Stochastic methods for solving partial differential equations in high dimension

Marie Billaud-Friess
Joint work with : A. Macherey, A. Nouy & C. Prieur

marie.billaud-friess@ec-nantes.fr
Centrale Nantes, Laboratoire de Mathématiques Jean Leray

July 2nd, 2018

13th International Conference in
Monte Carlo & Quasi-Monte Carlo Methods in Scientific Computing

High-dimensional problem

Find u solution of

$$\begin{aligned} \mathcal{L}(u) &= g \quad \text{in } D, \\ u &= f \quad \text{on } \partial D. \end{aligned} \tag{1}$$

- $u : \overline{D} \to \mathbb{R}$ **multivariate function** of $\boldsymbol{x} = (x_1, \ldots, x_d)$ on $D \subset \mathbb{R}^d$
- $\mathcal{L}$ linear differential operator
- $f, g : \overline{D} \to \mathbb{R}$ boundary condition and source term respectively

High-dimensional problem

Find u solution of

$$\begin{aligned} \mathcal{L}(u) &= g \quad \text{in } D, \\ u &= f \quad \text{on } \partial D. \end{aligned} \tag{1}$$

- $u : \overline{D} \to \mathbb{R}$ **multivariate function** of $\boldsymbol{x} = (x_1, \ldots, x_d)$ on $D \subset \mathbb{R}^d$
- $\mathcal{L}$ linear differential operator
- $f, g : \overline{D} \to \mathbb{R}$ boundary condition and source term respectively

Main challenges :

1. How to approximate u up to precision ε with reasonable computational cost ?
2. In particular for high dimensions ($d \gg 1$) while overcoming the so-called *curse of dimensionality*

Solving high dimensional problems

Deterministic way...

Framework : Nonlinear and sparse approximation

Solving high dimensional problems

Deterministic way...

Framework : Nonlinear and sparse approximation

① **Tensor based methods :**

- General tensor networks with application in physics [Verstarete,Vidal] ...
- Low-rank approximation methods [Bachmayer, Dahmen, Grasedyck, Hackbush, Khoromskij, Kressner, Matthies, Nouy, Oseledets, Schwab, Schneider, Uschmajew ...]

Solving high dimensional problems

Deterministic way...

Framework : Nonlinear and sparse approximation

① **Tensor based methods :**

- General tensor networks with application in physics [Verstarete,Vidal] ...
- Low-rank approximation methods [Bachmayer, Dahmen, Grasedyck, Hackbush, Khoromskij, Kressner, Matthies, Nouy, Oseledets, Schwab, Schneider, Uschmajew ...]

② **Sparse (tensor) approximation :**

Contributions for parameter dependent PDEs [Chkifa, Cohen, De Vore, Nobile, Schwab,...]

$$u(\boldsymbol{x}) \approx u_\Lambda(\boldsymbol{x}) = \sum_{\nu\in\Lambda} \alpha_\nu \varphi_\nu(\boldsymbol{x}) = \sum_{\nu\in\Lambda} \alpha_\nu \prod_{j\geq 1} \varphi^\nu_{\nu_j}(x_j),$$

⋆ **How to compute u_Λ ?**

- Polynomial expansions e.g [Chkifa'13, Cohen'15]
- Projection based methods : Galerkin e.g. with multilevel FE, wavelets [Schwab'11], least-square or polynomial interpolation [Chkifa'14]

Solving high dimensional problems

Deterministic way...

Framework : Nonlinear and sparse approximation

① **Tensor based methods :**

- General tensor networks with application in physics [Verstarete,Vidal] ...
- Low-rank approximation methods [Bachmayer, Dahmen, Grasedyck, Hackbush, Khoromskij, Kressner, Matthies, Nouy, Oseledets, Schwab, Schneider, Uschmajew ...]

② **Sparse (tensor) approximation :**

Contributions for parameter dependent PDEs [Chkifa, Cohen, De Vore, Nobile, Schwab,...]

$$u(\boldsymbol{x}) \approx u_\Lambda(\boldsymbol{x}) = \sum_{\nu\in\Lambda} \alpha_\nu \boldsymbol{x}^\nu = \sum_{\nu\in\Lambda} \alpha_\nu \prod_{j\geq 1} x_j^{\nu_j},$$

⋆ **How to compute u_Λ ?**

- Polynomial expansions e.g [Chkifa'13, Cohen'15]
- Projection based methods : Galerkin e.g. with multilevel FE, wavelets [Schwab'11], least-square or polynomial interpolation [Chkifa'14]

⋆ **Sparse polynomial interpolation :** $\varphi_\nu(\boldsymbol{x}) = \boldsymbol{x}^\nu$

- **Sample based methods** requiring evaluation of u at some points of D
- **Adaptive selection** of Λ leading to **sparse polynomial space** [Chkifa'13, Chkifa'14]

Solving high dimensional problems

Stochastic way...

Single point estimations of u at $\boldsymbol{x} \in \overline{D}$: Combine **probabilistic representation of $u(\boldsymbol{x})$** together with **Monte Carlo** estimation [Graham'13, Gobet'13]

$$u(\boldsymbol{x}) \approx u_M(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} \psi(f, g, X^{\boldsymbol{x}}(\omega_m)),$$

with $X^{\boldsymbol{x}}(\omega_m)$ a realization of $X^{\boldsymbol{x}}$ the diffusion process starting from $\boldsymbol{x}$ at $t = 0$.

Solving high dimensional problems

Stochastic way...

Single point estimations of u at $\boldsymbol{x} \in \overline{D}$: Combine **probabilistic representation of $u(\boldsymbol{x})$** together with **Monte Carlo** estimation [Graham'13, Gobet'13]

$$u(\boldsymbol{x}) \approx u_M(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} \psi(f, g, X^{\boldsymbol{x}}(\omega_m)),$$

with $X^{\boldsymbol{x}}(\omega_m)$ a realization of $X^{\boldsymbol{x}}$ the diffusion process starting from $\boldsymbol{x}$ at $t = 0$.

Toward a global approximation of u over $\overline{D}$:

- **Interpolation** combined together with **sequential variance reduction** technique [Gobet'04, Gobet'09]
  ✗Limited to small dimension !
- **Deep learning based on artificial neural network approximations** for linear and nonlinear parabolic **high-dimensional problems** [Beck'17, Weinan'17, Beck'18]
  ✗Efficient method, but no rigorous analysis !

Outline

Goal

Gather a probabilistic approach for **pointwise estimation of the solution** together with a **sparse interpolation method** to compute global approximation to solution of high-dimensional partial differential equations.

1. **A sequential algorithm for variance reduction**
2. **A sequential algorithm in high dimension**
3. **A perturbed sparse adaptive algorithm**

Outline

Pointwise estimation of $u(x)$

Let define $\mathcal{L}$ in (1) by

$$\mathcal{L} = \left(-\frac{1}{2} \sum_{i,j=1}^{d} (\sigma(\boldsymbol{x})\sigma(\boldsymbol{x})^T)_{ij} \partial^2_{x_i x_j} + \sum_{i=1}^{d} b_i(\boldsymbol{x}) \partial_{x_i} u(\boldsymbol{x}) + k(\boldsymbol{x}) \right)$$

as the **infinitesimal generator** of $\boldsymbol{X}^{\boldsymbol{x}}$ the d-dimensional **diffusion process** given by

$$\mathrm{d}\boldsymbol{X}_t^{\boldsymbol{x}} = b(\boldsymbol{X}_t^{\boldsymbol{x}})\mathrm{d}t + \sigma(\boldsymbol{X}_t^{\boldsymbol{x}}) d\boldsymbol{W}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x} \in \overline{D}.$$

where $\boldsymbol{W}$ is a d-dimensional brownian motion.

Pointwise estimation of $u(x)$

Let define $\mathcal{L}$ in (1) by

$$\mathcal{L} = \left(-\frac{1}{2} \sum_{i,j=1}^{d} (\sigma(\boldsymbol{x})\sigma(\boldsymbol{x})^T)_{ij} \partial^2_{x_i x_j} + \sum_{i=1}^{d} b_i(\boldsymbol{x}) \partial_{x_i} u(\boldsymbol{x}) + k(\boldsymbol{x}) \right)$$

as the **infinitesimal generator** of $\boldsymbol{X}^{\boldsymbol{x}}$ the d-dimensional **diffusion process** given by

$$\mathrm{d}\boldsymbol{X}^{\boldsymbol{x}}_t = b(\boldsymbol{X}^{\boldsymbol{x}}_t)\mathrm{d}t + \sigma(\boldsymbol{X}^{\boldsymbol{x}}_t) d\boldsymbol{W}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x} \in \overline{D}.$$

where $\boldsymbol{W}$ is a d-dimensional brownian motion.

Feynman-Kac formula

Assuming that

i) b, σ are lipschitz,

ii) $f, g : \overline{D} \to \mathbb{R}$ are continuous functions,

iii) there exists $u : \overline{D} \to \mathbb{R}$ in $\mathcal{C}^2$ on all open subsets of D solution of (1),

then for $\boldsymbol{x} \in \overline{D}$ we have

$$u(\boldsymbol{x}) = \mathbb{E}(\psi(f, g, \boldsymbol{X}^{\boldsymbol{x}})) := \mathbb{E}\left(f(\boldsymbol{X}^{\boldsymbol{x}}_{\tau_{\boldsymbol{x}}}) e^{-\int_0^{\tau_{\boldsymbol{x}}} k(\boldsymbol{X}^{\boldsymbol{x}}_r)\mathrm{d}t} + \int_0^{\tau_{\boldsymbol{x}}} g(\boldsymbol{X}^{\boldsymbol{x}}_s) e^{-\int_0^s k(\boldsymbol{X}^{\boldsymbol{x}}_r)\mathrm{d}r} \mathrm{d}s \right).$$

with $\tau_{\boldsymbol{x}} = \inf\{s > 0 : \boldsymbol{X}^{\boldsymbol{x}}_s \notin D\}$ the first exit time of D.

Pointwise estimation of $u(x)$

Let define $\mathcal{L}$ in (1) by

$$\mathcal{L} = \left(-\frac{1}{2} \sum_{i,j=1}^{d} (\sigma(\boldsymbol{x})\sigma(\boldsymbol{x})^T)_{ij} \partial^2_{x_i x_j} + \sum_{i=1}^{d} b_i(\boldsymbol{x}) \partial_{x_i} u(\boldsymbol{x}) + k(\boldsymbol{x}) \right)$$

as the **infinitesimal generator** of $\boldsymbol{X}^{\boldsymbol{x}}$ the d-dimensional **diffusion process** given by

$$\mathrm{d}\boldsymbol{X}^{\boldsymbol{x}}_t = b(\boldsymbol{X}^{\boldsymbol{x}}_t)\mathrm{d}t + \sigma(\boldsymbol{X}^{\boldsymbol{x}}_t) d\boldsymbol{W}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x} \in \overline{D}.$$

where $\boldsymbol{W}$ is a d-dimensional brownian motion.

Feynman-Kac formula

Assuming that

i) b, σ are lipschitz,

ii) $f, g : \overline{D} \to \mathbb{R}$ are continuous functions,

iii) there exists $u : \overline{D} \to \mathbb{R}$ in $\mathcal{C}^2$ on all open subsets of D solution of (1),

then for $\boldsymbol{x} \in \overline{D}$ we have

$$u(\boldsymbol{x}) = \mathbb{E}(\phi(u, \boldsymbol{X}^{\boldsymbol{x}})) := \mathbb{E}\left(u(\boldsymbol{X}^{\boldsymbol{x}}_{\tau_{\boldsymbol{x}}}) e^{-\int_0^{\tau_{\boldsymbol{x}}} k(\boldsymbol{X}^{\boldsymbol{x}}_r)\mathrm{d}t} + \int_0^{\tau_{\boldsymbol{x}}} \mathcal{L}u(\boldsymbol{X}^{\boldsymbol{x}}_s) e^{-\int_0^s k(\boldsymbol{X}^{\boldsymbol{x}}_r)\mathrm{d}r} \mathrm{d}s \right).$$

with $\tau_{\boldsymbol{x}} = \inf\{s > 0 : \boldsymbol{X}^{\boldsymbol{x}}_s \notin D\}$ the first exit time of D.

In practice $u(\boldsymbol{x}) \approx u_{\Delta t, M}(\boldsymbol{x})$

- Estimation of the expectation by Monte-Carlo simulation
- Noting $0 = t^0 < t^1 < \dots$ with $t^n = n\Delta t, n = 0, 1, \dots,$ let $\boldsymbol{X}^{\boldsymbol{x}} \approx \boldsymbol{X}^{\boldsymbol{x},\Delta t}$ where $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}}$ is given by Euler-Maruyama scheme

$$\boldsymbol{X}_{n+1}^{\boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}} + \Delta t\ b(\boldsymbol{X}_n^{\boldsymbol{x}}) + \sigma(\boldsymbol{X}_n^{\boldsymbol{x}})\Delta \boldsymbol{W}_n,$$

  where $\Delta \boldsymbol{W}_n = \boldsymbol{W}_{t^{n+1}} - \boldsymbol{W}_{t^n}$.

- Let $\left\{\boldsymbol{X}^{\Delta t, \boldsymbol{x}}(\omega_m)\right\}_{m=1}^{M}$, M independent realisations of $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}}$

$$u_{\Delta t, M}(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} u(\boldsymbol{X}_{\tau_{\boldsymbol{x}}^{\Delta t}}^{\Delta t, \boldsymbol{x}}(\omega_m)) + \Delta t \sum_{i=0}^{N-1} \mathcal{L}(u)(\boldsymbol{X}_{t^i}^{\Delta t, \boldsymbol{x}}(\omega_m)) \mathbf{1}_{t^i \leq \tau_{\boldsymbol{x}}^{\Delta t}}.$$

In practice $u(\boldsymbol{x}) \approx u_{\Delta t, M}(\boldsymbol{x})$

- Estimation of the expectation by Monte-Carlo simulation
- Noting $0 = t^0 < t^1 < \dots$ with $t^n = n\Delta t, n = 0, 1, \dots,$ let $\boldsymbol{X}^{\boldsymbol{x}} \approx \boldsymbol{X}^{\boldsymbol{x}, \Delta t}$ where $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}}$ is given by Euler-Maruyama scheme

$$\boldsymbol{X}_{n+1}^{\boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}} + \Delta t \; b(\boldsymbol{X}_n^{\boldsymbol{x}}) + \sigma(\boldsymbol{X}_n^{\boldsymbol{x}}) \Delta \boldsymbol{W}_n,$$

  where $\Delta \boldsymbol{W}_n = \boldsymbol{W}_{t^{n+1}} - \boldsymbol{W}_{t^n}$.

- Let $\left\{ \boldsymbol{X}^{\Delta t, \boldsymbol{x}}(\omega_m) \right\}_{m=1}^{M}$, M independent realisations of $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}}$

$$u_{\Delta t, M}(\boldsymbol{x}) := \frac{1}{M} \sum_{m=1}^{M} \phi(u, \boldsymbol{X}^{\Delta t, \boldsymbol{x}})(\omega_m).$$

In practice $u(\boldsymbol{x}) \approx u_{\Delta t, M}(\boldsymbol{x})$

- Estimation of the expectation by Monte-Carlo simulation
- Noting $0 = t^0 < t^1 < \dots$ with $t^n = n\Delta t, n = 0, 1, \dots,$ let $\boldsymbol{X}^{\boldsymbol{x}} \approx \boldsymbol{X}^{\boldsymbol{x}, \Delta t}$ where $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}}$ is given by Euler-Maruyama scheme

$$\boldsymbol{X}_{n+1}^{\boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}} + \Delta t \; b(\boldsymbol{X}_n^{\boldsymbol{x}}) + \sigma(\boldsymbol{X}_n^{\boldsymbol{x}}) \Delta \boldsymbol{W}_n,$$

  where $\Delta \boldsymbol{W}_n = \boldsymbol{W}_{t^{n+1}} - \boldsymbol{W}_{t^n}$.

- Let $\left\{ \boldsymbol{X}^{\Delta t, \boldsymbol{x}}(\omega_m) \right\}_{m=1}^{M}$, M independent realisations of $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}}$

$$u_{\Delta t, M}(\boldsymbol{x}) := \frac{1}{M} \sum_{m=1}^{M} \phi(u, \boldsymbol{X}^{\Delta t, \boldsymbol{x}})(\omega_m).$$

Error : Integration error $O(\sqrt{\Delta t})$ + MC error $O\left(\frac{1}{\sqrt{M}}\right)$

$\rightsquigarrow$ Slow convergence w.r.t. M with large error for large variance $\mathbb{V}(u_{\Delta t, M}(\boldsymbol{x}))$

In practice $u(\boldsymbol{x}) \approx u_{\Delta t, M}(\boldsymbol{x})$

- Estimation of the expectation by Monte-Carlo simulation
- Noting $0 = t^0 < t^1 < \dots$ with $t^n = n\Delta t, n = 0, 1, \dots$, let $\boldsymbol{X}^{\boldsymbol{x}} \approx \boldsymbol{X}^{\boldsymbol{x}, \Delta t}$ where $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}}$ is given by Euler-Maruyama scheme

$$\boldsymbol{X}_{n+1}^{\boldsymbol{x}} = \boldsymbol{X}_n^{\boldsymbol{x}} + \Delta t \; b(\boldsymbol{X}_n^{\boldsymbol{x}}) + \sigma(\boldsymbol{X}_n^{\boldsymbol{x}}) \Delta \boldsymbol{W}_n,$$

  where $\Delta \boldsymbol{W}_n = \boldsymbol{W}_{t^{n+1}} - \boldsymbol{W}_{t^n}$.

- Let $\left\{ \boldsymbol{X}^{\Delta t, \boldsymbol{x}}(\omega_m) \right\}_{m=1}^{M}$, M independent realisations of $\boldsymbol{X}_{t^n}^{\Delta t, \boldsymbol{x}}$

$$u_{\Delta t, M}(\boldsymbol{x}) := \frac{1}{M} \sum_{m=1}^{M} \phi(u, \boldsymbol{X}^{\Delta t, \boldsymbol{x}})(\omega_m).$$

Error : Integration error $O(\sqrt{\Delta t})$ + MC error $O\left(\frac{1}{\sqrt{M}}\right)$

$\rightsquigarrow$ Slow convergence w.r.t. M with large error for large variance $\mathbb{V}(u_{\Delta t, M}(\boldsymbol{x}))$

Improving the convergence : Multilevel MC [Giles'08], Variance reduction : control variate, importance sampling, antithetic sampling [Gobet'13, Graham'13], ...

Gobet-Maire Algorithm [Gobet'04, Gobet'09]

Notations : Let v be a smooth real-valued function defined on $\overline{D}$ the solution of an EDP under the form (1)

- Stochastic approximation of v at $\boldsymbol{x} \in \overline{D}$: $v_{\Delta t, M}(\boldsymbol{x})$
- Approximation of v at step k of the algorithm : $\tilde{v}^k$
- Linear approximation (e.g. interpolant) of v using **pointwise evaluations** at $\{\boldsymbol{z}_i\}_{i=1}^N \subset \overline{D}$ for some basis functions $\{\ell_i\}_{i=1}^N$:

$$\mathcal{I}(v) = \sum_{i=1}^{N} v(\boldsymbol{z}_i)\ell_i(\boldsymbol{x}).$$

Algorithm 1. Gobet & Maire algorithm

1. Set $\tilde{u}^0 = 0$.
2. **For** $k = 1, \dots, K$ **:**
   - Compute $e^k_{\Delta t, M}(\boldsymbol{z}_i) \approx e^k(\boldsymbol{z}_i), i = 1, \dots, N$ where $e^k = u - \tilde{u}^k$ is s.t.

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}) e^k(\boldsymbol{x}) &= g(\boldsymbol{x}) - \mathcal{L}\tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in D, \\ e^k(\boldsymbol{x}) &= f(\boldsymbol{x}) - \tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial D. \end{aligned}$$

   - Define $\tilde{e}^k = \sum_{i=1}^{N} e^k_{\Delta t, M}(\boldsymbol{z}_i) \ell_i(\boldsymbol{x})$
   - Update $\tilde{u}^{k+1} = \tilde{u}^k + \tilde{e}^k$

Algorithm 1. Gobet & Maire algorithm

1. Set $\tilde{u}^0 = 0$.
2. **For** $k = 1, \dots, K$ **:**
   - Compute $e^k_{\Delta t, M}(\boldsymbol{z}_i) \approx e^k(\boldsymbol{z}_i), i = 1, \dots, N$ where $e^k = u - \tilde{u}^k$ is s.t.

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}) e^k(\boldsymbol{x}) &= g(\boldsymbol{x}) - \mathcal{L}\tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in D, \\ e^k(\boldsymbol{x}) &= f(\boldsymbol{x}) - \tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial D. \end{aligned}$$

   - Define $\tilde{e}^k = \sum_{i=1}^{N} e^k_{\Delta t, M}(\boldsymbol{z}_i) \ell_i(\boldsymbol{x})$
   - Update $\tilde{u}^{k+1} = \tilde{u}^k + \tilde{e}^k$

Remarks :

- For Δt small enough, $\max_{i=1,\dots,N} |\mathbb{E}(\tilde{u}^k(\boldsymbol{z}_i) - \boldsymbol{z}_i)|$ and $\mathbb{V}(\tilde{u}^k(\boldsymbol{x}_i))$ **converge geometrically** with k, up to threshold term. [Gobet'09].
- Algorithm designed for small d.

Outline

Going to high dimension

Sparse interpolation (in brief) [Chkifa'14]

Let $u : \overline{D} \to \mathbb{R}$ where $D = [-1, 1]^d$.

Going to high dimension

Sparse interpolation (in brief) [Chkifa'14]

Let $u : \overline{D} \to \mathbb{R}$ where $D = [-1, 1]^d$.

① Given a finite set of multi-indices $\nu = (\nu_1, \dots, \nu_d)$ noted $\Lambda \subset \mathbb{N}^d$ that is **downward closed**

$$\forall \nu \in \Lambda, \mu \leq \nu \Rightarrow \mu \in \Lambda,$$

construct by **tensorisation** the multivariate polynomial P_ν associated to ν

$$P_\nu(\boldsymbol{x}) = \prod_{i=1}^{d} p_{\nu_i}(x_i)$$

where $x_i \mapsto p_{\nu_i}(x_i)$ are **univariate polynomial basis** (e.g. Legendre).

Sparse interpolation (in brief) [Chkifa'14]

Let $u : \overline{D} \to \mathbb{R}$ where $D = [-1,1]^d$.

① Given a finite set of multi-indices $\nu = (\nu_1, \dots, \nu_d)$ noted $\Lambda \subset \mathbb{N}^d$ that is **downward closed**

$$\forall \nu \in \Lambda, \mu \leq \nu \Rightarrow \mu \in \Lambda,$$

construct by **tensorisation** the multivariate polynomial P_ν associated to ν

$$P_\nu(\boldsymbol{x}) = \prod_{i=1}^{d} p_{\nu_i}(x_i)$$

where $x_i \mapsto p_{\nu_i}(x_i)$ are **univariate polynomial basis** (e.g. Legendre).
② The interpolant $\mathcal{I}_\Lambda(u)$ of u in $\mathbb{P}_\Lambda$ is given by

$$\mathcal{I}_\Lambda(u) = \sum_{\nu \in \Lambda} \alpha_\nu P_\nu.$$

- It is associated with **interpolation points** (e.g. Leja, Magic Points [Maday'09]) $\{\boldsymbol{z}_\nu\}_{\nu \in \Lambda} \subset [-1,1]^d$ **unisolvent** for $\mathbb{P}_\Lambda = \text{span}\{P_\nu, \nu \in \Lambda\}$,

Sparse interpolation (in brief) [Chkifa'14]

Let $u : \overline{D} \to \mathbb{R}$ where $D = [-1,1]^d$.

① Given a finite set of multi-indices $\nu = (\nu_1, \dots, \nu_d)$ noted $\Lambda \subset \mathbb{N}^d$ that is **downward closed**

$$\forall \nu \in \Lambda, \mu \leq \nu \Rightarrow \mu \in \Lambda,$$

construct by **tensorisation** the multivariate polynomial P_ν associated to ν

$$P_\nu(\boldsymbol{x}) = \prod_{i=1}^{d} p_{\nu_i}(x_i)$$

where $x_i \mapsto p_{\nu_i}(x_i)$ are **univariate polynomial basis** (e.g. Legendre).
② The interpolant $\mathcal{I}_\Lambda(u)$ of u in $\mathbb{P}_\Lambda$ is given by

$$\mathcal{I}_\Lambda(u) = \sum_{\nu \in \Lambda} u(\boldsymbol{z}_\nu)\ell_\nu.$$

- It is associated with **interpolation points** (e.g. Leja, Magic Points [Maday'09]) $\{\boldsymbol{z}_\nu\}_{\nu\in\Lambda} \subset [-1,1]^d$ **unisolvent** for $\mathbb{P}_\Lambda = \text{span}\{P_\nu, \nu \in \Lambda\}$,
- and $\{\ell_\nu\}_\nu$ is a basis of $\mathbb{P}_\Lambda$ s.t. $\ell_\nu(\boldsymbol{z}_\mu) = \delta_{\nu\mu}$.

Going to high dimension

Adaptive selection of Λ and construction of $\mathcal{I}_\Lambda(u)$ [Chkifa'13, Chkifa'14]

Algorithm 2. Adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\mathcal{I}_{\Lambda^\star}(u)$.
   - Select $N_n = \{\nu \in \mathcal{M}_n; \mathcal{E}_\nu(\mathcal{I}_{\Lambda^\star}(u)) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\mathcal{I}_{\Lambda^\star}(u))\}$
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\mathcal{I}_{\Lambda_n}(u)$ and ε^n
   - Update $n = n + 1$

Going to high dimension

Adaptive selection of Λ and construction of $\mathcal{I}_\Lambda(u)$ [Chkifa'13, Chkifa'14]

Algorithm 2. Adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\mathcal{I}_{\Lambda^\star}(u)$.
   - Select $N_n = \{\nu \in \mathcal{M}_n; \mathcal{E}_\nu(\mathcal{I}_{\Lambda^\star}(u)) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\mathcal{I}_{\Lambda^\star}(u))\}$
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\mathcal{I}_{\Lambda_n}(u)$ and ε^n
   - Update $n = n + 1$

Remarks :

① Using the **reduced margin** of Λ_n we define

$$\mathcal{M}_n = \{\nu \notin \Lambda_{n-1}, \nu_j \neq 0 \Rightarrow \nu - e_j \in \Lambda_{n-1}\}.$$

Adaptive selection of Λ and construction of $\mathcal{I}_\Lambda(u)$ [Chkifa'13, Chkifa'14]

Algorithm 2. Adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\mathcal{I}_{\Lambda^\star}(u)$.
   - Select $N_n = \{\nu \in \mathcal{M}_n; \mathcal{E}_\nu(\mathcal{I}_{\Lambda^\star}(u)) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\mathcal{I}_{\Lambda^\star}(u))\}$
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\mathcal{I}_{\Lambda_n}(u)$ and ε^n
   - Update $n = n + 1$

Remarks :

② N_n selected using a **bulk chasing algorithm**, ensuring Λ_n **downward closed**, where

$$\mathcal{E}_\mathcal{S}(\mathcal{I}_{\Lambda^\star}(u)) = \sum_{\nu^i \in \mathcal{S}} \beta_{\nu^i}^2$$

measures the **norm** of the interpolant coefficients $\{\beta_{\nu^j}\}_{\nu^j \in \Lambda^\star}$ of $\mathcal{I}_{\Lambda^\star}(u)$ ordered by decreasing values, associated to multi-indices in $\mathcal{S}$.

Going to high dimension

Adaptive selection of Λ and construction of $\mathcal{I}_\Lambda(u)$ [Chkifa'13, Chkifa'14]

Algorithm 2. Adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\mathcal{I}_{\Lambda^\star}(u)$.
   - Select $N_n = \{\nu \in \mathcal{M}_n; \mathcal{E}_\nu(\mathcal{I}_{\Lambda^\star}(u)) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\mathcal{I}_{\Lambda^\star}(u))\}$
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\mathcal{I}_{\Lambda_n}(u)$ and ε^n
   - Update $n = n + 1$

Remarks :

③ Here

$$\varepsilon^n = \frac{\sum_{\nu \in \mathcal{M}_n} \alpha_\nu^2}{\sum_{\nu \in \Lambda^\star} \alpha_\nu^2}$$

Algorithm 1. (High dimension Gobet & Maire algorithm)

1. Set $\tilde{u}^0 = 0$.
2. **For** $k = 1, \dots, K$ **:**
   - Set $e^k = u - \tilde{u}^k$ is s.t.

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}) e^k(\boldsymbol{x}) &= g(\boldsymbol{x}) - \mathcal{L}\tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in D, \\ e^k(\boldsymbol{x}) &= f(\boldsymbol{x}) - \tilde{u}^k(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial D. \end{aligned}$$

   - Interpolate $\tilde{e}^k$
   - Update

$$\tilde{u}^{k+1} = \tilde{u}^k + \tilde{e}^k$$

Remark : Here $\tilde{e}^k = \mathcal{I}^{\varepsilon}_{\Lambda^k}(e^k_{\Delta t, M})$ computed using Algorithm 2 for given precision ε using realizations $\{e^k_{\Delta t, M}(\boldsymbol{z}_\nu)\}_\nu$.

Problem setting

Laplacian in d dimensions in $D = [-1, 1]^d$

$$\begin{aligned} -\triangle u(\boldsymbol{x}) &= g(\boldsymbol{x}) \quad \boldsymbol{x} \in D, \\ u(\boldsymbol{x}) &= f(\boldsymbol{x}) \quad \boldsymbol{x} \in \partial D, \end{aligned} \tag{2}$$

Problem setting

Laplacian in d dimensions in $D = [-1, 1]^d$

$$\begin{aligned} -\triangle u(\boldsymbol{x}) &= g(\boldsymbol{x}) \quad \boldsymbol{x} \in D, \\ u(\boldsymbol{x}) &= f(\boldsymbol{x}) \quad \boldsymbol{x} \in \partial D, \end{aligned} \tag{2}$$

Tested methods : Let $\Lambda = \{\nu \in \mathbb{N}^d, |\nu| \leq p\}$.

- **Method 1 :** No adaptive selection of multi-indices :

$$\tilde{e}^k = \mathcal{I}_\Lambda(e^k_{\Delta t, M})$$

- **Method 2 :** Adaptive selection of multi-indices with $\Lambda_k \subset \Lambda$ s.t. $\theta = 0.5$:

$$\tilde{e}^k = \mathcal{I}^\varepsilon_{\Lambda^k}(e^k_{\Delta t, M})$$

where $\varepsilon = 0.05$.

Problem setting

Laplacian in d dimensions in $D = [-1, 1]^d$

$$\begin{aligned} -\triangle u(\boldsymbol{x}) &= g(\boldsymbol{x}) \quad \boldsymbol{x} \in D, \\ u(\boldsymbol{x}) &= f(\boldsymbol{x}) \quad \boldsymbol{x} \in \partial D, \end{aligned} \tag{2}$$

Tested methods : Let $\Lambda = \{\nu \in \mathbb{N}^d, |\nu| \leq p\}$.

- **Method 1 :** No adaptive selection of multi-indices :

$$\tilde{e}^k = \mathcal{I}_\Lambda(e^k_{\Delta t, M})$$

- **Method 2 :** Adaptive selection of multi-indices with $\Lambda_k \subset \Lambda$ s.t. $\theta = 0.5$:

$$\tilde{e}^k = \mathcal{I}^\varepsilon_{\Lambda^k}(e^k_{\Delta t, M})$$

  where $\varepsilon = 0.05$.

Test configurations :

- **Test 1 :** $u(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2$, and $d = 5, p = 2$
- **Test 2 :** $u(\boldsymbol{x}) = x_1^2 + \sin(x_2) + \exp(x_3) + \sin(x_4)(x_5 + 1)$, and $d = 5, p = 10$.

Numerical results

Method 1, test n° 1, $d = 5, p = 2, \#\Lambda = 11$

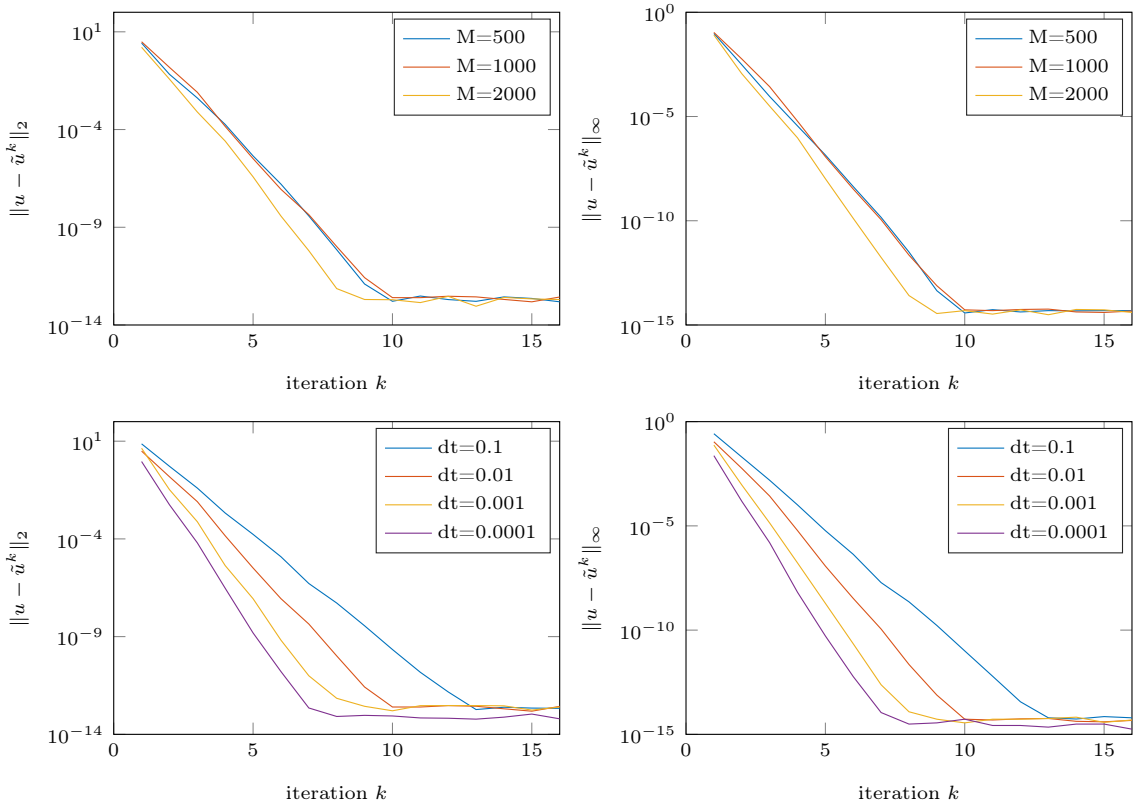

Figure – Test n°1 : Evolution of errors w.r.t. Δt (left) and M (right).

1. Convergence in few iterations
2. Error decreases with respect to M and Δt but finally stagnates

Numerical results

Method 2, test n°1 $d = 5, p = 2$

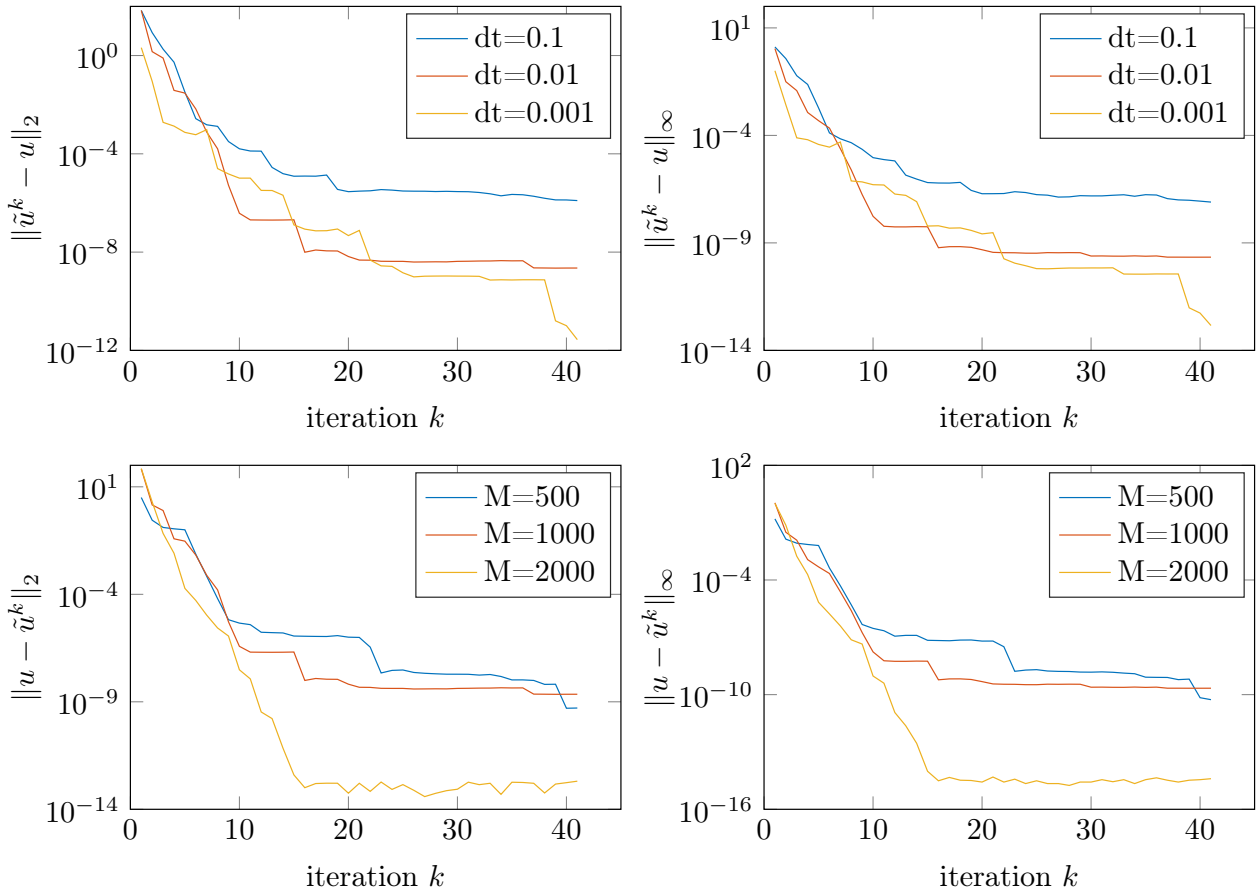

Figure – Test n°1 : Evolution of errors w.r.t. Δt (left) and M (right).

1. Convergence with respect to M and Δt
2. Slower convergence w.r.t. to k then for non adapted Λ

Numerical results

Method 2, test n° 2 $d = 5, p = 10$ *and* $\Delta t = 0.001, M = 1000$

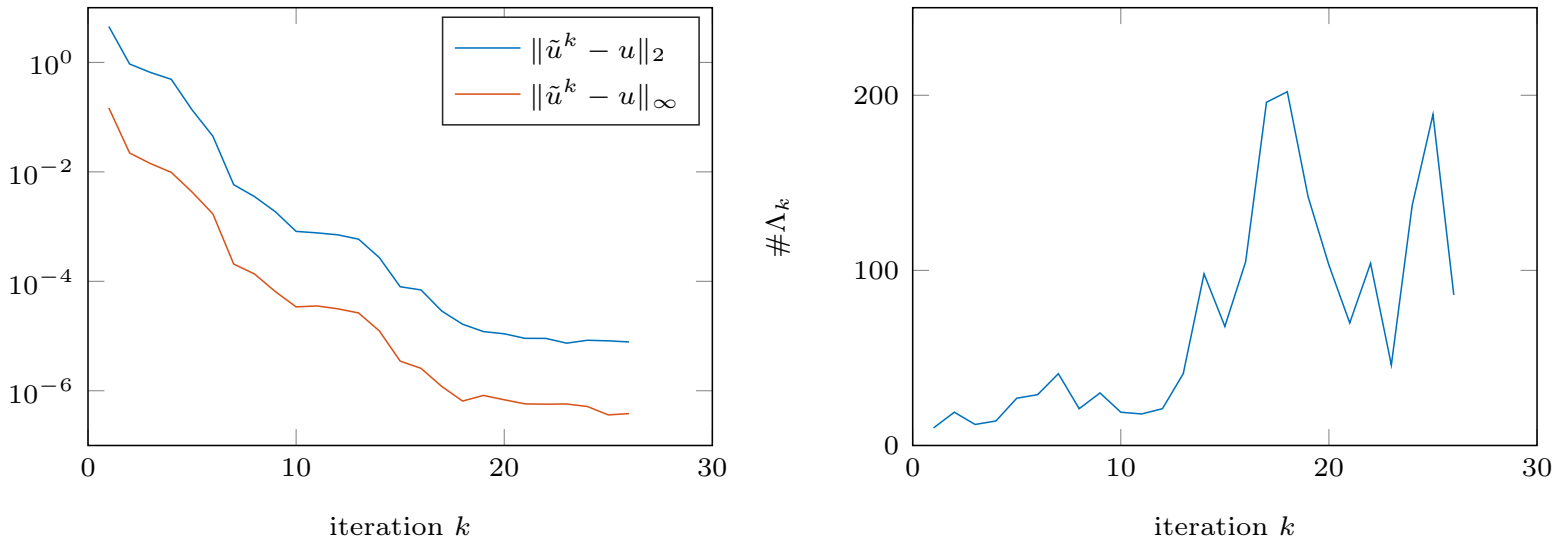

Figure – Test n°2 : Errors (left) and evolution of $\#\Lambda_k$ (right).

1. The method converges with respect to k
2. Reasonnable $\#\Lambda_k \leq 200$ during iterations

Error analysis for $\Lambda^k = \Lambda$

① *Pointwise error*

Notations : Let $a : \overline{D} \to \mathbb{R}$ be a smooth function, the **integration error** is

$$e(a, \Delta t, \boldsymbol{x}) = \mathbb{E}(\phi(a, \boldsymbol{X}^{\Delta t, \boldsymbol{x}}) - \phi(a, \boldsymbol{X}^{\boldsymbol{x}})).$$

Error analysis for $\Lambda^k = \Lambda$

① *Pointwise error*

Notations : Let $a : \overline{D} \to \mathbb{R}$ be a smooth function, the **integration error** is

$$e(a, \Delta t, \boldsymbol{x}) = \mathbb{E}(\phi(a, \boldsymbol{X}^{\Delta t, \boldsymbol{x}}) - \phi(a, \boldsymbol{X}^{\boldsymbol{x}})).$$

Pointwise error : We have

$$\mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu)) = \sum_{\nu' \in \Lambda} \mathbb{E}(u(\boldsymbol{z}_\nu) - \tilde{u}^k(\boldsymbol{z}_\nu)) e(\ell_{\nu'}, \Delta t, \boldsymbol{z}_\nu) + e(u - \mathcal{I}_\Lambda(u), \Delta t, \boldsymbol{z}_\nu)$$

Error analysis for $\Lambda^k = \Lambda$

① *Pointwise error*

Notations : Let $a : \overline{D} \to \mathbb{R}$ be a smooth function, the **integration error** is

$$e(a, \Delta t, \boldsymbol{x}) = \mathbb{E}(\phi(a, \boldsymbol{X}^{\Delta t, \boldsymbol{x}}) - \phi(a, \boldsymbol{X}^{\boldsymbol{x}})).$$

Pointwise error : We have

$$\mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu)) = \sum_{\nu' \in \Lambda} \mathbb{E}(u(\boldsymbol{z}_\nu) - \tilde{u}^k(\boldsymbol{z}_\nu)) e(\ell_{\nu'}, \Delta t, \boldsymbol{z}_\nu) + e(u - \mathcal{I}_\Lambda(u), \Delta t, \boldsymbol{z}_\nu)$$

Taking absolute value and the supremum over ν

$$m_{k+1} \leq m_k \rho_m + r(\Delta t, u - \mathcal{I}_\Lambda(u)). \tag{3}$$

with

- $m_k = \sup_\nu |\mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu))|$,
- $\rho_m = \sup_\nu \sum_{\nu' \in \Lambda} |e(\ell'_\nu, \Delta t, \boldsymbol{z}_\nu)|$,
- $r(\Delta t, u - \mathcal{I}_\Lambda(u)) = \sup_\nu |e(u - \mathcal{I}_\Lambda(u), \Delta t, \boldsymbol{z}_\nu)|$

Error analysis for $\Lambda^k = \Lambda$

① *Pointwise error*

Notations : Let $a : \overline{D} \to \mathbb{R}$ be a smooth function, the **integration error** is

$$e(a, \Delta t, \boldsymbol{x}) = \mathbb{E}(\phi(a, \boldsymbol{X}^{\Delta t, \boldsymbol{x}}) - \phi(a, \boldsymbol{X}^{\boldsymbol{x}})).$$

Pointwise error : We have

$$\mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu)) = \sum_{\nu' \in \Lambda} \mathbb{E}(u(\boldsymbol{z}_\nu) - \tilde{u}^k(\boldsymbol{z}_\nu)) e(\ell_{\nu'}, \Delta t, \boldsymbol{z}_\nu) + e(u - \mathcal{I}_\Lambda(u), \Delta t, \boldsymbol{z}_\nu)$$

Taking absolute value and the supremum over ν

$$m_{k+1} \leq m_k \rho_m + r(\Delta t, u - \mathcal{I}_\Lambda(u)). \tag{3}$$

Convergence theorem [Gobet'09]

For Δt small enough s.t. $\rho_m < 1$, $\{m_k\}_{k \geq 0}$ converges geometrically at rate ρ_m up to a threshold term that vanishes as $e(u - \mathcal{I}(u), \Delta t, \boldsymbol{z}_\nu)$ goes to 0 :

$$\limsup_{k \to \infty} m_k \leq \frac{1}{1 - \rho_m} r(\Delta t, u - \mathcal{I}_\Lambda(u)).$$

Error analysis for $\Lambda^k = \Lambda$

② *Global error*

Starting from

$$\mathbb{E}\left(\tilde{u}^{k+1} - \mathcal{I}_\Lambda(u)(\boldsymbol{x})\right) = \sum_{\nu\in\Lambda} \mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu))\ell_\nu(\boldsymbol{x}) \tag{4}$$

we get by taking the absolute value and then the supremum over $\overline{D}$

$$\sup_{x\in D}\left|\mathbb{E}(\tilde{u}^{k+1} - \mathcal{I}_\Lambda(u))(\boldsymbol{x})\right| \leq m_{k+1}\mathcal{L}_\Lambda^\infty \tag{5}$$

were $\mathcal{L}_\Lambda^\infty = \sup_{x\in D}\sum_{\nu\in\Lambda} |\ell_\nu(x)|$ denotes the Lebesgue constant. Then

$$\sup_{x\in D}\left|\mathbb{E}(\tilde{u}^{k+1} - u)(\boldsymbol{x})\right| \leq m_{k+1}\mathcal{L}_\Lambda^\infty + \|\mathcal{I}_\Lambda(u) - u\|_{\infty,D}$$

② *Global error*

Starting from

$$\mathbb{E}\left(\tilde{u}^{k+1} - \mathcal{I}_\Lambda(u)(\boldsymbol{x})\right) = \sum_{\nu\in\Lambda} \mathbb{E}(\tilde{u}^{k+1}(\boldsymbol{z}_\nu) - u(\boldsymbol{z}_\nu))\ell_\nu(\boldsymbol{x}) \tag{4}$$

we get by taking the absolute value and then the supremum over $\overline{D}$

$$\sup_{x\in D}\left|\mathbb{E}(\tilde{u}^{k+1} - \mathcal{I}_\Lambda(u))(\boldsymbol{x})\right| \leq m_{k+1}\mathcal{L}_\Lambda^\infty \tag{5}$$

were $\mathcal{L}_\Lambda^\infty = \sup_{x\in D}\sum_{\nu\in\Lambda} |\ell_\nu(x)|$ denotes the Lebesgue constant. Then

$$\sup_{x\in D}\left|\mathbb{E}(\tilde{u}^{k+1} - u)(\boldsymbol{x})\right| \leq m_{k+1}\mathcal{L}_\Lambda^\infty + \|\mathcal{I}_\Lambda(u) - u\|_{\infty,D}$$

Remark : When Δt is small enough s.t. $\rho_m < 1$ the approximation error converges geometrically with k up to the a threshold term i.e.

$$\limsup_k \left(\sup_{x\in D}\left|\mathbb{E}(\tilde{u}^{k+1} - u)(\boldsymbol{x})\right|\right) \leq \frac{\mathcal{L}_\Lambda^\infty}{1-\rho_m} r(\Delta t, u - \mathcal{I}(u)) + \|\mathcal{I}_\Lambda(u) - u\|_{\infty,D}.$$

Summary

Pros and cons :

	Method 1 Λ *fixed*	Method 2 Λ^k *adapted at each step*
Error analysis	✓Convergence up to threshold	✗Require additional assumptions
Convergence	✓Convergence in few iterations	✗Slower convergence
Problems	✗Only small d	✓Adapted for large d

Summary

Pros and cons :

	Method 1 Λ *fixed*	Method 2 Λ^k *adapted at each step*
Error analysis	✓Convergence up to threshold	✗Require additional assumptions
Convergence	✓Convergence in few iterations	✗Slower convergence
Problems	✗Only small d	✓Adapted for large d

⇝ **Alternative strategy : Perturbed sparse adaptive algorithm**

- FK based evaluation instead of the true solution provided via GM algorithm
- Possible control up of the error of the approximation to a precision ε (?)

Outline

Algorithm 3. Perturbed adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and set $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\tilde{u}_{\Lambda^\star}$.
   - Select $N_n = \{\nu \in \mathcal{M}_n; \mathcal{E}_\nu(\tilde{u}_{\Lambda^\star}) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\tilde{u}_{\Lambda^\star})\}$.
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\tilde{u}_{\Lambda_n}$ and ε^n.
   - Set $n = n + 1$.

Algorithm 3. Perturbed adaptive sparse interpolation algorithm

1. Set $\Lambda_1 = \{\mathbf{0}_d\}$ and set $n = 2$.
2. **While** $n < N$ **and** $\varepsilon^{n-1} > \varepsilon$ **:**
   - Define $\mathcal{M}_n$.
   - Set $\Lambda^\star = \Lambda_{n-1} \cup \mathcal{M}_n$ and compute $\tilde{u}_{\Lambda^\star}$.
   - Select $N_n = \{\nu \in \mathcal{M}_n ; \mathcal{E}_\nu(\tilde{u}_{\Lambda^\star}) \geq \theta \mathcal{E}_{\mathcal{M}_n}(\tilde{u}_{\Lambda^\star})\}$.
   - Update $\Lambda_n = \Lambda_{n-1} \cup N_n$.
   - Compute $\tilde{u}_{\Lambda_n}$ and ε^n.
   - Set $n = n + 1$.

Remarks :

- Here $\tilde{u}_{\Lambda^\star}, \tilde{u}_{\Lambda_n}$ can be computed with Algorithm 1. stopped either for a stopping criterion based on fixed number of iterations K or given precision δ.
- For error analysis, the second choice is preferable but we need a practical error estimate $\tilde{\varepsilon}$ (e.g. variance ?).

First numerical results : exact vs. perturbed algorithm

Test n° 2 : $\Delta t = 0.001, M = 2000, \delta = \varepsilon = 0.0001$

$\#\Lambda_n$	ε^n	$\tilde{\varepsilon}^n$	K
1	6.608586e-01	6.610627e-01	20
4	4.251625e-01	4.253583e-01	20
6	2.602440e-01	2.604017e-01	20
8	1.133414e-01	1.134104e-01	20
9	1.794418e-02	1.811954e-02	20
10	9.022909e-03	8.695473e-03	20
11	1.342954e-02	1.349724e-02	20
12	3.735804e-03	3.636647e-03	20
13	1.144930e-02	1.159909e-02	20
14	2.868766e-03	2.797837e-03	20
15	7.670605e-03	7.672113e-03	20
16	1.898420e-03	1.917487e-03	3
17	2.647237e-04	2.715048e-04	4
19	1.895775e-04	1.889428e-04	3
20	9.337434e-05	9.796734e-05	4

Table – $\#\Lambda_n, \varepsilon^n, \tilde{\varepsilon}^n$ and K w.r.t. $\#\Lambda_n$

Error	L_2-norm	L_∞-norm
$u - \mathcal{I}_{\Lambda_{20}}(u)$	4.766896e-02	2.353517e-03
$u - \tilde{u}^\delta_{\Lambda_{20}}$	4.889541e-02	2.436026e-03

Table – Error to exact solution

Remarks :

- Since $\delta = \varepsilon$, approximation $\tilde{u}^\delta_{\Lambda_{20}}$ as accurate as $\mathcal{I}_{\Lambda_{20}}(u)$.
- 15 **first iterates :** low impact of the error due to $\tilde{u}^\delta_{\Lambda_n}$ since governed by the interpolation error ($> \delta$)
- **Last iterates :** δ is reached with few iterations using exact error stopping criterion

Conclusion

Summary : Stochastic approaches for computing global approximation to solution of high-dimensional partial differential equations.

Conclusion

Summary : Stochastic approaches for computing global approximation to solution of high-dimensional partial differential equations.

Ongoing work :

- ? Clarify error analysis : for fixed Λ especially for the stagnation terms (w.r.t. $\Delta t, M$) and for the variance, for varying Λ^k (nested sets)
- ? Improve **Algorithm 3.** with better control of the error for the estimate provided by the approximation $\tilde{u}_{\Lambda_n}$
- ? Study the convergence of the perturbed sparse adaptive interpolation algorithm

Summary : Stochastic approaches for computing global approximation to solution of high-dimensional partial differential equations.

Ongoing work :

- ? Clarify error analysis : for fixed Λ especially for the stagnation terms (w.r.t. $\Delta t, M$) and for the variance, for varying Λ^k (nested sets)
- ? Improve **Algorithm 3.** with better control of the error for the estimate provided by the approximation $\tilde{u}_{\Lambda_n}$
- ? Study the convergence of the perturbed sparse adaptive interpolation algorithm

Thanks for attention !

References I

[Bachmayer'16] **M., Bachmayr, R. Schneider & A., Uschmajew** Tensor Networks and Hierarchical Tensors for the Solution of High-Dimensional Partial Differential Equations, *Found Comput Math.* 2016

[Beck'18] **C. Beck, S. Becker, P. Grohs, N. Jaafari & A. Jentzen** Solving stochastic differential equations and Kolmogorov equations by means of deep learning *ArXiv* 2018

[Beck'17] **C. Beck & A. Jentzen** Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations *ArXiv* 2017 Christian Beck, Weinan E, Arnulf Jentzen

[Chkifa'13] **A. Chkifa, A., Cohen, R., DeVore, R., & C. Schwab**, Sparse adaptive Taylor approximation algorithms for parametric and stochastic elliptic PDEs. *ESAIM : Mathematical Modelling and Numerical Analysis.* 2013.

[Chkifa'14] **A. Chkifa, A. Cohen & C.Schwab**, High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs, *Found. Comput. Math.* 2014

[Cohen'15] **A., Cohen, A., & R. DeVore** Approximation of high-dimensional parametric PDEs, *Acta Numerica.* 2015

[Nouy'17] **A. Nouy**, Low-Rank Methods for High-Dimensional Approximation and Model Order Reduction *Model Reduction and Approximation, Chapter 4.* 2017

[Giles'08] **M. Giles** Multi-level Monte Carlo path simulation. *Operations research*, 2008

References II

[Grasedyck'13] **L. Grasedyck, D. Kressner & C. Tobler**, A literature survey of low- rank tensor approximation techniques, *GAMM-Mitteilungen.* 2013

[Gobet'04] **E. Gobet and S. Maire**, A spectral Monte Carlo method for the Poisson equation, *Monte Carlo Methods Appl.* 2004

[Gobet'09] **E. Gobet and S. Maire**, Sequential control variates for functionals of Markov processes, *SIAM Journal on Numerical Analysis.* 2009

[Gobet'13] **E. Gobet**, Méthodes de Monte-Carlo et processus stochastiques : du linéaire au non linéaire, *Editions de l'école polytechnique* 2013

[Graham'13] **G. Graham & D. Talay**, Stochastic simulation and Monte Carlo methods *Stochastic Modelling and Applied Probability, Springer.* 2013

[Hackbush'14] **W. Hackbusch.** Numerical tensor calculus *Acta numerica* 2014

[Khoromskij'12] **B. Khoromskij**, Tensors-structured numerical methods in scientific computing : Survey on recent advances, *Chemometrics and Intelligent Laboratory Systems.* 2012

[Kolda'09] **T. G. Kolda & B. W. Bader**, Tensor decompositions and applications. *SIAM Review.* 2009

[Kloeden'99] **P.E. Kloeden & E. Platen**, Numerical Solution of Stochastic Differential Equations, *Springer Verlag* 1999

References III

[Maday'09] **Y. Maday, N. C. Nguyen, A. T. Patera, & G. S. H. Pau**. A general multipurpose interpolation procedure : The magic points, *Communications on Pure & Applied Analysis*, 2009

[Oseledets'11] **I. Oseledets**. Tensor-train decomposition. *SIAM J. Sci. Comput.* 2011

[Schwab'11] **C. Schwab, & C. J. Gittelson**, Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs, *Acta Numerica.* 2011

[Weinan'17] **E. Weinan, H. Jiequn & A. Jentzen**, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations *Arxiv.* 2017