

# A hybrid parareal Monte Carlo algorithm for parabolic problems \*

Jad Dabaghi, Yvon Maday, Andrea Zoia

## ► To cite this version:

Jad Dabaghi, Yvon Maday, Andrea Zoia. A hybrid parareal Monte Carlo algorithm for parabolic problems \*. 2021. hal-03143554v2

**HAL Id: hal-03143554**

**<https://hal.archives-ouvertes.fr/hal-03143554v2>**

Preprint submitted on 11 Mar 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A hybrid parareal Monte Carlo algorithm for parabolic problems \*

Jad Dabaghi <sup>†‡</sup>      Yvon Maday <sup>†</sup>      Andrea Zoia <sup>‡</sup>

March 11, 2021

## Abstract

In this work, we propose a novel hybrid Monte Carlo/deterministic “parareal-in-time” approach dedicated to further speed up to solution time of unsteady Monte Carlo simulations over massively parallel computing environments.

This parareal approach iterates on two different solvers: a low-cost “coarse” solver based on a very cheap deterministic Galerkin scheme and a “fine” solver based on a precise Monte Carlo resolution.

In a set of benchmark numerical experiments based on a toy model concerning the time-dependent diffusion equation, we compare our hybrid parareal strategy with a standard full Monte Carlo solution. 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

Several physical phenomena are described by partial differential equations (PDEs) whose analytical treatment is often out of reach. In this context, numerical simulations appear to be in general the only viable approach to approximating a solution. Among the wide range of numerical methods for solving PDEs, we mention for instance the finite element methods [1, 2, 3], the finite volume methods [4, 5, 6], or the discontinuous Galerkin methods [7, 8, 9]. These are deterministic methods that have been classically used in industrial contexts for years. For a certain class of problems, e.g. in large dimension, probabilistic approaches based on Monte Carlo methods have been advocated for more than 40 years now [10, 11, 12] and have become the approach of choice in several domains: they are preferred to deterministic methods in specific applications like radiation transport [12] or molecular dynamics [13]. For instance, in the simulation of neutron transport, a fine deterministic discretization of the phase space variables would involve a tremendous number of unknowns and correspondingly an unaffordable computational cost and memory burden. Monte Carlo methods are weakly dependent on the dimensionality of the system for high-dimensional problems, are natively implemented over massively parallel computing environments [14, 15].

The Monte Carlo algorithm consists in approximating the solution (denoted for instance by  $u$ ) by sampling a finite number  $M > 1$  of random walks whose estimated density at point  $\mathbf{x}$  and time  $t$  converges in the limit of large  $M$  to  $u(\mathbf{x}, t)$ . It is well known that the Monte Carlo method displays a  $1/\sqrt{M}$  convergence as a result of the Central Limit Theorem [10]. Obtaining a numerical solution sufficiently close to the exact one requires to consider a very large number of random walks, which demands a high computational cost.

Up to now, Monte Carlo methods have been applied almost exclusively to the solution of stationary (i.e. time-independent) problems, mainly due to their high computational cost [10, 12]. An application of utmost importance is the simulation of time-dependent neutron transport in nuclear reactor cores, in order

---

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

<sup>†</sup>Sorbonne Université, CNRS, Université de Paris, Laboratoire Jacques-Louis Lions (LJLL), F-75005 Paris, France & Institut Universitaire de France

<sup>‡</sup>Université Paris-Saclay, CEA, Service d’études des réacteurs et de mathématiques appliquées, 91191, Gif-sur-Yvette, France

to take into account transient and/or accidental regimes for safety issues : the main scientific challenge is to take into account the very different time scales of prompt and delayed neutrons in long transients ("kinetic" Monte Carlo). Kinetic methods have been the subject of major research efforts in recent years [16]. Thanks to the increasing available computer power, in particular based on incredibly large number of cores that are in excess with respect to the number  $M$  of replicas, non stationary problems where the full knowledge of what occurs during the evolution by Monte Carlo simulation can be tackled. This situation is similar to what occurs for deterministic approaches when domain decomposition methods are used and where the number of core is much larger than the number of subdomains. The idea is to decompose the time direction into slices where each of them can be handled in parallel. In this respect, several strategies have been proposed in order to efficiently cope the parallelisation in time [17, 18, 19, 20]. Among these numerical methods, the parareal algorithm [20] is based on the idea of solving the time evolution of dynamical systems in a parallel fashion. It involves two propagators  $\mathcal{F}$  and  $\mathcal{G}$  that integrate a given system of partial differential equations at the continuous level. 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 ( $\delta t$  for  $\mathcal{F}$  being much smaller than  $\Delta t$  for  $\mathcal{G}$ ), but the model that  $\mathcal{G}$  approximates may also be a simplified one.

Let  $T_0 = 0 < T_1 < \dots < T_N = T$  be a sequence of times. For the sake of simplicity, we choose here  $T_n = n\Delta T$  for some appropriate time interval  $\Delta T$ . The parareal algorithm constructs a sequence  $\mathbf{u}_k := (\mathbf{u}_k^n)_{1 \leq n \leq N}$  such that  $\forall k \geq 0$ ,  $\mathbf{u}_k^n$  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  $\Delta T$  (we denote by  $\mathcal{G}_{\Delta T}$  such a coarse evolution over a time window of size  $\Delta T$ ) :

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

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

$$u_{k+1}^{n+1} := \underbrace{\mathcal{G}_{\Delta T}(u_{k+1}^n)}_{\text{Prediction}} + \underbrace{[\mathcal{F}_{\Delta T}(u_k^n) - \mathcal{G}_{\Delta T}(u_k^n)]}_{\text{Correction}}, \quad (1.1)$$

where  $u_{k+1}^0 := u_0$ . Note that the coarse solver  $\mathcal{G}$  is sequential, whereas the fine solver  $\mathcal{F}$  computes at the end of each step  $k$  the corrections in parallel. When  $k \rightarrow \infty$ , equation (1.1) 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  $\mathcal{F}$ .

In general, the two solvers involved in the parareal procedure are deterministic. For instance a parareal procedure for the Navier–Stokes equation in the context of finite elements and spectral methods has been proposed in [21]. A micro-macro version of the parareal algorithm for singularly perturbed systems of ordinary differential equations (ODEs) has been illustrated in [22], coupling a coarse propagator based on an approximate macroscopic model with fewer degrees of freedom to a fine propagator that accurately simulates the full microscopic dynamics. For other applications, see [23, 24] for kinetic transport problems or [25] for reservoir simulation. For a convergence study of the parareal algorithm we refer to [20, 26, 27, 28]. In particular, a superlinear bound on the convergence on bounded time intervals for the diffusion equation and the advection equation has been demonstrated in [28].

The parareal-in-time strategy has been also applied to Monte Carlo methods, as in [29] where a parallelization of the Least-Square Longstaff–Sartz Monte Carlo algorithm dedicated to the pricing in american options has been discussed. The kernel transport used in that survey is a standard Brownian motion [30], where the coarse and fine propagators are Monte Carlo solvers with different time steps. A parareal in time version of a micro-macro Monte Carlo algorithm where the involved propagators have different time scales has been also proposed in [31], by closely following the ideas of [22].

In this work, we propose a novel hybrid version of the parareal algorithm, with a predictor based on a deterministic solver and a corrector based on a Monte Carlo solver. To the best of our knowledge, this scheme has never been considered before and may serve as a starting point for extensions to diverse applications. In order to illustrate the general features of this strategy, we apply this method to a simple benchmark problem based on the time-dependent diffusion equation, used as a prototype model of evolution 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 illustrate the features of the proposed approach.

## 2 Model problem and setting

Let  $\Omega \subset \mathbb{R}^d$ ,  $d = \{1, 2, 3\}$ , 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  $\Omega$ . Let  $H^1(\Omega)$  be the space of  $L^2$  functions on the domain  $\Omega$  which admit a weak gradient in  $L^2(\Omega)$  and let  $H_0^1(\Omega)$  be its zero-trace subspace. We denote by  $H^{-1}(\Omega)$  the dual space of  $H_0^1(\Omega)$  with the duality pairing  $\langle \cdot, \cdot \rangle_{H^{-1}(\Omega), H_0^1(\Omega)}$ . We consider the time-dependent diffusion equation with homogeneous Dirichlet boundary conditions: Find  $u$  such that

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

Here,  $\mathcal{D} > 0$  is a diffusion tensor supposed constant for the sake of simplicity, and  $u_0 \in L^2(\Omega)$  is the initial condition.

The weak formulation associated to (2.1) reads as follows: find  $u \in L^2(0, T; H_0^1(\Omega))$  such that  $\partial_t u \in L^2(0, T; H^{-1}(\Omega))$  and satisfying for almost all  $t \in ]0, T[$  and for all  $v \in L^2(0, T; H_0^1(\Omega))$

$$\langle \partial_t u, v \rangle_{H^{-1}(\Omega), H_0^1(\Omega)} + \mathcal{D} \int_{\Omega} \nabla u \cdot \nabla v \, dx = 0. \quad (2.2)$$

We mention the fundamental books of Lions [32], Dautray and Lions [33], and Brezis [34], for a complete analysis of parabolic problems.

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

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

For the space discretization, we consider a conforming simplicial mesh  $\mathcal{T}_h$  of the domain  $\Omega$ , i.e.  $\mathcal{T}_h$  is a set of elements  $K$  verifying  $\bigcup_{K \in \mathcal{T}_h} \bar{K} = \bar{\Omega}$ , where the intersection of the closure of two elements of  $\mathcal{T}_h$  is either an empty set, a vertex, or a  $l$ -dimensional face,  $0 \leq l \leq d-1$ . Denote by  $h_K$  the diameter of the generic element  $K \in \mathcal{T}_h$  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  $\mathcal{V}_h^{\text{int}}$  and the boundary nodes  $\mathcal{V}_h^{\text{ext}}$ . Similarly, the nodes of an element  $K \in \mathcal{T}_h$  are collected in the set  $\mathcal{V}_K$  and we denote respectively by  $\mathcal{V}_K^{\text{int}}$  and  $\mathcal{V}_K^{\text{ext}}$  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  $N_h^{\text{int}}$ . We denote by  $\mathcal{E}_h$  the set of mesh faces. Boundary faces are collected in the set  $\mathcal{E}_h^{\text{ext}} = \{\sigma \in \mathcal{E}_h; \sigma \subset \partial\Omega\}$  and internal faces are collected in the set  $\mathcal{E}_h^{\text{int}} = \mathcal{E}_h \setminus \mathcal{E}_h^{\text{ext}}$ . To each face  $\sigma \in \mathcal{E}_h$ , we

associate a unit normal vector  $\mathbf{n}_\sigma$ ; for  $\sigma \in \mathcal{E}_h^{\text{int}}$ ,  $\sigma = K \cap L$ ,  $\mathbf{n}_\sigma$  points from  $K$  towards  $L$  and for  $\sigma \in \mathcal{E}_h^{\text{ext}}$  it coincides with the outward unit normal vector  $\mathbf{n}_\Omega$  of  $\Omega$ . The jump operator  $[\![\cdot]\!]_\sigma$  yielding the difference of the argument from the two mesh elements that share  $\sigma \in \mathcal{E}_h^{\text{int}}$  is defined as  $[\![v]\!]_\sigma := (v|_K - v|_L) \mathbf{n}_\sigma$ , where  $\sigma = \overline{K} \cap \overline{L}$  and the average of  $v$  on  $\sigma \in \mathcal{E}_h^{\text{int}}$  is defined as  $\{\{v\}\}_\sigma := \frac{1}{2}(v|_K + v|_L)$ . When  $\sigma \in \mathcal{E}_h^{\text{ext}}$ ,  $\sigma = \partial K \cap \partial\Omega$ ,  $[\![v]\!]_\sigma := v \mathbf{n}_\Omega$  and  $\{\{v\}\}_\sigma = v$ . We also denote by  $N_{\text{sp}}$  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  $\mathbf{U}^{n-1} \in \mathbb{R}^m$ ,  $m \geq 1$ , the unknown vector  $\mathbf{U}^n \in \mathbb{R}^m$  satisfies

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

where  $\mathcal{G}_{\Delta t_n}(\mathbf{U})$  stands for the coarse discrete evolution over a time range of  $\Delta t_n$  (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  $\Delta t_n$  of the coarse propagator is smaller than the window  $\Delta T$ .

### 3.2 The Lagrange finite element propagator

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

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

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  $\mathbf{x}_l \in \mathcal{V}_h$ . We recall that  $\psi_{h,l}(\mathbf{x}_{l'}) = \delta_{l,l'}$  (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  $v_h \in X_{0h}^p$

$$\frac{1}{\Delta t_n} \int_{\Omega} (u_h^n - u_h^{n-1}) v_h \, dx + \mathcal{D} \int_{\Omega} \nabla u_h^n \cdot \nabla v_h \, dx = 0. \quad (3.2)$$

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

$$\mathbb{A}^n \mathbf{U}^n = \mathbf{F}^{n-1}. \quad (3.3)$$

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

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

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

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

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

$$[\mathbf{F}^{n-1}]_l := \frac{1}{\Delta t_n} \int_{\Omega} u_h^{n-1} \psi_{h,l} \, dx \quad \forall 1 \leq l \leq N_h^{\text{int}}.$$

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:

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

### 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  $\mathbf{x}_K \in K$  (the center of the cell) and for all faces  $\sigma \in \mathcal{E}_h$  there exists a point  $\mathbf{x}_\sigma \in \sigma$  (the center of the face) such that, for all faces  $\sigma \in \mathcal{E}_K$ , the line segment joining  $\mathbf{x}_K$  with  $\mathbf{x}_\sigma$  is orthogonal to  $\sigma$  [5]. For an interior face  $\sigma \in \mathcal{E}_h^{\text{int}}$  shared by two elements  $K$  and  $L$  (denoted in the sequel by  $\bar{\sigma} = \bar{K} \cap \bar{L}$ ) we define the distance between these elements  $d_{KL} := \text{dist}(\mathbf{x}_K, \mathbf{x}_L)$ . For an exterior face  $\sigma \in \mathcal{E}_h^{\text{ext}}$  the distance between the element  $K$  and the face  $\sigma$  is denoted by  $d_{K\sigma} := \text{dist}(\mathbf{x}_K, \mathbf{x}_\sigma)$ . To approximate the space gradient we use

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

Here, the notation  $\mathbf{n}_{K,\sigma}$  stands for the outward unit normal vector to the element  $K$  on the face  $\sigma$ . Obviously, when  $\sigma \in \mathcal{E}_h^{\text{int}}$ ,  $\sigma = \bar{K} \cap \bar{L}$ ,  $\mathbf{n}_{K,\sigma} = -\mathbf{n}_{L,\sigma}$ . Furthermore,  $v_K \approx \frac{1}{|K|} \int_K v \, dx$  is an approximation of  $v$  in the volume  $K$  and  $v_\sigma$  is an approximation of  $v$  on the face  $\sigma$ . Using the cell-centered finite volume method, the unknown of the model is discretized using a single constant value per cell:  $\forall n, 1 \leq n \leq N_t - 1$  we let

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

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

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

Here, the conservative numerical flux  $\mathfrak{F}_{K,\sigma}^n = -\mathfrak{F}_{L,\sigma}^n$ , for an internal face  $\sigma \in \mathcal{E}_h^{\text{int}}$ ,  $\sigma = \bar{K} \cap \bar{L}$ , 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

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

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

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

Here,  $\mathbb{A}^{n-1} \in \mathbb{R}^{N_{\text{sp}}, N_{\text{sp}}}$  has a sparse structure and the right-hand side vector  $\mathbf{F}^{n-1} \in \mathbb{R}^{N_{\text{sp}}}$  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  $N_h^{\text{int}}$  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

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

Next, we introduce the bilinear form

$$a_h(v_h, w_h) := \sum_{K \in \mathcal{T}_h} \int_K \nabla v_h \cdot \nabla w_h \, dx + \mathcal{A}_h(v_h, w_h). \quad (3.6)$$

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  $X_{0h}^p$ :

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

where  $h_\sigma$  is the diameter of the face  $\sigma$ . We mention for instance the SIPG method [35]: For  $\gamma > 0$  (large enough),  $\forall v_h \in X_h^p$  and  $\forall w_h \in X_h^p$

$$\begin{aligned} \mathcal{A}_h(v_h, w_h) := & - \sum_{\sigma \in \mathcal{E}_h} \int_\sigma (\{\{\nabla w_h\}\}_\sigma [[v_h]]_\sigma + \{\{\nabla v_h\}\}_\sigma [[w_h]]_\sigma) \, ds \\ & + \sum_{\sigma \in \mathcal{E}_h} \frac{\gamma}{h_\sigma} \int_\sigma [[w_h]]_\sigma [[v_h]]_\sigma \, ds, \end{aligned} \quad (3.8)$$

or the NIPG method [36]: For  $\gamma > 0$ ,  $\forall v_h \in X_h^p$  and  $\forall w_h \in X_h^p$

$$\begin{aligned} \mathcal{A}_h(v_h, w_h) := & - \sum_{\sigma \in \mathcal{E}_h} \int_\sigma (\{\{\nabla w_h\}\}_\sigma [[v_h]]_\sigma - \{\{\nabla v_h\}\}_\sigma [[w_h]]_\sigma) \, ds \\ & + \sum_{\sigma \in \mathcal{E}_h} \frac{\gamma}{h_\sigma} \int_\sigma [[w_h]]_\sigma [[v_h]]_\sigma \, ds. \end{aligned} \quad (3.9)$$

Other choices are also possible [7, 8, 9].

Considering the bilinear form  $a_h$  defined by (3.6), (3.8), and (3.9) the discontinuous Galerkin scheme reads

$$\sum_{K \in \mathcal{T}_h} \frac{1}{\Delta t_n} \int_K (u_h^n - u_h^{n-1}) v_h \, dx + \mathcal{D} a_h(u_h^n, v_h) = 0. \quad (3.10)$$

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

$$\mathbb{A}^n \mathbf{U}^n = \mathbf{F}^{n-1} \quad (3.11)$$

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

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

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

$$\begin{aligned} \mathbb{A}_{l,l'}^n := & \sum_{K \in \mathcal{T}_h} \left( \frac{1}{\Delta t_n} \int_K \psi_{h,l} \psi_{h,l'} \, dx + \mathcal{D} \int_K \nabla \psi_{h,l} \cdot \nabla \psi_{h,l'} \, dx \right) \\ & + \mathcal{A}_h(\psi_{h,l}, \psi_{h,l'}) \quad \forall 1 \leq l, l' \leq N_h^{\text{int}} \end{aligned}$$

and the right-hand side vector satisfies

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

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

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

**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 [37], the PCG [38] and the multigrid algorithm [39]. 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  $N_{\text{sp}}$  (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  $\Omega$  is partitionned into intervals  $E_i := [x_{i-1}, x_i]$ ,  $1 \leq i \leq N_{\text{sp}}$  where  $\bar{\Omega} = \cup_{i=1}^N \bar{E}_i$  and that  $\bar{\Omega} = [x_0, x_{N_{\text{sp}}}]$ . We recall that a PDF  $f$  satisfies:  $f > 0$  a.e. on  $\Omega$  and  $\int_{\Omega} f(y) dy = 1$ .

#### 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  $\Omega$ . Define the cumulative function  $F : \Omega \rightarrow [0, 1]$  associated to the PDF  $f : \Omega \rightarrow \mathbb{R}_+^*$  by

$$F(x) := \int_{x_0}^x f(y) dy. \quad (4.1)$$

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:

$$\bar{x} = F^{-1}(\xi). \quad (4.2)$$

Equation (4.2) is also known as the inversion theorem of the cumulative [12]. 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

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



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

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

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  $\hat{g} : \Omega \rightarrow \mathbb{R}_+^*$  “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  $\alpha(x) = \frac{f(x)}{k\hat{g}(x)}$ . First, we compute the cumulative distribution  $G : \Omega \rightarrow [0, 1]$  associated to the density  $\hat{g}$  following (4.1). Next, we sample according to the PDF  $\hat{g}$  the position  $\bar{x}$  employing the direct inversion of the cumulative procedure (4.2) and we compute  $\alpha(\bar{x})$ . Then, we consider a uniform random number  $\xi$  following the uniform law on the interval  $[0, 1]$ :  $\xi \sim \mathcal{U}([0, 1])$ . If  $\xi \leq \alpha(\bar{x})$ , then accept  $\bar{x}$ . 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 [12, 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  $\mathbf{X}^0 \in \mathbb{R}^M$ . If  $u_0 > 0$  is not a probability density function, i.e.  $\int_{\Omega} u_0(x) dx \neq 1$ , we sample from the PDF  $\tilde{u}_0$  defined by  $\tilde{u}_0(x) := \frac{u_0(x)}{\int_{\Omega} u_0(y) dy}$ . Obviously,  $\tilde{u}_0 > 0$  and  $\int_{\Omega} \tilde{u}_0(x) dx = 1$ . We obtain a collection of particle positions that we denote by  $\tilde{\mathbf{X}}^0$ . Finally, to obtain a population of particles  $\mathbf{X}^0 \in \mathbb{R}^M$  corresponding to the initial condition  $u_0$ , we consider the population  $\tilde{\mathbf{X}}^0$  and we attribute to each particle  $i$  a statistical weight equal to  $\omega_i = \int_{\Omega} u_0(y) dy$ . 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  $[\tilde{t}_{n-1}, \tilde{t}_n]$ ,  $1 \leq n \leq N^*$  such that  $0 = \tilde{t}_0 < \tilde{t}_1 < \dots < \tilde{t}_{N^*} = T$ . The time steps are denoted by  $\delta t_n = \tilde{t}_n - \tilde{t}_{n-1}$ ,  $n = 1, \dots, N^*$ . The underlying stochastic process for the diffusion equation whose diffusion coefficient is  $\mathcal{D} > 0$  is a Brownian motion or a Wiener process [30, 41, 42]. The variation of a Wiener process, over each time interval ( $\delta t_n \rightarrow 0$ ), is a continuous Gaussian probability density function. We simulate the Brownian motion at times  $\tilde{t}_n$ ,  $n = 0, \dots, N^* - 1$ . Knowing the position  $\mathbf{x}'$  at time  $t' > 0$  of a given particle, we determine its subsequent position  $\mathbf{x}$  at time  $t > t'$  by sampling the Gaussian transition kernel

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

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

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

$$\mathbf{X}^t = \mathbf{X}^{t'} + \sqrt{2\mathcal{D}(t-t')} \mathcal{S} \quad (4.4)$$

where  $\mathbf{S} \in \mathbb{R}^M$  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 [43] for more details.

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

$$\int_K u^n(x) dx \approx \frac{1}{M} \sum_{i=1}^M \mathbf{1}_{i \in K}^n \times \omega_i. \quad (4.5)$$

Here,  $\mathbf{1}_{i \in K}^n$  denotes the characteristic function such that  $\mathbf{1}_{i \in K}^n = 1$  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  $M \rightarrow +\infty$ , 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  $x = x_{N_{\text{sp}}}$  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 [44]. 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  $Z_{K,j}^n$  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  $j$ :

$$Z_{K,j}^n := \frac{1}{M'} \sum_{i=1}^{M'} \mathbf{1}_{i \in K,j}^n \times \omega_i \quad (4.6)$$

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

$$\int_K u^n(x) dx \approx \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n. \quad (4.7)$$

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  $\text{Var}(Z_K^n)$  is defined in  $K \in \mathcal{T}_h$  by

$$\text{Var}(Z_K^n) := \frac{1}{p-1} \left( \frac{1}{p} \sum_{j=1}^p (Z_{K,j}^n)^2 - \left( \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n \right)^2 \right).$$

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

$$\hat{\sigma}^n := \sqrt{\text{Var}(Z_K^n)}.$$

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

$$\hat{I}_K^n := \left[ \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n - 1\hat{\sigma}^n, \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n + 1\hat{\sigma}^n \right], \quad (4.8)$$

$$I_K^n := \left[ \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n - 2\hat{\sigma}^n, \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n + 2\hat{\sigma}^n \right], \quad (4.9)$$

$$\tilde{I}_K^n := \left[ \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n - 3\hat{\sigma}^n, \frac{1}{p} \sum_{j=1}^p Z_{K,j}^n + 3\hat{\sigma}^n \right]. \quad (4.10)$$

From the probability theory [45], the probability for the exact solution  $u$  at time  $T_n$  to be in  $\hat{I}_K^n$  is approximately equal to 68%, the probability to be in  $I_K^n$  is approximately equal to 95%, and the probability for the exact solution to be in  $\tilde{I}_K^n$  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  $\mathbf{U}^n \in \mathbb{R}^m$  be the deterministic solution. When the finite element method and the discontinuous Galerkin method are employed  $m = N_h^{\text{int}}$ , and when the cell-centered finite volume method is considered,  $m = N_{\text{sp}}$ . The statistical representation of  $\mathbf{U}^n$  is still denoted by  $\mathbf{X}^n$  and to simplify the notations, we denote by  $\mathcal{F}_{\Delta T}(\mathbf{U}^n)$  the whole Monte Carlo computation. In fact, in this notation is gathered the statistical representation  $\mathbf{X}^n$  of  $\mathbf{U}^n$ , the fine discrete evolution over a time range of  $\Delta T$ , and the averaging step (4.7). Let  $1 \leq k \leq K$ , be the parareal index such that  $\mathbf{U}_k^{n+1}$  is the approximation of  $\mathbf{U}^{n+1}$ .

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

$$\mathbf{U}_k^{n+1} := \frac{1}{p} \sum_{j=1}^p \mathbf{U}_{k,j}^{n+1}. \quad (5.1)$$

In the sequel,  $M' > 1$  is the number of particles. The hybrid Monte Carlo algorithm that we propose is the following.

---

**Algorithm 1** Hybrid Monte Carlo Algorithm

---

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

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

**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  $\mathbf{X}_{0,j}^n \in \mathbb{R}^{M'}$  using the sampling procedure 4.1 from  $\mathbf{U}_0^n$  with uniform weights.

**else**

                When  $n = 0$ , use the statistical representation  $\mathbf{X}_{0,j}^0$  with uniform weights. Otherwise, employ the statistical representation obtained before the average  $\mathcal{F}_{\Delta T}(\mathbf{U}_{k-2,j}^{n-1})$  and modify each particle weight according to (5.5).

**end**

**3.** Compute the Monte Carlo propagation and the average.

**4.** Compute the correction term:

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

**end**

**for**  $n = 0 : N-1$

**5.** Compute the prediction term  $\mathcal{G}_{\Delta T}(\mathbf{U}_{k,j}^n)$  and next the hybrid solution at the time observable  $(n+1)\Delta T$ :

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

**6.** Update the statistical weights:

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

        where  $[\tilde{\omega}_{k-1,j}^n]_{i \in K}$  is the weight of the particle  $i \in K$  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)

**7.** Compute  $\mathbf{U}_k^{n+1} := \frac{1}{p} \sum_{j=1}^p \mathbf{U}_{k,j}^{n+1}$ . Check the stopping criterion:

    sup  $|\mathbf{U}_k^{n+1} - \mathbf{U}_{k-1}^{n+1}| \leq \frac{C}{\sqrt{M}}$  with  $C > 0$  a fixed parameter. If satisfied, set  $\mathbf{U}^{n+1} = \mathbf{U}_k^{n+1}$ . 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  $\mathbf{X}_{k=0,j}^{n+1}$  are different in each batch  $j \in [1, p]$ . For the sake of clarity, we have used the notation  $\mathbf{U}_{k=0,j}^{n+1}$  to indicate the presence of the sampling procedure on a given batch when where used the fine propagator. Next, observe that  $\mathbf{U}_{k,j}^1$  is

constant  $\forall k \geq 1$  since

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

**Remark 5.2.** Also, observe that our hybrid numerical solution computed from (5.4) is different from the one presented in (1.1). 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),  $\mathcal{G}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$  should be nonzero in all mesh elements. To tackle this difficulty, one can add to  $\mathcal{G}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$  the quantity  $\varepsilon \approx C/M$ .

**Remark 5.3.** On a given batch  $j \in [1, p]$ , when the hybrid solution  $\mathbf{U}_{k,j}^{n+1}$  is computed we need its statistical version  $\mathbf{X}_{k,j}^{n+1}$  to compute the subsequent parareal solution  $\mathbf{U}_{k+1,j}^{n+2}$ . Instead of sampling  $\mathbf{U}_{k,j}^{n+1}$  and thus introducing a bias in the Monte Carlo solver, we employ the statistical version of the deterministic object  $\mathcal{F}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$  (that we denote for instance by  $\tilde{\mathcal{F}}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$ ) 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  $\tilde{\mathcal{F}}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$  weighted by  $\omega_{k,j}^{n+1}$  provided by (5.5) gives the deterministic representation  $\mathbf{U}_{k,j}^{n+1}$ . 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  $\mathcal{F}_{\Delta T}(\mathbf{U}_{k-1,j}^n)$  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  $\Omega$  consisting in a segment of length  $L = 5$  m. We then solve the model

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

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

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

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.

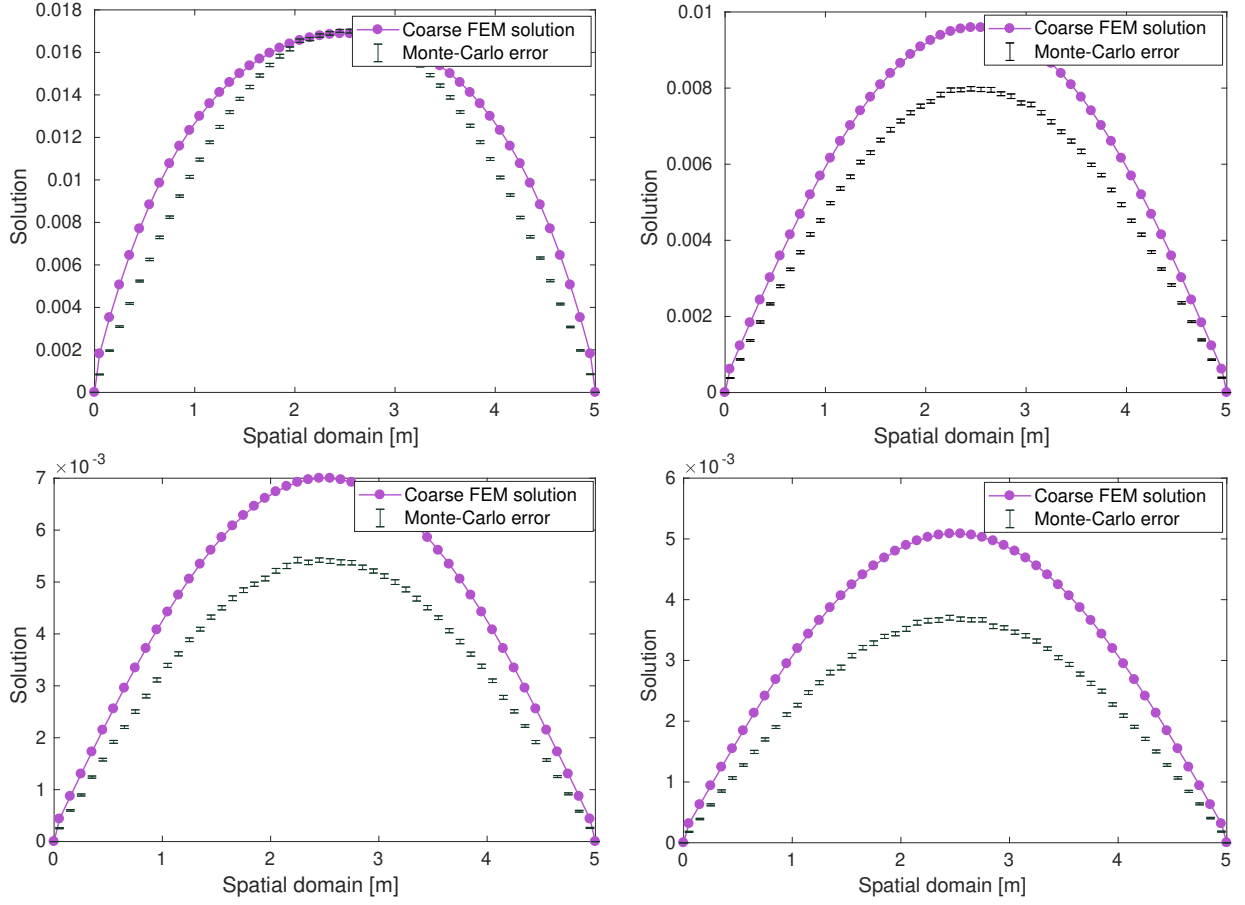


Figure 1: Coarse finite element resolution and statistical Monte Carlo error at time step  $n = 1$  (top left),  $n = 3$  (top right),  $n = 4$  (bottom left), and  $n = 5$  (bottom right).

## 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  $0.5 \text{ m}^2 \cdot \text{s}^{-1}$ . The initial condition  $u_0$  is taken as  $u_0(x) := \frac{1}{L}$  such that  $\int_0^L u_0(x) dx = 1$ .

1. Each particle  $i$  of the statistical representation of  $u_0$  has a statistical weight  $\omega_i = 1$ .

In Figure 1, we display the shape of the coarse finite element solution  $u^{\text{FEM}}$  for four selected time values:  $t = 2$  s,  $t = 6$  s,  $t = 8$  s and  $t = 10$  s. We observe that the coarse solution  $u^{\text{FEM}}$  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  $u^{\text{MC}}$  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  $u^{\text{FEM}}$ .

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

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

and

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

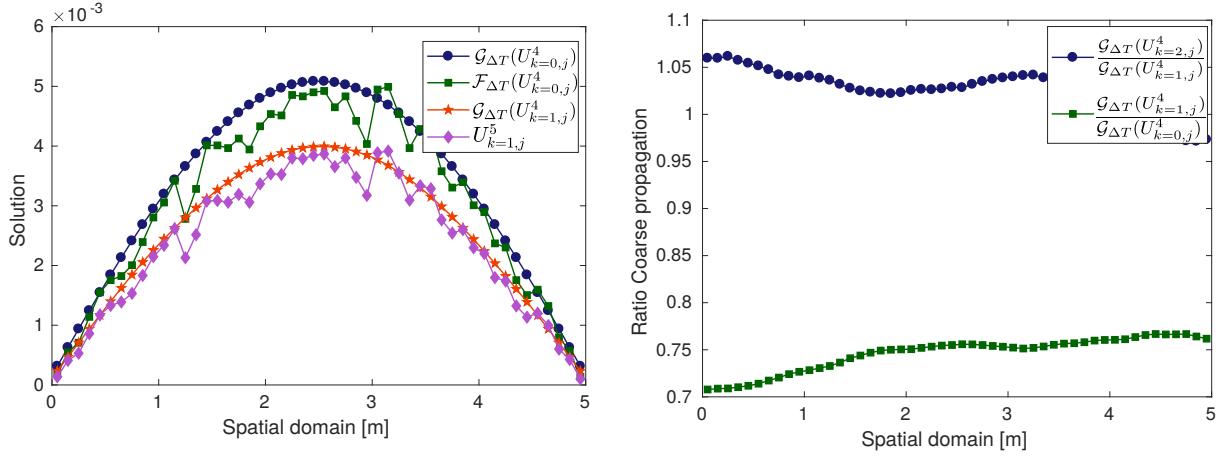


Figure 2: Construction of the hybrid solution  $U_{k=1,j}^5$  (left), and coarse propagators (right) at time iteration  $n = 5$  for the batch  $j = 1$ .

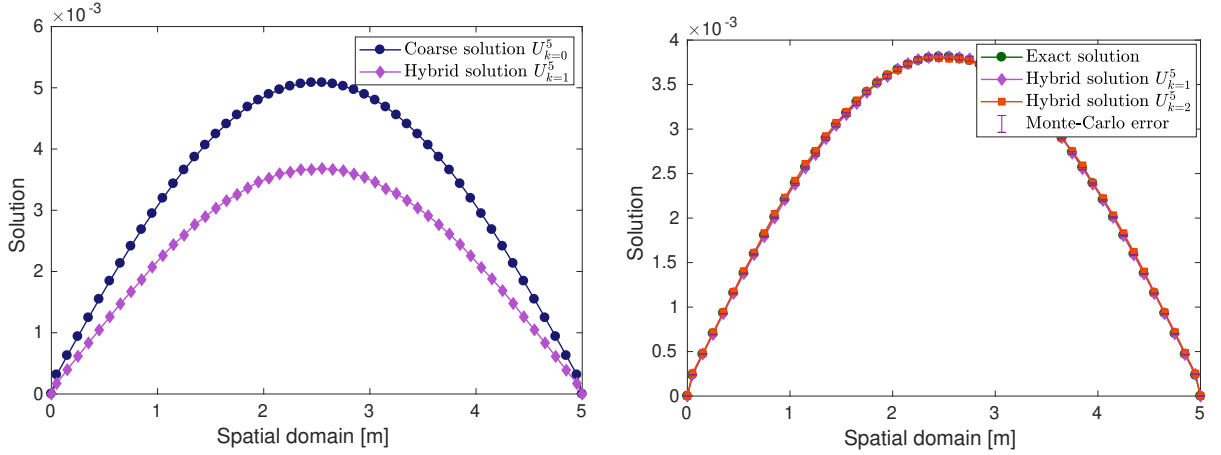


Figure 3: Shape of the hybrid solution at  $k = 0$  and  $k = 1$  (left) and shape of the exact solution and hybrid solution at  $k = 1$  and  $k = 2$  (right).

In the left Figure 2, the red curve displaying  $\mathcal{G}_{\Delta T}(U_{k=1,j}^4)$  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  $\mathcal{F}_{\Delta T}(U_{k=0,j}^4)$  and the blue curve  $\mathcal{G}_{\Delta T}(U_{k=0,j}^4)$  is the correction term. It lifts the coarse approximation  $\mathcal{G}_{\Delta T}(U_{k=1,j}^4)$  to obtain the numerical solution  $U_{k=1,j}^5$ . Also observe that the shape of the hybrid solution follows the shape of the result of the fine propagator  $\mathcal{F}_{\Delta T}(U_{k=0,j}^4)$  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  $\frac{\mathcal{G}_{\Delta T}(U_{k=2,j}^4)}{\mathcal{G}_{\Delta T}(U_{k=1,j}^4)}$  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  $U_{k=0}^5$  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,  $U_{k=1}^5 := \frac{1}{p} \sum_{j=1}^p U_{k=1,j}^5$ . Furthermore, observe that  $\|(u - u^{\text{HYB}})(x, T)\|_{L^\infty(\Omega)} \approx 3 \times 10^{-4}$  which is in agreement with the Monte Carlo

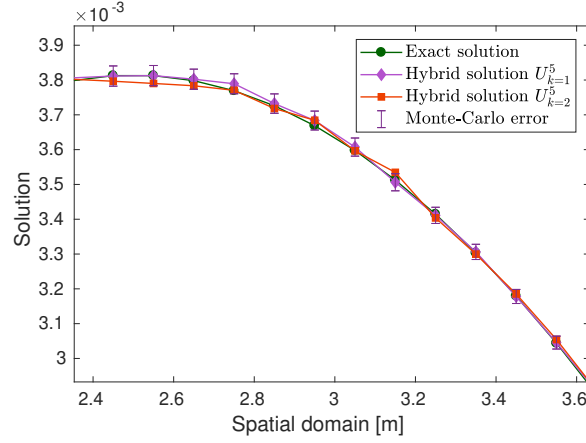


Figure 4: Hybrid solution at time iteration  $n = 5$  and at parareal iteration  $k = 1, k = 2$ , and Monte Carlo error bar.

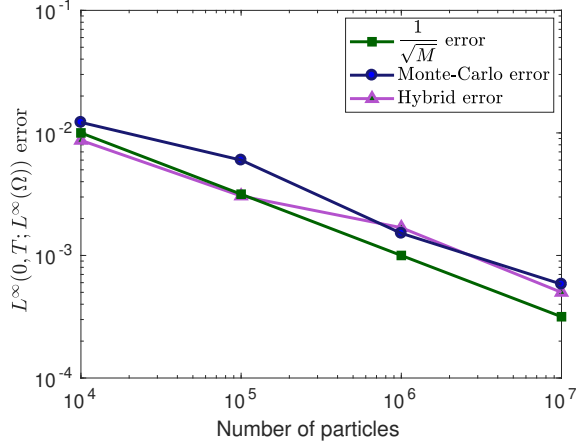


Figure 5: Error in the  $L^\infty(0, T; L^\infty(\Omega))$  norm between  $u$  and  $u^{\text{MC}}$ , and error in the  $L^\infty(0, T; L^\infty(\Omega))$  between  $u$  and  $u^{\text{HYB}}$  as a function of the number of particles.

convergence speed. Next, we observe that the solutions  $U_{k=1}^5$  and  $U_{k=2}^5$  coincide on the spatial domain  $\Omega$  in the sense that the stopping criterion 7. 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  $U_{k=1}^5$  and  $U_{k=2}^5$  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

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

and

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

We observe that the curves of the Monte Carlo error and of the hybrid error behave like  $\frac{1}{\sqrt{M}}$  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



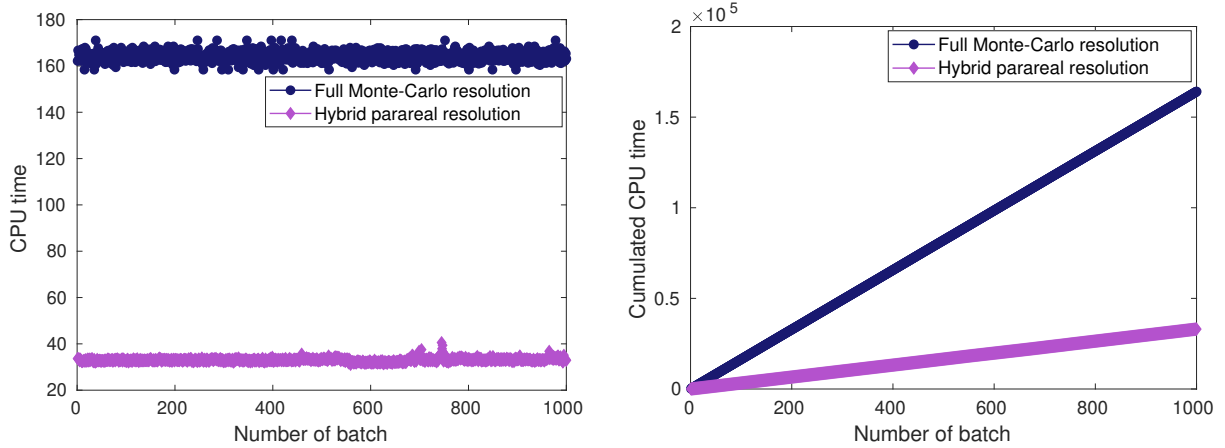


Figure 6: CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

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

parallelization per batch is used (left Figure 6), and when no parallelization 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  $3.3 \times 10^4$  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 completeness, 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  $\mathcal{D} = 0.5 m^2.s^{-1}$  for the fine propagator and  $\mathcal{D} = 0.48 m^2.s^{-1}$  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

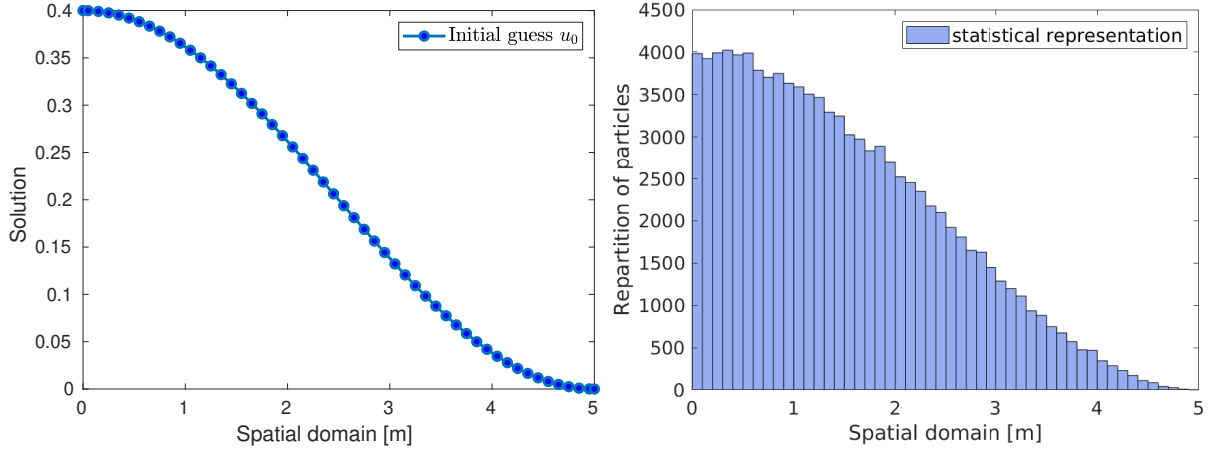


Figure 7: Solution at initial time  $t = 0$  (left) and histogram of the statistical population at initial time  $t = 0$  (right) for the second test case.

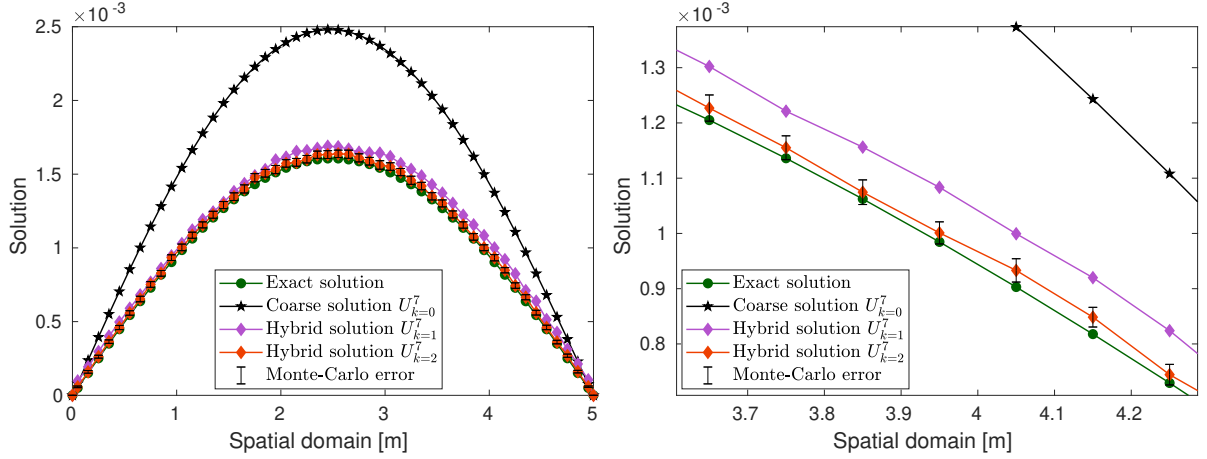


Figure 8: Shape of the hybrid solution at  $n = 7$  and  $k = 0$ ,  $k = 1$ ,  $k = 2$ , and shape of the exact solution.

$M = 10^7$ . 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  $\Delta t$  is chosen equal to the observable window  $\Delta T$ . 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  $U_k^7$  for  $k = 0$ ,  $k = 1$ , and  $k = 2$ . Note that the solution  $U_{k=0}^7$  corresponds to the coarse solution obtained using the coarse solver while  $U_{k=1}^7$  and  $U_{k=2}^7$  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

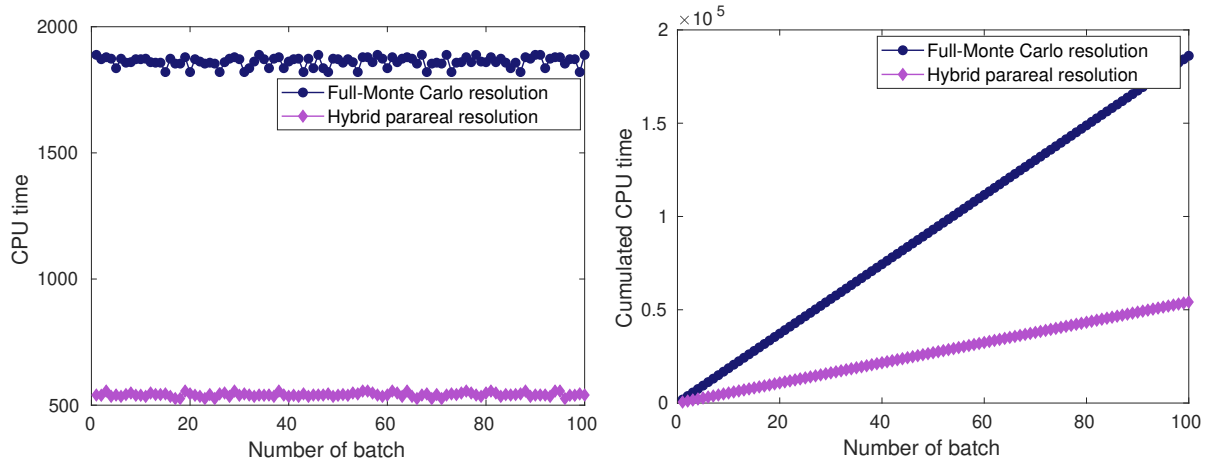


Figure 9: CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

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 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 strategy 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] A. Quarteroni, A. Valli, Numerical approximation of partial differential equations, Vol. 23 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1994.
- [2] C. Bernardi, Y. Maday, F. Rapetti, Discrétisations variationnelles de problèmes aux limites elliptiques, Vol. 45 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer-Verlag, Berlin, 2004.
- [3] S. C. Brenner, L. R. Scott, [The mathematical theory of finite element methods](#), Vol. 15 of Texts in Applied Mathematics, Springer-Verlag, New York, 1994. doi:10.1007/978-1-4757-4338-8. URL <https://doi.org/10.1007/978-1-4757-4338-8>
- [4] E. Godlewski, P.-A. Raviart, [Numerical approximation of hyperbolic systems of conservation laws](#), Vol. 118 of Applied Mathematical Sciences, Springer-Verlag, New York, 1996. doi:10.1007/978-1-4612-0713-9. URL <https://doi.org/10.1007/978-1-4612-0713-9>

- [5] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, in: Handbook of numerical analysis, Vol. VII, Handb. Numer. Anal., VII, North-Holland, Amsterdam, 2000, pp. 713–1020.
- [6] B. Després, [Numerical methods for Eulerian and Lagrangian conservation laws](#), Frontiers in Mathematics, Birkhäuser/Springer, Cham, 2017. doi:10.1007/978-3-319-50355-4.  
URL <https://doi-org/10.1007/978-3-319-50355-4>
- [7] 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, theory and implementation. doi:10.1137/1.9780898717440.  
URL <https://doi.org/10.1137/1.9780898717440>
- [8] D. A. Di Pietro, A. Ern, [Mathematical aspects of discontinuous Galerkin methods](#), Vol. 69 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer, Heidelberg, 2012. doi:10.1007/978-3-642-22980-0.  
URL <https://doi.org/10.1007/978-3-642-22980-0>
- [9] V. Dolejší, M. Feistauer, [Discontinuous Galerkin method](#), Vol. 48 of Springer Series in Computational Mathematics, Springer, Cham, 2015, analysis and applications to compressible flow. doi:10.1007/978-3-319-19267-3.  
URL <https://doi-org.accesdistant.sorbonne-universite.fr/10.1007/978-3-319-19267-3>
- [10] M. H. Kalos, P. A. Whitlock, [Monte Carlo methods. Vol. I](#), A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1986, basics. doi:10.1002/9783527617395.  
URL <https://doi.org/10.1002/9783527617395>
- [11] G. Ökten, [Solving linear equations by Monte Carlo simulation](#), SIAM J. Sci. Comput. 27 (2) (2005) 511–531. doi:10.1137/04060500X.  
URL <https://doi.org/10.1137/04060500X>
- [12] I. Lux, L. Koblinger, [Monte Carlo particle transport methods: neutron and photon calculations](#), CRC Press, 1991.  
URL <https://books.google.fr/books?id=4u7vAAAAMAAJ>
- [13] M. Mascagni, N. A. Simonov, [Monte Carlo methods for calculating some physical properties of large molecules](#), SIAM J. Sci. Comput. 26 (1) (2004) 339–357. doi:10.1137/S1064827503422221.  
URL <https://doi.org/10.1137/S1064827503422221>
- [14] Y. I. Khlopkov, Z. Y. M. Myint, A. Y. Khlopkov, Monte Carlo method and its parallel computing technique in molecular gas dynamics, International Journal of Educational Research and Information Science 2 (1) (2015) 1.
- [15] H. Zaidi, C. Labbé, 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 (9) (1998) 1523 – 1536. doi:[https://doi.org/10.1016/S0167-8191\(98\)00069-6](https://doi.org/10.1016/S0167-8191(98)00069-6).  
URL <http://www.sciencedirect.com/science/article/pii/S0167819198000696>
- [16] M. Faucher, D. Mancusi, A. Zoia, New kinetic simulation capabilities for Tripoli-4®: Methods and applications, Ann. Nucl. Energy 120 (2018) 74–88. doi:10.1016/j.anucene.2018.05.030.
- [17] J. Nievergelt, [Parallel methods for integrating ordinary differential equations](#), Comm. ACM 7 (1964) 731–733. doi:10.1145/355588.365137.  
URL <https://doi-org/10.1145/355588.365137>
- [18] W. L. Miranker, W. Liniger, [Parallel methods for the numerical integration of ordinary differential equations](#), Math. Comp. 21 (1967) 303–320. doi:10.2307/2003233.  
URL <https://doi-org/10.2307/2003233>

- [19] P. Chartier, B. Philippe, [A parallel shooting technique for solving dissipative ODEs](#), Computing 51 (3-4) (1993) 209–236. doi:[10.1007/BF02238534](#).  
URL <https://doi-org/10.1007/BF02238534>
- [20] J.-L. Lions, Y. Maday, G. Turinici, [Résolution d'EDP par un schéma en temps "pararéel"](#), C. R. Acad. Sci. Paris Sér. I Math. 332 (7) (2001) 661–668. doi:[10.1016/S0764-4442\(00\)01793-6](#).  
URL [https://doi.org/10.1016/S0764-4442\(00\)01793-6](https://doi.org/10.1016/S0764-4442(00)01793-6)
- [21] P. F. Fischer, F. Hecht, Y. Maday, [A parareal in time semi-implicit approximation of the Navier-Stokes equations](#), in: Domain decomposition methods in science and engineering, Vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 433–440. doi:[10.1007/3-540-26825-1\\_44](#).  
URL [https://doi-org/10.1007/3-540-26825-1\\_44](https://doi-org/10.1007/3-540-26825-1_44)
- [22] F. Legoll, T. Lelièvre, G. Samaey, [A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations](#), SIAM J. Sci. Comput. 35 (4) (2013) A1951–A1986. doi:[10.1137/120872681](#).  
URL <https://doi.org/10.1137/120872681>
- [23] A.-M. Baudron, J.-J. Lautard, Y. Maday, 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.
- [24] A.-M. Baudron, J.-J. Lautard, Y. Maday, M. K. Riahi, J. Salomon, [Parareal in time 3D numerical solver for the LWR benchmark neutron diffusion transient model](#), J. Comput. Phys. 279 (2014) 67–79. doi:[10.1016/j.jcp.2014.08.037](#).  
URL <https://doi.org/10.1016/j.jcp.2014.08.037>
- [25] I. Garrido, M. S. Espedal, G. E. Fladmark, A convergent algorithm for time parallelization applied to reservoir simulation, in: T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, J. Xu (Eds.), Domain Decomposition Methods in Science and Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 469–476.
- [26] L. Baffico, S. Bernard, Y. Maday, G. Turinici, G. Zérah, [Parallel-in-time molecular-dynamics simulations](#), Phys. Rev. E 66 (2002) 057701. doi:[10.1103/PhysRevE.66.057701](#).  
URL <https://link.aps.org/doi/10.1103/PhysRevE.66.057701>
- [27] 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. doi:[10.1007/3-540-26825-1\\_43](#).  
URL [https://doi.org/10.1007/3-540-26825-1\\_43](https://doi.org/10.1007/3-540-26825-1_43)
- [28] M. J. Gander, S. Vandewalle, [Analysis of the parareal time-parallel time-integration method](#), SIAM J. Sci. Comput. 29 (2) (2007) 556–578. doi:[10.1137/05064607X](#).  
URL <https://doi.org/10.1137/05064607X>
- [29] G. Pagès, O. Pironneau, G. Sall, [The parareal algorithm for american options](#), SIAM J. Financial Math. 9 (3) (2018) 966–993. doi:[10.1137/17M1138832](#).  
URL <https://doi.org/10.1137/17M1138832>
- [30] L. Infeld, On the theory of Brownian motion, University of Toronto Studies, Applied Mathematics Series, no. 4, University of Toronto Press, Toronto, Ont., 1940.
- [31] F. Legoll, T. Lelièvre, K. Myerscough, G. Samaey, [Parareal computation of stochastic differential equations with time-scale separation: a numerical convergence study](#), Comput. Vis. Sci. 23 (1-4) (2020) Paper No. 9, 18. doi:[10.1007/s00791-020-00329-y](#).  
URL <https://doi-org/10.1007/s00791-020-00329-y>
- [32] J.-L. Lions, Quelques méthodes de résolution des problèmes aux limites non linéaires, Dunod; Gauthier-Villars, Paris, 1969.

- [33] R. Dautray, 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.
- [34] H. Brezis, *Functional analysis, Sobolev spaces and partial differential equations*, Universitext, Springer, New York, 2011.
- [35] D. N. Arnold, [An interior penalty finite element method with discontinuous elements](#), SIAM J. Numer. Anal. 19 (4) (1982) 742–760. doi:10.1137/0719052.  
URL <https://doi.org/10.1137/0719052>
- [36] B. Rivière, M. F. Wheeler, V. Girault, [A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems](#), SIAM J. Numer. Anal. 39 (3) (2001) 902–931. doi:10.1137/S003614290037174X.  
URL <https://doi.org/10.1137/S003614290037174X>
- [37] Y. Saad, [Iterative methods for sparse linear systems](#), 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003. doi:10.1137/1.9780898718003.  
URL <https://doi.org/10.1137/1.9780898718003>
- [38] 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.  
URL <https://doi.org/10.1137/1.9781611970944>
- [39] W. L. Briggs, *A multigrid tutorial*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1987.
- [40] C. P. Robert, G. Casella, [Introducing Monte Carlo methods with R](#), Use R!, Springer, New York, 2010. doi:10.1007/978-1-4419-1576-4.  
URL <https://doi-org/10.1007/978-1-4419-1576-4>
- [41] E. Nelson, *Dynamical theories of Brownian motion*, Princeton University Press, Princeton, N.J., 1967.
- [42] S. Umarov, M. Hahn, K. Kobayashi, [Beyond the triangle: Brownian motion, Ito calculus, and Fokker-Planck equation—fractional generalizations](#), World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2018. doi:10.1142/10734.  
URL <https://doi-org/10.1142/10734>
- [43] G. E. P. Box, M. E. Muller, [A note on the generation of random normal deviates](#), Ann. Math. Statist. 29 (2) (1958) 610–611. doi:10.1214/aoms/1177706645.  
URL <https://doi.org/10.1214/aoms/1177706645>
- [44] 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.  
URL <https://books.google.fr/books?id=MqEZAQAIAAJ>
- [45] J. Jacod, P. Protter, [Probability essentials](#), 2nd Edition, Universitext, Springer-Verlag, Berlin, 2003. doi:10.1007/978-3-642-55682-1.  
URL <https://doi-org/10.1007/978-3-642-55682-1>