Journal of Computational and Applied Mathematics 420 (2023) 114800

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A hybrid parareal Monte Carlo algorithm for parabolic problems[☆]

Jad Dabaghi[a,*], Yvon Maday[b,c], Andrea Zoia[d]

[a] Léonard de Vinci Pôle Universitaire, Research Center, 92 916 Paris La Défense, France
[b] Sorbonne Université, CNRS, Université Paris Cité, Laboratoire Jacques-Louis Lions (LJLL), F-75005, Paris, France
[c] Institut Universitaire de France, France
[d] Université Paris-Saclay, CEA, Service d'études des réacteurs et de mathématiques appliquées, 91191, Gif-sur-Yvette, France

ARTICLE INFO

Article history:
Received 11 March 2021
Received in revised form 23 June 2022

MSC:
65N30
65N75
65F10
65H10
68W10

Keywords:
Parareal-in-time algorithm
Time-dependent problems
Predictor–corrector
Galerkin schemes
Monte Carlo method

ABSTRACT

In this work, we propose a hybrid Monte Carlo/deterministic "parareal-in-time" approach devoted to accelerating Monte Carlo simulations over massively parallel computing environments for the simulation of time-dependent problems.

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 high-fidelity 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.

1. Introduction

Several physical phenomena are described by partial differential equations (PDEs) whose analytical solution is often out of reach. In this context, numerical simulations appear to be in general the only viable approach to approximate a solution. Among the wide range of numerical methods for solving PDEs, we mention for instance the finite element methods [1–3], the finite volume methods [4–6], or the discontinuous Galerkin methods [7–9]. All these approaches belong to the class of deterministic methods. For a certain class of problems, e.g. in large dimension, probabilistic approaches based on Monte Carlo methods have been advocated [10–12] and have become the approach of choice in several fields of application, 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

☆ This project has received funding from the ANR project "Ciné-Para" (ANR-15-CE23-0019). This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 810367 - project EMC2) (YM) and from the European High Performance Computing Joint Undertaking (EuroHPC JU) under the European Union's Horizon 2020 research and innovation program (grant agreement No 955701 - project TIME-X) (YM).

* Corresponding author.
E-mail addresses: jad.dabaghi@devinci.fr (J. Dabaghi), yvon.maday@sorbonne-universite.fr (Y. Maday), andrea.zoia@cea.fr (A. Zoia).

https://doi.org/10.1016/j.cam.2022.114800
0377-0427/© 2022 Elsevier B.V. All rights reserved.

correspondingly an unaffordable computational cost and memory burden. Monte Carlo methods are weakly dependent on the dimensionality and are natively implemented over massively parallel computing environments [14,15].

The Monte Carlo approach consists in approximating the sought solution $u(\boldsymbol{x}, t)$ by sampling a large number $M \gg 1$ of random walks whose estimated density at point $\boldsymbol{x}$ and time t converges in the limit of large M to $u(\boldsymbol{x}, t)$. It is well known that the statistical uncertainty of the average quantities estimated by 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 therefore a very large number of random walks, which demands a high computational cost. These random walks (or histories) are independent, so that the Monte Carlo sampling can be performed in parallel.

In the context of nuclear reactor physics, where stochastic methods are used to establish reference solutions to be compared to faster but approximated solutions obtained by deterministic methods, until recently 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]. However, the growth in available computer power stimulates the application of Monte Carlo to the simulation of time-dependent neutron transport, in order 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 for neutron transport [16]). Similar efforts aimed at making non-stationary problems increasingly accessible by Monte Carlo methods are carried out in many other disciplines, such as molecular dynamics [17]. When addressing time-dependent problems, the limiting factor affecting the total simulation time for a given amount of available processors is the fact that the time variable has a natural flow and cannot be trivially parallelized. However, in recent years a few methods have been introduced for this purpose: the key idea behind the time parallelization is to decompose the time direction into "slices" where each interval can be handled in parallel. In this respect, several strategies have been proposed in order to efficiently cope with the parallelization in time [18–21]. Among these numerical methods, the parareal algorithm [21] relies 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 approximately integrate a given system of partial differential equations. The propagator $\mathcal{F}$ is a fine, accurate and thus expensive propagator, which approximates the exact solution u with high accuracy; on the contrary, the propagator $\mathcal{G}$ is a coarse propagator, which is a less accurate approximation of the exact solution u and much less expensive than $\mathcal{F}$. These solvers can, e.g., be based on different time steps (δt for $\mathcal{F}$ being typically much smaller than Δt for $\mathcal{G}$), but the model that $\mathcal{G}$ approximates may also be a simplified version of the underlying set of partial differential equations describing the model.

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 $\boldsymbol{u}_k := \left(\boldsymbol{u}_k^n\right)_{1\le n\le N}$ such that, for each iteration $k \geq 0$, $\boldsymbol{u}_k^n$ 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 over a time window of size ΔT) :

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

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

$$\forall n \geq 0,\ u_{k+1}^{n+1} := \underbrace{\mathcal{G}_{\Delta T}(u_{k+1}^n)}_{\text{Prediction}} + \underbrace{\left[\mathcal{F}_{\Delta T}(u_k^n) - \mathcal{G}_{\Delta T}(u_k^n)\right]}_{\text{Correction}}, \tag{1.1}$$

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 the algorithm converges, $k \to \infty$, Eq. (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 $\mathcal{F}$ with a weaker restitution clock time. In most of the publications on applications of the parareal algorithm, 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 [22]. A micro-macro version of the parareal algorithm for singularly perturbed systems of ordinary differential equations (ODEs) has been illustrated in [23], 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 [24,25] for kinetic transport problems or [26] for reservoir simulation. For a convergence study of the parareal algorithm we refer to [21,27–29]. In particular, a superlinear bound on the convergence on bounded time intervals for the diffusion equation and the advection equation has been demonstrated in [29]. The parareal-in-time strategy has been also applied to Monte Carlo methods, as in [30] where a parallelization of the Least-Square Longstaff–Schartz Monte Carlo algorithm dedicated to the pricing in american options has been discussed. The transport kernel used in that survey is a standard Brownian motion [31], 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 also been proposed in [32], by closely following the ideas of [23].

In this work, we explore the behavior of a novel hybrid version of the parareal algorithm, with a predictor based on a deterministic finite element solver and a corrector based on a Monte Carlo solver. Our approach, inspired by the groundbreaking work of Legoll et al. [32], makes the use of a finite element coarse propagator and a communication scheme between the coarse and the fine propagator within the parareal iterations that avoids additional discretization errors. This work contributes to the generalization of the parareal algorithm to the framework of mixed stochastic–deterministic discretizations in order to address, in the full term, the neutronics problems presented above.

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. We propose an answer to the following question: given a fixed precision, can we speed up a standard Monte Carlo resolution using our novel hybrid parareal approach ? To be more explicit, assume that a supercomputer has a very large number of processors M. A classical Monte Carlo resolution employs all the processors to simulate M random walks on a given time interval $[0, T]$. The precision of this method is of order $1/\sqrt{M}$. In the parareal procedure, we propose to allocate N clusters of processors to the time parallelization and in each of these clusters of processors we employ M fine propagations so that the total number of processors is equal to $M \times N$. The statistical precision is preserved in the parareal resolution and offers important computational savings when the number $\overline{k}$ of required parareal iterations to reach convergence is small. In the best scenario, the reduction factor is close to $\dfrac{N}{\overline{k}}$ which means that we could simulate the physical phenomenon on a time interval $[0, N \times T]$ with a resolution clock time equal to a pure Monte Carlo simulation over the time interval $[0, T]$. Then, this approach would be appealing in the context of radiation transport as it enables to simulate long simulation times and to treat the much longer time-scale of the delayed neutrons. This paper is organized as follows. First, in Section 2, we detail our model problem and settings. Next, in Section 3 we present the deterministic coarse propagator. 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 Ω. Let $H^1(\Omega)$ be the space of functions in $L^2(\Omega)$ 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

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

Here, $\mathcal{D} > 0$ 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 $\partial_t u \in L^2\left(0, T; H^{-1}(\Omega)\right)$ and satisfying for almost all $t \in]0, T[$ and for all $v \in H_0^1(\Omega)$

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

We mention the fundamental books of Lions [33], Dautray and Lions [34], and Brezis [35], 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 is 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 $0 = t_0 < t_1 < \cdots < t_{N_t} = T$. The time steps are denoted by $\Delta t_n = t_n - t_{n-1}$, $n = 1, \ldots, N_t$. 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 Ω, i.e. $\mathcal{T}_h$ is a set of simplicial elements 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 common vertex, or a common 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 partitioned into the interior nodes $\mathcal{V}_h^{\text{int}}$ and the boundary nodes $\mathcal{V}_h^{\text{ext}}$. 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^{int}. In the following, we define the coarse propagator $\mathcal{G}_{\Delta t_n}$ associated to the discretization of problem (2.1) corresponding to the finite element discretization.

3.2. The Lagrange finite element propagator

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

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

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 $\left(\psi_{h,l}\right)_{1\leq l\leq N_h}$ 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 $u_h^0 \in L^2(\Omega)$, where u_h^0 is some approximation of u_0 in X_{0h}^p, 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 $v_h \in X_{0h}^p$

$$\frac{1}{\Delta t_n} \int_\Omega \left(u_h^n - u_h^{n-1}\right) v_h \,\mathrm{d}x + \mathcal{D} \int_\Omega \nabla u_h^n \cdot \nabla v_h \,\mathrm{d}x = 0. \tag{3.1}$$

Expressing u_h^n in the Lagrange basis $\left(\psi_{h,l}\right)_{1\leq l\leq N_h^{\text{int}}}$, problem (3.1) reads

$$\mathbb{A}^n \boldsymbol{U}^n = \boldsymbol{F}^{n-1}. \tag{3.2}$$

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

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

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

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

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

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

In practice, we choose for the sake of simplicity, all the time steps to be equal: $\Delta t_n = \Delta t$. Thus, applying the propagator $\mathcal{G}_{\Delta t_n}$ amounts to solving (3.2) and yields:

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

Remark 3.1. 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 [36], the PCG [37] and the multigrid algorithm [38]. 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 and the number of unknowns is set to N_h.

4. The Monte Carlo solver as a fine propagator

In this section, we describe the 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 finite element method). Each particle carries a statistical weight, assigned according to some appropriate rules (see Section 5), and representative of the contribution of the particle to the approximate solution. 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 sample 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 probability density function (PDF) denoted by f. We recall several techniques available in the literature and we focus on the case $d = 1$ since sampling methods in higher dimensions are often decomposed into simpler one-dimensional sampling procedures. When $d = 1$, the domain Ω is partitioned into intervals $K_i := [x_{i-1}, x_i]$, $1 \leq i \leq N_e$ where $\overline{\Omega} = \cup_{i=1}^{N} \overline{K_i}$ and $\overline{\Omega} = \left[x_0, x_{N_e}\right]$. Here, N_e denotes the number of intervals. We recall that a PDF f satisfies: $f \geq 0$ on Ω and $\displaystyle\int_{\Omega} f(y)\,\mathrm{d}y = 1$.

4.1.1. Direct inversion of the cumulative distribution

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

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

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:

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

Then $\bar{x}$ is distributed according to the PDF f. Eq. (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 variables obeying the PDF f.

Remark 4.1. In some cases the cumulative function F is hard to invert.

4.1.2. The table lookup method

When the inversion of F is not possible analytically, numerical methods can be used instead. 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 $\xi \sim \mathcal{U}([0, 1])$. There exists a unique $i \in [1, N_e]$ 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.

Remark 4.2. Alternative sampling methods are available in the literature, as the rejection method and we refer to [12, Theorem 2.5] for a complete description.

4.2. Sampling of the initial condition

Suppose now that the initial condition u_0 of the PDE given in (2.1) is a probability density function. We use one of the sampling procedures given above to obtain a population of particles $\boldsymbol{X}^0 \in \mathbb{R}^M$. If $u_0 \geq 0$ is not normalized, i.e. $\int_{\Omega} u_0(x)\,\mathrm{d}x \neq 1$, we sample from the PDF $\tilde{u}_0$ defined by $\tilde{u}_0(x) := \dfrac{u_0(x)}{\int_{\Omega} u_0(y)\,\mathrm{d}y}$. Then, $\tilde{u}_0 \geq 0$ 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 sample particle i 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 $0 = \tilde{t}_0 < \tilde{t}_1 < \cdots < \tilde{t}_{N^\star} = T$. The time steps are denoted by $\delta t_n = \tilde{t}_n - \tilde{t}_{n-1}$, $n = 1, \dots, N^\star$. The underlying stochastic process for the diffusion equation with diffusion coefficient $\mathcal{D} > 0$ is a Brownian motion [31,39,40]. The displacement of a Brownian motion, over each time interval ($\delta t_n \to 0$), obeys a continuous Gaussian probability density function. We simulate the Brownian motion at times $\tilde{t}_n$, $n = 0, \dots N^\star - 1$. Knowing the position x' at time $t' > 0$ of a given particle, we determine its subsequent position x at time $t > t'$ by sampling the Gaussian transition kernel

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

The particle displacement $x - x'$ follows the normal distribution law whose mean is 0 and whose standard deviation is $\mathcal{D}(t - t')$ i.e. $x - x' \sim \mathcal{N}(0, \mathcal{D}(t - t'))$.

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 [41] for more details.

Finally, the approximation of the particle density $\int_K u^n(x)\,\mathrm{d}x$ 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 belongs to the element K at time $n\Delta T$ and is otherwise equal to 0. In the limit of $M \to \infty$, the law of large numbers ensures that the Monte Carlo sampling will yield an unbiased estimate of the particle density $u(x, t)$ integrated over each mesh element.

To take into account Dirichlet boundary conditions, the particles that cross the spatial boundaries $x = x_0$ or $x = x_{N_e}$ are killed, meaning that their simulation is abandoned and they will not be considered in the histogram. The variance and the standard deviation can be computed to determine the confidence intervals associated to the average estimates provided by (4.5); see also [42].

Remark 4.3. Concerning homogeneous 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. 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 unbiased 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

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

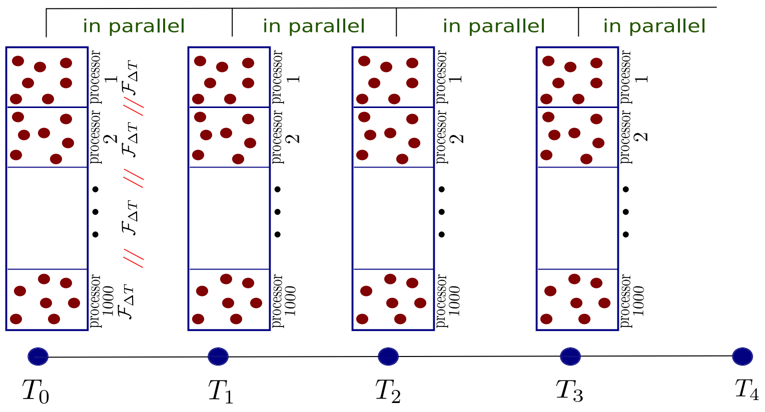

in parallel
in parallel
in parallel
in parallel
processor 1
processor 2
processor 1000
$\mathcal{F}_{\Delta T}$ // $\mathcal{F}_{\Delta T}$ // $\mathcal{F}_{\Delta T}$ // $\mathcal{F}_{\Delta T}$
T_0
T_1
T_2
T_3
T_4

Fig. 1. Time parallelization procedure for the hybrid algorithm.

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

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

We also define the 2-sigmas Monte Carlo error bars in each mesh element $K \in \mathcal{T}_h$:

$$I_K^n := \left[\frac{1}{p} \sum_{j=1}^{p} Z_{K,j}^n - 2\widehat{\sigma}^n, \frac{1}{p} \sum_{j=1}^{p} Z_{K,j}^n + 2\widehat{\sigma}^n \right]. \tag{4.8}$$

From the probability theory [43], the probability for the exact solution u at time T_n to be in I_K^n is approximately equal to 95%.

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, closely following the ideas of Legoll et al. [32]. Let $\boldsymbol{U}^n \in \mathbb{R}^m$ be the deterministic solution. Here, $m = N_h^{\mathrm{int}}$ since the finite element method is employed. 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 we gather 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 $1 \leq k \leq \overline{k}$, be the parareal index such that $\boldsymbol{U}_k^{n+1}$ 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 (refer to Algorithm 1)

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

In the sequel, $M' > 1$ is the number of particles.

In Fig. 1, we present the conceptual scheme of our parareal strategy. In this example, the time interval $[0, T]$ is partitioned into 4 time windows: $0 = T_0 < T_1 < T_2 < T_3 < T_4 = T$. We want to reach a statistical precision equal to $\dfrac{1}{\sqrt{10^7}}$. We consider arbitrarily $M = 4 \times 10^3$ processors. Here, 4 groups of processors are allocated to the time parallelization. Within each parallel-in-time propagation, 10^3 processors are available. Inside each of these processors, $M' = 10^4$ independent random walks (particles) are simulated. The precision of this method is of order $\dfrac{1}{\sqrt{10^7}}$. In a standard Monte Carlo resolution, 10^7 independent random walks are simulated (10^3 replicas and 10^4 particles per replica) so that the sought precision is ensured. Furthermore, the cost of the parareal strategy depends on both the number of parareal windows N and the required iteration number k to achieve accuracy. If for instance $k = 1$, it means that the required clock time of our parareal strategy is 4 times less than a pure Monte Carlo simulation, yielding a significant computational speed-up. Another option, following the discussion in Section 1, is that we can explore the Brownian motion simulation on a larger time interval $\left[T_0, T_1, T_2, T_3, T_4, \ldots, T_\sharp\right]$ with $T_\sharp > 0$. The hybrid Monte Carlo algorithm that we propose is the following (see Algorithm 1).

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 $(n+1)\,\Delta T$, $0 \le n \le N-1$, by the coarse propagator

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

for $k = 1 : \overline{k}$
    for $j = 1 : p$ (in parallel)
        for $n = 0 : N-1$ (in parallel)
            2. if $k = 1$
                Compute the statistical representation $\boldsymbol{X}_{k,j}^0 = \boldsymbol{X}_{0,j}^0 \in \mathbb{R}^{M'}$ using the sampling procedure 4.1 from $\boldsymbol{U}_0^n$
                with uniform weights.
              else
                When $n = 0$, use the statistical representation $\boldsymbol{X}_{0,j}^0$ with Otherwise, employ for $\boldsymbol{X}_{k-1,j}^n$ 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:

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

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

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

            6. Update the statistical weights:

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

            where $\left[\tilde{\omega}_{k-1,j}^n\right]|_{i\in K}$ 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 Section 5.1.
        end
    end
    for $n = 0 : N-1$ (loop for parareal update)
        7. Compute $\boldsymbol{U}_k^{n+1} := \frac{1}{p}\sum_{j=1}^{p} \boldsymbol{U}_{k,j}^{n+1}$. Check the stopping criterion: $\sup\left|\boldsymbol{U}_k^{n+1} - \boldsymbol{U}_{k-1}^{n+1}\right| \le \frac{C}{\sqrt{M}}$ 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 $\forall k \ge 1$ 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. We want to point out that our hybrid numerical solution computed from (5.4), is different – in the form but not in the substance – from the one presented in (1.1). Indeed the original idea is based on the general idea of an additive group framework. As originally noticed in [44] the structure of the solution may require another group operation. In [44] the natural set of solution was the manifold of norm 1 functions and the group of transformation was thus the set of rotations, here, we notice, as also proposed in [32, Section 4.2] in the context of the Fokker–Planck equation, that the solution of the diffusion model (2.1) should be nonnegative and the multiplicative group framework given by the formula

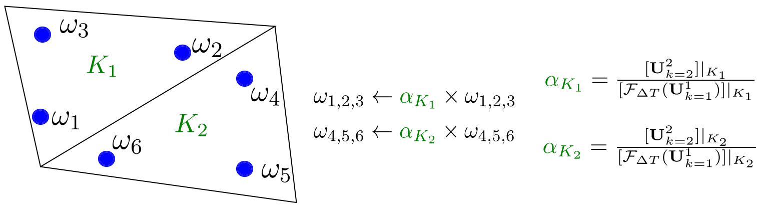

ω_3
K_1
ω_2
ω_4
ω_1
K_2
ω_6
ω_5
$\omega_{1,2,3} \leftarrow \alpha_{K_1} \times \omega_{1,2,3}$
$\omega_{4,5,6} \leftarrow \alpha_{K_2} \times \omega_{4,5,6}$
$\alpha_{K_1} = \frac{[\mathbf{U}^2_{k=2}]|_{K_1}}{[\mathcal{F}_{\Delta T}(\mathbf{U}^1_{k=1})]|_{K_1}}$
$\alpha_{K_2} = \frac{[\mathbf{U}^2_{k=2}]|_{K_2}}{[\mathcal{F}_{\Delta T}(\mathbf{U}^1_{k=1})]|_{K_2}}$

Fig. 2. Evolution of the statistical weights.

(5.4) is more convenient. Note also that in the definition of the correction factor (5.3), $\mathcal{G}_{\Delta T}(\boldsymbol{U}^n_{k-1,j})$ should be nonzero in all mesh elements. To tackle this difficulty, one can add to $\mathcal{G}_{\Delta T}(\boldsymbol{U}^n_{k-1,j})$ the small quantity $\varepsilon \approx C/M$.

5.1. Correction of the statistical weights

On a given batch $j \in [1, p]$, when the hybrid solution $\boldsymbol{U}^{n+1}_{k,j}$ is computed we need its statistical version $\boldsymbol{X}^{n+1}_{k,j}$ to compute the subsequent parareal solution $\boldsymbol{U}^{n+2}_{k+1,j}$. Instead of sampling $\boldsymbol{U}^{n+1}_{k,j}$ and thus introducing a discretization error, we employ the statistical version of the deterministic object $\mathcal{F}_{\Delta T}(\boldsymbol{U}^n_{k-1,j})$ (that we denote $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}^n_{k-1,j})$) which is available but we modify each associated weight as in (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 $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}^n_{k-1,j})$ weighted by $\omega^{n+1}_{k,j}$ provided by (5.5) gives the deterministic representation $\boldsymbol{U}^{n+1}_{k,j}$.

An example of this procedure is provided in Fig. 2 where 6 particles are distributed in a domain composed of 2 elements K_1, K_2. In this example, we assume that we have computed the parareal solution

$$\boldsymbol{U}^2_{k=2,j} := \mathcal{G}_{\Delta T}(\boldsymbol{U}^1_{k=2,j}) \times \frac{\mathcal{F}_{\Delta T}(\boldsymbol{U}^1_{k=1,j})}{\mathcal{G}_{\Delta T}(\boldsymbol{U}^1_{k=1,j})}.$$

Instead of sampling $\boldsymbol{U}^2_{k=2,j}$ for the computation of the subsequent parareal solution $\boldsymbol{U}^3_{k=3,j}$, we consider the statistical representation $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}^1_{k=1,j})$ and we multiply each of its particle weights ω_i by a correction factor α_{K_i} where i is the index of the element containing the particle i. With such a procedure we observe that the normalized histogram of the population $\tilde{\mathcal{F}}_{\Delta T}(\boldsymbol{U}^1_{k=1,j})$ using the new weights provides an estimate of the deterministic quantity $\boldsymbol{U}^2_{k=2,j}$. Note that, in [32] other "iterators" and "matching" operators have been proposed and tested. In our study, the simple multiplicative one has revealed sufficient for a rapid convergence of the parareal algorithm.

5.2. Discussion on the CPU cost

In terms of CPU cost, the Monte Carlo algorithm presented in Section 4.4 employs p processors. Therefore, if $\widetilde{T} > 0$ is the required simulation time for a collection of M' particles to reach the time horizon $T > 0$ on one dedicated processor, the overall computational time is roughly equal to $\widetilde{T}$. We recall here that the total number of particles M satisfies $M = p \times M'$ and the accuracy of the method is $1/\sqrt{M}$. In the hybrid parareal resolution, assume we have a larger number of processors, say $N \times p$. We devote N groups of processors for the time parallelization (we recall that the time discretization involves N time observables) and p processors within each of these N groups. One processor is assimilated to one replica (or batch) so that p independent Monte Carlo simulations are performed in parallel. We simulate M' random walks in each replica so that the cumulated number of simulated particles is $M = M' \times p$. This approach displays again a $\sqrt{M}$ convergence. In terms of CPU time, the hybrid strategy requires a CPU time roughly equal to $\frac{\overline{k}}{N} \times \widehat{T}$ where $\widehat{T} \approx \widetilde{T}$ since the numerical cost of the fine propagator $\mathcal{F}_{\Delta T}$ is approximately the same in the hybrid resolution and the standard Monte Carlo resolution. A second possibility is to simulate the brownian motion on a larger time interval $[0, T^{\star}]$ (as would be required for taking into account the effect of delayed neutrons in the neutron transport that we indicated in the introduction) with $T^{\star} := N \times T > T$ in a clock time equal to $\overline{k} \times \widehat{T}$ that is smaller than what would be achieved sequentially i.e. $N \times \widehat{T}$ provided $\overline{k} \ll N$.

6. Numerical benchmark experiments

This section illustrates numerically the behavior of the hybrid parareal scheme proposed above. The numerical results have been implemented using MATLAB.

We consider a one dimensional domain Ω consisting in a segment of length $L = 5$ m. We then solve the model

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

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. Concerning the spatial discretization, we consider a uniform space step $\Delta x := 0.1\ m$. For the standard Monte Carlo resolution, we consider $p = 10^3$ processors so that 10^3 independent replicas of simulation are performed in parallel. In each replica, we consider $M' := 10^4$ particles so that the total number of simulated particles is $M := M' \times p = 10^7$. We denote by T_i^{MC} the simulation time associated to the replica i and by T^{MC} the overall simulation time. We simulate in MATLAB the parallelism in the sense that $T^{\mathrm{MC}} := \max\limits_{1\leq i\leq p} T_i^{\mathrm{MC}}$.

In the hybrid parareal resolution, as shown in Algorithm 1 and also in the discussion of Section 5.2, we allocate N groups of processors to the time parallelization and $p = 10^3$ processors for each parallel-in-time propagation. We thus have for each parallel-in-time propagation $p = 10^3$ Monte Carlo replicas of simulations and $M' = 10^4$ particles are simulated in each of these replicas. The required simulation time for the parareal strategy is more intricate. We denote by $T_{i,n}^{\mathrm{HYB}}$ the simulation time associated to a fixed replica i and for the time step n. Thanks to the time parallelization, the overall simulation time for the replica i is $T_i^{\mathrm{HYB}} := \max\limits_{1\leq n\leq N} T_{i,n}^{\mathrm{HYB}}$. The replicas are also parallelized so that the required simulation time for $\overline{k}$ opened parareal iteration is given by $T^{\mathrm{HYB}} := \overline{k} \max\limits_{1\leq i\leq p} T_i^{\mathrm{HYB}}$. In general, $\overline{k}$ is small and so the k-dependency is removed. Furthermore, the parameter C in the stopping criterion **7.** of Algorithm 1 is chosen as $C = 1$. 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 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.

6.1. A first test case

The final simulation time is set to $T = 10$ s and then the number of time steps is $N = N_t = 5$. 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 $\displaystyle\int_0^L u_0(x)\,\mathrm{d}x = 1$. To sample the initial distribution u_0, we employ the inversion theorem as described in Section 4.1.1. Each particle i of the statistical representation of u_0 has a statistical weight $\omega_i = 1$.

In Fig. 3, 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. It corresponds to solutions at the initial parareal stage $k = 0$. We observe that the coarse solution u^{FEM} is not within the Monte Carlo uncertainty defined in (4.8). These are expected results as the time step Δt associated to the coarse propagator $\mathcal{G}$ is big and yields a poor prediction. Here, the Monte Carlo solution u^{MC} is considered as the reference method. In Fig. 4, 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

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

and

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

In Fig. 4, (left) the red curve displaying $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=1,j}^4)$ represents the prediction terms, as computed by the coarse propagator based on the FEM method. It is computed fastly and its numerical cost is negligible. Next, the ratio of the green curve $\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k=0,j}^4)$ and the blue curve $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=0,j}^4)$ is the correction term. It shifts the coarse approximation $\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=1,j}^4)$ to obtain the numerical solution $\boldsymbol{U}_{k=1,j}^5$. Also observe that the shape of the hybrid solution follows the shape of the result of the fine propagator $\mathcal{F}_{\Delta T}(\boldsymbol{U}_{k=0,j}^4)$. Furthermore, we observe that the ratio $\dfrac{\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=2,j}^4)}{\mathcal{G}_{\Delta T}(\boldsymbol{U}_{k=1,j}^4)}$ 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 Fig. 5, 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 $\boldsymbol{U}_{k=0}^5$ is far from the exact solution u and lies outside the Monte Carlo uncertainty. Such

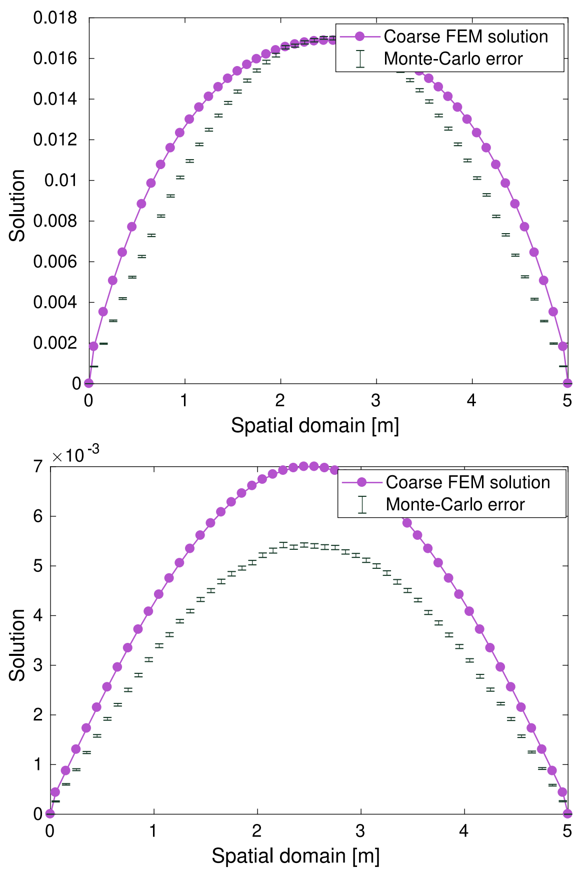


Fig. 3. Coarse finite element resolution and statistical Monte Carlo error at time step $n = 1$ (top left), $n = 3$ (top right), $n = 4$ (bottom left), and $n = 5$ (bottom right).

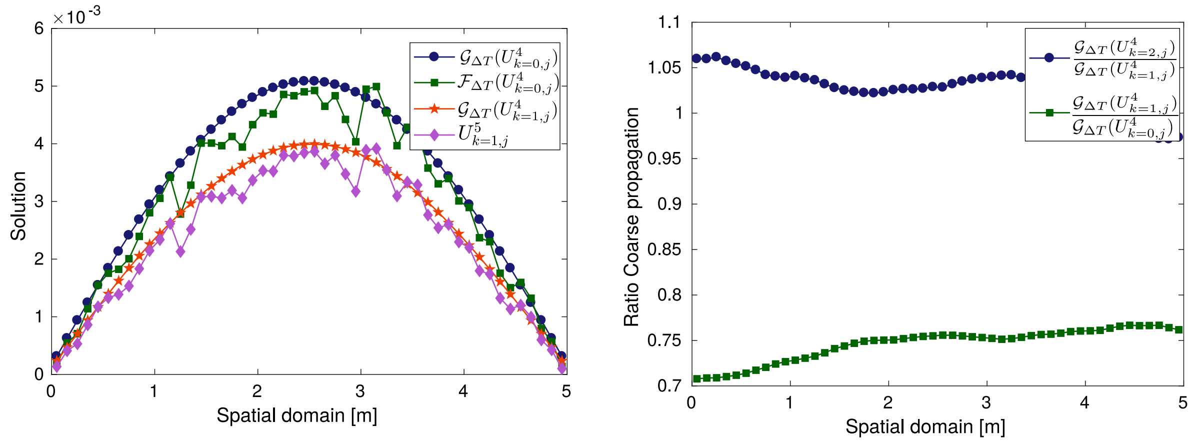
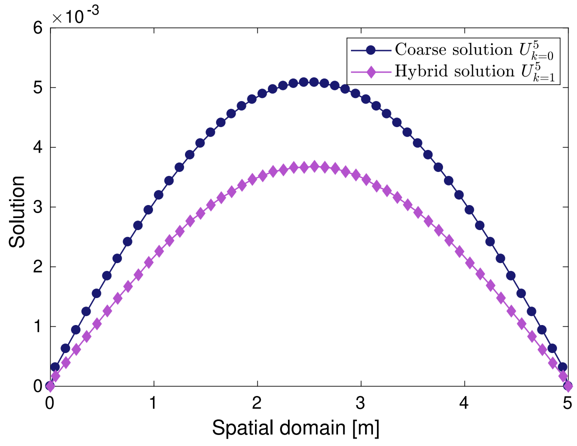


Fig. 4. Construction of the hybrid solution $\boldsymbol{U}^5_{k=1,j}$ (left), and coarse propagators (right) at time iteration $n = 5$ for the batch $j = 1$.

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 of Fig. 5. 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 precision. Next, we observe that the solutions $\boldsymbol{U}^5_{k=1}$ and $\boldsymbol{U}^5_{k=2}$ coincide on the spatial domain Ω in the sense



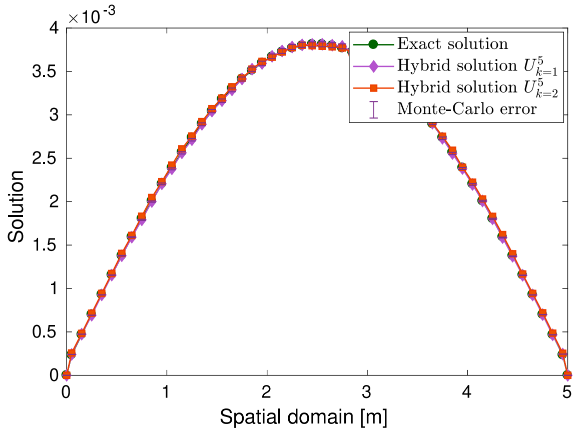
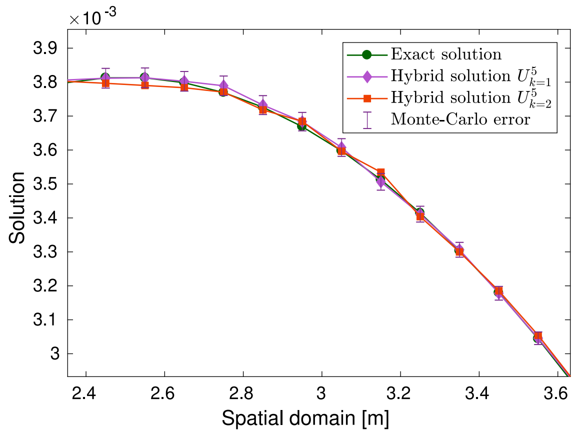


Fig. 5. Shape of the hybrid solution at $k = 0$ and $k = 1$ (left) and shape of the exact solution and hybrid solution at $k = 1$ and $k = 2$ (right).



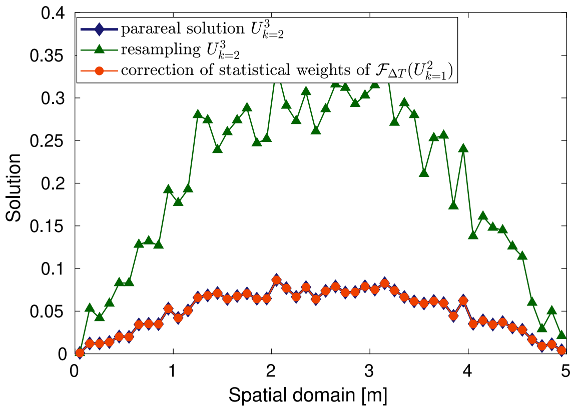


Fig. 6. Hybrid solution at time iteration $n = 5$ and at parareal iteration $k = 1$, $k = 2$ (left), and correction of the statistical weights (right).

that the stopping criterion **7.** of Algorithm 1 is satisfied. Thus our hybrid strategy requires only 2 parareal iterations to converge for the current time step.

Fig. 6 is a crucial complement to Fig. 5, 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 right cells of Fig. 5. Furthermore, we have represented on the right the effect of the correction of the statistical weights as explained in Section 5.1. The blue curve represents the deterministic parareal solution $\boldsymbol{U}^3_{k=2,j}$. Sampling a population of particles from $\boldsymbol{U}^3_{k=2,j}$ denoted by $\boldsymbol{X}^3_{k=2,j}$ involves a discretization error as the normalized histogram of $\boldsymbol{X}^3_{k=2,j}$ (green curve) does not give back $\boldsymbol{U}^3_{k=2,j}$. However, employing the available population $\mathcal{F}_{\Delta T}(\boldsymbol{U}^2_{k=1})$ for which we modify the weight of all the particles contained thereby (refer to Section 5.1) enables to fully recover the deterministic representation $\boldsymbol{U}^3_{k=2}$.

In Fig. 7, 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, i.e. $\left\|u - u^{\mathrm{MC}}\right\|_{L^\infty(0,T;L^\infty(\Omega))}$ and $\left\|u - u^{\mathrm{HYB}}\right\|_{L^\infty(0,T;L^\infty(\Omega))}$. Recall that

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

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 Fig. 8 and Table 1 and 2. We compared in Fig. 8 the global CPU time of the simulation for two different strategies: when the parallelization per batch is used (left Fig. 8), and when no parallelization per batch is used (right Fig. 8). More precisely, on the left Figure, one processor is assigned to each batch. For the full Monte Carlo algorithm, 10^3 processors are available. For the hybrid parareal strategy

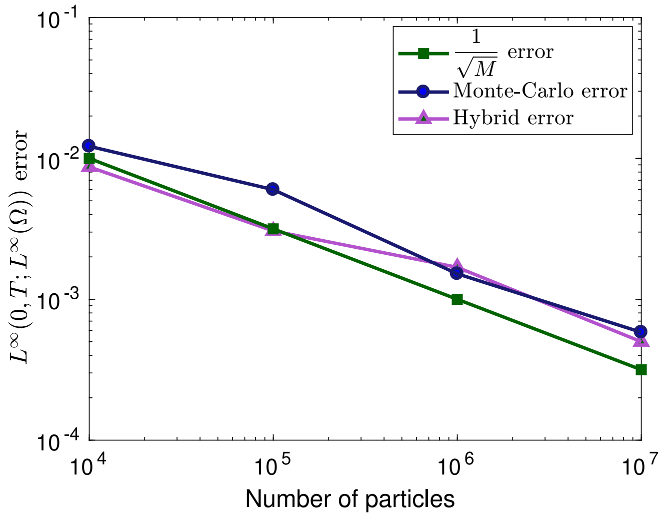


Fig. 7. Error in the $L^\infty(0,T;L^\infty(\Omega))$ norm between u and u^{MC}, and error in the $L^\infty(0,T;L^\infty(\Omega))$ between u and u^{HYB} as a function of the number of particles.

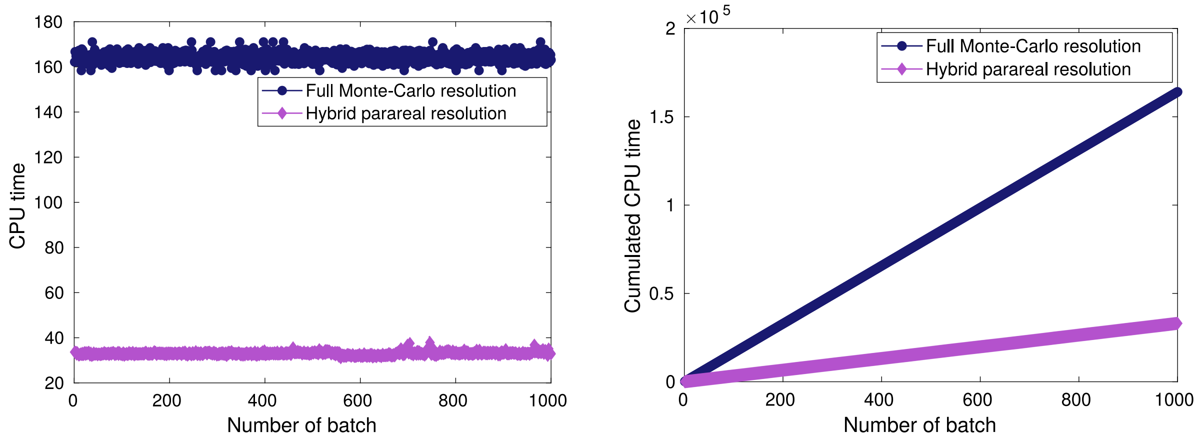


Fig. 8. CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

Table 1
Computational cost of the full Monte Carlo resolution.

Monte Carlo resolution		
Number of particles	Number of processors	CPU time
10^5	10^2	1653.4 s
10^4	10^3	164.09 s
10^3	10^4	16.86 s
10^2	10^5	1.78 s

5×10^3 processors are available : 5 groups for the time parallelization times 10^3 for the replicas. 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 T^{MC} 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, at $k = 1$, 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. It yields a gain factor equal to 4.96 in the overall CPU time which is very close to ideal scaling equal to $\frac{N}{k} = 5$. When no parallelization per batch occurs (right Figure) the standard Monte Carlo

Table 2
Computational cost of the hybrid parareal resolution.

Hybrid parareal resolution						
Number of processors time parallelization	Number of replicas for each parallel-in-time propagation	Number of particles for one replica j	CPU time $k = 1$	CPU time $k = 2$	Gain factor $k = 1$	Gain factor $k = 2$
5	10^2	10^5	335.76 s	537.16 s	4.92	3.04
5	10^3	10^4	33.05 s	53.1 s	4.96	3.09
5	10^4	10^3	3.39 s	5.49 s	4.97	3.07
5	10^5	10^2	0.35 s	0.58 s	5.08	3.02

algorithm computes the expectation sequentially (only one processor is available). However, the hybrid strategy employs 5 processors for the correction step which makes this method faster. 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. For the sake of completeness, we also test in Table 1 and 2 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 $k = 1$ parareal iteration is performed and roughly equal to 3 when $k = 2$ parareal iterations are performed. It is close to the ideal scaling equal to 2.5. Finally, we proved that the excess of processors enables to speed-up a Monte Carlo resolution and may serves to simulate longer time domains as shown in Section 6.3.

6.2. A second test case

For this second example, we consider a different initial solution and we choose a coarse propagator that weakly degrades the physics of the problem. 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. "The artificial" degradation of the coarse propagator in this numerical experiment, in the same spirit as the one proposed in Ref. [32], aims at mimicking the situation where, for a more complex problem (e.g. neutronics), the macroscopic model is less accurate than in our toy model. 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. In this test case, we still consider a statistical precision of order $\frac{1}{\sqrt{10^7}}$. In the Monte Carlo procedure, we consider $M' = 10^5$ particles and we simulate $p = 10^2$ independent replicas (batches) so that the total number of particles is $M = 10^7$. In the hybrid parareal algorithm, 7 groups of processors are allocated to the time parallelization. For each fine parallel-in-time propagation, 10^2 processors are available. In each of these processors (replicas), 10^5 particles are simulated. Therefore, the cumulated number of simulated particles is also equal to 10^7. 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 Fig. 9 we represent the shape of the initial guess and its statistical version. 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 (see Section 4.1.2) to find numerically the intervals where each particle lives.

Fig. 10 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 within the Monte Carlo error bar. In Fig. 11 we represent for the two strategies the required CPU time. Similarly 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 method 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 of 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.

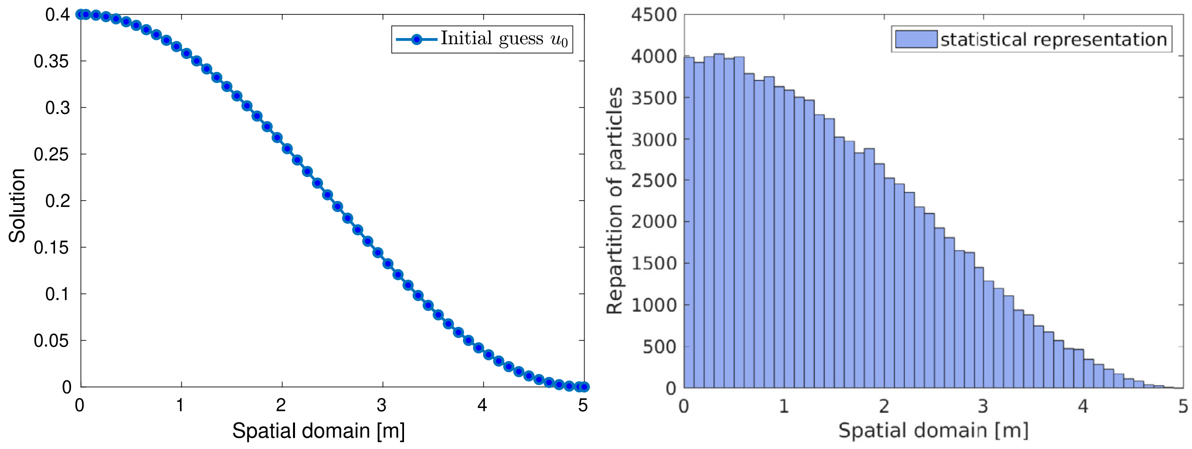


Fig. 9. Solution at initial time $t = 0$ (left) and histogram of the statistical population at initial time $t = 0$ (right) for the second test case.

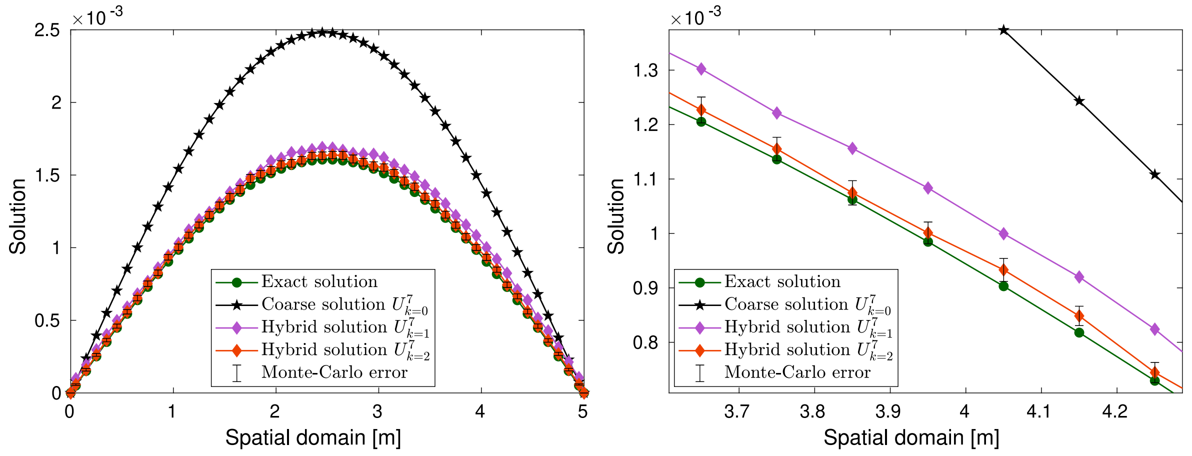


Fig. 10. Shape of the hybrid solution at $n = 7$ and $k = 0$, $k = 1$, $k = 2$, and shape of the exact solution.

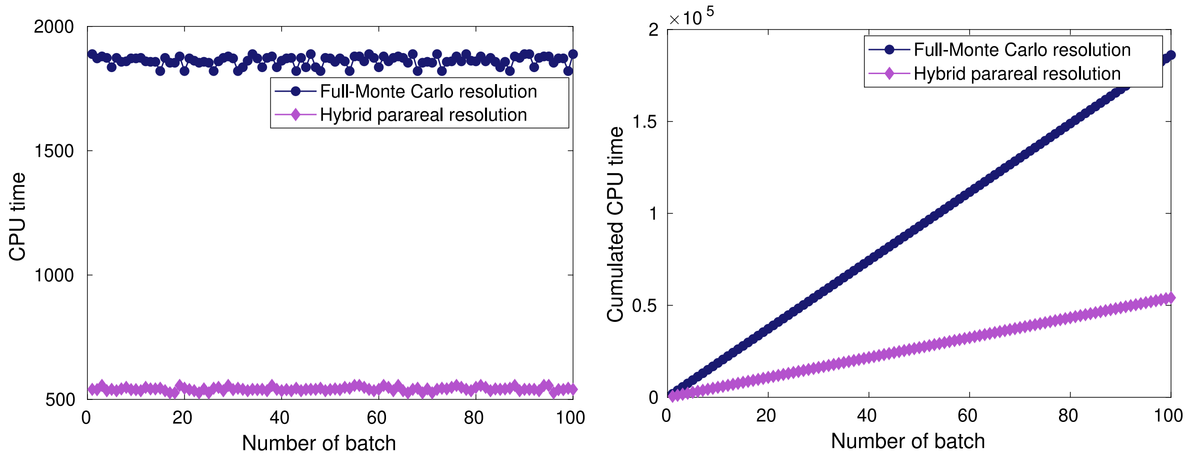


Fig. 11. CPU time for each batch (left) and cumulated CPU time with no parallelization per batch (right).

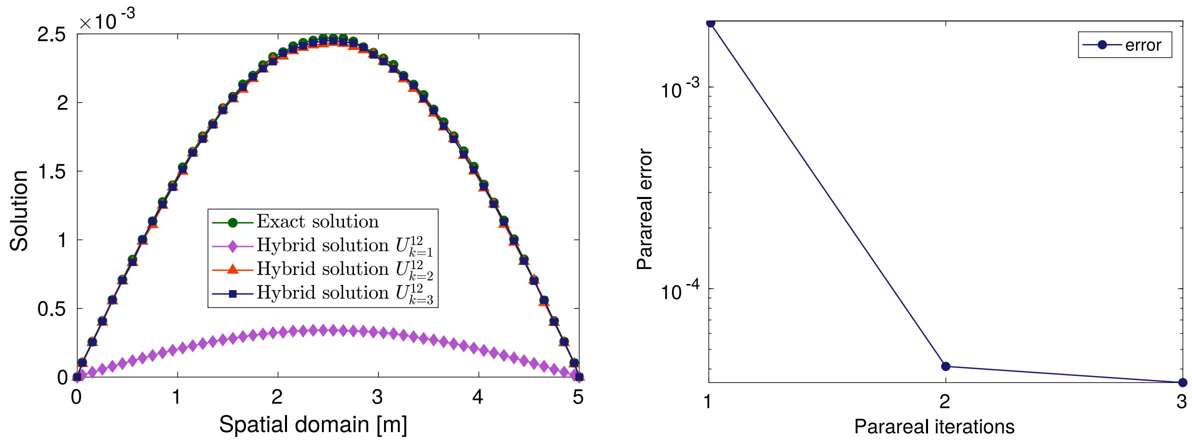


Fig. 12. Shape of the hybrid solution at $n = 12$ and $k = 1$, $k = 2$, $k = 3$, and shape of the exact solution.

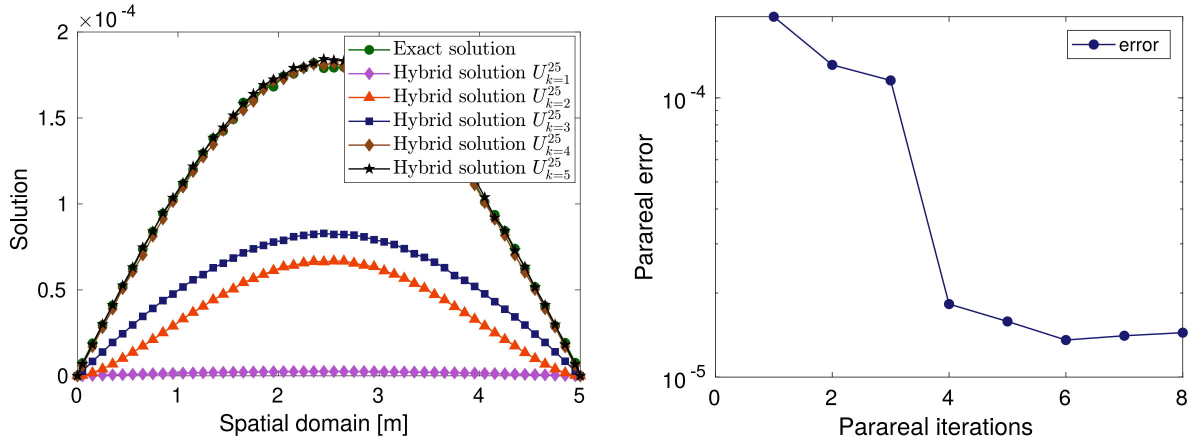


Fig. 13. Shape of the hybrid solution at $n = 25$ and shape of the exact solution.

6.3. *A third test case: extension to long simulation times with higher precision*

For this third example, we consider the two different situations described in Section 5.2. First, we consider a longer simulation time $T = 50$ *s*. The initial condition u_0 is the same as in the first test case: $u_0(x) := \frac{1}{L}$ and the inverse of the cumulative procedure is used to sample u_0. The diffusion coefficient is set to $\mathcal{D} = 0.25$ m^2 s^{-1} for both propagators to avoid having solutions getting too small because of the Dirichlet boundary conditions. The coarse propagator is still a $\mathbb{P}_1$ finite element solver with constant time step $\Delta t := 2$ s and the fine propagator is a Monte Carlo solver with constant time step $\delta t = 2 \times 10^{-3}$ s. Here also, the coarse time step Δt is chosen equal to be the observable time window ΔT. The statistical precision for both numerical schemes is set to $\frac{1}{\sqrt{10^8}}$. In the Monte Carlo resolution, we consider 10^3 processors (replicas) and 10^5 particles per replicas. In our hybrid parareal resolution, 25 groups of processors are affected to the time parallelization. Therefore, 25 replicas of simulation are launched in parallel (one per each time observable). In each time replica, 10^3 processors are available. We decide to simulate $M' = 10^5$ random walks in each of these 10^3 processors.

In Fig. 12 we display the shape of the hybrid parareal solution at the time step $n = 12$ at $k = 1$, $k = 2$, and $k = 3$. We observe that $k = 3$ parareal iterations are required to converge in the sense of the stopping criterion (see point **7.** of Algorithm 1). The right Figure shows the behavior of the error $\left\|u - u^{\text{HYB}}\right\|_{L^\infty(\Omega)}$ at $n = 12$. We thus see that from the third parareal iteration the error stagnates.

The Fig. 13 is a complement to Fig. 12. We represented the shape of the numerical solution at the final simulation time. We observe that $k = 5$ parareal iterations are required to satisfy the stopping criterion **7.** of Algorithm 1. Therefore, as for the two previous test cases, few parareal iterations are required to converge, leading to an important computational speed-up.

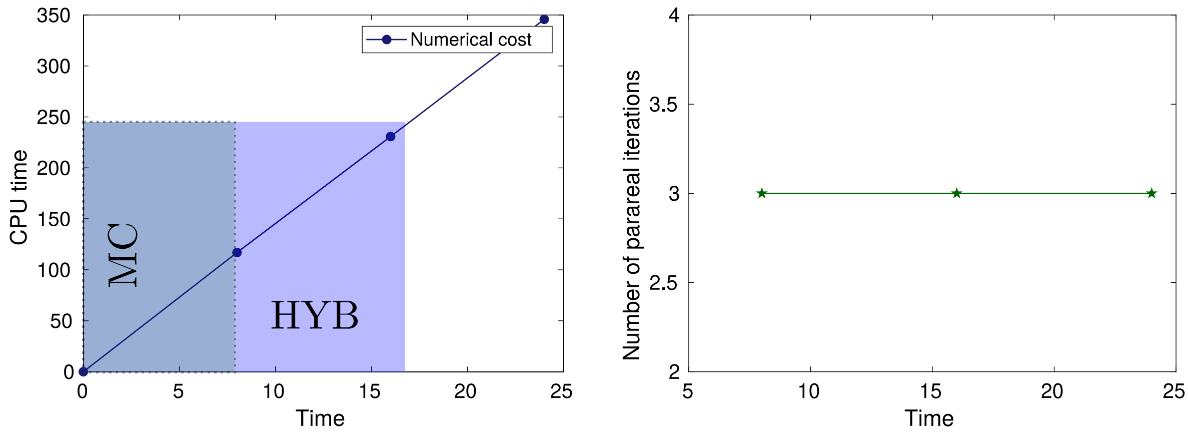


Fig. 14. CPU time for the hybrid resolution (left) and number of parareal iterations (right).

Finally, in Fig. 14 we illustrate the second feature of the hybrid algorithm as described in Section 5.2. We have represented the cumulated CPU time for the hybrid scheme as a function of the time steps (left) and the required number of parareal iterations to converge for each of these time steps (right). In the standard Monte Carlo method the final simulation time is taken equal to $T = 8$ s and the required simulation time to reach this time horizon T is approximately equal to 245 s. In the left Figure we show that our parareal procedure enables to reach a time horizon $T^{\star} \approx 17\ s$ with the same CPU cost as the pure Monte Carlo resolution. For each time slice, the hybrid resolution requires 3 parareal iterations to satisfy the stopping criterion **7.** of Algorithm 1. Thus, our parareal strategy is a very interesting approach as it enables to simulate a diffusion process over a longer time interval for a price equal to the one involved by a pure Monte Carlo simulation on a shorter interval.

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. Future work will concern the application of the proposed hybrid scheme to more sophisticate transport models: in particular, we will replace the diffusion equation by the 7-dimensional Boltzmann equation for neutron propagation, including the full physics of scattering, capture and fission events.

Acknowledgments

This work was funded by the ANR project “Ciné-Para” (ANR-15-CE23-0019). This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No 810367 - project EMC2) (YM) and from the European High Performance Computing Joint Undertaking (EuroHPC JU) under the European Union’s Horizon 2020 research and innovation program (grant agreement No 955701 - project TIME-X) (YM). 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, in: Springer Series in Computational Mathematics, Vol. 23, Springer-Verlag, Berlin, 1994, p. xvi+543.
[2] C. Bernardi, Y. Maday, F. Rapetti, Discrétisations Variationnelles de Problèmes Aux Limites Elliptiques, in: Mathématiques & Applications (Berlin) [Mathematics & Applications], Vol. 45, Springer-Verlag, Berlin, 2004, p. xii+310.
[3] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, in: Texts in Applied Mathematics, Vol. 15, Springer-Verlag, New York, 1994, p. xii+294, http://dx.doi.org/10.1007/978-1-4757-4338-8.
[4] E. Godlewski, P.-A. Raviart, Numerical Approximation of Hyperbolic Systems of Conservation Laws, in: Applied Mathematical Sciences, Vol. 118, Springer-Verlag, New York, 1996, p. viii+509, http://dx.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, in: Handb. Numer. Anal., VII, North-Holland, Amsterdam, 2000, pp. 713–1020.
[6] B. Després, Numerical Methods for Eulerian and Lagrangian Conservation Laws, in: Frontiers in Mathematics, Birkhäuser/Springer, Cham, 2017, p. xvii+349, http://dx.doi.org/10.1007/978-3-319-50355-4.
[7] B. Rivière, Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations, in: Frontiers in Applied Mathematics, Vol. 35, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008, p. xxii+190, http://dx.doi.org/10.1137/1.9780898717440, Theory and implementation.

[8] D.A. Di Pietro, A. Ern, Mathematical Aspects of Discontinuous Galerkin Methods, in: Mathématiques & Applications (Berlin) [Mathematics & Applications], Vol. 69, Springer, Heidelberg, 2012, p. xviii+384, http://dx.doi.org/10.1007/978-3-642-22980-0.
[9] V. Dolejší, M. Feistauer, Discontinuous Galerkin Method, in: Springer Series in Computational Mathematics, Vol. 48, Springer, Cham, 2015, p. xiv+572, http://dx.doi.org/10.1007/978-3-319-19267-3, Analysis and applications to compressible flow.
[10] M.H. Kalos, P.A. Whitlock, Monte Carlo Methods. Vol. I, in: A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1986, p. xii+186, http://dx.doi.org/10.1002/9783527617395, Basics.
[11] G. Ökten, Solving linear equations by Monte Carlo simulation, SIAM J. Sci. Comput. 27 (2) (2005) 511–531, http://dx.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, http://dx.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, Int. J. Educ. Res. Inf. Sci. 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 Comput. 24 (9) (1998) 1523–1536, http://dx.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, http://dx.doi.org/10.1016/j.anucene.2018.05.030.
[17] N. Castin, G. Bonny, A. Bakaev, C. Ortiz, A. Sand, D. Terentyev, Object kinetic Monte Carlo model for neutron and ion irradiation in tungsten: Impact of transmutation and carbon impurities, J. Nucl. Mater. 500 (2018) 15–25, http://dx.doi.org/10.1016/j.jnucmat.2017.12.014, URL https://www.sciencedirect.com/science/article/pii/S0022311517312813.
[18] J. Nievergelt, Parallel methods for integrating ordinary differential equations, Comm. ACM 7 (1964) 731–733, http://dx.doi.org/10.1145/355588.365137.
[19] W.L. Miranker, W. Liniger, Parallel methods for the numerical integration of ordinary differential equations, Math. Comp. 21 (1967) 303–320, http://dx.doi.org/10.2307/2003233.
[20] P. Chartier, B. Philippe, A parallel shooting technique for solving dissipative ODEs, Computing 51 (3–4) (1993) 209–236, http://dx.doi.org/10.1007/BF02238534.
[21] 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, http://dx.doi.org/10.1016/S0764-4442(00)01793-6.
[22] 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, in: Lect. Notes Comput. Sci. Eng., Vol. 40, Springer, Berlin, 2005, pp. 433–440, http://dx.doi.org/10.1007/3-540-26825-1_44.
[23] 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, http://dx.doi.org/10.1137/120872681.
[24] 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, in: Lect. Notes Comput. Sci. Eng., Vol. 98, Springer, Cham, 2014, pp. 437–445.
[25] 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, http://dx.doi.org/10.1016/j.jcp.2014.08.037.
[26] 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.
[27] L. Baffico, S. Bernard, Y. Maday, G. Turinici, G. Zérah, Parallel-in-time molecular-dynamics simulations, Phys. Rev. E 66 (2002) 057701, http://dx.doi.org/10.1103/PhysRevE.66.057701, URL https://link.aps.org/doi/10.1103/PhysRevE.66.057701.
[28] 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, in: Lect. Notes Comput. Sci. Eng., Vol. 40, Springer, Berlin, 2005, pp. 425–432, http://dx.doi.org/10.1007/3-540-26825-1_43.
[29] M.J. Gander, S. Vandewalle, Analysis of the parareal time-parallel time-integration method, SIAM J. Sci. Comput. 29 (2) (2007) 556–578, http://dx.doi.org/10.1137/05064607X.
[30] G. Pagès, O. Pironneau, G. Sall, The parareal algorithm for american options, SIAM J. Financial Math. 9 (3) (2018) 966–993, http://dx.doi.org/10.1137/17M1138832.
[31] L. Infeld, On the Theory of Brownian Motion, in: University of Toronto Studies, Applied Mathematics Series, no. 4, University of Toronto Press, Toronto, Ont., 1940, p. 42.
[32] 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) http://dx.doi.org/10.1007/s00791-020-00329-y, Paper No. 9, 18.
[33] J.-L. Lions, Quelques Méthodes de Résolution Des Problèmes Aux Limites Non Linéaires, Dunod; Gauthier-Villars, Paris, 1969, p. xx+554.
[34] R. Dautray, J.-L. Lions, Mathematical Analysis and Numerical Methods for Science and Technology. Vol. 2, Springer-Verlag, Berlin, 1988, p. xvi+561, http://dx.doi.org/10.1007/978-3-642-61566-5, Functional and variational methods, With the collaboration of Michel Artola, Marc Authier, Philippe Bénilan, Michel Cessenat, Jean Michel Combes, Hélène Lanchon, Bertrand Mercier, Claude Wild and Claude Zuily, Translated from the French by Ian N. Sneddon.
[35] H. Brezis, Functional Analysis, Sobolev Spaces and Partial Differential Equations, in: Universitext, Springer, New York, 2011, p. xiv+599.
[36] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003, p. xviii+528, http://dx.doi.org/10.1137/1.9780898718003.
[37] C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations, in: Frontiers in Applied Mathematics, Vol. 16, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995, p. iv+165, With separately available software.
[38] W.L. Briggs, A Multigrid Tutorial, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1987, p. x+88.
[39] E. Nelson, Dynamical Theories of Brownian Motion, Princeton University Press, Princeton, N.J., 1967, p. iii+142.
[40] 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, p. xiii+177, http://dx.doi.org/10.1142/10734.
[41] G.E.P. Box, M.E. Muller, A note on the generation of random normal deviates, Ann. Math. Stat. 29 (2) (1958) 610–611, http://dx.doi.org/10.1214/aoms/1177706645.
[42] S.C. Port, Theoretical Probability for Applications, in: Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York, 1994, p. xviii+894, https://books.google.fr/books?id=MqEZAQAAIAAJ, A Wiley-Interscience Publication.
[43] J. Jacod, P. Protter, Probability Essentials, second ed., in: Universitext, Springer-Verlag, Berlin, 2003, p. x+254, http://dx.doi.org/10.1007/978-3-642-55682-1.
[44] Y. Maday, G. Turinici, A parallel in time approach for quantum control: the parareal algorithm, in: Proceedings of the 41st IEEE Conference on Decision and Control, 2002, Vol. 1, IEEE, 2002, pp. 62–66.