A hybrid parareal Monte-Carlo algorithm for parabolic problems *

Jad Dabaghi, Yvon Maday, Andrea Zoia

A hybrid parareal Monte-Carlo algorithm for parabolic problems*

Jad Dabaghi [†‡] Yvon Maday [‡] Andrea Zoia[†]

February 16, 2021

Abstract

In this work, we examine a hybrid Monte-Carlo/deterministic approach for a toy model based on the parabolic time-dependent diffusion equation.
We consider two different solvers: a low-cost "coarse" solver based on a deterministic Galerkin scheme and a "fine" solver based on a Monte-Carlo resolution. We use a hybrid "parareal-in-time" algorithm based on these two solvers to reduce the computational cost with respect to a full Monte-Carlo simulation.
In a set of benchmark numerical experiments, we compare our hybrid parareal strategy with a standard full Monte-Carlo solution of the time-dependent diffusion equation. In particular, we show that for a large number of processors, our hybrid strategy significantly reduces the computational time of the simulation while preserving its accuracy. The convergence properties of the proposed Monte-Carlo/deterministic parareal strategy are also discussed.

Keywords: parareal-in-time algorithm, time-dependent problems, predictor-corrector, Galerkin schemes, Monte-Carlo method.

1 Introduction

Let $\Omega \subset \mathbb{R}^d$, $d = \{1, 2\}$, be a polygonal domain and $T > 0$ be the upper boundary of the time domain $[0, T] \in \mathbb{R}$. Let $L^2(\Omega)$ be the Hilbert space of square integrable functions on Ω. Consider the following system of parabolic equations: find u such that

$$\begin{aligned} \partial_t u - \mathcal{D}\Delta u = g \quad &\text{in} \quad]0, T[\times \Omega \\ u(0, \cdot) = u_0 \quad &\text{in} \quad \Omega. \end{aligned} \tag{1.1}$$

Here, $\mathcal{D} > 0$ is a diffusion tensor supposed constant for the sake of simplicity, $g \in L^2(]0, T[\times \Omega)$ is a known function, and $u_0 \in L^2(\Omega)$ the initial condition. Boundary conditions prescribed on the boundary domain $\partial\Omega$ are arbitrary: for instance, when homogeneous Dirichlet boundary conditions are prescribed, $u(t, \boldsymbol{x}) = 0$ on $]0, T[\times \partial\Omega$; when homogeneous Neumann boundary conditions are prescribed, $\dfrac{\partial u}{\partial \boldsymbol{n}}(t, \boldsymbol{x}) = 0$ on $]0, T[\times \partial\Omega$; and when homogeneous Robin boundary conditions are applied, $\dfrac{\partial u}{\partial \boldsymbol{n}}(t, \boldsymbol{x}) + u(t, \boldsymbol{x}) = 0$ on $]0, T[\times \partial\Omega$. Here, $\boldsymbol{n}$ stands for the outward unit

*This project has received funding from the ANR project "Ciné-Para" (ANR-15-CE23-0019).

[†]Université Paris-Saclay, CEA, Service d'études des réacteurs et de mathématiques appliquées, 91191, Gif-sur-Yvette, France

[‡]Sorbonne Université, CNRS, Université de Paris, Laboratoire Jacques-Louis Lions (LJLL), F-75005 Paris, France & Institut Universitaire de France

normal vector to the boundary $\partial\Omega$. We mention the fundamental books of Lions [28], Dautray and Lions [12], and Brezis [10], for a complete analysis. Solving analytically Problem (1.1) for arbitrary domains may be challenging and numerical simulations appear to be often the only viable approach. Among the wide range of numerical methods for solving (1.1), we mention the deterministic approach, based for instance on the finite element methods [7, 9, 37], the finite volume methods [14, 17, 20], or the discontinuous Galerkin methods [15, 16, 38], and the probabilistic approach based on Monte-Carlo methods [1,13,18,23,26,32,34,42,44], see also the references therein. Monte-Carlo algorithms are advantageous for high-dimensional problems and are natively implemented over massively parallel computing environments [25, 45]. In general, the Monte-Carlo algorithm associated to problem (1.1) consists in approximating the solution u by sampling a finite number M of random walks whose estimated density at point $\boldsymbol{x}$ and time t converges in the limit of a large number of particles to $u(\boldsymbol{x}, t)$. It is well known that the Monte-Carlo method displays a $1/\sqrt{M}$ convergence as a result of the Central Limit Theorem [22,36]. Obtaining a numerical solution sufficiently close to the exact one requires to consider a very large number of sampled points, wich demands a high computational cost. In the last decades, Monte-Carlo methods have been applied in several domains such as radiation transport [30] or molecular dynamics [31]. In the present work, we are interested in reducing the numerical cost of a Monte-Carlo simulation by proposing a hybrid version of a Monte-Carlo solver based on the parareal-in-time algorithm.

The parareal algorithm was presented by Lions, Maday, and Turinici [29] as a numerical method to solve the time evolution of dynamical systems in a parallel fashion. It involves two propagators $\mathcal{F}$ and $\mathcal{G}$ that integrate (1.1). The propagator $\mathcal{F}$ is a fine, accurate and thus expensive propagator, which approximates the exact solution u with high precision, whereas the propagator $\mathcal{G}$ is a coarse propagator, which is a less accurate approximation of the exact solution u and thus much less expensive than $\mathcal{F}$. These solvers can, e.g., be based on different time steps (δt for $\mathcal{F}$ being much smaller than Δt for $\mathcal{G}$), but the model that $\mathcal{G}$ approximates may also be a simplified one.

Let $T_0 = 0 < T_1 < \cdots < T_N = T$ be a sequence of times. For the sake of simplicity, we choose here $T_n = n\Delta T$ for some appropriate time interval ΔT. The parareal algorithm constructs a sequence $\boldsymbol{u}_k := (\boldsymbol{u}_k^n)_{1\leq n\leq N}$ such that $\forall k \geq 0$, $\boldsymbol{u}_k^n$ is an approximation of $u^n := u(n\Delta T)$. For the iteration $k = 0$, the initial approximation is obtained at each time step n using the coarse propagator $\mathcal{G}$ over a propagation length of fixed size ΔT (we denote by $\mathcal{G}_{\Delta T}$ such a coarse evolution) :

$$u_{k=0}^{n+1} := \mathcal{G}_{\Delta T}(u_{k=0}^n),$$

where $u_{k=0}^0 := u_0$. Next, we perform a prediction, followed by a correction iteration

$$u_{k+1}^{n+1} := \underbrace{\mathcal{G}_{\Delta T}(u_{k+1}^n)}_{\text{Prediction}} + \underbrace{[\mathcal{F}_{\Delta T}(u_k^n) - \mathcal{G}_{\Delta T}(u_k^n)]}_{\text{Correction}}, \tag{1.2}$$

where $u_{k+1}^0 := u_0$. Note that the coarse solver $\mathcal{G}$ is sequential, whereas the fine solver $\mathcal{F}$ computes for a fixed step k the corrections in parallel. When $k \to \infty$, equation (1.2) yields $u_{k+1}^{n+1} = \mathcal{F}_{\Delta T}(u_k^n)$ and thus the final approximation is achieved by the accuracy of the fine propagator $\mathcal{F}$. The convergence of the parareal algorithm is described in [3, 4, 19, 29]. In particular, in [19] is proven a superlinear bound on the convergence on bounded time intervals for the diffusion equation and the advection equation. Parareal methods have been successfully applied to deterministic neutron transport problems [5, 6], pricing in American options [35], molecular dynamics simulations [3], and multiscale-in-time systems [27]. In the present work, we consider a fine Monte-Carlo propagator and a coarse Galerkin scheme propagator, thus leading to a hybrid version of a parareal algorithm devoted to parabolic problems. To the best of our knowledge, this scheme has never been considered before. Our hybrid approach is presented here in the context of the time-dependent diffusion equation.

This paper is organized as follows. First, in Section 2, we expose our model problem and settings. Next, in Section 3 we present some admissible deterministic coarse propagators. Section 4 focuses on the fine solver in terms of a standard Monte-Carlo algorithm. In Section 5, we introduce our hybrid parareal scheme for the time-dependent diffusion equation. Finally, in Section 6 we present a set of benchmark numerical experiments so as to ascertain the features of the proposed approach.

2 Model problem and setting

Let $H^1(\Omega)$ be the space of L^2 functions on the domain Ω which admit a weak gradient in $L^2(\Omega)$ and let $H^1_0(\Omega)$ be its zero-trace subspace. We denote by $H^{-1}(\Omega)$ the dual space of $H^1_0(\Omega)$ with the duality pairing $\langle \cdot, \cdot \rangle_{H^{-1}(\Omega),H^1_0(\Omega)}$. We consider the time-dependent diffusion equation with homogeneous Dirichlet boundary conditions: Find u such that

$$\begin{aligned} \partial_t u - \mathcal{D}\Delta u = 0 \quad &\text{in} \quad]0,T[\times \Omega, \\ u = 0 \quad &\text{on} \quad]0,T[\times \partial\Omega, \\ u(0,\cdot) = u_0 \quad &\text{in} \quad \Omega. \end{aligned} \tag{2.1}$$

Here, $u_0 \in L^2(\Omega)$ is the initial condition. The weak formulation associated to (2.1) reads as follows: find $u \in L^2(0,T;H^1_0(\Omega))$ such that $\partial_t u \in L^2\left(0,T;H^{-1}(\Omega)\right)$ and satisfying for almost all $t \in]0,T[$ and for all $v \in L^2(0,T;H^1_0(\Omega))$

$$\langle \partial_t u, v \rangle_{H^{-1}(\Omega),H^1_0(\Omega)} + \mathcal{D} \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}x = 0. \tag{2.2}$$

3 Discretization methods and deterministic propagators

In this section, we present the numerical discretization of problem (2.2) to define the coarse propagator. In particular, we specify the coarse grid for the discretization in time. The Lagrange finite element method, the cell-centered finite volume method, and the discontinuous Galerkin method are presented.

3.1 Setting

For the time discretization, we introduce a division of the interval $[0,T]$ into subintervals $I_n := [t_{n-1}, t_n]$, $1 \le n \le N_t$, such that $0 = t_0 < t_1 < \cdots < t_{N_t} = T$. The time steps are denoted by $\Delta t_n = t_n - t_{n-1}$, $n = 1, \cdots, N_t$. For a function v with sufficient regularity, we denote $v^n := v(t^n)$, $0 \le n \le N_t$, and we define the approximation of the first-order time derivative thanks to the backward Euler scheme as follows:

$$\partial_t v^n := \frac{v^n - v^{n-1}}{\Delta t_n} \quad \forall\, 1 \le n \le N_t.$$

For the space discretization, we consider a conforming simplicial mesh $\mathcal{T}_h$ of the domain Ω, i.e. $\mathcal{T}_h$ is a set of triangles K verifying $\bigcup_{K \in \mathcal{T}_h} \overline{K} = \overline{\Omega}$, where the intersection of the closure of two elements of $\mathcal{T}_h$ is either an empty set, a vertex, or an edge. Denote by h_K the diameter of a triangle K and $h := \max_{K \in \mathcal{T}_h} h_K$. We denote by $\mathcal{V}_h$ the set of Lagrange nodes of $\mathcal{T}_h$. This set is partitionned into the interior nodes $\mathcal{V}_h^{\mathrm{int}}$ and the boundary nodes $\mathcal{V}_h^{\mathrm{ext}}$. Similarly, the nodes of an element $K \in \mathcal{T}_h$ are collected in the set $\mathcal{V}_K$ and we denote respectively by $\mathcal{V}_K^{\mathrm{int}}$ and $\mathcal{V}_K^{\mathrm{ext}}$ the set of the Lagrange nodes in $K \cap \Omega$ and in $K \cap \partial\Omega$. The number of Lagrange nodes of $\mathcal{T}_h$

is denoted by N_h and the number of internal Lagrange nodes is denoted by $N_h^{\rm int}$. We denote by $\mathcal{E}_h$ the set of mesh edges. Boundary edges are collected in the set $\mathcal{E}_h^{\rm ext} = \{\sigma \in \mathcal{E}_h; \sigma \subset \partial\Omega\}$ and internal edges are collected in the set $\mathcal{E}_h^{\rm int} = \mathcal{E}_h \backslash \mathcal{E}_h^{\rm ext}$. To each edge $\sigma \in \mathcal{E}_h$, we associate a unit normal vector $\boldsymbol{n}_\sigma$; for $\sigma \in \mathcal{E}_h^{\rm int}$, $\sigma = K \cap L$, $\boldsymbol{n}_\sigma$ points from K towards L and for $\sigma \in \mathcal{E}_h^{\rm ext}$ it coincides with the outward unit normal vector $\boldsymbol{n}_\Omega$ of Ω. The jump operator $[\![\cdot]\!]_\sigma$ yielding the difference of the argument from the two mesh elements that share $\sigma \in \mathcal{E}_h^{\rm int}$ is defined as $[\![v]\!]_\sigma := (v|_K - v|_L)\,\boldsymbol{n}_\sigma$, where $\sigma = \overline{K} \cap \overline{L}$ and the average of v on $\sigma \in \mathcal{E}_h^{\rm int}$ is defined as $\{\{v\}\}_\sigma := \frac{1}{2}(v|_K + v|_L)$. When $\sigma \in \mathcal{E}_h^{\rm ext}$, $\sigma = \partial K \cap \partial\Omega$, $[\![v]\!]_\sigma := v\boldsymbol{n}_\Omega$ and $\{\{v\}\}_\sigma = v$. We also denote by $N_{\rm sp}$ the number of elements in the mesh $\mathcal{T}_h$. For any spatial discretization method, the numerical unknown associated to (2.2) is obtained via the following procedure. Knowing $\boldsymbol{U}^{n-1} \in \mathbb{R}^m$, $m \geq 1$, the unknown vector $\boldsymbol{U}^n \in \mathbb{R}^m$ satisfies

$$\boldsymbol{U}^n = \mathcal{G}_{\Delta t_n}(\boldsymbol{U}^{n-1}). \tag{3.1}$$

where $\mathcal{G}_{\Delta t_n}(\boldsymbol{U})$ stands for the coarse discrete evolution over a time range of Δt_n (i.e. one coarse step).

In the following, we explicit for various discretization schemes the definition of the coarse propagator $\mathcal{G}_{\Delta t_n}$ associated to (2.1). Note that the time step Δt_n of the coarse propagator is smaller than the window ΔT.

3.2 The Lagrange finite element propagator

In this section, we assume that $p \geq 1$. We define the conforming spaces

$$\begin{aligned} X_h^p &:= \left\{v_h \in \mathcal{C}^0(\overline{\Omega});\ v_h|_K \in \mathbb{P}_p(K)\ \ \forall K \in \mathcal{T}_h\right\} \subset H^1(\Omega), \\ X_{0h}^p &:= X_h^p \cap H_0^1(\Omega), \end{aligned}$$

where $\mathbb{P}_p(K)$ stands for the set of polynomials of total degree less than or equal to p on the element K. The Lagrange basis functions of X_h^p are denoted by $(\psi_{h,l})_{1\leq l\leq N_h}$ for $\boldsymbol{x}_l \in \mathcal{V}_h$. We recall that $\psi_{h,l}(\boldsymbol{x}_l') = \delta_{l,l'}$ (the Kronecker symbol) for all $1 \leq l, l' \leq N_h$. Given the data $u_h^0 := u_0 \in L^2(\Omega)$, the discrete weak formulation associated to (2.2) consists in searching, for all $1 \leq n \leq N_t - 1$, $u_h^n \in X_{0h}^p$ such that for all $v_h \in X_{0h}^p$

$$\frac{1}{\Delta t_n}\int_\Omega \left(u_h^n - u_h^{n-1}\right) v_h \,\mathrm{d}x + \mathcal{D}\int_\Omega \boldsymbol{\nabla} u_h^n \cdot \boldsymbol{\nabla} v_h \,\mathrm{d}x = 0. \tag{3.2}$$

Expressing u_h^n in the Lagrange basis $(\psi_{h,l})_{1\leq l\leq N_h^{\rm int}}$, problem (3.2) reads

$$\mathbb{A}^n \boldsymbol{U}^n = \boldsymbol{F}^{n-1}. \tag{3.3}$$

Here, $\boldsymbol{U}^n \in \mathbb{R}^{N_h^{\rm int}}$ is the unknown vector expressed nodewise, satisfying

$$u_h^n := \sum_{l=1}^{N_h^{\rm int}} (\boldsymbol{U}^n)_l \,\psi_{h,l},$$

and $\mathbb{A}^n \in \mathbb{R}^{N_h^{\rm int}, N_h^{\rm int}}$ is a sparse matrix defined by

$$\mathbb{A}_{l,l'}^n := \frac{1}{\Delta t_n}\int_\Omega \psi_{h,l}\psi_{h,l'} \,\mathrm{d}x + \mathcal{D}\int_\Omega \boldsymbol{\nabla}\psi_{h,l} \cdot \boldsymbol{\nabla}\psi_{h,l'} \,\mathrm{d}x \quad \forall 1 \leq l, l' \leq N_h^{\rm int}.$$

The right-hand side vector $\boldsymbol{F}^{n-1} \in \mathbb{R}^{N_h^{\rm int}}$ is defined as

$$\left[\boldsymbol{F}^{n-1}\right]_l := \frac{1}{\Delta t_n}\int_\Omega u_h^{n-1}\psi_{h,l} \,\mathrm{d}x \quad \forall 1 \leq l \leq N_h^{\rm int}.$$

In practice, we choose for the sake of simplicity, all the time steps to be equal : $\Delta t_n = \Delta t$, and thus applying the propagator $\mathcal{G}_{\Delta t_n}$, amounts to solving (3.3) that yields:

$$\boldsymbol{U}^n = \mathcal{G}_{\Delta t_n}\left(\boldsymbol{U}^{n-1}\right) \quad \text{with} \quad \mathcal{G}_{\Delta t_n}\left(\boldsymbol{U}^{n-1}\right) := \mathbb{A}^{-1} \times \boldsymbol{F}^{n-1}.$$

3.3 The cell-centered finite volume propagator

In the case of the cell-centered finite volume method, we assume that the family $\mathcal{T}_h$ is superadmissible, in the sense that for all cells $K \in \mathcal{T}_h$ there exists a point $\boldsymbol{x}_K \in K$ (the center of the cell) and for all edges $\sigma \in \mathcal{E}_h$ there exists a point $\boldsymbol{x}_\sigma \in \sigma$ (the center of the edge) such that, for all edges $\sigma \in \mathcal{E}_K$, the line segment joining $\boldsymbol{x}_K$ with $\boldsymbol{x}_\sigma$ is orthogonal to σ [17]. For an interior edge $\sigma \in \mathcal{E}_h^{\rm int}$ shared by two elements K and L (denoted in the sequel by $\overline{\sigma} = \overline{K} \cap \overline{L}$) we define the distance between these elements $d_{KL} := \mathrm{dist}(\boldsymbol{x}_K, \boldsymbol{x}_L)$. For an exterior edge $\sigma \in \mathcal{E}_h^{\rm ext}$ the distance between the element K and the edge σ is denoted by $d_{K\sigma} := \mathrm{dist}(\boldsymbol{x}_K, \boldsymbol{x}_\sigma)$. To approximate the space gradient we use

$$\begin{aligned}
&\int_\sigma \boldsymbol{\nabla} v \cdot \boldsymbol{n}_{K,\sigma} \,\mathrm{ds} \approx |\sigma| \frac{v_L - v_K}{d_{KL}}, \quad \sigma \in \mathcal{E}_K^{\rm int}, \quad \sigma = \overline{K} \cap \overline{L}, \\
&\int_\sigma \boldsymbol{\nabla} v \cdot \boldsymbol{n}_{K,\sigma} \,\mathrm{ds} \approx -|\sigma| \frac{v_K}{d_{K\sigma}}, \qquad \sigma \in \partial\Omega.
\end{aligned}$$

Here, the notation $\boldsymbol{n}_{K,\sigma}$ stands for the outward unit normal vector to the element K on the edge σ. Obviously, when $\sigma \in \mathcal{E}_h^{\rm int}$, $\sigma = \overline{K} \cap \overline{L}$, $\boldsymbol{n}_{K,\sigma} = -\boldsymbol{n}_{L,\sigma}$. Furthermore, $v_K \approx \dfrac{1}{|K|} \displaystyle\int_K v \,\mathrm{d}x$ is an approximation of v in the volume K and v_σ is an approximation of v on the edge σ. Using the cell-centered finite volume method, the unknown of the model is discretized using a single constant value per cell: $\forall n,\ 1 \le n \le N_t - 1$ we let

$$\boldsymbol{U}^n := (u_K^n)_{K \in \mathcal{T}_h} \in \mathbb{R}^{N_{\rm sp}}.$$

By integration of (2.1) over the element K and using the Green's formula we obtain

$$\frac{|K|}{\Delta t_n} u_K^n + \mathcal{D} \sum_{\sigma \in \mathcal{E}_K^{\rm int}} \mathfrak{F}_{K,\sigma}^n - |\sigma| \frac{u_K^n}{d_{K\sigma}} = Q_K^{n-1} \quad \forall K \in \mathcal{T}_h. \tag{3.4}$$

Here, the conservative numerical flux $\mathfrak{F}_{K,\sigma}^n = -\mathfrak{F}_{L,\sigma}^n$, for an internal edge $\sigma = \mathcal{E}_h^{\rm int}$, $\sigma = \overline{K} \cap \overline{L}$, is given by

$$\mathfrak{F}_{K,\sigma}^n := -|\sigma| \frac{u_L^n - u_K^n}{d_{KL}}.$$

The elementwise right-hand side Q_K^{n-1} is defined by

$$Q_K^{n-1} := -\frac{|K|}{\Delta t_n} u_K^{n-1}. \tag{3.5}$$

Problem (3.4) reads then as follows: Find $\boldsymbol{U}^n \in \mathbb{R}^{N_{\rm sp}}$ such that $\boldsymbol{U}^n = \mathcal{G}_{\Delta t}(\boldsymbol{U}^{n-1})$, where (under the same hypothesis $\Delta t_n = \Delta t$)

$$\mathcal{G}_{\Delta t}(\boldsymbol{U}^{n-1}) := \mathbb{A}^{-1} \times \boldsymbol{F}^{n-1}.$$

Here, $\mathbb{A}^{n-1} \in \mathbb{R}^{N_{\rm sp}, N_{\rm sp}}$ has a sparse structure and the right-hand side vector $\boldsymbol{F}^{n-1} \in \mathbb{R}^{N_{\rm sp}}$ is defined locally by (3.5).

3.4 The discontinuous Galerkin propagator

In this section, we consider the discontinuous Galerkin method (or DG method) for solving problem (2.1). This time, the unknowns are local and $N_h^{\rm int}$ denotes the total number of local internal degrees of freedom which is different and higher than for the finite element method. We define the discontinuous Galerkin space

$$\begin{aligned}
X_h^p &:= \left\{ v_h \in L^2(\Omega);\ v_h|_K \in \mathbb{P}_p(K)\ \ \forall K \in \mathcal{T}_h \right\} \not\subset H^1(\Omega), \\
X_{0h}^p &:= \left\{ v_h \in L^2(\Omega);\ v_h|_K \in \mathbb{P}_p(K)\ \ \forall K \in \mathcal{T}_h,\ \ v_h|_{\partial\Omega} = 0 \right\} \not\subset H_0^1(\Omega).
\end{aligned}$$

Next, we introduce the bilinear form

$$a_h(v_h, w_h) := \sum_{K\in\mathcal{T}_h} \int_K \boldsymbol{\nabla} v_h \cdot \boldsymbol{\nabla} w_h \,\mathrm{d}x + \mathcal{A}_h(v_h, w_h). \tag{3.6}$$

Here, the bilinear symmetric form $\mathcal{A}_h$ consists of all consistency and stability terms. Several choices are possible for the bilinear symmetric form $\mathcal{A}_h$, provided that the bilinear form a_h is coercive and bounded with respect to the following norm on X_{0h}^p:

$$\|v_h\|_{X_{0h}^p}^2 := \sum_{K\in\mathcal{T}_h} \|\boldsymbol{\nabla} v_h\|_K^2 + \sum_{\sigma\in\mathcal{E}_h} \frac{1}{h_\sigma} \|[\![v_h]\!]_\sigma\|_\sigma^2 \tag{3.7}$$

where h_σ is the diameter of the edge σ. We mention for instance the SIPG method [2]: For $\gamma > 0$ (large enough), $\forall v_h \in X_h^p$ and $\forall w_h \in X_h^p$

$$\mathcal{A}_h(v_h, w_h) := -\sum_{\sigma\in\mathcal{E}_h} \int_\sigma \left(\{\{\boldsymbol{\nabla} w_h\}\}_\sigma [\![v_h]\!]_\sigma + \{\{\boldsymbol{\nabla} v_h\}\}_\sigma [\![w_h]\!]_\sigma\right) \mathrm{d}s + \sum_{\sigma\in\mathcal{E}_h} \frac{\gamma}{h_\sigma} \int_\sigma [\![w_h]\!]_\sigma [\![v_h]\!]_\sigma \,\mathrm{d}s, \tag{3.8}$$

or the NIPG method [39]: For $\gamma > 0$, $\forall v_h \in X_h^p$ and $\forall w_h \in X_h^p$

$$\mathcal{A}_h(v_h, w_h) := -\sum_{\sigma\in\mathcal{E}_h} \int_\sigma \left(\{\{\boldsymbol{\nabla} w_h\}\}_\sigma [\![v_h]\!]_\sigma - \{\{\boldsymbol{\nabla} v_h\}\}_\sigma [\![w_h]\!]_\sigma\right) \mathrm{d}s + \sum_{\sigma\in\mathcal{E}_h} \frac{\gamma}{h_\sigma} \int_\sigma [\![w_h]\!]_\sigma [\![v_h]\!]_\sigma \,\mathrm{d}s. \tag{3.9}$$

Other choices are also possible [15, 16, 38].
Considering the bilinear form a_h defined by (3.6), (3.8), and (3.9) the discontinuous Galerkin scheme reads

$$\sum_{K\in\mathcal{T}_h} \frac{1}{\Delta t_n} \int_K \left(u_h^n - u_h^{n-1}\right) v_h \,\mathrm{d}x + \mathcal{D}\; a_h(u_h^n, v_h) = 0. \tag{3.10}$$

Expressing u_h^n in the Lagrange basis $(\psi_{h,l})_{1\le l\le N_h^{\mathrm{int}}}$, problem (3.10) reads

$$\mathbb{A}^n \boldsymbol{U}^n = \boldsymbol{F}^{n-1} \tag{3.11}$$

where the unknown vector $\boldsymbol{U}^n \in \mathbb{R}^{N_h^{\mathrm{int}}}$ is defined by

$$\boldsymbol{U}^n := (\boldsymbol{U}_K^n)_{K\in\mathcal{T}_h}, \quad \text{and} \quad u_h^n|_K := \sum_{l=1}^{\dim(\mathbb{P}_p(K)))} (\boldsymbol{U}_K^n)_l \,\psi_{h,l} \quad \forall \boldsymbol{x}_l \in \mathcal{V}_K^{\mathrm{int}}.$$

The sparse matrix $\mathbb{A}^n \in \mathbb{R}^{N_h^{\mathrm{int}}, N_h^{\mathrm{int}}}$ is defined by

$$\mathbb{A}_{l,l'}^n := \sum_{K\in\mathcal{T}_h} \left(\frac{1}{\Delta t_n} \int_K \psi_{h,l}\psi_{h,l'} \,\mathrm{d}x + \mathcal{D} \int_K \boldsymbol{\nabla}\psi_{h,l} \cdot \boldsymbol{\nabla}\psi_{h,l'} \,\mathrm{d}x\right) + \mathcal{A}_h(\psi_{h,l}, \psi_{h,l'}) \quad \forall 1 \le l, l' \le N_h^{\mathrm{int}}$$

and the right-hand side vector satisfies

$$\boldsymbol{F}^{n-1} := \left(\boldsymbol{F}_K^{n-1}\right)_{K\in\mathcal{T}_h} \quad \text{with} \quad \left[\boldsymbol{F}_K^{n-1}\right]_l := \frac{1}{\Delta t_n} \int_K u_h^{n-1}\psi_{h,l} \,\mathrm{d}x \quad \forall \boldsymbol{x}_l \in \mathcal{V}_K^{\mathrm{int}}.$$

Finally, (3.11) is solved by inverting $\mathbb{A}$ (still under the hypothesis $\Delta t_n = \Delta t$)

$$\boldsymbol{U}^n = \mathcal{G}_{\Delta t_n}\left(\boldsymbol{U}^{n-1}\right) \quad \text{with} \quad \mathcal{G}_{\Delta t_n}\left(\boldsymbol{U}^{n-1}\right) := \mathbb{A}^{-1} \times \boldsymbol{F}^{n-1}.$$

Remark 3.1. *Note that for each of the discretization methods described above, the resulting linear system could also be solved by an iterative algebraic solver, which is a popular approach to speed up the numerical resolution. We mention for instance the GMRES [41], the PCG [24] and the multigrid algorithm [11]. In the present case, the matrix $\mathbb{A}$ is symmetric positive definite: then the fastest iterative solver would be the multigrid algorithm.*

Remark 3.2. *The resolution of (2.1) is also possible for different boundary conditions. For instance, when Neumann boundary conditions are used, $H^1(\Omega)$ is the set of test functions. Concerning the finite element discretization, the number of unknowns is set to N_h. When the cell-centered finite volume method is used, the number of unknowns is still equal to $N_{\rm sp}$ (except that in (3.4) the boundary terms are eliminated). For the discontinuous Galerkin method, the total number of unknowns is the total number of degrees of freedom N_h.*

4 The Monte-Carlo solver as a fine propagator

In this section, we propose a resolution of (2.1) by the Monte-Carlo method. In the Monte-Carlo procedure, we propagate a finite number of particles following an appropriate stochastic process over a time window. The positions of these particles at the end of the window are then used to estimate the solution of (2.1) in each element K of a given mesh $\mathcal{T}_h$ (the properties of this mesh $\mathcal{T}_h$ are for instance similar to the one of the DG method). Each particle carries a statistical weight, assigned according the appropriate rules (see Section 5), and representative of the contribution of the particle to the sought response and that may vary. We consider in the sequel "analog" Monte-Carlo methods where the weights are fixed in the course of the simulation. Note also that statistical weights could evolve and in this case the Monte-Carlo methods are called "non analog", see [40]. In the sequel, $M \geq 1$ denotes the number of particles. At the beginning of a Monte-Carlo computation, we have to sample a particle population corresponding to the initial condition u_0. Next, we have to specify how to describe each particle history starting from the initial condition and until the particle either leaves the viable domain or attains the final simulation time.

4.1 Sampling

In this section, we detail the sampling procedure according to a given strictly positive probability density function (PDF) denoted by f. We recall several techniques available in the literature and we detail the case $d = 1$ as all multidimensional samplings are extensions to the monodimensional sampling. Note that when $d = 1$, the domain Ω is partitionned into intervals $E_i := [x_{i-1}, x_i]$, $1 \leq i \leq N_{\rm sp}$ where $\overline{\Omega} = \cup_{i=1}^{N} \overline{E_i}$ and that $\overline{\Omega} = \left[x_0, x_{N_{\rm sp}}\right]$. We recall that a PDF f satisfies: $f > 0$ a.e. on Ω and $\displaystyle\int_\Omega f(y)\,{\rm d}y = 1$.

4.1.1 Direct inversion of the cumulative distribution

In this section, we recall a theoretical tool to sample from a probability density function f defined over the domain Ω. Define the cumulative function $F : \Omega \to [0, 1]$ associated to the PDF $f : \Omega \to \mathbb{R}_+^*$ by

$$F(x) := \int_{x_0}^{x} f(y)\,{\rm d}y. \tag{4.1}$$

Let $\xi \sim \mathcal{U}([0, 1])$ be a random variable that obeys a uniform law on the interval $[0, 1]$. Compute the inverse function:

$$\overline{x} = F^{-1}(\xi). \tag{4.2}$$

Equation (4.2) is also known as the inversion theorem of the cumulative [30]. We repeat this procedure M times to obtain a collection of M independent and identically distributed particles obeying the PDF f.

Remark 4.1. *The direct inversion method often involves complicated functions and in some cases the cumulative function F is hard to invert.*

4.1.2 The table lookup method

In this case, we invert numerically the cumulative function F (associated to the PDF f) whose inverse is often hard to compute analytically. First, we construct for each interval E_i the associated cumulative distribution

$$F_i := \sum_{j=1}^{i} \int_{x_{j-1}}^{x_j} f(x)\,\mathrm{d}x.$$

Let $\xi \sim \mathcal{U}([0,1])$. Obviously, there exists a unique $i \in [1, N_{\mathrm{sp}}]$ such that $F_{i-1} \leq \xi < F_i$. By a linear interpolation, the approximate solution of the equation $F(\overline{x}) = \xi$ is given by

$$\overline{x} := \frac{(x_i - x_{i-1})\,\xi - x_i F_{i-1} + x_{i-1} F_i}{F_i - F_{i-1}}.$$

We repeat this procedure M times to obtain a set of sampled particle positions obeying the interpolant of f.

4.1.3 Rejection method

Next, we present the rejection method, often used when a simpler PDF is available. In this case, we assume there exists a PDF $\hat{g} : \Omega \to \mathbb{R}_+^*$ "easy" to sample from, such that

$$f(x) \leq k\hat{g}(x), \quad \forall x \in \Omega$$

with $k \geq 1$ a constant.

We set $\alpha(x) = \dfrac{f(x)}{k\hat{g}(x)}$. First, we compute the cumulative distribution $G : \Omega \to [0,1]$ associated to the density $\hat{g}$ following (4.1). Next, we sample according to the PDF $\hat{g}$ the position $\overline{x}$ employing the direct inversion of the cumulative procedure (4.2) and we compute $\alpha(\overline{x})$. Then, we consider a uniform random number ξ following the uniform law on the interval $[0,1]$: $\xi \sim \mathcal{U}([0,1])$. If $\xi \leq \alpha(\overline{x})$, then accept $\overline{x}$. Otherwise, reject it and go back to the first step. This method provides a sequence of values x obeying the PDF f that is hard to simulate [30, Theorem 2.5].

4.2 Sampling of the initial condition

Suppose now that the initial condition u_0 is a probability density function. We use a sampling procedure given above to obtain a population of particles $\boldsymbol{X}^0 \in \mathbb{R}^M$. If $u_0 > 0$ is not a probability density function, i.e. $\int_\Omega u_0(x)\,\mathrm{d}x \neq 1$, we sample from the PDF $\tilde{u}_0$ defined by $\tilde{u}_0(x) := \dfrac{u_0(x)}{\int_\Omega u_0(y)\,\mathrm{d}y}$. Obviously, $\tilde{u}_0 > 0$ and $\int_\Omega \tilde{u}_0(x)\,\mathrm{d}x = 1$. We obtain a collection of particle positions that we denote by $\tilde{\boldsymbol{X}}^0$. Finally, to obtain a population of particles $\boldsymbol{X}^0 \in \mathbb{R}^M$ corresponding to the initial condition u_0, we consider the population $\tilde{\boldsymbol{X}}^0$ and we attribute to each particle i a statistical weight equal to $\omega_i = \int_\Omega u_0(y)\,\mathrm{d}y$. Note that, when u_0 is already a PDF, we assign to each particle i (after sampling) a statistical weight $\omega_i = 1$.

4.3 Simulation of a Brownian motion

Concerning the fine Monte-Carlo solver we consider a subdivision of the interval $[0,T]$ into subintervals $[\tilde{t}_{n-1},\tilde{t}_n]$, $1 \leq n \leq N^\star$ such that $0 = \tilde{t}_0 < \tilde{t}_1 < \ldots < \tilde{t}_{N^\star} = T$. The time steps are denoted by $\delta t_n = \tilde{t}_n - \tilde{t}_{n-1}$, $n = 1, \cdots, N^\star$. The underlying stochastic process for the diffusion equation whose diffusion coefficient is $\mathcal{D} > 0$ is a Brownian motion or a Wiener process [21, 33, 43]. The variation of a Wiener process, over each time interval ($\delta t_n \to 0$), is a continuous Gaussian probability density function. We simulate the Brownian motion at times $\tilde{t}_n$, $n = 0, \cdots N^\star - 1$. Knowing the position $\boldsymbol{x}'$ at time $t' > 0$ of a given particle, we determine its subsequent position $\boldsymbol{x}$ at time $t > t'$ by sampling the Gaussian transition kernel

$$T(\boldsymbol{x}', t' \to \boldsymbol{x}, t) := \frac{1}{\sqrt{2\pi\mathcal{D}\,(t-t')}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2\mathcal{D}\,(t-t')}\right). \tag{4.3}$$

We say that the particle displacement $\boldsymbol{x} - \boldsymbol{x}'$ follows the normal distribution law whose mean is 0 and whose standard deviation is $\mathcal{D}(t-t')$ i.e. $\boldsymbol{x} - \boldsymbol{x}' \sim \mathcal{N}(0, \mathcal{D}(t-t'))$. For the sake of clarity, the notation $(\boldsymbol{x}', t' \to \boldsymbol{x}, t)$ means that we randomly displace the particle having the position $\boldsymbol{x}'$ at time t' to position $\boldsymbol{x}$ and time t.

Based on (4.3), the position of the particles at time t denoted by $\boldsymbol{X}^t$ when we know the positions of the particles at time t' denoted by $\boldsymbol{X}^{t'}$ is determined by the formula,

$$\boldsymbol{X}^t = \boldsymbol{X}^{t'} + \sqrt{2\mathcal{D}(t-t')}\,\boldsymbol{\mathcal{S}} \tag{4.4}$$

where $\boldsymbol{\mathcal{S}} \in \mathbb{R}^M$ is a standard normal Gaussian vector.

A popular approach to generate Gaussian random variables $\mathcal{N}(0,1)$ is the Box-Muller algorithm. We refer to [8] for more details.

Finally, the approximation of the particle density $\displaystyle\int_K u^n(x)\,\mathrm{d}x$ at any time step n (where we recall that $u^n := u(n\Delta T)$), is given by

$$\int_K u^n(x)\,\mathrm{d}x \approx \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}^n_{i\in K} \times \omega_i. \tag{4.5}$$

Here, $\mathbf{1}^n_{i\in K}$ denotes the characteristic function such that $\mathbf{1}^n_{i\in K} = 1$ if the particle i (after its random walk) belongs to the element K at time $n\Delta T$ and is otherwise equal to 0. Note that for an infinite number of simulated particles $M \to +\infty$, the law of large numbers provides an exact computation in (4.5).

To take into account Dirichlet boundary conditions, the particles that cross the spatial boundaries $x = x_0$ or $x = x_{N_{\mathrm{sp}}}$ are killed. The Monte-Carlo computation presented in (4.5) is sequential in the sense that one processor is employed throughout the simulation. The variance and the standard deviation can be computed to determine the confidence intervals [36]. We present in the following Section the parallelized Monte-Carlo algorithm which is more practical and competitive.

Remark 4.2. *Concerning Neumann boundary conditions, the particles crossing the spatial boundary are reflected inside the domain ensuring mass conservation.*

4.4 Parallelized Monte-Carlo

In practice, we realize the Monte-Carlo computation $p > 1$ times independently. We simulate $p > 1$ batches or replicas of M' particles so that $M = p \times M'$. This is a more convenient manner to compute the Monte-Carlo expectation (4.5) as it allows parallelization per batches. More precisely, one processor is devoted to one batch. For a given M, the greater the number of

batches, the faster the computation. Furthermore, the Monte-Carlo computation per batches is a practical way to compute the variance (see below). In this case, we denote by $Z^n_{K,j}$ the score associated to the element $K \in \mathcal{T}_h$ in the batch $j \in [1,p]$ at the fixed time step n. This score is defined as the result of the Monte-Carlo computation (4.5) for the batch j:

$$Z^n_{K,j} := \frac{1}{M'} \sum_{i=1}^{M'} \mathbf{1}^n_{i\in K,j} \times \omega_i \tag{4.6}$$

Here, $\mathbf{1}^n_{i\in K,j}$ denotes the characteristic function for the replica j such that $\mathbf{1}^n_{i\in K,j} = 1$ if the particle i belongs to the element K, and is otherwise equal to 0. Then, the particle density is approximated by

$$\int_K u^n(x)\,\mathrm{d}x \approx \frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j}. \tag{4.7}$$

Note that for a very large number of processors the previous Monte-Carlo computation is exact as a result of the law of large numbers. The variance denoted by $\mathrm{Var}(Z^n_K)$ is defined in $K \in \mathcal{T}_h$ by

$$\mathrm{Var}(Z^n_K) := \frac{1}{p-1}\left(\frac{1}{p}\sum_{j=1}^{p}\left(Z^n_{K,j}\right)^2 - \left(\frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j}\right)^2\right).$$

The standard deviation denoted by $\widehat{\sigma}^n$ is defined by

$$\widehat{\sigma}^n := \sqrt{\mathrm{Var}(Z^n_K)}.$$

We also define the Monte-Carlo error bars in each mesh element $K \in \mathcal{T}_h$:

$$\hat{I}^n_K := \left[\frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} - 1\widehat{\sigma}^n, \frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} + 1\widehat{\sigma}^n\right], \tag{4.8}$$

$$I^n_K := \left[\frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} - 2\widehat{\sigma}^n, \frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} + 2\widehat{\sigma}^n\right], \tag{4.9}$$

$$\tilde{I}^n_K := \left[\frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} - 3\widehat{\sigma}^n, \frac{1}{p}\sum_{j=1}^{p} Z^n_{K,j} + 3\widehat{\sigma}^n\right]. \tag{4.10}$$

From the probability theory [22], the probability for the exact solution u at time T_n to be in $\hat{I}^n_K$ is approximately equal to 68%, the probability to be in I^n_K is approximately equal to 95%, and the probability for the exact solution to be in $\tilde{I}^n_K$ is approximately equal to 99.7%.

5 A hybrid parareal Monte-Carlo algorithm

We want to build a hybrid parareal scheme with a coarse propagator $\mathcal{G}$ given by a deterministic solver, and a fine propagator $\mathcal{F}$ given by a Monte-Carlo solver. Let $\boldsymbol{U}^n \in \mathbb{R}^m$ be the deterministic solution. When the finite element method and the discontinuous Galerkin method are employed $m = N^{\mathrm{int}}_h$, and when the cell-centered finite volume method is considered, $m = N_{\mathrm{sp}}$. The statistical representation of $\boldsymbol{U}^n$ is still denoted by $\boldsymbol{X}^n$ and to simplify the notations, we denote by $\mathcal{F}_{\Delta T}(\boldsymbol{U}^n)$ the whole Monte-Carlo computation. In fact, in this notation is gathered the statistical representation $\boldsymbol{X}^n$ of $\boldsymbol{U}^n$, the fine discrete evolution over a time range of ΔT, and the averaging step (4.7). Let $1 \le k \le K$, be the parareal index such that $\boldsymbol{U}^{n+1}_k$ is the approximation of $\boldsymbol{U}^{n+1}$.

The numerical solution obtained for a batch $j \in [1, p]$ at parareal iteration k is denoted by $\boldsymbol{U}_{k,j}^{n+1}$. Then, the final solution $\boldsymbol{U}_k^{n+1}$ is obtained by the averaging

$$\boldsymbol{U}_k^{n+1} := \frac{1}{p} \sum_{j=1}^{p} \boldsymbol{U}_{k,j}^{n+1}. \tag{5.1}$$

The hybrid Monte-Carlo algorithm that we propose is the following.

Algorithm 1 Hybrid Monte-Carlo Algorithm

1. Initialization: Choose an initial vector $\boldsymbol{U}^0 \in \mathbb{R}^m$ and compute a coarse approximation $\boldsymbol{U}_{k=0}^{n+1} \in \mathbb{R}^m$ of the numerical unknown $\boldsymbol{U}^{n+1} \in \mathbb{R}^m$ at each time observable $(n+1)\,\Delta T$, $0 \leq n \leq N-1$, by the coarse propagator

$$\boldsymbol{U}_{k=0}^{n+1} := \mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=0}^{n}) \quad \text{where} \quad \boldsymbol{U}_{k=0}^{0} := \boldsymbol{U}^0. \tag{5.2}$$

Consider $M' > 1$ particles.

for $k = 1 : K$
  for $j = 1 : p$ (in parallel)
    for $n = 0 : N-1$ (in parallel)

2. if $k = 1$
Compute the statistical representation $\boldsymbol{X}_{0,j}^{n} \in \mathbb{R}^{M'}$ employing the sampling procedure (see Section 4.1) from $\boldsymbol{U}_0^n$ with uniform weights.
else
When $n = 0$, use the statistical representation $\boldsymbol{X}_{0,j}^{0}$ with uniform weights. Otherwise, employ the statistical representation obtained before the average $\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-2,j}^{n-1})$ and modify each particle weight according to (5.5).
end

3. Compute the Monte-Carlo propagation and the average.
4. Compute the correction term:

$$\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})/\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n}). \tag{5.3}$$

    end
    for $n = 0 : N-1$
5. Use the coarse solver to compute the prediction term $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k,j}^{n})$.
6. Compute the hybrid solution at the time observable $(n+1)\,\Delta T$

$$\boldsymbol{U}_{k,j}^{n+1} := \mathcal{G}_{\Delta T}(\boldsymbol{U}_{k,j}^{n}) \times \left(\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})/\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})\right). \tag{5.4}$$

7. Update the statistical weights:

$$\left[\omega_{k,j}^{n+1}\right]|_{i\in K} := \left[\tilde{\omega}_{k-1,j}^{n}\right]|_{i\in K} \times \left(\boldsymbol{U}_{k,j}^{n+1}/\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})\right)|_K \tag{5.5}$$

where $\left[\tilde{\omega}_{k-1,j}^{n}\right]|_{i\in K}$ is the weight of the particle i belonging to the element K at time step n and parareal step $k-1$ after the use of the kernel transport (4.4) (before averaging). See also Remark 5.3.

    end
  end
  for $n = 0 : N-1$ (loop for parareal update)
8. Compute $\boldsymbol{U}_k^{n+1} := \dfrac{1}{p}\displaystyle\sum_{j=1}^{p} \boldsymbol{U}_{k,j}^{n+1}$. Check the stopping criterion: $\sup\left|\boldsymbol{U}_k^{n+1} - \boldsymbol{U}_{k-1}^{n+1}\right| \leq \frac{C}{\sqrt{M}}$ with $C > 0$ a fixed parameter. If satisfied, set $\boldsymbol{U}^{n+1} = \boldsymbol{U}_k^{n+1}$. If not, set $k := k+1$ and go back to the loop indexed by k.
  end
end

Remark 5.1. *The first stage of Algorithm 1 provides a coarse approximation of the solution at each observation time $T_n = n\Delta T$. Furthermore, the initial coarse deterministic approximations*

provided by (5.2) are equal in all batch $j \in [1,p]$. However, the corresponding statistical versions $\boldsymbol{X}_{k=0,j}^{n+1}$ are different in each batch $j \in [1,p]$. For the sake of clarity, we have used the notation $\boldsymbol{U}_{k=0,j}^{n+1}$ to indicate the presence of the sampling procedure on a given batch when where used the fine propagator. Next, observe that $\boldsymbol{U}_{k,j}^{1}$ is constant $\forall k \geq 1$ since

$$\boldsymbol{U}_{k+1,j}^{1} = \mathcal{G}_{\Delta T}(\boldsymbol{U}_{k+1,j}^{0}) \times \frac{\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k,j}^{0})}{\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k,j}^{0})} = \mathcal{F}_{\Delta T}(\boldsymbol{U}^{0}).$$

Remark 5.2. *Also, observe that our hybrid numerical solution computed from (5.4) is different from the one presented in (1.2). Indeed, the solution of the diffusion model (2.1) should be nonnegative and thus the formula (5.4) is more convenient. Note also that in the definition of the correction factor (5.3), $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$ should be nonzero in all mesh elements. To tackle this difficulty, one can add to $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$ the quantity $\varepsilon \approx C/M$.*

Remark 5.3. *On a given batch $j \in [1,p]$, when the hybrid solution $\boldsymbol{U}_{k,j}^{n+1}$ is computed we need its statistical version $\boldsymbol{X}_{k,j}^{n+1}$ to compute the subsequent parareal solution $\boldsymbol{U}_{k+1,j}^{n+2}$. Instead of sampling $\boldsymbol{U}_{k,j}^{n+1}$ and thus introducing a bias in the Monte-Carlo solver, we employ the statistical version of the deterministic object $\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$ (that we denote for instance by $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$) which is available but we modify each of its particles weight as provided by (5.5). With this construction, the normalized histogram (i.e. the number of particles that fall in each element divided by the total number of particles) of the population $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$ weighted by $\omega_{k,j}^{n+1}$ provided by (5.5) gives the deterministic representation $\boldsymbol{U}_{k,j}^{n+1}$. The formula (5.5) could also be seen as the evolution of the statistical weights.*

Remark 5.4. *In terms of CPU cost, the Monte-Carlo algorithm presented in Section 4.4 employs in its best performance p processors. The hybrid parareal strategy presented here computes at a fixed iteration k the propagations $\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k-1,j}^{n})$ in parallel and the prediction term which has a low cost. This strategy provides a total number of processors equal to $p \times (N-1)$ which allows a computational speed-up of the standard Monte-Carlo resolution.*

6 Numerical benchmark experiments

This section illustrates numerically the behavior of the hybrid parareal scheme proposed above. We consider a one dimensional domain Ω consisting in a segment of length $L = 5\ m$. We then solve the model

$$\begin{aligned} \partial_t u - \mathcal{D}\partial_{xx}^2 u = 0 \quad &\text{in} \quad \Omega \times]0,T[\,, \\ u = 0 \quad &\text{on} \quad \partial\Omega \times]0,T[\,, \\ u(x,0) = u_0(x) \quad &\text{in} \quad \Omega. \end{aligned} \tag{6.1}$$

We test our hybrid strategy with a coarse $\mathbb{P}_1$ finite element propagator where the time steps are supposed constant $\Delta t_n = \Delta t = \Delta T := 2\ s\ \forall 1 \leq n \leq N_t$, and a fine diffusion Monte-Carlo propagator having a constant time step $\delta t_n = \delta t := 2 \times 10^{-4}\ s$. The definition of N_t is provided in the two following test cases. We consider $M' := 10^4$ particles and we use $p := 10^3$ independent replicas so that the total number of simulated particles is $M := M' \times p = 10^7$. Here, Algorithm 1 is thus repeated $p = 10^3$ times independently. Furthermore, the parameter C in the stopping criterion **8.** of Algorithm 1 is chosen as $C = 2$. For the sake of clarity, the exact solution of (2.1) is denoted by u, the solution obtained by a full Monte-Carlo algorithm is denoted by u^{MC}, the solution provided by the coarse finite element solver is denoted by

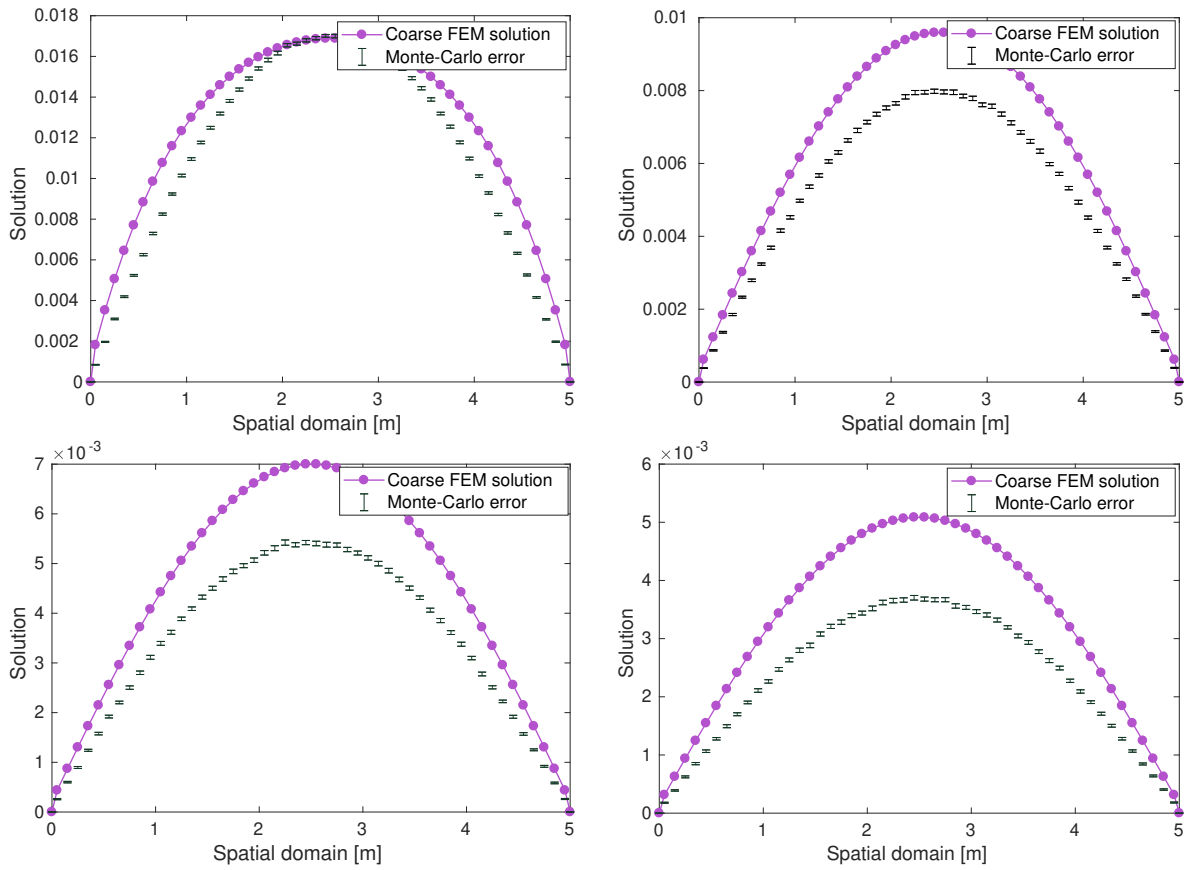

Figure 1: Coarse finite element resolution and statistical Monte-Carlo error at time step $n = 1$ (top left), $n = 3$ (top right), $n = 4$ (bottom left), and $n = 5$ (bottom right).

u^{FEM}, and the solution given by our hybrid strategy is denoted by u^{HYB}. The exact solution $u \in \mathcal{C}^0([0,T] \times \Omega)$ for this benchmark problem can be obtained explicitly and reads

$$u(x,t) := \frac{2}{L} \int_0^L u_0(\xi) \sum_{n=1}^{\infty} \sin\left(\frac{n\pi x}{L}\right) \sin\left(\frac{n\pi \xi}{L}\right) \exp\left(-\frac{\mathcal{D} n^2 \pi^2 t}{L^2}\right) \mathrm{d}\xi. \tag{6.2}$$

We compare the performances of the full Monte-Carlo algorithm and the one of the hybrid parareal algorithm. We suppose that p processors are available to speed-up the Monte-Carlo algorithm (one processor for each batch). It means that on one dedicated processor, we study the displacement of $M' = 10^4$ particles and we compute the Monte-Carlo expectation (4.7). Besides, when the hybrid parareal solver is used, $p \times (N_t - 1)$ processors are available (p for the number of batches, $N_t - 1$ for the number of time steps where the parallelization occurs in Algorithm 1). To sample the initial distribution u_0, we employ the inverse of the cumulative procedure as described in Section 4.1.1.

6.1 A first test case

The final simulation time is set to $T = 10\ s$ and then the number of time steps $N_t = 5$. The diffusion coefficient $\mathcal{D}$ is equal to $0.5\, m^2.s^{-1}$. The initial condition u_0 is taken as $u_0(x) := \dfrac{1}{L}$ such that $\displaystyle\int_0^L u_0(x)\, \mathrm{d}x = 1$. Each particle i of the statistical representation of u_0 has a statistical weight $\omega_i = 1$.

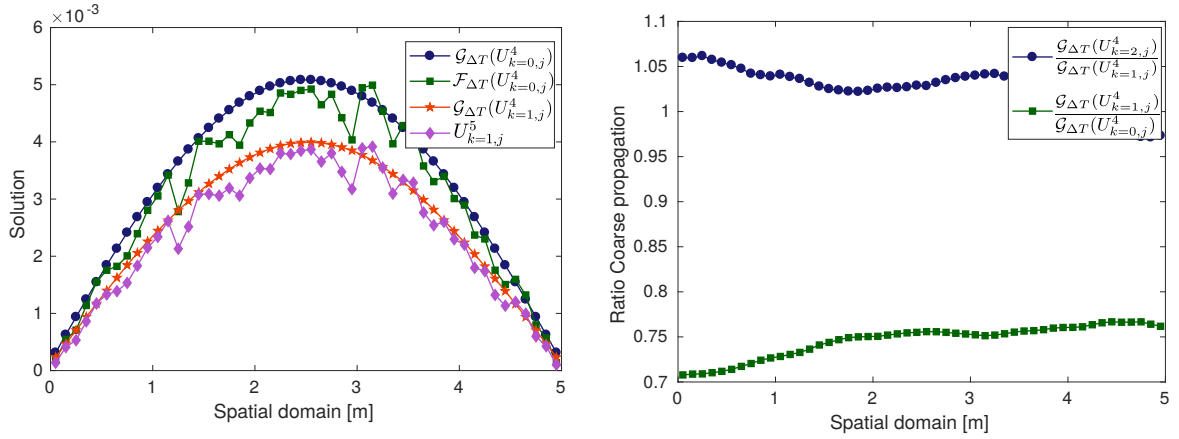

Figure 2: Construction of the hybrid solution $\boldsymbol{U}^5_{k=1,j}$ (left), and coarse propagators (right) at time iteration $n = 5$ for the batch $j = 1$.

In Figure 1, we display the shape of the coarse finite element solution u^{FEM} for four selected time values: $t = 2$ s, $t = 6$ s, $t = 8$ s and $t = 10$ s. We observe that the coarse solution u^{FEM} is not compatible with the Monte-Carlo uncertainty defined in (4.9). It thus proves that the coarse propagator $\mathcal{G}_{\Delta T}$ is a poor predictor. Here, the Monte-Carlo solution u^{MC} is considered as the reference method. The analytical solution u is obviously contained in the Monte-Carlo uncertainty given by (4.9) which is not the case for u^{FEM}.

In Figure 2, we have represented for one selected batch ($j = 1$) the behavior of the coarse propagator $\mathcal{G}_{\Delta T}$ and the behavior of the fine propagator $\mathcal{F}_{\Delta T}$ at the time step $n = 5$. Recall that

$$\boldsymbol{U}^5_{k=1,j} := \underbrace{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})}_{\text{prediction}} \times \underbrace{\frac{\mathcal{F}_{\Delta T}(\boldsymbol{U}^4_{k=0,j})}{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=0,j})}}_{\text{correction}} \quad \text{and} \quad \boldsymbol{U}^5_{k=2,j} := \underbrace{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=2,j})}_{\text{prediction}} \times \underbrace{\frac{\mathcal{F}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})}{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})}}_{\text{correction}}.$$

In the left Figure 2, the red curve displaying $\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})$ represents the prediction terms, as computed by the coarse propagator based on the FEM method. It is computed fastly. Next, the ratio of the green curve $\mathcal{F}_{\Delta T}(\boldsymbol{U}^4_{k=0,j})$ and the blue curve $\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=0,j})$ is the correction term. It lifts the coarse approximation $\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})$ to obtain the numerical solution $\boldsymbol{U}^5_{k=1,j}$. Also observe that the shape of the hybrid solution follows the shape of the result of the fine propagator $\mathcal{F}_{\Delta T}(\boldsymbol{U}^4_{k=0,j})$ up to a small constant shift. This latter is given by the ratio provided in the second graph of Figure 2 (green curve). Furthermore, we observe that the ratio $\dfrac{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=2,j})}{\mathcal{G}_{\Delta T}(\boldsymbol{U}^4_{k=1,j})}$ tends to 1, which means that, from $k = 2$, the approximation is achieved by the accuracy of the fine propagator $\mathcal{F}_{\Delta T}$.

In Figure 3, we have displayed for the time step $n = 5$, the shape of the hybrid solution when $k = 0$, $k = 1$, $k = 2$, and the shape of the exact solution u given by (6.2). At $k = 0$, which corresponds to the initial step of the hybrid Algorithm 1, we observe that the solution $\boldsymbol{U}^5_{k=0}$ is far from the exact solution u and lies outside the Monte-Carlo uncertainty. Such observation is coherent with the fact that the initial stage corresponds to a poor prediction of the numerical solution. Next, at $k = 1$, a correction step is performed yielding an accurate numerical solution as we can see in the second graph. Note that here, $\boldsymbol{U}^5_{k=1} := \dfrac{1}{p}\displaystyle\sum_{j=1}^{p} \boldsymbol{U}^5_{k=1,j}$. Furthermore, observe that $\left\| \left(u - u^{\mathrm{HYB}}\right)(x,T) \right\|_{L^\infty(\Omega)} \approx 3 \times 10^{-4}$ which is in agreement with the Monte-Carlo convergence speed. Next, we observe that the solutions $\boldsymbol{U}^5_{k=1}$ and $\boldsymbol{U}^5_{k=2}$ coincide

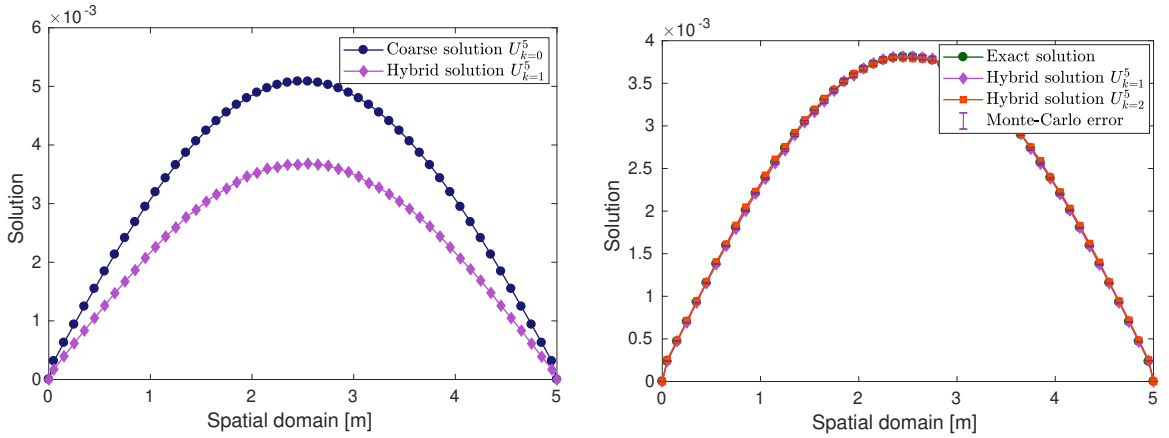

Figure 3: Shape of the hybrid solution at $k = 0$ and $k = 1$ (left) and shape of the exact solution and hybrid solution at $k = 1$ and $k = 2$ (right).

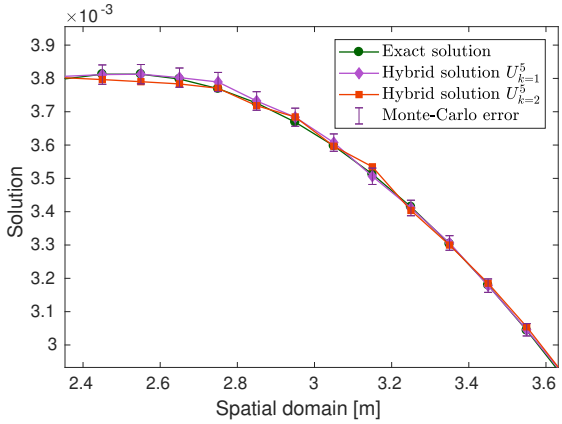

Figure 4: Hybrid solution at time iteration $n = 5$ and at parareal iteration $k = 1$, $k = 2$, and Monte-Carlo error bar.

on the spatial domain Ω in the sense that the stopping criterion **8.** of Algorithm 1 is satisfied. Thus our hybrid strategy requires only 1 parareal iteration to converge for the current time step. Note that we observed numerically the same phenomenon for all time intervals $t_n = n\Delta t$.

Figure 4 is a crucial complement to Figure 3, as it shows that the hybrid solutions $\boldsymbol{U}^5_{k=1}$ and $\boldsymbol{U}^5_{k=2}$ lies within the Monte-Carlo uncertainty. We zoomed on the the right cells of Figure 3.

In Figure 5, we represented for the full Monte-Carlo resolution and the hybrid parareal resolution the error in the $L^\infty(0,T;L^\infty(\Omega))$ norm as a function of the number of particles. Recall that

$$\left\|u - u^{\mathrm{MC}}\right\|_{L^\infty(0,T;L^\infty(\Omega))} = \max_{t\in[0,T]} \max_{x\in\Omega} \left|\left(u - u^{\mathrm{MC}}\right)(x,t)\right| \tag{6.3}$$

and

$$\left\|u - u^{\mathrm{HYB}}\right\|_{L^\infty(0,T;L^\infty(\Omega))} = \max_{t\in[0,T]} \max_{x\in\Omega} \left|\left(u - u^{\mathrm{HYB}}\right)(x,T)\right|. \tag{6.4}$$

We observe that the curves of the Monte-Carlo error and of the hybrid error behave like $\dfrac{1}{\sqrt{M}}$ which is in agreement with the Monte-Carlo convergence rate.

The details of the efficiency of our Hybrid strategy can finally be appreciated in Figure 6 and Table 1. We compared in Figure 6 the global CPU time of the simulation for two different strategies: when the parallelization per batch is used (left Figure 6), and when no parallelization

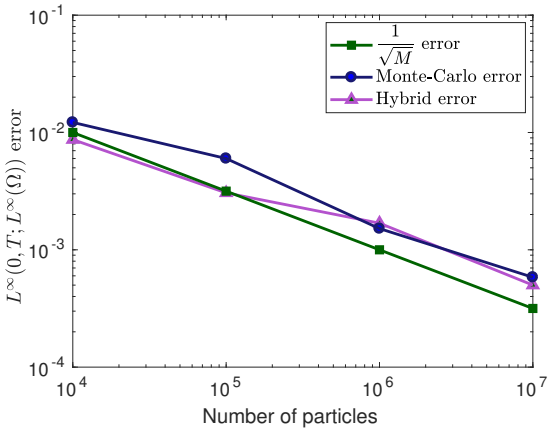

Figure 5: Error in the $L^\infty(0,T;L^\infty(\Omega))$ norm between u and u^{MC}, and error in the $L^\infty(0,T;L^\infty(\Omega))$ between u and u^{HYB} as a function of the number of particles.

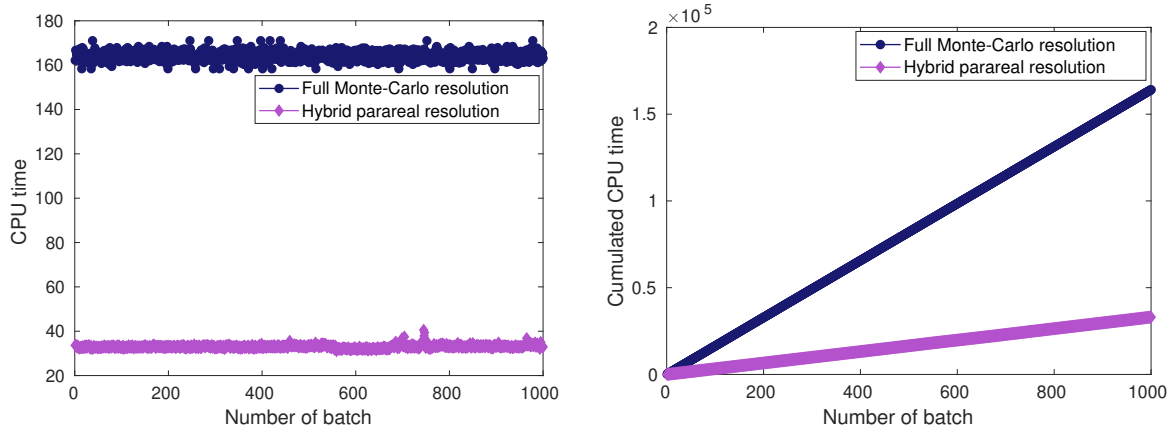

Figure 6: CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

Number of particles	Number of batch	Parallelized Monte-Carlo	Hybrid parallelized Monte-Carlo $k=1$	Hybrid parallelized Monte-Carlo $k=2$	Gain factor $k=1$	Gain factor $k=2$
10^5	10^2	1653.4 s	335.76 s	534.16 s	4.92	3.04
10^4	10^3	164.09 s	33.05 s	7.97 s	4.96	3.09
10^3	10^4	16.86 s	3.39 s	0.83 s	4.97	3.07
10^2	10^5	1.78 s	0.35 s	0.11 s	5.08	3.02

Table 1: Computational cost of the full Monte-Carlo resolution and of the hybrid resolution

per batch is used (right Figure 6). More precisely, on the left Figure, one processor is assigned to each batch. For the full Monte-Carlo algorithm, 10^3 processors are therefore available. For the hybrid parareal strategy $p \times (N_t - 1)$ processors are available for the computation of the numerical solution. When no parallelization per batch occurs (right Figure) the standard Monte-Carlo algorithm computes the expectation sequentially (only one processor is available). However, the hybrid strategy employs $N_t - 1$ processors for the correction step. In both situations, the hybrid parareal strategy employs more processors. We observe from the left Figure that the full Monte-Carlo expectation is computed in each batch after roughly 164 seconds. The average CPU time for the standard Monte-Carlo simulation (with parallelization) is equal to about 164 seconds (see Table 1). It is roughly 1000 times less than the sequential Monte-Carlo resolution (right Figure). Concerning our hybrid parareal strategy, the solution for each batch is computed only after roughly 33 seconds (see the left Figure) which is much faster than the parallelized Monte-Carlo resolution. Besides, the total cumulated CPU time when no parallelization per batches occurs is equal to 3.3×10^4 seconds which is also less expensive than the sequential Monte-Carlo algorithm. Finally, when parallelized Monte-Carlo is considered, our hybrid parareal strategy yields a gain factor in the overall CPU time of around 5. For the sake of completness, we also test in Table 1 the influence of the number of particles/batches on the CPU time. We see that the hybrid resolution is always much faster than the standard full Monte-Carlo resolution with a gain factor roughly equal to 5 when 1 parareal iteration is performed and equal to 3 when 2 parareal iterations are performed. The ideal scaling is equal to N_t/k.

6.2 A second test case

For this second example, the final simulation time is $T = 14\ s$. The diffusion coefficient is set to $\mathcal{D} = 0.5$ for the fine propagator and $\mathcal{D} = 0.48$ for the coarse propagator. We consider $M' = 10^5$ particles and we simulate $p = 10^2$ independant replicas (batches) so that the total number of particles is $M = 10^7$. The coarse propagator is a $\mathbb{P}_1$ finite element solver with constant time step $\Delta t := 2$ s and the fine propagator is the Monte-Carlo solver with constant time step $\delta t = 2 \times 10^{-4}$ s. Furthermore, the coarse time step Δt is chosen equal to the observable window ΔT. The initial condition u_0 is chosen as

$$u_0(x) = \frac{1}{L}\left(1 + \cos\left(\frac{\pi x}{L}\right)\right).$$

Here, u_0 is a PDF, and to each particle i of the statistical representation of u_0 is assigned the statistical weight $\omega_i = 1$. In Figure 7 we represent the shape of the initial guess and its statistical version. Note that here, the repartition of the particles in each intervals follows the methodology of the inversion of the cumulative function (see Section 4.1.1.) However, before employing the fine solver in the parareal stages we use the table lookup method to find numerically the intervals where each particle live.

Figure 8 displays at the final time step $n = 7$, the shape of the analytical solution and the parareal sequence $\boldsymbol{U}_k^7$ for $k = 0$, $k = 1$, and $k = 2$. Note that the solution $\boldsymbol{U}_{k=0}^7$ corresponds to the coarse solution obtained using the coarse solver while $\boldsymbol{U}_{k=1}^7$ and $\boldsymbol{U}_{k=2}^7$ correspond to corrected solutions as the fine solver is applied. We observe from the left figure that the coarse solution (black curve) is far from the analytical solution. Next, we see that a first parareal step will bring the solution (purple curve) closer to the analytical solution (green curve) but is not sufficiently close to stop the parareal iterations. Indeed, in the right figure we perform a zoom on several cells and we observe that a second parareal iteration (red curve) is required to converge to the analytical solution and to have the analytical solution present in the Monte-Carlo error bar. In Figure 9 we represent for the two strategies the required CPU time. As for the first test case, we observe that the hybrid resolution with this time $k = 2$ parareal

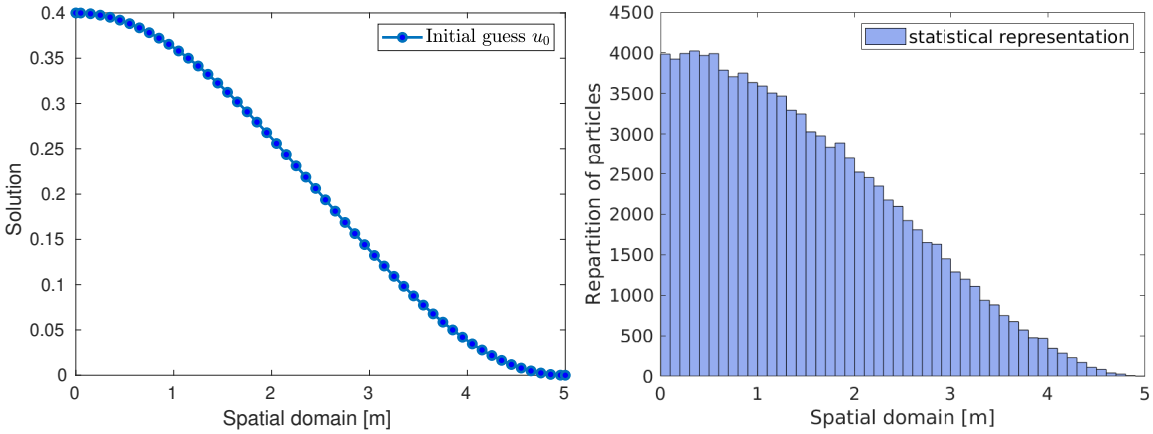

Figure 7: Solution at initial time $t = 0$ (left) and histogram of the statistical population at initial time $t = 0$ (right) for the second test case.

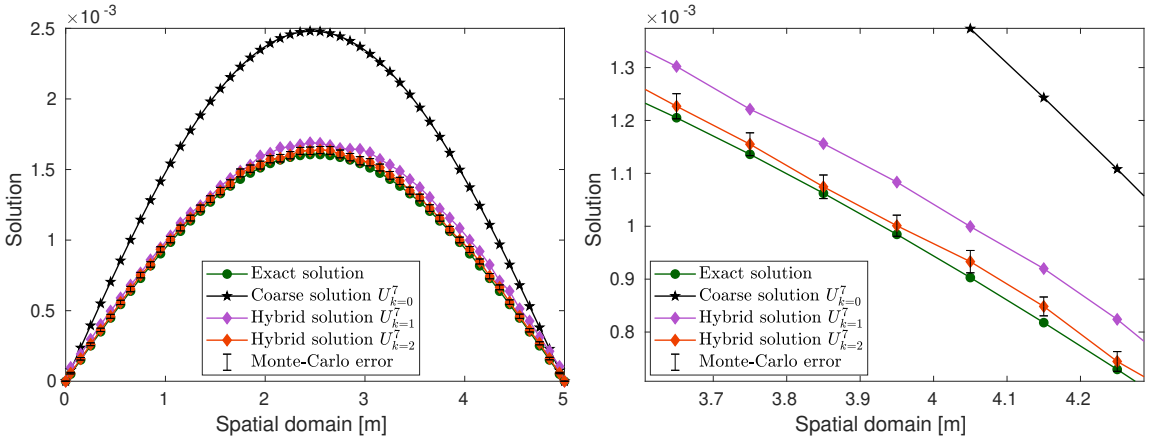

Figure 8: Shape of the hybrid solution at $n = 7$ and $k = 0$, $k = 1$, $k = 2$, and shape of the exact solution.

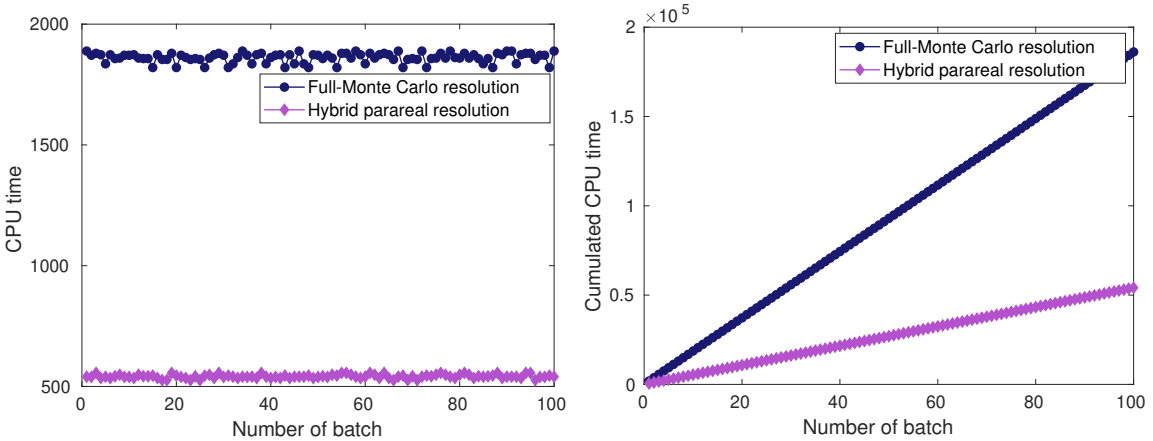

Figure 9: CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

iterations is less expensive than the classical Monte-Carlo resolution. The hybrid resolution computes the expectation in each batch after roughly 540 seconds whereas the classical Monte-Carlo expectation computes the expectation in each batch after roughly 1810 seconds. Then, when parallelized Monte-Carlo is considered, our hybrid stratedy yields a gain factor in the overall CPU time at around 3.44. The figure on the right is a complement and shows that if no parallelization per batch occurs the hybrid strategy is still better and reduces significantly the computational cost.

7 Conclusions

In this work, we have designed a parareal hybrid version of a Monte-Carlo algorithm. We used the parareal-in-time procedure to speed-up a Monte-Carlo algorithm. We showed numerically that our strategy requires few parareal iterations to reach convergence. Besides, our hybrid resolution is very fast in terms of CPU time compared to a standard full Monte-Carlo resolution. Implementation of Boltzmann transport kernel and extension to transport of Neutron in nuclear reactors are under investigation.

Acknowledgments: This work was funded by the ANR project "Ciné-Para" (ANR-15-CE23-0019). We thank Tony Lelièvre (CERMICS and École des Ponts ParisTech) for many useful comments.

References

[1] J. A. Acebrón, M. P. Busico, P. Lanucara, and R. Spigler, *Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and quasi-Monte Carlo methods*, SIAM J. Sci. Comput., 27 (2005), pp. 440–457, https://doi.org/10.1137/030600692.

[2] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM J. Numer. Anal., 19 (1982), pp. 742–760, https://doi.org/10.1137/0719052.

[3] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E, 66 (2002), p. 057701, https://doi.org/10.1103/PhysRevE.66.057701.

[4] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, in Domain decomposition methods in science and engineering, vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 425–432, https://doi.org/10.1007/3-540-26825-1_43.

[5] A.-M. Baudron, J.-J. Lautard, Y. Maday, and O. Mula, *The parareal in time algorithm applied to the kinetic neutron diffusion equation*, in Domain decomposition methods in science and engineering XXI, vol. 98 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2014, pp. 437–445.

[6] A.-M. Baudron, J.-J. Lautard, Y. Maday, M. K. Riahi, and J. Salomon, *Parareal in time 3D numerical solver for the LWR benchmark neutron diffusion transient model*, J. Comput. Phys., 279 (2014), pp. 67–79, https://doi.org/10.1016/j.jcp.2014.08.037.

[7] C. Bernardi, Y. Maday, and F. Rapetti, *Discrétisations variationnelles de problèmes aux limites elliptiques*, vol. 45 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer-Verlag, Berlin, 2004.

[8] G. E. P. Box and M. E. Muller, *A note on the generation of random normal deviates*, Ann. Math. Statist., 29 (1958), pp. 610–611, https://doi.org/10.1214/aoms/1177706645.

[9] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 15 of Texts in Applied Mathematics, Springer-Verlag, New York, 1994, https://doi.org/10.1007/978-1-4757-4338-8.

[10] H. Brezis, *Functional analysis, Sobolev spaces and partial differential equations*, Universitext, Springer, New York, 2011.

[11] W. L. Briggs, *A multigrid tutorial*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1987.

[12] R. Dautray and J.-L. Lions, *Analyse mathématique et calcul numérique pour les sciences et les techniques. Vol. 8*, INSTN: Collection Enseignement. [INSTN: Teaching Collection], Masson, Paris, 1988. Évolution: semi-groupe, variationnel. [Evolution: semigroups, variational methods], Reprint of the 1985 edition.

[13] J. M. DeLaurentis and L. A. Romero, *A Monte Carlo method for Poisson's equation*, J. Comput. Phys., 90 (1990), pp. 123–140, https://doi.org/10.1016/0021-9991(90)90199-B.

[14] B. Després, *Numerical methods for Eulerian and Lagrangian conservation laws*, Frontiers in Mathematics, Birkhäuser/Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-50355-4.

[15] D. A. Di Pietro and A. Ern, *Mathematical aspects of discontinuous Galerkin methods*, vol. 69 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer, Heidelberg, 2012, https://doi.org/10.1007/978-3-642-22980-0.

[16] V. Dolejší and M. Feistauer, *Discontinuous Galerkin method*, vol. 48 of Springer Series in Computational Mathematics, Springer, Cham, 2015, https://doi.org/10.1007/978-3-319-19267-3. Analysis and applications to compressible flow.

[17] R. Eymard, T. Gallouët, and R. Herbin, *Finite volume methods*, in Handbook of numerical analysis, Vol. VII, Handb. Numer. Anal., VII, North-Holland, Amsterdam, 2000, pp. 713–1020.

[18] G. E. Forsythe and R. A. Leibler, *Matrix inversion by a Monte Carlo method*, Math. Tables Aids Comput., 4 (1950), pp. 127–129.

[19] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578, https://doi.org/10.1137/05064607X.

[20] E. Godlewski and P.-A. Raviart, *Numerical approximation of hyperbolic systems of conservation laws*, vol. 118 of Applied Mathematical Sciences, Springer-Verlag, New York, 1996, https://doi.org/10.1007/978-1-4612-0713-9.

[21] L. Infeld, *On the theory of Brownian motion*, University of Toronto Studies, Applied Mathematics Series, no. 4, University of Toronto Press, Toronto, Ont., 1940.

[22] J. Jacod and P. Protter, *Probability essentials*, Universitext, Springer-Verlag, Berlin, second ed., 2003, https://doi.org/10.1007/978-3-642-55682-1.

[23] M. H. Kalos and P. A. Whitlock, *Monte Carlo methods. Vol. I*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1986, https://doi.org/10.1002/9783527617395. Basics.

[24] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, vol. 16 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. With separately available software.

[25] Y. I. Khlopkov, Z. Y. M. Myint, and A. Y. Khlopkov, *Monte Carlo method and its parallel computing technique in molecular gas dynamics*, International Journal of Educational Research and Information Science, 2 (2015), p. 1.

[26] R. Kornhuber and E. Youett, *Adaptive multilevel Monte Carlo methods for stochastic variational inequalities*, SIAM J. Numer. Anal., 56 (2018), pp. 1987–2007, https://doi.org/10.1137/16M1104986.

[27] F. Legoll, T. Lelièvre, and G. Samaey, *A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations*, SIAM J. Sci. Comput., 35 (2013), pp. A1951–A1986, https://doi.org/10.1137/120872681.

[28] J.-L. Lions, *Quelques méthodes de résolution des problèmes aux limites non linéaires*, Dunod; Gauthier-Villars, Paris, 1969.

[29] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668, https://doi.org/10.1016/S0764-4442(00)01793-6.

[30] I. Lux and L. Koblinger, *Monte Carlo particle transport methods: neutron and photon calculations*, CRC Press, 1991.

[31] M. Mascagni and N. A. Simonov, *Monte Carlo methods for calculating some physical properties of large molecules*, SIAM J. Sci. Comput., 26 (2004), pp. 339–357, https://doi.org/10.1137/S1064827503422221.

[32] W. J. Morokoff and R. E. Caflisch, *A quasi-Monte Carlo approach to particle simulation of the heat equation*, SIAM J. Numer. Anal., 30 (1993), pp. 1558–1573, https://doi.org/10.1137/0730081.

[33] E. Nelson, *Dynamical theories of Brownian motion*, Princeton University Press, Princeton, N.J., 1967.

[34] G. Ökten, *Solving linear equations by Monte Carlo simulation*, SIAM J. Sci. Comput., 27 (2005), pp. 511–531, https://doi.org/10.1137/04060500X.

[35] G. Pagès, O. Pironneau, and G. Sall, *The parareal algorithm for american options*, SIAM J. Financial Math., 9 (2018), pp. 966–993, https://doi.org/10.1137/17M1138832.

[36] S. C. Port, *Theoretical probability for applications*, Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York, 1994. A Wiley-Interscience Publication.

[37] A. Quarteroni and A. Valli, *Numerical approximation of partial differential equations*, vol. 23 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1994.

[38] B. Rivière, *Discontinuous Galerkin methods for solving elliptic and parabolic equations*, vol. 35 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008, https://doi.org/10.1137/1.9780898717440. Theory and implementation.

[39] B. Rivière, M. F. Wheeler, and V. Girault, *A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems*, SIAM J. Numer. Anal., 39 (2001), pp. 902–931, https://doi.org/10.1137/S003614290037174X.

[40] C. P. Robert and G. Casella, *Introducing Monte Carlo methods with R*, Use R!, Springer, New York, 2010, https://doi.org/10.1007/978-1-4419-1576-4.

[41] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003, https://doi.org/10.1137/1.9780898718003.

[42] M. Ullrich, *A Monte Carlo method for integration of multivariate smooth functions*, SIAM J. Numer. Anal., 55 (2017), pp. 1188–1200, https://doi.org/10.1137/16M1075557.

[43] S. Umarov, M. Hahn, and K. Kobayashi, *Beyond the triangle: Brownian motion, Ito calculus, and Fokker-Planck equation—fractional generalizations*, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2018, https://doi.org/10.1142/10734.

[44] W. R. Wasow, *A note on the inversion of matrices by random walks*, Math. Tables Aids Comput., 6 (1952), pp. 78–81.

[45] H. Zaidi, C. Labbé, and C. Morel, *Implementation of an environment for Monte Carlo simulation of fully 3-d positron tomography on a high-performance parallel platform*, Parallel Computing, 24 (1998), pp. 1523 – 1536, https://doi.org/https://doi.org/10.1016/S0167-8191(98)00069-6.