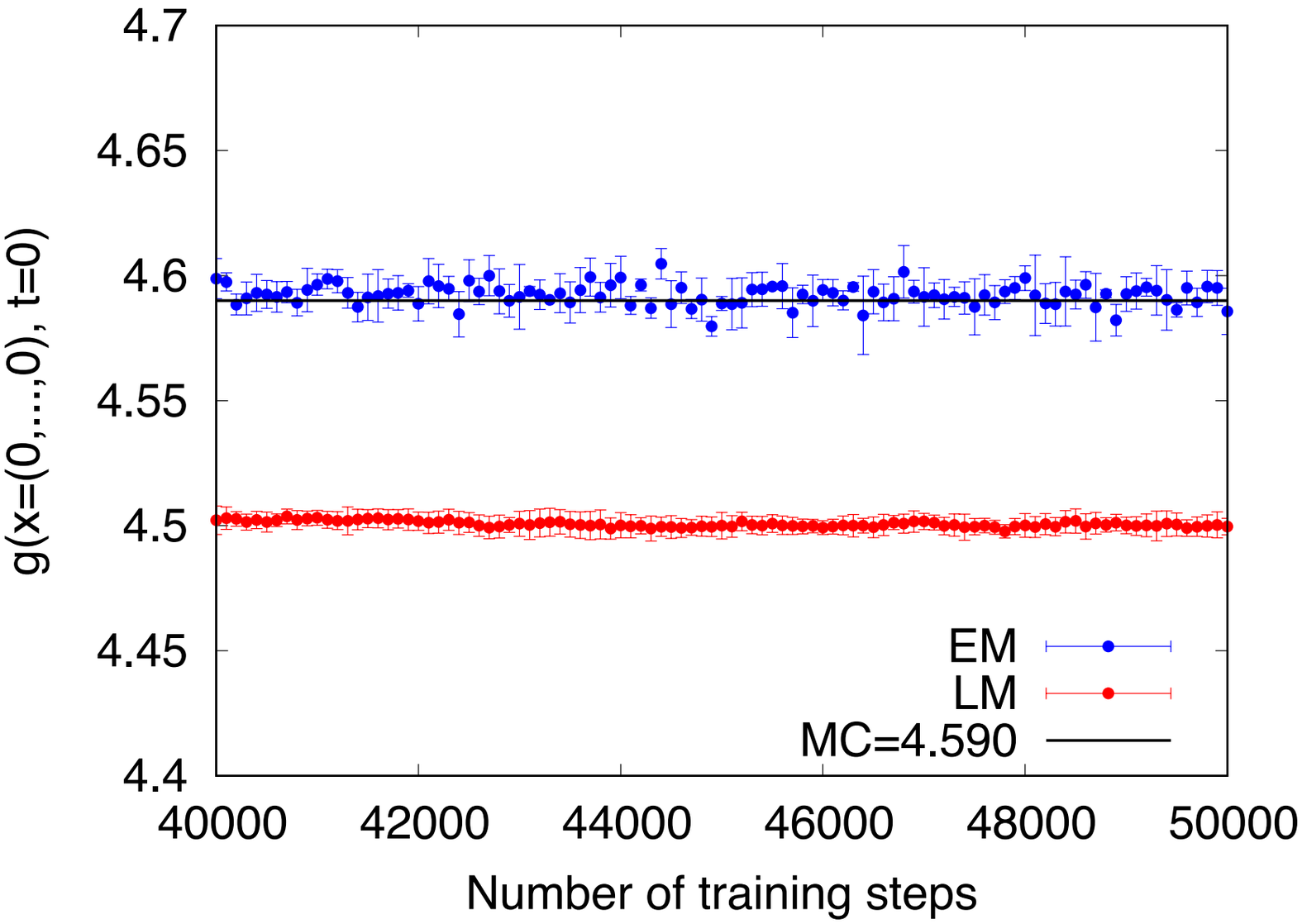
Exact solution: $g(\vec{x}, t) = -\lambda^{-1} \ln(\mathbb{E}[-\lambda g(\vec{x} + \sqrt{2}\vec{W}_{T-t})])$

where $\vec{W}_t$ is a standard Brownian motion.

GOAL:

$$g(\vec{x}, t = 0), \vec{x} = (0,...0) \in \mathbb{R}^{100}, \lambda = 1$$

Results II — (backward) Hamilton-Jacobi-Bellman equation in d=100+1

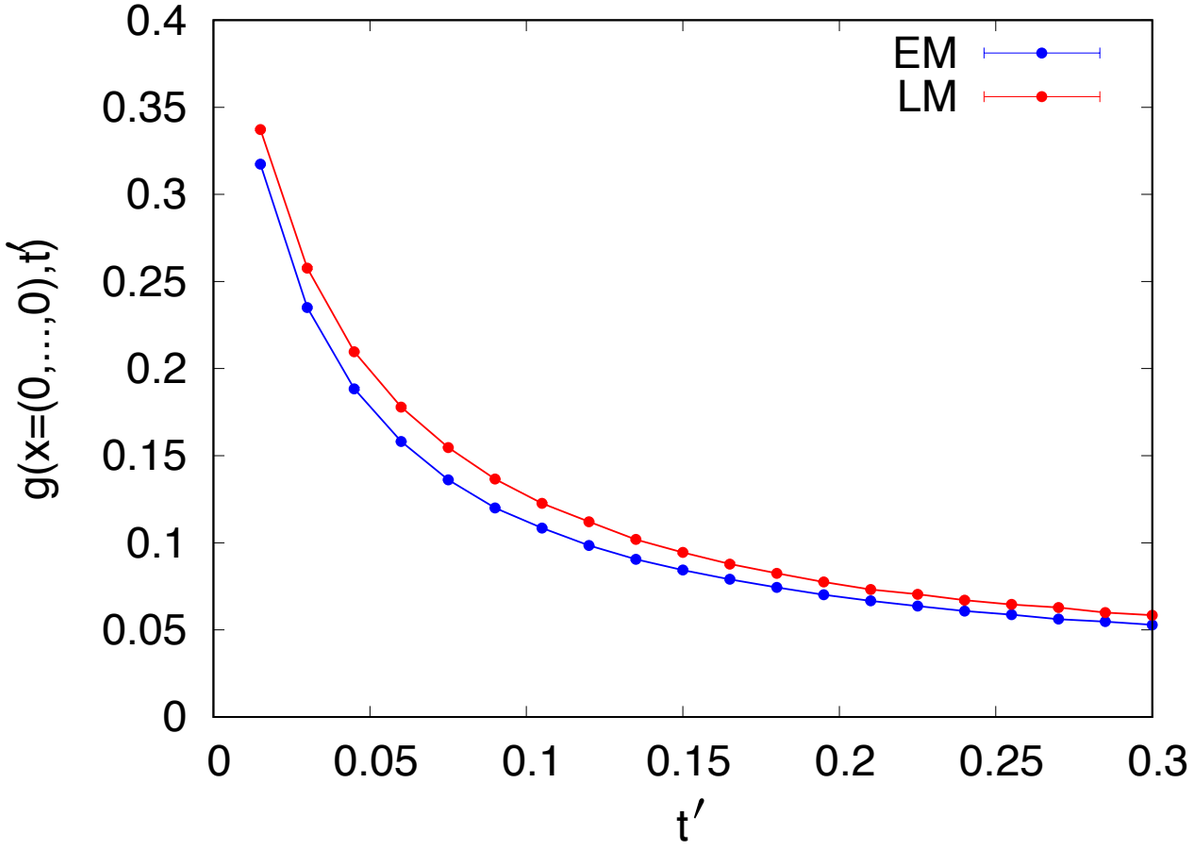


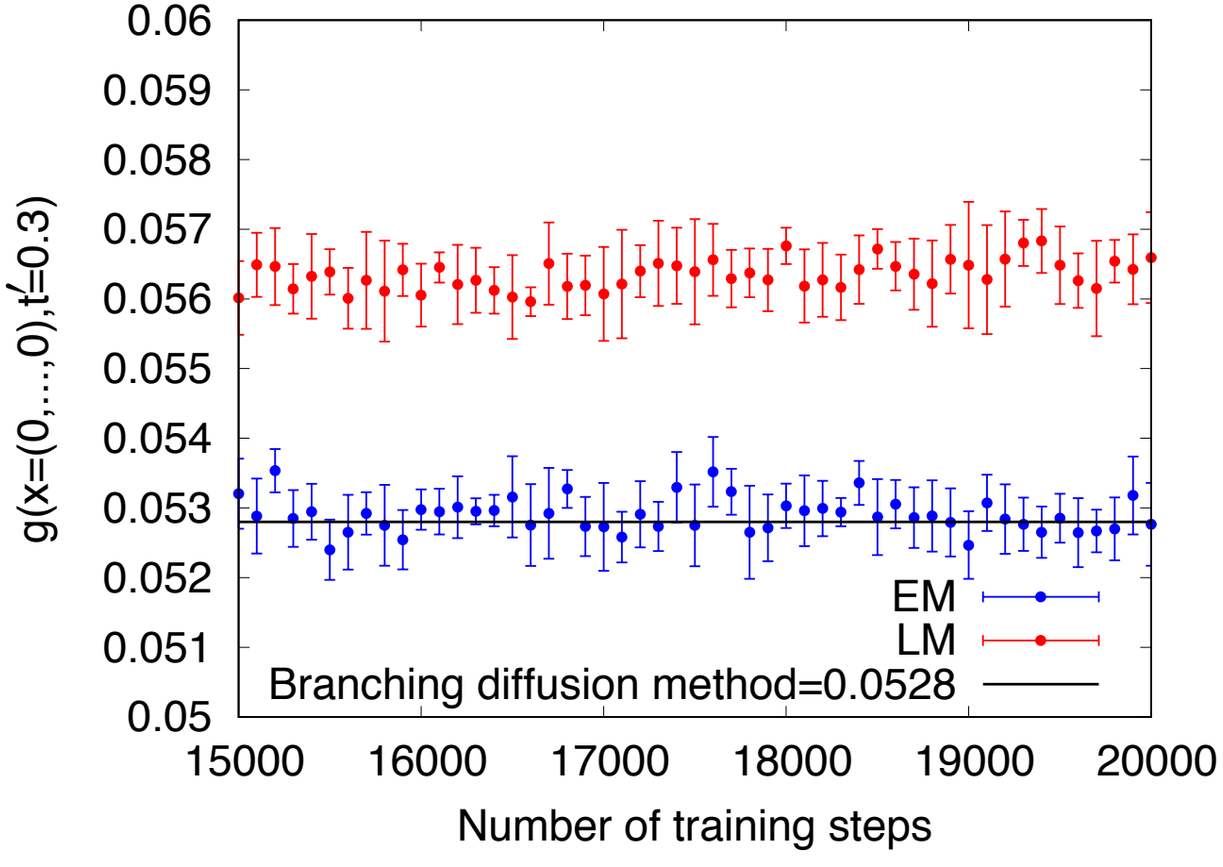
The figure shows the comparison between the Euler-Maruyama (blue points) and the Leimkuhler-Mattews (red points) schemes in approximating the solution $g(\overrightarrow{x}, t = 0)$, with $\lambda = 1$, at the origin of $\mathbb{R}^{100}$, as a function of the training steps. Each point is the average over 5 experiments. Error bars are standard deviations. The black line is the exact value obtained by standard Monte Carlo on the exact solution.

Results III — (forward) Allan-Cahn equation in d=100 + 1

$$\frac{\partial}{\partial t'} g(\vec{x}, t') = \Delta g(\vec{x}, t') + (g(\vec{x}, t') - g(\vec{x}, t')^3)$$

Initial conditions: $g(\overrightarrow{x},0) = \phi(\overrightarrow{x}) = (2 + 0.4||\overrightarrow{x}||^2)^{-1}$





Results IV — (backward) An exactly solvable non-linear diffusion equation

Problem:

$$\frac{\partial g(\overrightarrow{x},t)}{\partial t} + D\Delta_x g(\overrightarrow{x},t) = -\left[2Dg(\overrightarrow{x},t) - \frac{1}{d} - D\right]\sum_{i=1}^{d}\frac{\partial g(\overrightarrow{x},t)}{\partial x_i}$$

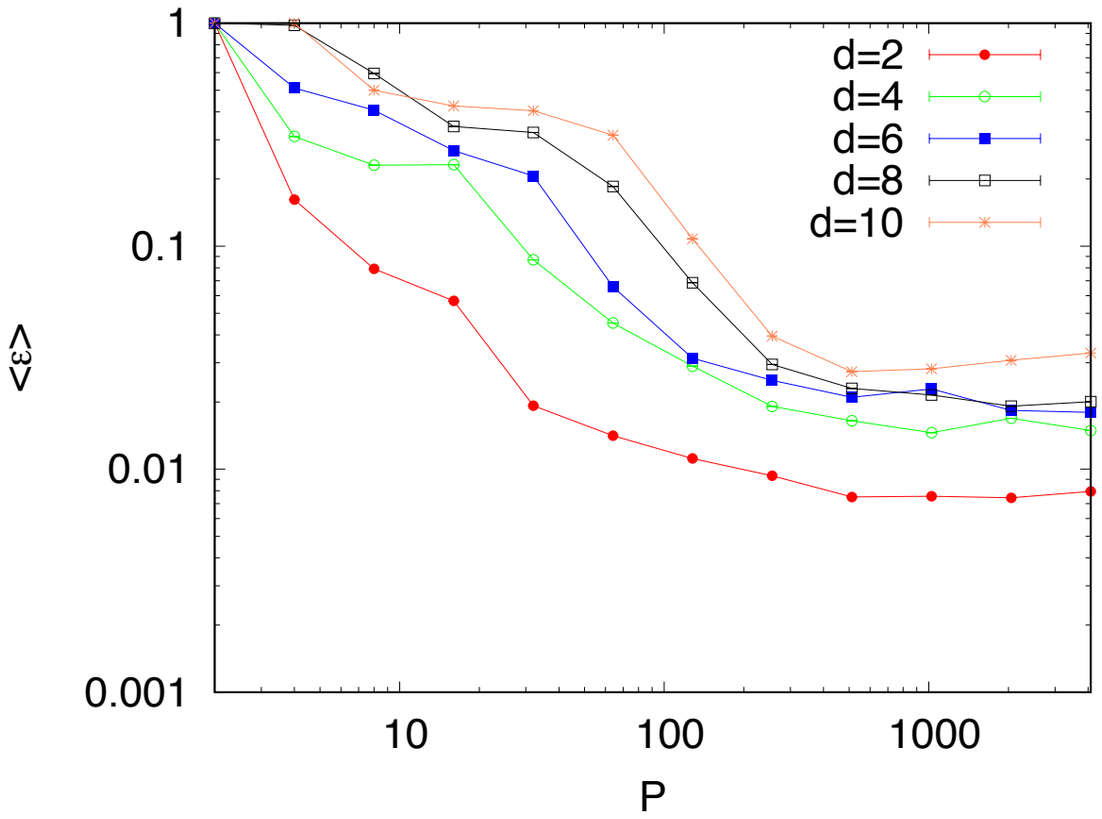
With terminal condition $g(\overrightarrow{x},T) = \frac{1}{1+e^{-T-\sum_{i=1}^{d}x_i}}$

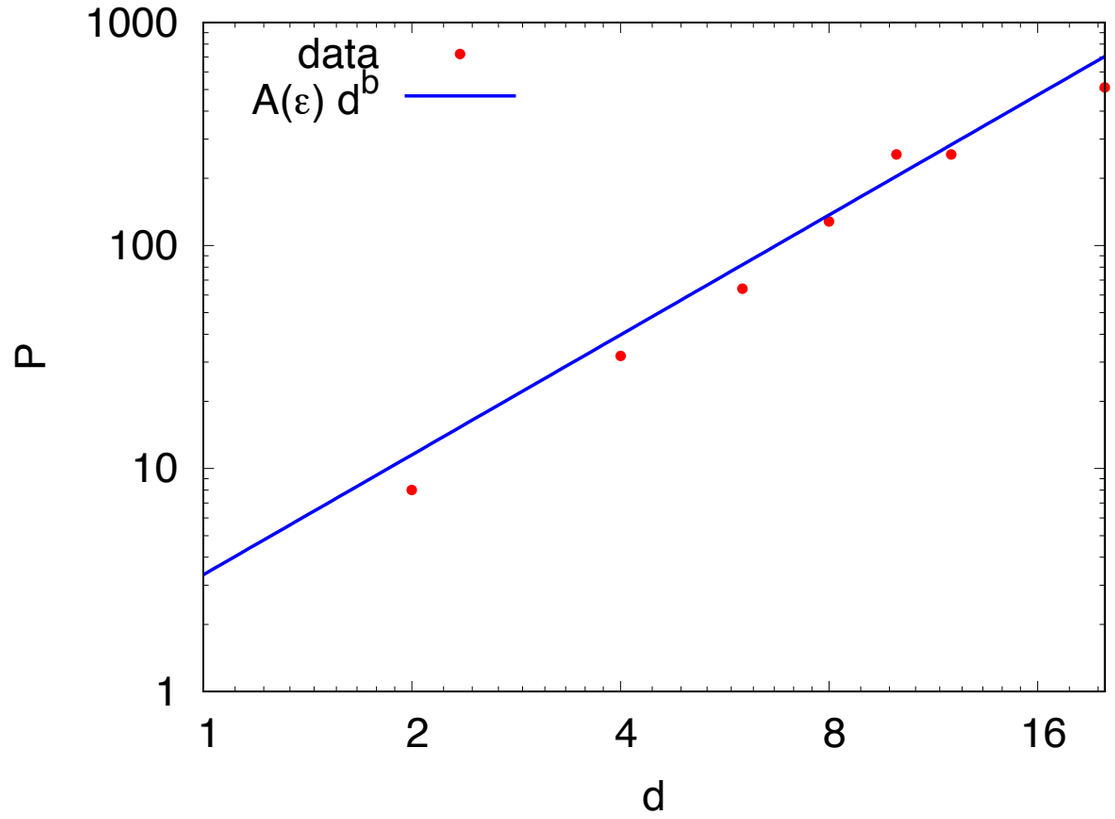
Exact solution: $g(\overrightarrow{x},t) = \frac{1}{1+e^{-t-\sum_{i=1}^{d}x_i}}$

We want to compute the computational complexity of the algorithm $O(d^b)$.
For doing so, we analyze:

$$\langle\epsilon\rangle = \frac{1}{M}\sum_{m=1}^{M}\left|\frac{g(\overrightarrow{x}_m,0) - \mathcal{NN}(\overrightarrow{x}_m,0\,|\,\overrightarrow{\theta})}{g(\overrightarrow{x}_m,0)}\right|$$

Results IV — (backward) An exactly solvable non-linear diffusion equation





Left: Average relative error $\langle \epsilon \rangle$ as a function of number of initial points of stochastic trajectories $P = 4,8,16,\cdots,1024,2048,4096$, on a log-log scale for different values of $d = 2,4,6,8,10$.
Right: Computational complexity, given an average relative error $\langle \epsilon \rangle < 5\,\%$, measured by P as a function of $d = 2,4,6,8,10,12,18$, on a log-log scale the slope is $b \approx 1.78$. The computational complexity obtained is proportional to $A(\epsilon)d^2$, with $A(\epsilon) \sim 0.20$. The results are obtained by fixing $D = \frac{\sigma^2}{2}$, where $\sigma = 0.25$, and $T = 0.01$.

THANK YOU !