A time-step-robust algorithm to compute particle trajectories in 3-D unstructured meshes for Lagrangian stochastic methods

Guilhem Balvet, Jean-Pierre Minier, Christophe Henry, Yelva Roustan, Martin Ferrand

A time-step-robust algorithm to compute particle trajectories in 3-D unstructured meshes for Lagrangian stochastic methods

G. Balvet[*,1,2], J.-P. Minier[1,2], C. Henry[3], Y. Roustan[2], M. Ferrand[1,2]

[*] contact address: guilhem.balvet@edf.fr
[1] EDF R&D, Fluid Mechanics, Energy and Environment Dept., 6 Quai Watier,78400, Chatou, France
[2] CEREA, École des Ponts, Île-de-France, France
[3] Université Côte d'Azur, Inria, CNRS, Sophia-Antipolis, France

DOI: 10.1515/mcma-2023-2002

Abstract

The purpose of this paper is to propose a time-step-robust cell-to-cell integration of particle trajectories in 3-D unstructured meshes in particle/mesh Lagrangian stochastic methods. The main idea is to dynamically update the mean fields used in the time integration by splitting, for each particle, the time step into sub-time steps, such that each of these sub-steps corresponds to particle cell residence times. This reduces the spatial discretization error. Given the stochastic nature of the models, a key aspect is to derive estimations of the residence times that do not anticipate on the future of the Wiener process. To that effect, the new algorithm relies on a virtual particle, attached to each stochastic one, whose mean conditional behavior provides free-of-statistical-bias predictions of residence times. After consistency checks, this new algorithm is validated on two representative test cases: particle dispersion in a statistically uniform flow and particle dynamics in a non-uniform flow.

MSC 2010 class: 76M35

keyword: Lagrangian stochastic modeling, particle-mesh PDF, temporal integration, trajectory in 3-D unstructured mesh, time-splitting methods, anticipation error

1 Introduction

As it transpires from their name, Lagrangian probability density function (PDF) methods consist in simulating the PDF of a number of variables, which have been selected to describe relevant information about single-phase or disperse two-phase flows. From the Lagrangian PDF, the Eulerian PDF is immediately deduced which gives access to the statistical moments of interest. In a weak formulation in which only statistics are of interest, Lagrangian PDF methods are equivalent to particle stochastic models. Consequently, these methods appear as stochastic equations written for a set of variables attached to each particle which form the particle state vector (often composed of the particle position, its velocity and other variables of interest). These methods present strong advantages for modeling and simulating turbulent reactive particle-laden flows. In particular, they can treat without approximation local source terms, however complex or non-linear, when these source terms are given as known expressions of the variables attached to each particle (provided that these variables are included in the particle state vector). Consequently, PDF methods can solve closure problems related to the average of non-linear local terms, such as chemical source terms or the drag force which enters the particle equation of motion. At the same time, the Lagrangian point of view ensures that convection is treated accurately. This explains that these methods are well-suited to simulate polydisperse non-homogeneous two-phase flows, as well as single-phase reactive flows, while still being interesting models for inert single-phase turbulent flows.

Lagrangian PDF models started being developed in the 1980s and 1990s, mostly by Pope and co-workers [11, 33, 31, 34, 35]) and were later extended to the disperse two-phase flow situation by Minier and co-workers [24, 37, 21, 23, 16, 17, 18]. Over the years, there has been continuous effort to analyze stochastic models [38, 16, 17] as well as to come up with useful guidelines for their construction [20]. This is particularly true in stochastic models for the velocity of the fluid seen in disperse turbulent two-phase flows, which is a subject of ongoing research [13] (see also a recent proposal of a general methodology in [18]). Meanwhile, the situation seems to have reached a more mature state of affairs for single-phase turbulent flows with the formulation of general Langevin models (GLM) by Pope and various co-authors [11, 34, 32, 10, 14]. On the numerical side, there has been specific works dedicated to the analysis of the various issues related to the numerical implementation, which is often carried out in the context of particle/mesh methods [42, 29]. At this stage, it is worth reminding that many of such developments have been made in research codes mostly with a view towards validating new stochastic models hence using structured meshes in relatively simple geometries. To the authors' knowledge, less attention has been devoted to the question of particle tracking methods within complex realistic domains (e.g., for environmental purposes) used in combination with appropriate time integration schemes.

To integrate the stochastic differential equations (SDE) that model particle dynamics, a specific time-integration numerical scheme was introduced in [29, 22]. Building on the existence of relaxation terms in the particle velocity equations, this scheme was obtained, first by considering the integrated form involving exponential terms, and, second by discretizing these integrated expressions rather than the SDEs themselves. For this reason, it is referred to as an exponential numerical scheme. Note that this scheme is not applicable to any system of SDEs, since it assumes relaxation terms, but it is nevertheless of great interest for a wide range of modeled equations for particle dynamics in turbulent flows. This exponential scheme has interesting properties: by construction, it corresponds to the exact solution when all mean fields and timescales are constant (when the SDEs are linear), it remains explicit and unconditionally stable (thanks to the exponential factors in its formulation). Furthermore, careful discretization of the stochastic integrals allows to correctly reproduce the asymptotic regimes, when one or several of the timescales entering the relaxation terms become much smaller than the time step. As a typical example of key practical interest, this means that the diffusive regime is well captured without having to switch to a different numerical scheme. An in-depth description and analysis of this exponential scheme can be found in an extensive review on the subject and we refer interested readers to the available literature for further details [29].

The unconditional stability of the time-integration exponential scheme, combined with its ability to capture the correct asymptotic limits, allow large time steps to be used in practical simulations. However, in that case, a particle can typically cross a distance over which the mean fields entering its modeled dynamical equation, such as the fluid mean velocity field, can vary significantly. Yet, in the spirit of Euler schemes, the first-order-in-time exponential scheme retains only the values of these mean fields at the initial particle location. Note that the same limitation exists even for the second-order scheme, which is based on an extension of prediction-correction steps and retains only two values of the mean fields (at the initial and at the predicted particle positions). In the case of strong gradients of these mean fields, this leads to a spatial

discretization error that reduces the overall accuracy of the numerical scheme and of the statistics of interest extracted from the particle dynamics simulation. New methods are therefore required to couple particle tracking with the time-integration numerical scheme in complex geometries and non-uniform flows.

In this context, the present objective is to develop a new particle-tracking algorithm to reduce the spatial discretization error of the exponential scheme, especially for large time steps during which particles can cross several cells. The basic idea is to perform a cell-to-cell integration by introducing sub-time steps corresponding to the particle residence times in the cells that each particle crosses. This allows to update the mean fields and local timescales dynamically along particle trajectories during each time step. In essence, if a P_0 interpolation is retained (in which mean fields are considered as constant within a cell), this new time-integration method is meant to retrieve the exact solution when the exponential numerical scheme is applied. However, although this idea of using particle-attached local sub-time steps may seem rather straightforward and directly applicable to each particle trajectory, the estimation of the residence time of a particle within a given cell is actually a tricky question due to the stochastic nature of the dynamical models used to construct its trajectory. The estimations of the residence time have indeed to respect Itô's definition of stochastic integrals. This corresponds to the non-anticipating requirement, which is satisfied by the new algorithm detailed in this paper. Note that the idea of cell-to-cell integration was already proposed by [36] but was only tested with prescribed velocity fields given on structured meshes (therefore without explicit velocity stochastic models). Hence, this study can be seen as an extension of this work in the context of stochastic models (i.e., addressing the non-anticipating requirement) and for unstructured meshes.

The paper is organized as follows. The theoretical background on Lagrangian PDF, or particle stochastic, models is first recalled in Section 2, in which the limit case of fluid particles and the specific Langevin model retained are presented. In Section 3, an overview of the current numerical scheme is given to bring out the spatial discretization error. The new algorithm is then detailed in Section 4 and validated on two test cases in Section 5. Conclusions are then drawn in Section 6.

2 Theoretical background on stochastic models for turbulent flows

2.1 PDF models for turbulent flows

Disperse turbulent two-phase flows involve discrete elements, or 'particles', transported by turbulent fluid flows. For the sake of simplicity, we focus here on particle dynamics while leaving out thermal effects, changes in the particle radius or mass, and particle collision effects. To model such a system, a suitable modeling framework consists in adopting a Lagrangian stochastic point of view. In this approach, a system of stochastic differential equations (SDEs) is written to describe the evolution in time of the particle-attached variables making up the particle state-vector $\mathbf{Z}_p$. Here, the state vector is taken equal to $\mathbf{Z}_p = (\mathbf{X}_p, \mathbf{U}_p, \mathbf{U}_s)$, with $\mathbf{X}_p$ the particle position, $\mathbf{U}_p$ its velocity and $\mathbf{U}_s$ the velocity of the fluid seen. The system of SDEs is

$$\mathrm{d}X_{p,i} = U_{p,i}\,\mathrm{d}t\ , \tag{1a}$$

$$\mathrm{d}U_{p,i} = \frac{U_{s,i} - U_{p,i}}{\tau_p}\,\mathrm{d}t + A_{p,i}\,\mathrm{d}t\ , \tag{1b}$$

$$\mathrm{d}U_{s,i} = (\text{stochastic model})\ . \tag{1c}$$

In Eqs. (1), the velocity of the fluid seen $\mathbf{U}_s$ is defined as the instantaneous fluid velocity sampled at the particle location, $\mathbf{U}_s(t) = \mathbf{U}(t, \mathbf{X}_p(t))$ where $\mathbf{U}(t, \mathbf{x})$ is the fluid velocity field. Note that, for discrete particles in non-fully-resolved turbulent flows, the velocity of the fluid seen cannot be obtained from the reduced information available on the fluid velocity field (typically, its first and second moments). This means that $\mathbf{U}_s$ needs to be introduced as a separate particle-attached variable (see comprehensive discussions in [21, 16, 17]). On the right-hand side (RHS) of Eq. (1b), it is seen that $\mathbf{U}_s$ enters the expression of the drag force while the term $\mathbf{A}_p$ contains possible additional forces acting on discrete particles (such as gravity). This drag force is expressed in terms of the particle relaxation timescale τ_p, defined as

$$\tau_p = \frac{\rho_p}{\rho}\frac{4\,d_p}{3\,C_D|\,\mathbf{U}_s - \mathbf{U}_p\,|}\ , \tag{2}$$

with ρ and ρ_p the fluid and particle densities, respectively, d_p the particle diameter and C_D the drag coefficient. This timescale represents the typical time over which particle velocities adjust to the local fluid velocity seen

and is a measure of particle inertia. In the Stokes regime, valid when $\mathrm{Re}_p \leq 1$ (with Re_p the particle Reynolds number defined by $\mathrm{Re}_p = |\,\mathbf{U}_r\,|\,d_p/\nu$, with $\mathbf{U}_r = \mathbf{U}_s - \mathbf{U}_p$ and ν the fluid kinematic molecular viscosity), the drag coefficient is $C_D = 24/\mathrm{Re}_p$. In that case, the particle relaxation timescale is given by the Stokes formula:

$$\tau_p = \frac{\rho_p}{\rho}\frac{d_p^2}{18\nu_f}. \tag{3}$$

For general values of Re_p, the drag coefficient is obtained through empirical correlations such as [5, 3]

$$C_D = \begin{cases} \dfrac{24}{\mathrm{Re}_p}\left[\,1 + 0.15\mathrm{Re}_p^{0.687}\,\right] & \text{if } \mathrm{Re}_p \leq 1000, \\ 0.44 & \text{if } \mathrm{Re}_p \geq 1000. \end{cases} \tag{4}$$

It is worth recalling that stochastic particles, modeled by equations such as Eqs. (1), must be regarded as samples of a corresponding PDF and, consequently, the description can also be referred to as a two-phase PDF model [21, 17, 20]. In the present PDF framework, we are basically handling a two-time one-particle Lagrangian PDF, from which a one-time one-point Eulerian PDF is derived, allowing moment equations to be extracted in a straightforward manner [21, 17, 32]. This means that once a stochastic model, such as the one in Eqs. (1), is written, we obtain directly the corresponding expressions of the discrete Lagrangian $F^L_{p,N}$ and resulting Eulerian $F^E_{p,N}$ mass density functions (MDF) through

$$F^L_{p,N}(t\,;\,\mathbf{Y}_p,\mathbf{V}_p,\mathbf{V}_s) = \sum_{i=1}^{N} m_p^{(i)}\,\delta(\mathbf{Y}_p - \mathbf{X}_p^{(i)}(t))\,\delta(\mathbf{V}_p - \mathbf{U}_p^{(i)}(t))\,\delta(\mathbf{V}_s - \mathbf{U}_s^{(i)}(t))\ , \tag{5}$$

$$F^E_{p,N}(t,\mathbf{x}\,;\,\mathbf{V}_p,\mathbf{V}_s) = F^L_{p,N}(t\,;\,\mathbf{Y}_p = \mathbf{x},\mathbf{V}_p,\mathbf{V}_s)\ , \tag{6}$$

where m_p is the particle mass, N is the number of stochastic particles in the domain and $\mathbf{Y}_p$, $\mathbf{V}_p$ and $\mathbf{V}_s$ the sample space values corresponding to the random variables $\mathbf{X}_p(t)$, $\mathbf{U}_p(t)$ and $\mathbf{U}_s(t)$, respectively. It is then straightforward to derive the field values for the average $\langle H_p\rangle(t,\mathbf{x})$ of any particle variable $H_p(t;\mathbf{V}_p,\mathbf{V}_s)$,

$$\alpha_p(t,\mathbf{x})\,\rho_p\langle H_p\rangle(t,\mathbf{x}) = \int H_p(t;\mathbf{V}_p,\mathbf{V}_s)F_p^E(t,\mathbf{x};\mathbf{V}_p,\mathbf{V}_s)\,d\mathbf{V}_p\,d\mathbf{V}_s\ , \tag{7}$$

where $\alpha_p(t,\mathbf{x})$ is the mean particle volumetric fraction defined through a proper probabilistic normalization constraint [21, 30]. In a numerical simulation, these theoretical expressions imply that, in a small volume around a given location $\mathbf{x}$, mean values are estimated as the ensemble averages over the $N^p_{\mathbf{x}}$ particles present in that volume, or as Favre averages (or mass-weighted averages)

$$\langle H_p\rangle(t,\mathbf{x}) \simeq \frac{\displaystyle\sum_{i=1}^{N^p_{\mathbf{x}}} m_p^{(i)} H_p(t;\mathbf{V}_p^{(i)}(t),\mathbf{V}_s^{(i)}(t))}{\displaystyle\sum_{i=1}^{N^p_{\mathbf{x}}} m_p^{(i)}}\ . \tag{8}$$

For reasons explained elsewhere [21], the velocity of the fluid seen is generally represented by a Langevin model, which takes the following form (see extensive accounts in [21, 23, 16, 20])

$$\mathrm{d}U_{s,i} = -\frac{1}{\rho}\frac{\partial\langle P\rangle}{\partial x_i}\,\mathrm{d}t + \left(\langle U_{p,j}\rangle - \langle U_j\rangle\right)\frac{\partial\langle U_i\rangle}{\partial x_j}\mathrm{d}t + G^*_{ij}\left(U_{s,j} - \langle U_j\rangle\right)\mathrm{d}t + B^*_{ij}\,\mathrm{d}W_j\ . \tag{9}$$

where P is the fluid pressure and $\mathrm{d}\mathbf{W}$ the increments of a Wiener process (see more details in Section 2.2). In Eq. (9), the expressions retained to close the matrices G^*_{ij} and B^*_{ij} can be intricate and we refer to the relevant literature for their complete formulas [21, 16]. As explained in these works, the stochastic model for $\mathbf{U}_s$ is built as an extension of the ones used to simulate fluid particles. Starting from the general disperse two-phase model in Eqs. (1), the fluid limit case (this is also referred to as the tracer-particle limit) is indeed retrieved by considering the limit of vanishing particle timescale τ_p.

In the present work, we essentially focus on particle tracking for general non-homogeneous flows in complex geometries, and the emphasis is therefore on particle locations $\mathbf{X}_p$. To bring out the essential elements of the new algorithm introduced in Section 4, the stochastic model retained to model particle velocity is chosen

as simple as possible to avoid handling cumbersome expressions in the developments to follow. For these reasons, from now on, we consider only the limit case of fluid particles in turbulent flows and we choose to select the simplest possible Langevin model for the velocity of these fluid particles. However, the general context recalled in this section is meant to indicate that the new algorithm is indeed applicable to a much larger class of stochastic formulations than the one retained in the rest of the paper.

PDF models for single-phase turbulent flows As just indicated, the dynamic of fluid particles is based on the simplest Langevin model, the simplified Langevin model (SLM) [32, 9]. The particle state vector is now $\mathbf{Z} = (\mathbf{X}, \mathbf{U})$ and the evolution equations write

$$\mathrm{d}X_i = U_i \mathrm{d}t, \tag{10a}$$

$$\mathrm{d}U_i = -\frac{1}{\rho}\frac{\partial \langle P \rangle}{\partial x_i}\left(t, \mathbf{X}(t)\right)\mathrm{d}t - \frac{U_i - \langle U_i \rangle(t, \mathbf{X}(t))}{T_L(t, \mathbf{X}(t))}\mathrm{d}t + \sqrt{C_0 \epsilon(t, \mathbf{X}(t))}\,\mathrm{d}W_i \ . \tag{10b}$$

The RHS of Eq. (10b) involves several mean field values, including the turbulent dissipation rate ϵ , the fluid turbulent kinetic energy k and the Lagrangian integral time scale T_L. All these mean fields are to be evaluated locally in the vicinity of the particle, hence the local dependence that is explicitly written. The Lagrangian integral time scale is defined as:

$$T_L = \left(\frac{1}{\frac{1}{2} + \frac{3}{4}C_0}\right)\frac{k}{\epsilon} \ , \tag{11}$$

with C_0 the so-called Kolmogorov constant. The PDF formalism, outlined above, can then be followed to reveal that, in terms of resulting statistics for the fluid velocity, the SLM model corresponds directly to a second-order turbulence model, namely the Rotta model [35, 32]. Furthermore, we are considering only incompressible flows. Hence, since we associate the same mass δm to each fluid particles, this means that the particle concentration must remain uniform to respect the incompressible condition.

2.2 A reminder on stochastic integrals

The SLM for fluid particles is retained as the model of reference in this work, although, as indicated above, the numerical developments to follow in Section 4 can be extended to the case of dispersed turbulent two-phase flows by using the extended numerical integration scheme presented in [29]. Present ideas can also be applied to more general stochastic models where the particle state vector does include particle location and velocity as well as extra variables $\mathbf{Z} = (\mathbf{X}, \mathbf{U}, \mathbf{Z}')$, where $\mathbf{Z}'$ stands for additional variables attached to each stochastic particle to address more complex physical issue (in the rest of the paper, we have $\mathbf{Z}' = \varnothing$). Furthermore, other stochastic models than Langevin ones can be considered for the modeled particle velocity equation but we assume that it takes the form of a general stochastic diffusion process, implying white-noise terms (note that, in that case, the numerical scheme for time-integration has then to be adapted, see Section 4.2.1).

However, a few words are in order to recall the definition of stochastic integrals and the important issue of non-anticipating formulations. To that effect, we consider a general stochastic diffusion process for a particle state vector $\mathbf{Z} = (Z_i)$ with the form

$$\mathrm{d}Z_i = \underbrace{A_i\left(t, \mathbf{Z}, \langle \mathcal{F}(t, \mathbf{Z}) \rangle\right)\mathrm{d}t}_{\text{deterministic Part}} + \underbrace{B_{ij}\left(t, \mathbf{Z}, \langle \mathcal{G}(t, \mathbf{Z}) \rangle\right)\mathrm{d}W_j}_{\text{random Part}} \ . \tag{12}$$

In Eq. (12), $\mathbf{A} = (A_i)$ and $\mathbf{B} = (B_{ij})$ stand for the drift vector and diffusion matrix, respectively. They are written with the most general possible form, involving a possible dependence on mean values represented by the two functional terms, namely $\mathcal{F}$ and $\mathcal{G}$, which makes Eq. (12) a stochastic diffusion process of the McKean type (see [16, 17]). The Wiener process $\mathbf{W}$ is a continuous time Markov process whose increments follow independent Gaussian centered distributions of variance $\mathrm{d}t$ [28, 38]. Although continuous, its trajectories are highly erratic and even of infinite total variance in any interval (see the illustration in Figure 1), implying that classical integration rules cannot be applied.

The formulation of the SDE in Eq. (12) is actually a short-hand notation for the proper mathematical expression, which is its integrated version

$$Z_i(t) = Z_i(t_0) + \int_{t_0}^{t} A_i(s, \mathbf{Z}(s), \mathcal{F}(s, \mathbf{Z}(s)))\,\mathrm{d}s + \underbrace{\int_{t_0}^{t} B_{ij}(s, \mathbf{Z}(s), \mathcal{G}(s, \mathbf{Z}(s)))\,\mathrm{d}W_j(s)}_{\text{stochastic integral}} \ , \tag{13}$$

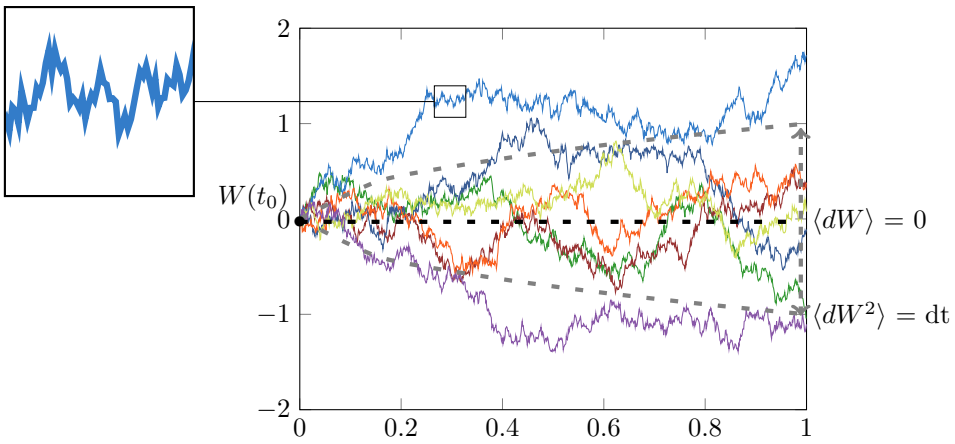


Figure 1: Some realizations of a conditional Wiener process, starting from 0 at time $t = 0$.

where the stochastic integral must be carefully defined. This is done by applying the Itô definition for non-anticipating processes. Using a partition $[t_k\,;\,t_{k+1}]$, $k = 1, \ldots, N$, of the interval $[t_0\,;\,t]$, the stochastic integral in the Itô sense is then obtained as

$$\int_{t0}^{t} B_{ij}(s, \mathbf{Z}(s), \mathcal{F}(s, \mathbf{Z}(s)))\,\mathrm{d}W_s = \text{ms-}\lim_{N\to\infty} \sum_{k=1}^{N} B_{ij}(t_k, \mathbf{Z}(t_k), \mathcal{F}(t_k, \mathbf{Z}(t_k)))\,(W_j(t_{k+1}) - W_j(t_k)) \ , \qquad (14)$$

where the limit must be understood as a limit in the mean-square sense (since a convergence trajectory by trajectory is not possible). The choice of this definition allows two fundamental properties of stochastic integrals to be always respected for any smooth-enough functions Ψ_1 and Ψ_2 (vector indexes and functional dependencies are left out here, for the sake of keeping simple notations):

- $\left\langle \int_{t_0}^{t} \Psi_1(\mathbf{Z}(s))\mathrm{d}W(s) \right\rangle = 0$,
- $\left\langle \left(\int_{t_0}^{t_2} \Psi_1(\mathbf{Z}(u))\mathrm{d}W(u) \right) \left(\int_{t_1}^{t_3} \Psi_2(\mathbf{Z}(v))\mathrm{d}W(v) \right) \right\rangle = \int_{t_1}^{t2} \langle \Psi_1(\mathbf{Z}(s))\Psi_2(\mathbf{Z}(s)) \rangle \mathrm{d}s$, $t_0 \le t_1 \le t_2 \le t_3$.

A key consequence of the non-anticipation nature of the definition of the stochastic integrals is that the mean conditional increment of the stochastic process $\mathbf{Z}$ in Eq. (12) over a small time increment Δt is governed only by the drift term, which means that we have

$$\langle \Delta Z_i \,|\, \mathbf{Z}(t_0) = \mathbf{z}_0 \rangle = A_i(t_0, \mathbf{z}_0, \mathcal{F}(t_0, \mathbf{z}_0))\,\Delta t \ . \qquad (15)$$

These relations play a central role in the development of the new algorithm in Section 4.

3 Current numerical algorithm

To obtain numerical solutions of the SDEs in Eqs. (10), three issues need to be addressed:

1) How do we calculate the mean fields at particle positions?

2) How do we integrate in time the SDEs?

3) How do we extract statistics from the ensemble of particles?

These three separate points are related to three sources of error: the first one corresponds to the spatial discretization error due to an approximate expression of mean fields at particle locations; the second one corresponds to the time error due to the integration scheme selected to update particle variables at discrete times; while the third one is related to the statistical error which is inherent in Monte Carlo methods. In a

complete numerical formulation, these errors can impact each other and can even induce a fourth one referred to as the bias error. However, for reasons set forth below, this bias error is not present in the present context and we can safely concentrate on the three ones mentioned above.

Although relatively few studies have been devoted to their numerical analysis, these questions have been analyzed in depth in at least two detailed investigations [42, 29], to which readers are referred to for further details. In this section, we limit ourselves to recalling the key aspects of the current algorithm putting the emphasis on the spatial discretization error so as to pave the way for the developments described in Section 4.

3.1 Hybrid FV/PDF approach and particle-mesh implementation

When dealing with fluid particles, a first possibility is to extract the mean fields entering Eqs. (10) directly from the particle simulation which are then fed back into the computation. This can be done using either an auxiliary grid to compute such statistics from the ensemble of particles or by resorting to a local kernel estimation centered around each particle position. Such numerical formulations are called PDF stand-alone codes and one interest is that they are consistent by construction while, on the other hand, the inherent statistical noise due to the use of a finite number of particles can induce a deterministic bias error [42, 29]. In the present study, we have, however, chosen to rely on a hybrid formulation in which the particle simulation is coupled to a classical FV (finite volume) solver, thereby leading to so-called hybrid FV/PDF methods. One interest of such formulations is that the mean fields provided to the particle solver are now free of statistical noise, thereby avoiding bias errors. On the other hand, this hybrid formulation raises immediately a consistency issue for fluid particles since, for example, the mean fluid velocity field is simulated twice (in the FV solver but also as the mean value of fluid particles located in a given small volume). As discussed in several works [16, 4], it is then important to couple FV and particle solvers that correspond to the same turbulence model. For instance, with the present simple Langevin model retained in Eqs. (10), the turbulence model within the FV solver must be the Rotta model with a constant consistent with the value of C_0 in Eq. (10b). These points have been discussed in detail in several papers [35, 16, 32, 4] and are not repeated here. A further reason behind the choice of such hybrid formulations is that, as pointed out in Section 2, we are interested in extending present developments to the case of discrete particles where, for example, the fluid mean velocity field cannot be extracted from the particle simulation (the statistics of the velocity of the fluid seen being different from the fluid ones [21, 16, 17]) and, in which, such hybrid formulations are of specific interest. Note that, although not addressed in the present work, the introduction of an underlying mesh in the particle solver is also useful to simulate particle collision effects [39, 40].

In practice, as displayed in Figure 2, the hybrid FV/PDF coupled formulation is translated into a particle/mesh simulation in which, during each iteration, the FV solver is first run and then provides the needed mean fields on a grid to the particle solver. The difficulty in such hybrid formulations is that the mean fields in the FV solver are known at determined spatial locations (typically at the center of each cell of the mesh for collocated discretizations). However, fluid tracers in the particle solver are continuously distributed within the domain and very rarely correspond to the cell center locations. We are then faced with an additional issue which is to determine how mean-field values are interpolated from the mesh to the particles. In the present study, we have retained the simplest possibility, which assigns the same cell-centered mean values to all particles located within that cell: this corresponds to a P_0 interpolation at particle positions. While this implies discontinuous profiles of mean quantities when a particle moves across an interface between two adjacent cells, this drawback is offset by the simplicity of the numerical implementation and, in particular, by the fact that it remains valid and easy to apply even in the case of unstructured meshes (sometimes made up by several overlapping meshes) in complex geometries for which more advanced methods, such a linear interpolation between neighboring cell centers quickly become very intricate and nearly intractable.

Based on this hybrid formulation and using the P_0 interpolation assumption, the particle equations, cf Eqs. (10), can be expressed as:

$$\mathrm{d}X_i = U_i \,\mathrm{d}t, \tag{16a}$$

$$\mathrm{d}U_i = -\frac{1}{\rho}\left[\frac{\partial \langle P\rangle}{\partial x_i}\right]_0 (t,\mathbf{X}(t))\mathrm{d}t - \frac{U_i - [\langle U_i\rangle]_0\,(t,\mathbf{X}(t))}{[T_L]_0\,(t,\mathbf{X}(t))}\mathrm{d}t + \sqrt{C_0\,[\epsilon]_0\,(t,\mathbf{X}(t))}\mathrm{d}W_i\ , \tag{16b}$$

where $[.]_0$ stands for the P_0 interpolation and will be omitted in the following for the sake of clarity. In this hybrid FV/PDF formulation, the same grid is used to extract particle statistics from the subset of particles present in a given cell at a given time. This corresponds to the nearest-grid-point (NGP) method in which

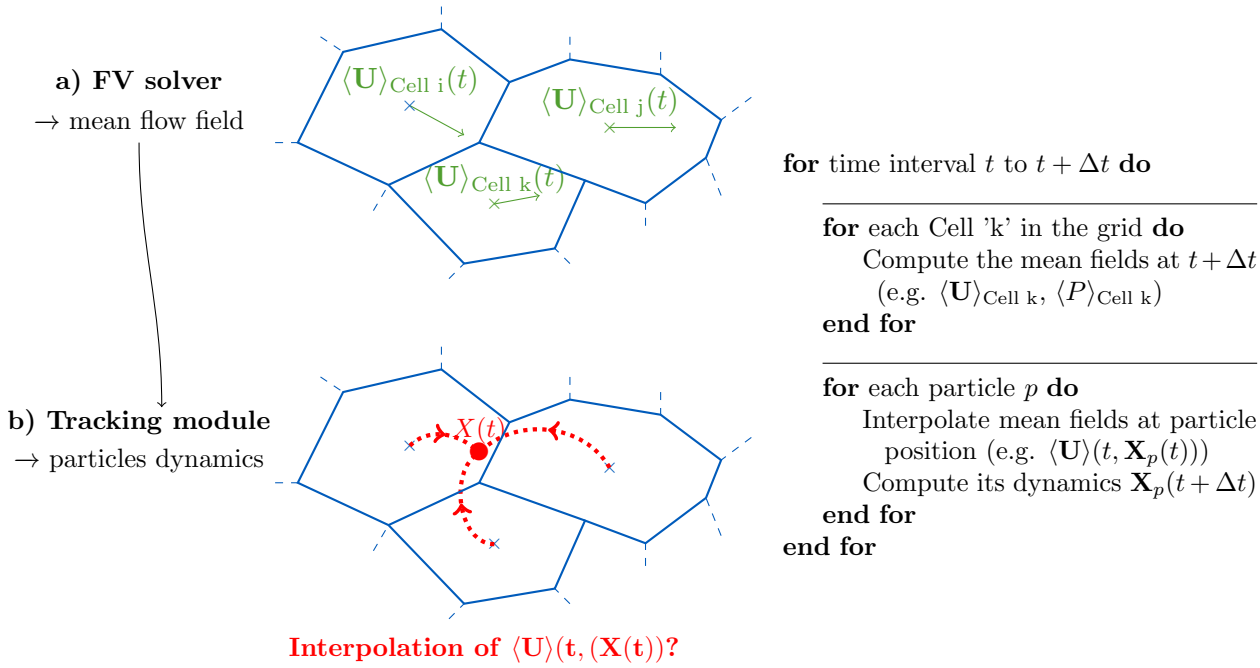


Figure 2: Sketch illustrating the principle of hybrid FV/PDF approaches: for each time step, the FV solver is first run to provide the needed mean fields on a grid and, then, the particle dynamics is solved with a PDF approach which implies to have an interpolation method to determine the mean field values at the particle position.

all the statistical weight associated to each particle is associated to the cell it belongs to, regardless of its position in that cell. This NGP method is actually consistent with the P_0 interpolation used to provide information from the grid to the particles and appears as its reverse operation, that is from the particles to the grid (see discussions in [42, 29]).

At this stage, a few comments are in order with respect to the issue of interpolating mean fields at particle positions. In stand-alone PDF models implemented as particle-mesh methods, the interpolation scheme and how particle statistics are extracted appear as two adjoint operators [42, 29]. This means that if, for example, a linear interpolation scheme is used to obtain interpolated mean-field values, then a cloud-in-cell (CIC) method must be selected to derive statistics from the particle set. In structured meshes, this corresponds to a classical approach [12]. However, the implementation of such a consistent method becomes quickly intricate in unstructured meshes. In present hybrid FV/PDF formulations for fluid particles, these two operators are decoupled since particle statistics are not injected in the stochastic model for particle dynamics. It is then tempting to resort to more precise interpolation schemes than P_0, which are likely to contribute to reducing the overall spatial discretization error. Nevertheless, such schemes remain delicate to implement in unstructured meshes. Furthermore, considering higher-order interpolations would essentially be beneficial if they can be applied in the different cells crossed by particles during one time step. In that sense, the interpolation issue appears as complementary to the development of cell-to-cell integration schemes. In the following, we want to concentrate on building such a cell-to-cell integration method. Therefore, we have chosen to retain the simplest case, namely the P_0 interpolation scheme for the sake of simplicity.

Therefore, questions (1) and (3) have been addressed and we can turn our attention to the question (2) and how time integration of the SDEs is performed.

3.2 Current numerical scheme for particle transport

Once mean fields at particle location have been determined, the next step consists in devising a numerical scheme to predict the fluid particle properties making up the particle state vector $\mathbf{Z} = (\mathbf{X}, \mathbf{U})$. This typically introduces the notion of a time step Δt for the integration of the particle equations of motion and discrete approximations of $\mathbf{Z}$ at times $t^n = n\Delta t$, noted $\mathbf{Z}^n$. These discrete approximations $\mathbf{Z}^n$ are obtained by

successive updates (i.e., computing $\mathbf{Z}^{n+1}$ from $\mathbf{Z}^n$), using an integrated form of the RHS of Eqs. (16). In the frame of present hybrid methods, the current algorithm involves a two-step process:

- A time-integration scheme, which allows to obtain $\mathbf{Z}^{n+1}$ from $\mathbf{Z}^n$ using known values of the mean fields at particle location entering the SDEs in Eqs. (16);
- A trajectory algorithm, which determines the location of each particle in the spatial domain. This second step is needed in hybrid FV/PDF approaches since we must know in which cell each particle is with respect to the mesh defined for the computation of the fluid phase. This information is necessary in order to update the values of the mean fields at each particle location for the next step of the time-integration process.

In the following, we briefly recall the details of the current algorithm used for each step.

3.2.1 Time-integration scheme to predict the particle state vector

The numerical scheme used to integrate the SDEs, cf. Eqs. (16), has been described in previous papers [29, 22], which provide comprehensive information on its derivation and main characteristics (see, in particular, the complete description of the different steps leading to its construction in [29]). Therefore, only the salient aspects are recalled here along with the resulting formulation.

Requirements The time-integration scheme has been developed according to the following guidelines (more details can be found in [29, 22]):

- The numerical scheme is explicit (for simplicity reasons);
- The numerical scheme is unconditionally stable (this is of key interest when the time step cannot be reduced, as in hybrid methods, to respect potential stability criteria);
- The numerical scheme corresponds to the exact solution when the mean fields and timescales entering the modeled equations are constant;
- The numerical scheme should capture the correct physical behaviors in the limit cases when the time step becomes much larger than the physical timescales [29].

The main reason behind these requirements is that the time step used in hybrid FV-PDF formulations is generally the same for the fluid solver and for the particle solver. This is mandatory for unsteady flows, where fluid and particle properties must be obtained at the same time. Consequently, since the time step is usually imposed by the fluid solver (to properly compute the flow field on a given mesh), it cannot always be reduced/adapted for the particle solver (as could be done in PDF stand-alone approaches). In complex flows in which mean fluid velocities can drastically vary from one area to another one, this means that we do not control the number of cells that are crossed by particles during a time step. Furthermore, when the fluid timescales are also widely different, it is important to properly capture the diffusive limit rather than limiting the time step for the whole particle set [29].

Chosen scheme The guiding principle underlying the present time-integration scheme is to express first the integrated form of the SDEs (using an assumption of constant properties) which leads directly to an exponential analytical form. In the general case, when mean fields and timescales are not constant but vary in space and time, the idea is to use Euler-like schemes by freezing their values at the beginning of the time step in the integrated form rather than in the SDEs (see a detailed description in [29]). Thanks to the formulation with exponential factors, unconditional stability is then automatically guaranteed. Although second-order scheme are quite possible using the same approach (see [29]), we limit ourselves to a first-order formulation in the following since our main concern is about the prediction of relevant mean fields in such

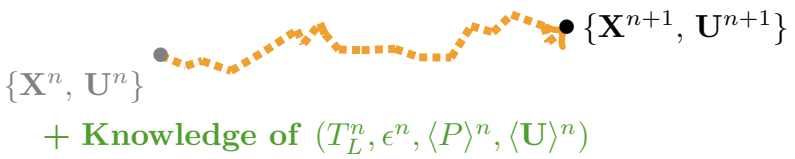


Figure 3: Sketch illustrating the time-integration with the current numerical scheme: the particle position X_i^{n+1} and velocity U_i^{n+1} at the next iteration are computed using information on the particle position and velocity at the previous iteration n as well as values of the fluid characteristics at the position X_i^n.

schemes. Consequently, the resulting discrete numerical scheme for fluid particles writes:

$$X_i^{n+1} = X_i^n + U_i^n T_L^n \left(1 - \exp\left(-\frac{\Delta t}{T_L^n}\right)\right) + C_i^n T_L^n \left(\Delta t - T_L^n \left(1 - \exp\left(-\frac{\Delta t}{T_L^n}\right)\right)\right) + \underbrace{\sqrt{\frac{C_0 \epsilon^n (T_L^n)^3}{2}} \frac{\left(1 - \exp\left(-\frac{\Delta t}{T_L^n}\right)\right)^2}{\sqrt{1 - \exp\left(-2\frac{\Delta t}{T_L^n}\right)}} \zeta_i^U + \sqrt{C_0 \epsilon^n (T_L^n)^2 \left(\Delta t - 2T_L^n \frac{1 - \exp\left(-\frac{\Delta t}{T_L^n}\right)}{1 + \exp\left(-\frac{\Delta t}{T_L^n}\right)}\right)} \zeta_i^X}_{I_i^X(\Delta t)}, \tag{17a}$$

$$U_i^{n+1} = U_i^n \exp\left(-\frac{\Delta t}{T_L^n}\right) + C_i^n T_L^n \left(1 - \exp\left(-\frac{\Delta t}{T_L^n}\right)\right) + \underbrace{\sqrt{C_0 \epsilon^n \frac{T_L^n}{2} \left(1 - \exp\left(-2\frac{\Delta t}{T_L^n}\right)\right)} \zeta_i^U}_{I_i^U(\Delta t)}. \tag{17b}$$

The two terms ζ_i^X and ζ_i^U correspond to independent random numbers sampled in a normalized Gaussian distribution $\mathcal{N}(0,1)$. As indicated in Eqs. (17), these random numbers intervene in the Choleski decomposition of the two correlated stochastic integrals, namely $I_i^U(\Delta t)$ and $I_i^X(\Delta t)$ (see descriptions in [29]).

As displayed in Figure 3, the time-integration scheme predicts the particle state vector at the next time step $(\mathbf{X}^{n+1}, \mathbf{U}^{n+1})$ using information on its values at the beginning of the time step $(\mathbf{X}^n, \mathbf{U}^n)$ and local fluid characteristics at the position $\mathbf{X}^n$.

3.2.2 Trajectory algorithm for spatial location

The time-integration scheme requires knowledge on the fluid mean fields at the particle position at a given time (e.g. , fluid velocity, turbulent dissipation rate), which are known on a given mesh. As a result, the time-integration scheme has to be supplemented by a trajectory algorithm to locate each particle and assign the corresponding cell.

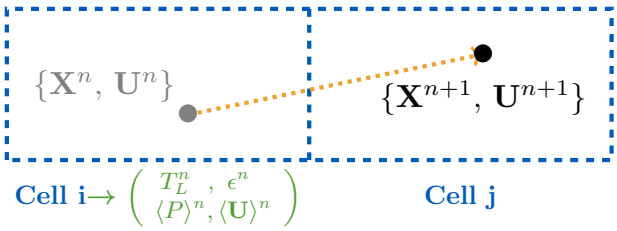
Assumptions The trajectory algorithm has to satisfy the following requirements:

- First, since we resort to a NGP approach, the trajectory algorithm is expected to provide the cell in which each particle is located.
- Second, the trajectory algorithm has to be applicable in the case of generic 3-D unstructured meshes, with cells being star-shaped around their center. This choice is motivated by the typical meshes used in the computation of the average flow-field.
- Third, the trajectory algorithm assumes a free-flight motion of particles during each time step. This means that, during each time step, particles are moving in a straight line from their initial position X_i^n to their final position X_i^{n+1}.

Chosen algorithm With respect to the previous requirements, we resort to an algorithm based on the successive neighbor searches. Such tracking algorithms determine the new cell inside which a particle is by using information only on its initial and final positions (see also Figure 4). The new cell is then determined by searching intersections between the trajectory vector (joining the initial and final position) and faces of the current cell. To do so, we rely on the Möller-Trumblemore algorithm [26] which has been developed to determine the intersections between a vector and triangular sub-faces. This process is then repeated to check if the final position is in the new cell or if another face is crossed by the trajectory vector. The details and validation of this tracking algorithm are provided in Appendix A.

Such tracking algorithms have been retained since they have been shown to be efficient for unstructured meshes (e.g. , for Finite Difference approaches in [15] or for Finite Volume approaches in [27, 41]). Moreover, in the case of unstructured meshes, these tracking algorithms are more adequate than simple locating algorithms (which determine the location only using the information on the final position).

At this point, it is worth mentioning that additional conditions can be taken into account for each of the faces that are crossed during the tracking algorithm. In particular, when the crossed face corresponds to a physical boundary, specific boundary conditions can be added: for instance, particles can be removed from the simulation when reaching outlet boundaries while boundary conditions can be applied to wall surfaces (interested readers are referred to [6, 25, 2]). Similarly, specific conditions can be added to properly treat periodicity by translation and/or periodicity by rotation.



for time interval n to $n+1$ **do**
  Integrate $\mathbf{X}^{n+1}$
  Draw vector $\delta\mathbf{X} = \mathbf{X}^{n+1} - \mathbf{X}^n$
  do
    Determine if $\delta\mathbf{X}$ leaves the cell through a face
    Update the current cell
    Increment the sub-iteration m
  while A face is crossed
  Track the final integrated position
end for

Figure 4: Current trajectory algorithm used after a single integration for the whole time step: the cell in which the particle resides at the end of the time step is tracked using a successive neighbor search algorithm.

4 A new algorithm based on cell-to-cell integration

In the context of present hybrid FV/PDF approaches, the P_0-interpolation assumption for mean-field values means that the exponential scheme described in Section 3.2 is not only stable but provides an exact solution for the particle state vector as long as particles remain in the same cell they started from during the whole time step. However, in such hybrid formulations, the time step is often imposed by the mean fluid flow computation and cannot always be reduced so as to respect this criterion. This is especially encountered in disperse two-phase flow simulations with discrete particles having inertia that can range over several orders of magnitude, as indicated above. In the present work, we limit ourselves to fluid particles but this general picture is an indication that we are faced with a situation where, during a given time step, a particle can cross several cells. The inherent assumption made in Euler schemes, and therefore also in the present exponential one, retains only mean-field values and timescales evaluated at the beginning of the time step, that is at the initial particle location. In non-uniform flows, with potential variations of mean flow field quantities, this implies that a spatial discretization error is introduced, as illustrated in Figure 5.

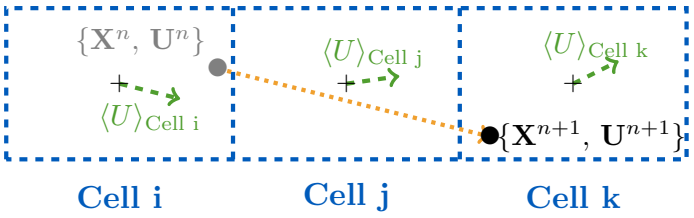


Figure 5: Sketch illustrating the issue with the current numerical scheme over large time steps in non-uniform flows: the particle can cross many cells in a single time step, meaning that the fluid characteristics (including velocities) in the intermediate cells are not taken into account in its trajectory.

Objectives The aim of this paper is to develop a new algorithm that remains accurate even for non-uniform flows in the case of large time steps when particles can cross several cells. For that purpose, we extend the numerical scheme described in Section 3.2 while keeping similar assumptions/requirements:

a) it should be explicit;

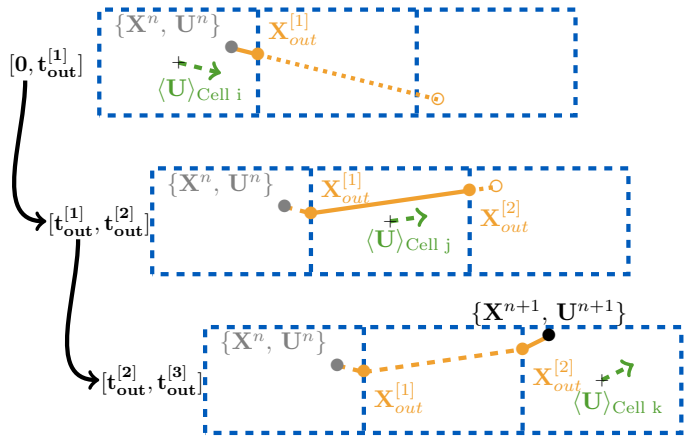
b) it should be unconditionally stable with respect to the time step;

c) it should be exact for constant flow mean fields (and thus within a given cell in line with the P_0 interpolation);

d) it should take into account spatial variations of flow mean fields encountered over large time steps.

4.1 Leading Principle: a cell-to-cell integration for large time steps

Principle To improve the numerical accuracy in non-uniform flows and in the case of large time steps, the idea is to extend the present exponential scheme with a cell-to-cell integration. As illustrated in Figure 6, the leading principle consists in splitting the time step in several sub-iterations: the integration is stopped every time the particle exits a cell, such that each sub-iteration corresponds to the motion of the particle within one cell. By doing so, it is straightforward to account for the change of flow characteristics every time a particle enters a new cell. Since we need to detect whenever a particle crosses an interface between two adjacent cells, the following two steps are applied for each sub-iteration : (1) a time-integration scheme and (2) a trajectory algorithm. When the time step is small enough for a particle to remain inside the same cell during a whole time step, the new scheme is the same as the one previously described.



for time interval n to $n+1$ **do**
  do
    Get relative intersection time $\theta_{[m+1]}$
    Integrate until the face
    Update cell and remaining time
    Increment sub-iteration m
  while A face is crossed
end for

Figure 6: Sketch illustrating the principle of a cell-to-cell integration for large time steps and the corresponding algorithm.

Additional points Compared to the two-step process described in Section 3.2, the new cell-to-cell integration brings out two additional questions:

- When does a particle exit a cell?
- Where does a particle exit a cell?

This means that the trajectory algorithm has to be modified since it only provides information on the new cell (simple tracking). The new trajectory algorithm should provide information on the exit time t_{out} and exit location $\mathbf{X}_{\text{out}}$. These two additional pieces of information are indeed required to compute the motion of the particle during the remaining part of the time step. In fact, as displayed in Figure 6, the motion of the particle after crossing the face is given by a time-integration scheme starting at $\mathbf{X}_{\text{out}}$ over the remaining time $\Delta t - t_{\text{out}}$. In that sense, the present algorithm draws on the cell-to-cell integration proposed by [36] while extending it to respect the non-anticipating requirement for stochastic integrals in the Itô sense (this is discussed in the next paragraph).

The key issue of non-anticipating estimations Implementing the principle of the cell-to-cell algorithm may seem straightforward. A naive formulation would indeed consider that, starting from a particle position $\mathbf{X}^n$ at time t^n, we apply the exponential scheme, cf. Eqs. (17), to predict $\mathbf{X}^{n+1}$ at time t^{n+1} and, then, use the trajectory algorithm to determine $\mathbf{X}_{\text{out}}$ and the exit time of that cell t_{out} (see the top figure in Figure 6). However, whatever the geometrical method used to determine these quantities, it is clear that the resulting time t_{out} will depend on the random numbers $\boldsymbol{\zeta}$ which, in Eqs. (17), represent the normalized increments of the Wiener process in the stochastic model, cf. Eqs. (10). When combining the different diffusion coefficients obtained from each sub-iteration (for instance, the three sub-iterations depicted in Figure 6), the resulting diffusion coefficient will then be a function of the future of the Wiener process since the random numbers $\boldsymbol{\zeta}$ stand for the normalized values of $\mathbf{W}^{n+1} - \mathbf{W}^n$. In other words, the misleadingly simple approach will yield diffusion coefficients that anticipate on the future of the Wiener process. This is in direct violation of the very definition of the stochastic integrals in the Itô sense, thereby inducing spurious overall drift and diffusion numerical values (see a similar analysis in [19]).

Assumptions Drawing on the aforementioned analysis, the new numerical scheme is designed so as to respect the following additional assumptions/constraints:

i) It should respect the non-anticipation rule with respect to the Wiener process;

ii) The trajectory algorithm is based on a neighbor search;

iii) The trajectory algorithm considers free-flight motion of particles within each sub-iteration.

4.2 Non-anticipating estimations of intermediate time steps using virtual partners

To obtain an estimate of the fraction of time spent by a particle in a given cell that remains independent of the future of the Wiener process entering the stochastic terms, a virtual partner is associated to each particle within each time step of the computation. This means that at the beginning of each time step $t = t^n$, this virtual partner starts from the same location as the real particle which is considered but moves in a deterministic manner (to be precised below) based only on the particle variables at time t^n and local mean-field values. The fraction of time spent by this virtual partner in the corresponding cell is therefore, by construction, free of any statistical dependence with the Wiener process driving the random terms and provides an estimate for the time spent by the real particle within that cell.

In the following, details about the algorithm are first provided in Section 4.2.1. The consistence of this algorithm with the current integration scheme is then demonstrated in Section 4.2.2.

4.2.1 Overview of the new algorithm

As indicated, the new algorithm is based on a cell-to-cell integration for large time steps. This implies that each time iteration is split in a series of sub-iterations, each one corresponding to the motion of a particle in a given cell. As a result, the algorithm combines a number of successive time-integration and trajectory steps. During each time step, the virtual partners are used in the trajectory steps to determine the cell in which the particle is as well as to estimate the exit time and location.

In practice, the algorithm is composed of the following steps, which are depicted in Figure 7:

step-1 Based on the particle initial location $\mathbf{X}^{[0]}$ and the knowledge of the fluid mean fields in that cell, a deterministic estimation of the particle position at the end of the time step Δt, noted $\widehat{\mathbf{X}^{[1]}}$, is made. Note that the superscript $[m]$ means that it corresponds to the sub-iteration number m within the cell-to-cell integration. This is achieved by forcing the stochastic integrals to zero in the exponential time-integration scheme, which gives the following equation for $\widehat{\mathbf{X}^{[1]}}$:

$$\widehat{\mathbf{X}^{[1]}} = \mathbf{X}^{[0]} + \mathbf{U}^{[0]} T_L^{[0]} \left\{ 1 - \exp\left(-\frac{\Delta t}{T_L^{[0]}} \right) \right\} + C_i^{[0]} T_L^{[0]} \left(\Delta t - T_L^{[0]} \left\{ 1 - \exp\left(-\frac{\Delta t}{T_L^{[0]}} \right) \right\} \right) \ . \qquad (18)$$

It is important to note that $\widehat{\mathbf{X}^{[1]}}$ corresponds to the mean conditional particle location at time Δt conditioned on the fact that its initial location is $\mathbf{X}^{[0]}$, that is $\widehat{\mathbf{X}^{[1]}} = \langle \mathbf{X}(\Delta t) \,|\, \mathbf{X}(0) = \mathbf{X}^{[0]} \rangle$. Therefore, if we release a number of particles at $\mathbf{X}^{[0]}$ at time $t = 0$, $\widehat{\mathbf{X}^{[1]}}$ represents the position of the center of mass of the released cloud at time $t = \Delta t$.

step-2 Then, the trajectory algorithm based on a free-flight assumption is applied to determine if the particle leaves/remains in the cell. To avoid any anticipation issue, the trajectory algorithm is applied on the virtual partner and not on the real particle. The motion of the virtual partner is assumed to follow a straight line between the first location of the virtual partner (initialized at the particle position $\widetilde{\mathbf{X}^{[0]}} = \mathbf{X}^{[0]}$) and $\widehat{\mathbf{X}^{[1]}}$. The trajectory algorithm is detailed in Appendix A. Its outcome allows to distinguish between two cases:

step-2a The virtual partner leaves the cell. In that case, the trajectory algorithm provides information on the face through which the virtual particle exits the cell but also on the exit time $\theta^{[1]}\Delta t$ (see the extension of the trajectory algorithm in Appendix A.1.2). Thanks to the free-flight assumption, the exit location of this virtual partner, $\widetilde{\mathbf{X}^{[1]}}$, is derived directly from the exit time. This case will activate a next sub-iteration to compute the motion of the particle during the remaining time (see Step-3a).

step-2b The virtual partner remains in the cell. In that case, $\theta^{[1]} = 1$, and the position of the virtual partner at the end of the time step is equal to the first estimated position $\widetilde{\mathbf{X}^{[1]}} = \widehat{\mathbf{X}^{[1]}}$. This case will not activate a new sub-iteration and the cell-to-cell integration will be stopped (see Step-3b).

step-3 The position $\mathbf{X}^{[1]}$ and velocity $\mathbf{U}^{[1]}$ of the particle at the end of this sub-iteration are then computed using the full time-integration scheme, i.e., including the stochastic integrals (see Eq. (17)). The only issue is to determine the amount of time that the particle has spent in the cell. From step-2, there are two possibilities:

step-3a If the virtual partner exits the cell (step-2a), the elapsed time retained for the prediction of ($\mathbf{X}^{[1]}$, $\mathbf{U}^{[1]}$) is taken as being equal to the fraction of time computed for the virtual partner $\theta^{[1]}\Delta t$.

Then, we have to pursue the computation of the particle motion during the remaining time $\Delta t^{[1]} = (1 - \theta^{[1]})\Delta t$. In the spirit of this cell-to-cell algorithm, this is achieved by repeating the previous three steps and updating the information required at each step (see sub-iterations 2 and 3 in Figure 7). In step-1 of the second sub-iteration, the mean conditional particle position $\widehat{\mathbf{X}^{[2]}}$ at the end of the new time step $\Delta t^{[1]}$ is estimated starting from the particle position $\mathbf{X}^{[1]}$ and particle velocity $\mathbf{U}^{[1]}$. The mean fields required to estimate $\widehat{\boldsymbol{X}}^{[2]}$ are now ($T_L(t, \widetilde{\boldsymbol{X}}^{[1]})$, $C_i(t, \widetilde{\boldsymbol{X}}^{[1]})$) taken in the cell in which the virtual partner is. Afterward, step-2 is applied to compute the trajectory of the virtual partner, assuming that it starts at the last exit location $\widetilde{\mathbf{X}^{[1]}}$ and moves towards the estimated mean conditional position $\widehat{\mathbf{X}^{[2]}}$. It is worth noting that the starting position is

taken as the last exit location so that the trajectory of the virtual partner remains continuous. Subsequently, from the estimation of θ_2 obtained in step-2, the position $\mathbf{X}^{[2]}$ and velocity $\mathbf{U}^{[2]}$ are computed in step-3 of this second sub-iteration.

This three-step process is actually repeated until the virtual partner reaches a cell in which it remains for the whole remaining time step $\Delta t^{[m]}$ (recall that the superscript $[m]$ corresponds to the sub-iteration number m).

step-3b If the virtual particle remains in the cell (step-2b), the elapsed time is actually equal to the remaining time step $\Delta t^{[m]}$ (i.e. $\theta^{[m]} = 1$). The position of the virtual partner is equal to the last estimated position made in step-1 $\widetilde{\mathbf{X}^{[m]}} = \widehat{\mathbf{X}^{[m]}}$. In that case, the cell-to-cell iterative integration is stopped.

Final step When the virtual partner finally reaches a cell in which it remains for the remaining time, we obtain the particle position at the end of the whole time step $\mathbf{X}(t + \Delta t)$. Yet, as can be seen in Figure 7, the particle might end up in a different cell than the virtual partner due to the stochastic terms. For that reason, we apply a last trajectory algorithm: we track the change of cell from the virtual partner last position $\widetilde{\mathbf{X}^{[m]}}$ to the particle position $\mathbf{X}(t + \Delta t)$. This last trajectory step ensures that the particle is associated to the correct cell inside which it resides at the end of the time step. This ensures that no error on the location is introduced for the next time step.

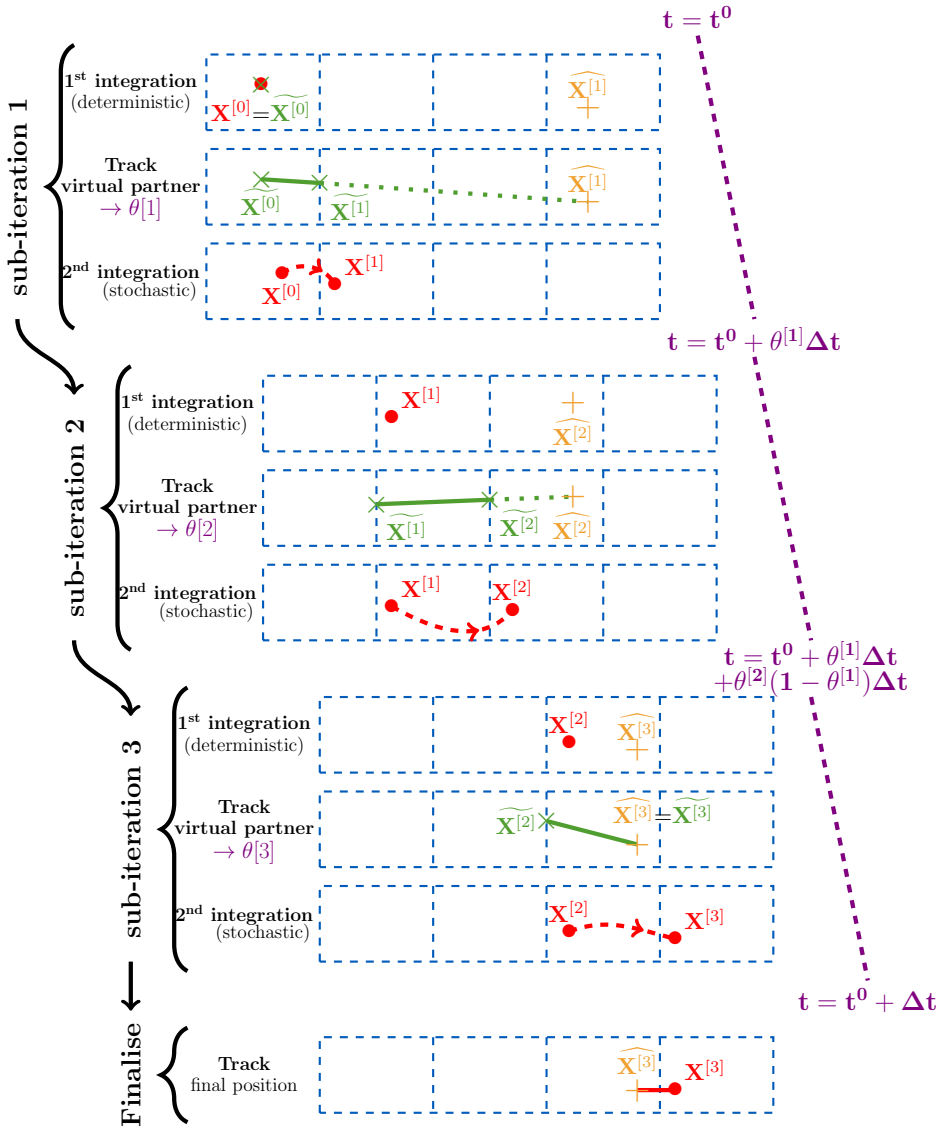
A few comments are in order. At first sight, it could be argued that the difference between the position of the virtual partner $\widetilde{\mathbf{X}^{[m]}}$ and the position of the particle $\mathbf{X}(t + \Delta t)$ at the last sub-iteration points to an error in the algorithm. However, it should be recalled that this difference is due to the stochastic terms that are taken into account when computing the particle position but not in the position of the virtual partner (which is fully deterministic). At this stage, it is worth emphasizing that, within each time step of the computation, the role of the virtual partner is essentially to provide information on the time spent in each cell and to determine the neighboring cells that a particle can cross during the time step, while ensuring that the anticipation issue is avoided. This further supports the choice of a tracking algorithm based on successive neighbor searches (see Section 3.2.2), since it naturally provides information on the successive cells crossed by a particle during a time step (contrary to simple localization algorithms). At each sub-iteration of the algorithm, the increments of this virtual partner represent the mean conditional displacement and, in that sense, remain coherent with the underlying physical process modeled. Each virtual partner is therefore an auxiliary in the calculation of the (true) particle motion. Furthermore, we are basically interested in developing weak approximations, which means capturing particle dynamics in a statistical sense. As a result, what really matters is to obtain accurate estimations of statistics extracted from particle dynamics through Monte Carlo methods (e.g. , average concentration, mean velocity), that is from an ensemble of particles. In other words, improvements in the prediction of individual particle variables (location as well as velocity) should always be assessed in a statistical sense. For that purpose, we now turn our attention to the resulting behavior of such particle statistics in the rest of the paper.

4.2.2 Consistency analysis of the time step decomposition

Before considering proper validation test cases in Section 5, it is important to check that the new algorithm meets the criterion (c) set forth at the beginning of Section 4.

Verification procedure To that effect, we consider a situation where all mean fields and timescales are constant. In that case, we already know that the current numerical scheme, cf. Eqs. (17) in which T_L, C_i and ϵ are now constant, provides the exact solution in the weak sense. We then consider the new algorithm and assume that one sub-iteration has taken place (the extension to several sub-iterations being immediate) at a time $\theta\Delta t$, implying that the particle state vector $\mathbf{Z}^{n+1}$ is now obtained from $\mathbf{Z}^n$ as the sum of two iterations (one over $\theta\Delta t$ and one over $(1-\theta)\Delta t$). The issue is then to check that, still in the weak sense, both predictions are identical.

In the following, we leave out the direction index i since this is basically a 1D formulation. Starting from a given particle state vector value $\mathbf{Z}^n$ at time t^n, predictions obtained in one iteration (i.e., the current algorithm) are indicated by the index [1], while those obtained with two sub-iterations (i.e., the new algorithm) are indicated by the index [2].



for time interval n to $n+1$ **do**

do

(a) Integrate $\widehat{\mathbf{X}^{[m+1]}}$ from $\mathbf{X}^{[m]}$ with $I^Z = 0$

(b) Compute the exit point $\widetilde{\mathbf{X}^{[m+1]}}$ using the vector $\widetilde{\mathbf{X}^{[m]}}\widehat{\mathbf{X}^{[m+1]}}$

Compute the exit time $\theta^{[m+1]}dt^{[m]}$ with $\theta^{[m+1]} = \frac{|\widetilde{\mathbf{X}^{[m]}}\widetilde{\mathbf{X}^{[m+1]}}|}{|\widehat{\mathbf{X}^{[m]}}\widehat{\mathbf{X}^{[m+1]}}|}$

(c) Integrate $\mathbf{X}^{[m+1]}$ from $\mathbf{X}^{[m]}$ using the exit time $\theta^{[m+1]}\mathrm{d}t^{[m]}$ and random numbers $(\zeta^Z)^{[m+1]}$

(d) Update cell, flow fields and remaining time $\mathrm{d}t^{[m+1]} = (1-\theta^{[m+1]})\mathrm{d}t^{[m]}$

Increment the sub-iteration m

while (A face is crossed)

Track the final integrated position

end for

Figure 7: Summary of the proposed algorithm.

Predictions in one iteration are expressed directly by Eqs. (17), which are rewritten here as

$$(X^{n+1})^{[1]} = X^n + U^n T_L \left(1 - \exp\left(-\frac{\Delta t}{T_L}\right)\right) + C T_L \left[\Delta t - T_L \left(1 - \exp\left(-\frac{\Delta t}{T_L}\right)\right)\right] + (I^X)^{[1]}(\Delta t)\ , \quad (19a)$$

$$(U^{n+1})^{[1]} = U^n \exp\left(-\frac{\Delta t}{T_L}\right) + C T_L \left[(1 - \exp\left(-\frac{\Delta t}{T_L}\right)\right] + (I^U)^{[1]}(\Delta t). \quad (19b)$$

The two stochastic integrals are represented by the two correlated centered Gaussian random variables

$(I^X)^{[1]}(\Delta t)$ and $(I^U)^{[1]}(\Delta t)$ which are fully determined by the matrix of their second-order moments [29]

$$\left\langle \left((I^U)^{[1]}(\Delta t)\right)^2 \right\rangle = C_0\epsilon\frac{T_L}{2}\left(1-\exp\left(-2\frac{\Delta t}{T_L}\right)\right) , \tag{20a}$$

$$\left\langle (I^U)^{[1]}(\Delta t)\,(I^X)^{[1]}(\Delta t) \right\rangle = C_0\epsilon\frac{T_L^2}{2}\left(1-\exp\left(-\frac{\Delta t}{T_L}\right)\right)^2 , \tag{20b}$$

$$\left\langle \left((I^X)^{[1]}(\Delta t)\right)^2 \right\rangle = C_0\epsilon T_L^2\left\{\Delta t - \frac{T_L}{2}\left[1-\exp\left(-\frac{\Delta t}{T_L}\right)\right]\left[3-\exp\left(-\frac{\Delta t}{T_L}\right)\right]\right\} . \tag{20c}$$

In the case of two iterations, a first estimation of $\mathbf{Z}(\theta\Delta t)$ is made with Eqs. (19) using the two correlated Gaussian random variables noted $(I^U)^{[2]}(\theta\Delta t)$ and $(I^X)^{[2]}(\theta\Delta t)$ and a second iteration is performed over the remaining time interval $(1-\theta)\Delta t$ to obtain the prediction of $\mathbf{Z}^{n+1}$ using two other correlated Gaussian random variables noted $(I^U)^{[2]}((1-\theta)\Delta t)$ and $(I^X)^{[2]}((1-\theta)\Delta t)$. Note that $\left[(I^U)^{[2]}((1-\theta)\Delta t),(I^X)^{[2]}((1-\theta)\Delta t)\right]$ are independent of $\left[(I^U)^{[2]}(\theta\Delta t),(I^X)^{[2]}(\theta\Delta t)\right]$. Combining the two gives

$$\begin{aligned}(X^{n+1})^{[2]} = X^n + U^nT_L\left(1-\exp\left(-\frac{\Delta t}{T_L}\right)\right) + CT_L\left[\Delta t-\exp\left(-\frac{\Delta t}{T_L}\right)\right]\\ + \underbrace{(I^X)^{[2]}(\theta\Delta t)+(I^X)^{[2]}((1-\theta)\Delta t)+T_L(I^U)^{[2]}(\theta\Delta t)\left(1-\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\right)}_{(\widetilde{I}^X)^{[2]}(\theta,\Delta t)},\end{aligned} \tag{21a}$$

$$\begin{aligned}(U^{n+1})^{[2]} = U^n\exp\left(-\frac{\Delta t}{T_L}\right) + CT_L\left(1-\exp\left(-\frac{\Delta t}{T_L}\right)\right)\\ + \underbrace{(I^U)^{[2]}(\theta\Delta t)\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)+(I^U)^{[2]}((1-\theta)\Delta t)}_{(\widetilde{I}^U)^{[2]}(\theta,\Delta t)}.\end{aligned} \tag{21b}$$

The variance of $(\widetilde{I}^U)^{[2]}(\theta,\Delta t)$ is easily calculated and is:

$$\left\langle \left((\widetilde{I}^U)^{[2]}(\theta,\Delta t)\right)^2 \right\rangle = \left\langle \left((I^U)^{[2]}(\theta\Delta t)\right)^2 \right\rangle \exp\left(-\frac{2(1-\theta)\Delta t}{T_L}\right) + \left\langle \left((I^U)^{[2]}((1-\theta)\Delta t)\right)^2 \right\rangle \tag{22a}$$

$$= C_0\epsilon\frac{T_L}{2}\left[\left(1-\exp\left(-\frac{2\theta\Delta t}{T_L}\right)\right)\exp\left(-\frac{2(1-\theta)\Delta t}{T_L}\right)+1-\exp\left(-\frac{2(1-\theta)\Delta t}{T_L}\right)\right] \tag{22b}$$

$$= C_0\epsilon\frac{T_L}{2}\left[1-\exp\left(-\frac{2\Delta t}{T_L}\right)\right] \tag{22c}$$

$$= \left\langle \left((I^U)^{[1]}(\Delta t)\right)^2 \right\rangle, \tag{22d}$$

where the last equality comes from Eq. (20a). Similarly, the variance of $(\widetilde{I}^X)^{[2]}(\theta,\Delta t)$ is obtained as:

$$\begin{aligned}\left\langle \left((\widetilde{I}^X)^{[2]}(\theta,\Delta t)\right)^2 \right\rangle &= T_L^2\left[1-\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\right]^2\left\langle \left((I^U)^{[2]}(\theta\Delta t)\right)^2 \right\rangle\\ &+2T_L\left[1-\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\right]\left\langle \left((I^X)^{[2]}(\theta\Delta t)(I^U)^{[2]}(\theta\Delta t)\right) \right\rangle\\ &+\left\langle \left((I^X)^{[2]}(\theta\Delta t)\right)^2 \right\rangle + \left\langle \left((I^X)^{[2]}((1-\theta)\Delta t)\right)^2 \right\rangle ,\end{aligned} \tag{23}$$

while the covariance is

$$\begin{aligned}\left\langle \left((\widetilde{I}^X)^{[2]}(\theta,\Delta t)(\widetilde{I}^U)^{[2]}(\theta,\Delta t)\right) \right\rangle = T_L\left[1-\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\right]\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\left\langle \left((I^U)^{[2]}(\theta\Delta t)\right)^2 \right\rangle\\ +\exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)\left\langle \left((I^X)^{[2]}(\theta\Delta t)(I^U)^{[2]}(\theta\Delta t)\right) \right\rangle + \left\langle \left((I^X)^{[2]}((1-\theta)\Delta t)(I^U)^{[2]}((1-\theta)\Delta t)\right) \right\rangle .\end{aligned} \tag{24}$$

The formulas in Eqs. (20) can then be applied, using either $\theta\Delta t$ or $(1-\theta)\Delta t$ as the proper time interval, and injected in Eqs. (23) and (24). Tedious but straightforward calculations show then that we have

$$\left\langle \left((\widetilde{I}^X)^{[2]}(\theta,\Delta t) \right)^2 \right\rangle = \left\langle \left((I^X)^{[1]}(\Delta t) \right)^2 \right\rangle \tag{25a}$$

$$\left\langle \left((\widetilde{I}^X)^{[2]}(\theta,\Delta t)(\widetilde{I}^U)^{[2]}(\theta,\Delta t) \right) \right\rangle = \left\langle (I^U)^{[1]}(\Delta t)(I^X)^{[1]}(\Delta t) \right\rangle , \tag{25b}$$

which, with Eq. (22d), proves that $\left[(\widetilde{I}^X)^{[2]}(\theta,\Delta t), (\widetilde{I}^U)^{[2]}(\theta,\Delta t)\right]$ is statistically equivalent to $\left[(I^X)^{[1]}(\Delta t), (I^U)^{[1]}(\Delta t)\right]$ and, consequently, that Eqs. (21) are identical to Eqs. (19).

An important remark These calculations remain valid even when θ is a function of $\mathbf{Z}^n$ and of the mean fields and timescale (here C and T_L). However, it is worth emphasizing that the fact that θ is independent of the random variables representing the stochastic integrals is a crucial point in the above verification and that these properties would break down otherwise. Indeed in this spurious case we would have:

$$\left\langle \exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right)(I^{Z_1})^{[2]}(\theta\Delta t)(I^{Z_2})^{[2]}(\theta\Delta t) \right\rangle \neq \left\langle \exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right) \right\rangle \left\langle (I^{Z_1})^{[2]}(\theta\Delta t)(I^{Z_2})^{[2]}(\theta\Delta t) \right\rangle, \tag{26}$$

with Z_1 and Z_2 either X or U and:

$$\left\langle \exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right) \right\rangle \neq \exp\left(-\frac{(1-\theta)\Delta t}{T_L}\right). \tag{27}$$

This reflects the non-anticipation issue brought out repeatedly throughout this section.

Having carried out this verification test case, we can now consider validation test cases. This is addressed in the next section.

5 Numerical results

Having checked that all the criteria set forth at the beginning of Section 4 are met, the new algorithm is now validated. For that purpose, it has been implemented in the open source CFD solver code_saturne [1] and numerical results are compared to analytical ones in two situations:

- The first case consists in checking that statistics obtained from particle-attached variables by Monte-Carlo methods are in line with analytical expressions when all mean fields and timescales are constant (i.e. a uniform mean flow). This is done in Section 5.1 using a point source particle dispersion in a stationary homogeneous isotropic turbulent flow.
- The second case corresponds to a simple but challenging non-uniform flow involving curved streamlines, which allows to assess the accuracy of the new algorithm in complex situations where classical particle-tracking algorithms often encounter limitations (see Section 5.2).

5.1 Validation in a uniform flow

Drawing on the verification test case carried out in Section 4.2.2, the aim is twofold: first, to check that, with the new proposed time-splitting method, numerical results are exact when flow properties are constant in time and space; second, to bring out discrepancies induced when using an anticipation method even in such a simple case. To that end, the dispersion of particles released from a point source is analyzed in a stationary homogeneous isotropic turbulent flow. This choice is motivated by the fact that, in such a context, analytical formulas can be derived [22].

In the following, we start by describing the system considered, including physical aspects (e.g., flow characteristics), as well as numerical ones (e.g., spatial discretization, time step) and theoretical results. Then, numerical results are validated with respect to these theoretical expressions.

5.1.1 System considered: point source dispersion in homogeneous isotropic turbulence

Physical aspects The case studied consists of a point-source (fluid) particle dispersion in a stationary homogeneous isotropic turbulent flow. This means that the flow is uniform and stationary within the domain considered and that it is periodic in all directions. Here, we have imposed a velocity based on the turbulent dispersion $U_\alpha = \sqrt{\langle u_\alpha^2 \rangle} = 1\,\mathrm{m\,s^{-1}}$ and a Lagrangian time scale $T_L = 1\,\mathrm{s}$.

Numerical aspects Simulations were carried out using the CFD software code_saturne. The mean flow field is imposed taking into account an extra artificial forcing to maintain a constant level of energy within the system [32]. This leads to a flow field with a constant dissipation rate equal to $\epsilon = \frac{2}{C_0}\frac{U_\alpha^2}{T_L}$ where the proper closure of T_L for homogeneous stationary flow is $T_L = 4k/(3C_0\epsilon)$ [32, 21] with $k = 3/2U_\alpha^2$, and C_0 the Kolomogorov constant in the diffusion coefficient of the Langevin model.

Once fluid mean fields are obtained, 100 000 fluid particles are injected at the center of the box. Their trajectories are computed using the new algorithm described in Section 4. The total size of the domain (box) is taken large enough so that the particles do not reach boundaries during each simulation.

Theoretical aspects As shown in Section 4.2.2, the algorithm is exact when all mean fields and timescales are constant. By resorting to Itô's calculus, it is then straightforward to derive analytical formulas for the variances $\langle X^2 \rangle$ and $\langle U^2 \rangle$ as well as for the covariance $\langle XU \rangle$ as a function of time, taking $t_0 = 0$ as the initial time when particles are released. From these analytical solutions, we can extract two limit cases: the ballistic (i.e., when $t \ll T_L$) and the diffusive one (i.e., when $t \gg T_L$). This was done in [22], yielding the following expressions in these two limit cases:

Ballistic limit case ($t \ll T_L$)		Diffusive limit case ($t \gg T_L$)	
$\langle X^2 \rangle \sim C_0 \frac{\epsilon t^2 T_L}{2} = U_\alpha^2 t^2,$	(28a)	$\langle X^2 \rangle \sim C_0 \epsilon T_L^2 t = 2U_\alpha^2 T_L t,$	(29a)
$\langle XU \rangle \sim \quad C_0 \epsilon\, t^2 \quad = 2U_\alpha^2 \frac{t^2}{T_L},$	(28b)	$\langle XU \rangle \sim \frac{C_0 \epsilon T_L^2}{2} = U_\alpha^2 T_L,$	(29b)
$\langle U^2 \rangle \sim \quad C_0 \epsilon\, t \quad = 2U_\alpha^2 \frac{t}{T_L}.$	(28c)	$\langle U^2 \rangle \sim \frac{C_0 \epsilon T_L}{2} = U_\alpha^2.$	(29c)

It is worth mentioning that, in the diffusive regime, the only physically-relevant statistic is $\langle X^2 \rangle$ since the instantaneous particle velocity does not play a role anymore in the particle position evolution equation. This corresponds to the notion of fast-variable elimination (here U becomes a fast variable that can be eliminated), which is addressed extensively in standard textbooks [8] as well as in previous works both from the theoretical standpoint [21, 17] and the numerical one [29]. The constant value of the velocity second-order moment still retains its typical kinetic energy meaning (showing that we are actually dealing with particles in contact with a kind of thermostat) but the correlation $\langle XU \rangle$, which is less physically-meaningful, is also considered since both results are useful to discuss statistical noise in Monte Carlo estimations.

5.1.2 Validation

In the following, we check that numerical results are in agreement with analytical formulas regardless of the number of occurrences the time-splitting algorithm is called in two cases:

- A first set of simulations was performed with a focus on the ballistic regime. For that purpose, the time step was taken equal to $\Delta t = 0.05 T_L$. Two spatial discretizations were considered: $\Delta x = U_\alpha \Delta t/50$ and $\Delta x = 10 U_\alpha \Delta t$. These two grid sizes were carefully chosen so that first one (resp. the second one) corresponds to the case where the average particle displacement during one time step is much smaller (resp. much larger) than the grid size Δx. The simulations were run for a total time equal to $6T_L$.

  Figures 8A-8C display the second-order moments as a function of the dimensionless time $t^* = t/T_L$, comparing results obtained with the new algorithm to analytical formulas. It can be seen that the outcomes of the new algorithm remain exact for both cases. This validates the new algorithm (which

has been activated on average 50 times every time step in the case where $\Delta x = U_\alpha \Delta t/50$). Small differences between results obtained with the cell-to-cell algorithm and analytical values are simply related to statistical errors using Monte Carlo methods (due to a finite number of fluid particles), as discussed below.

As can be seen in Figs. 8D-8F, completely different results are obtained when using an anticipation method, such as the naive formulation outlined in Section 4.1. This is especially revealed by the strong dependence on the cell size, which indeed governs how often the time-splitting algorithm is applied. Whereas results remain roughly acceptable in the case of a large-enough cell size (meaning that the new algorithm is seldom called), numerical predictions quickly deteriorate when smaller cell sizes are considered (i.e. using $\Delta x = U_\alpha \Delta t/50$), therefore inducing severe errors.

- A second set of simulations was carried out in the diffusive limit case. For that purpose, the time step was taken equal to $\Delta t = 200 T_L$. To assess that numerical results are independent of the number of times the new algorithm is applied, two spatial discretizations were also considered: $\Delta x = U_\alpha \Delta t/20$ and $\Delta x = 2.5\ U_\alpha \Delta t$. The simulations were run for a total time equal to $24\,000\ T_L$.

  Figures 9A-9C display the evolution of the second-order moments as a function of time, indicating that the new algorithm provides accurate results regardless of the number of occurrences the time-splitting algorithm is used within a time step. In Figure 9A, it is seen that $\langle X^2 \rangle$ follows a linear evolution right from the outset, as it should be in the diffusive regime, and that the slope is properly reproduced when using non anticipating methods. This demonstrates that the particle dispersion coefficient is well captured. As mentioned above, it is interesting to consider the numerical predictions of $\langle U^2 \rangle$ in Fig. 9B and $\langle XU \rangle$ in Fig. 9C to bring out the statistical noise inherent to Monte Carlo methods. When considering $\langle U^2 \rangle$, the variance of the Monte Carlo estimator is constant in time since $\mathrm{Var}(U^2) = 2\,\langle U^2 \rangle^2$ and the 99% confidence interval is indicated by the two horizontal lines shown in Figs. 9B and 9E. However, for the correlation $\langle XU \rangle$, the variance of the estimator is a function of time since $\mathrm{Var}(XU)(t) = (C_0\epsilon)^2 T_L^4/2\,(1 + t/T_L)$ and the envelope lines limiting the 99% confidence interval are now increasing with time, as displayed in Figs. 9C and 9F. Note again that the resulting increasing level of noise for $\langle XU \rangle$ is a mere observable and has no feedback effect on the particle simulation.

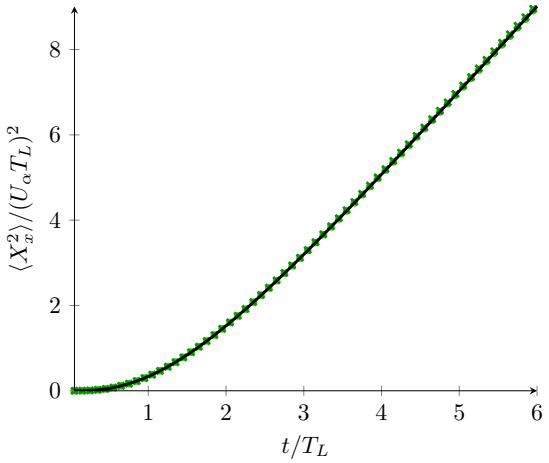
  On the other hand, as can be seen in Fig. 9D, an anticipating method, such as the naive formulation outlined in Section 4.1, is unable to reproduce the correct dispersion coefficient, especially when the time-splitting algorithm is often called (in Fig. 9D, this corresponds to the case $\Delta x = U_\alpha \Delta t/20$ where 20 cells are crossed per iteration on average). For the sake of completeness, we also display $\langle U^2 \rangle$ in Fig. 9E and the correlation $\langle UX \rangle$ in Fig. 9F, which further confirms that an anticipation method yields results that fluctuate but around incorrect averages.

At this stage, it is worth emphasizing that present results were obtained in a 1-D dispersion case. However, these results are directly generalized to 3-D dispersion cases since each direction is treated independently. Furthermore, similar behaviors were obtained using 3-D unstructured meshes instead of regular 1-D Cartesian meshes (details are provided in A.2).

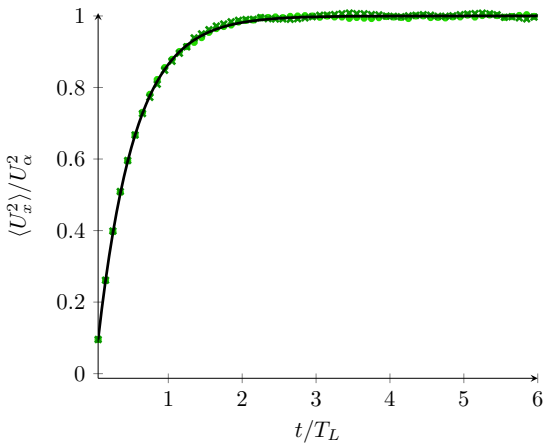
5.2 Validation in a non-uniform flow

Having validated the new algorithm in a uniform flow, the idea is to validate the algorithm in the case of non-uniform flows. To that end, a laminar flow is considered so that the stochastic terms in the time-integration part of the algorithm are equal to zero. This ensures that the time-integration part is deterministic, hence allowing to check that the computed trajectory in non-uniform flows is exact. In addition, a cylindrical Couette flow has been chosen since it is one of the simplest non-uniform flows. In that case, each fluid particle is indeed expected to follow a purely circular motion around the center of rotation of the cylinder (see Section 5.2).

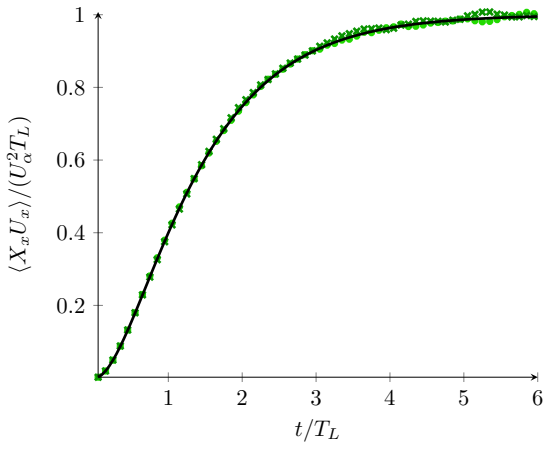
In the following, we start by describing the system considered, including both physical aspects (e.g., geometry of the rotating cylinders, flow characteristics) and numerical aspects (e.g., spatial discretization, time step). Then, the accuracy of the new trajectory algorithm is assessed. This validation is performed in two steps: first, we verify that the motion of a single particle actually follows a purely circular motion and, second, that statistics on particle concentration remain constant throughout the simulation time regardless of the time step used.



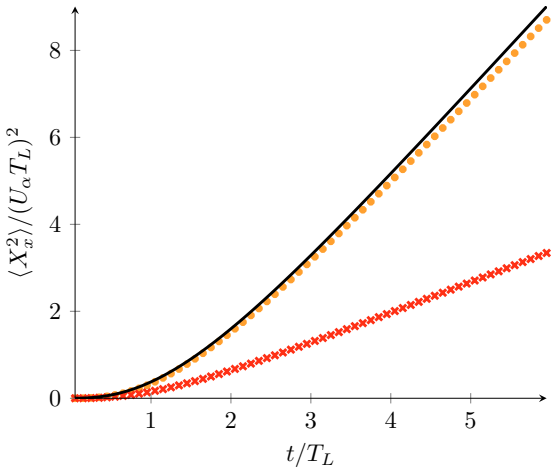
(A) $\langle XX\rangle$ using the proposed method.



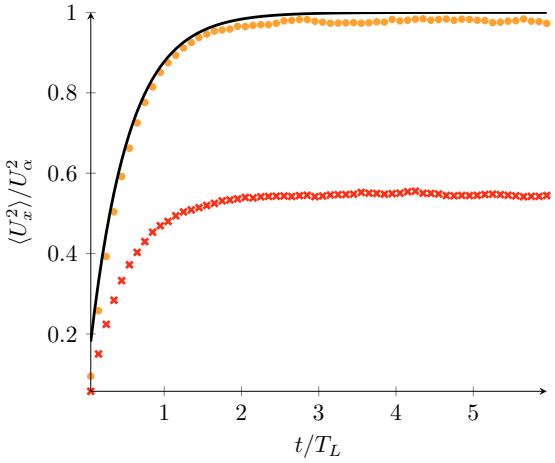
(B) $\langle UU\rangle$ using the proposed method.



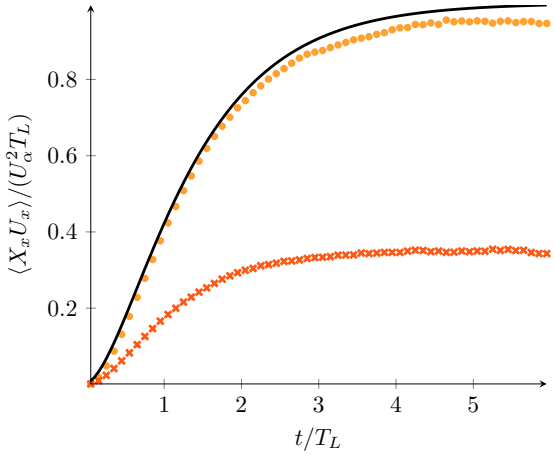
(C) $\langle XU\rangle$ using the proposed method.



(D) $\langle XX\rangle$ using an anticipating method.

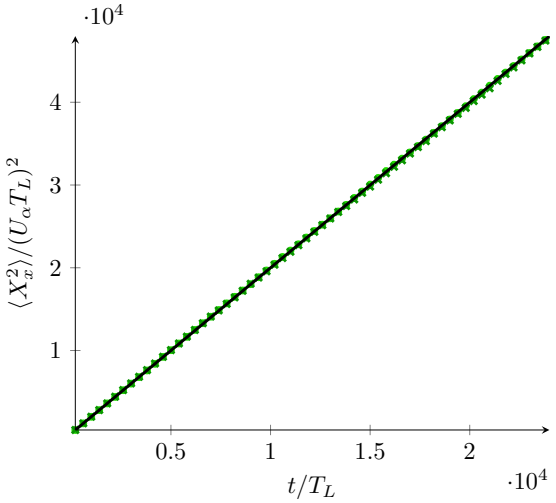


(E) $\langle UU\rangle$ using an anticipating method.

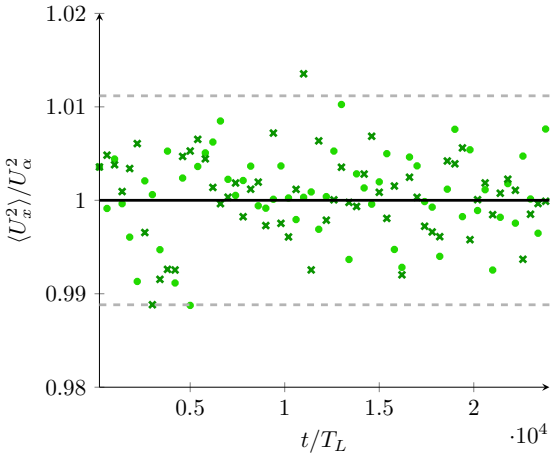


(F) $\langle XU\rangle$ using an anticipating method.

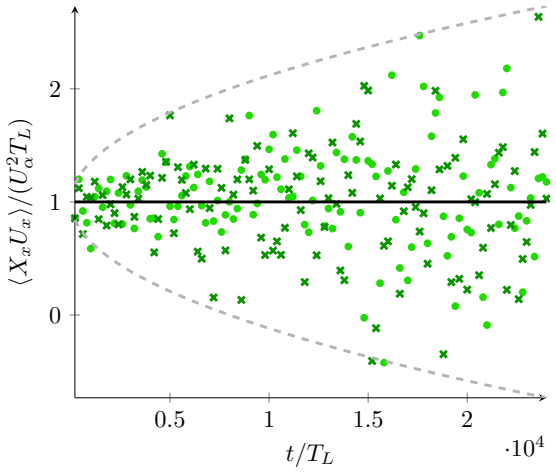
Figure 8: Evolution of $\langle XX\rangle$, $\langle UU\rangle$, $\langle UX\rangle$ as a function of the dimensionless time in the ballistic limit case with a time step $\Delta t = 0.05 T_L$. Two spatial refinements are considered: $\times$ (resp. $\bullet$) corresponds to simulations with a cell size $\Delta x = U_\alpha \Delta t/50$ (resp. $10\ U_\alpha \Delta t$). On the left part, Comparison between the analytical solution (black line) and numerical results obtained with an anticipating method. On the right part, errors obtained with the new algorithm. The dashed lines correspond to the envelop for the 99% confidence interval (analytical formula).



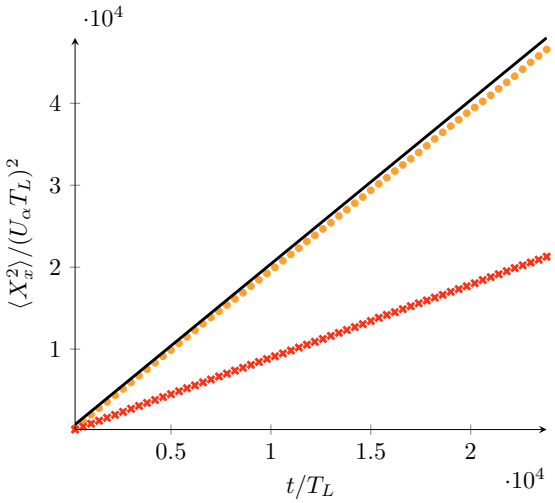
(A) $\langle XX \rangle$ using the proposed method.



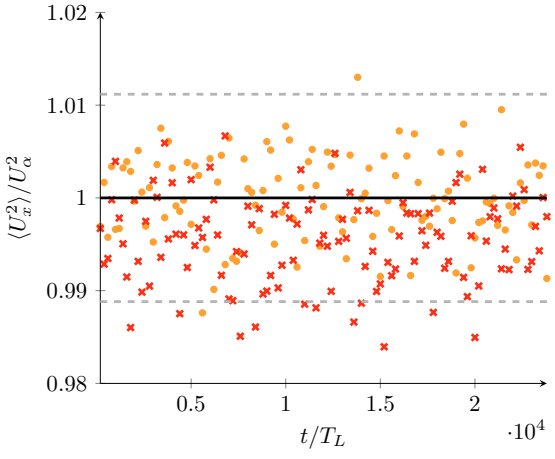
(B) $\langle UU \rangle$ using the proposed method.



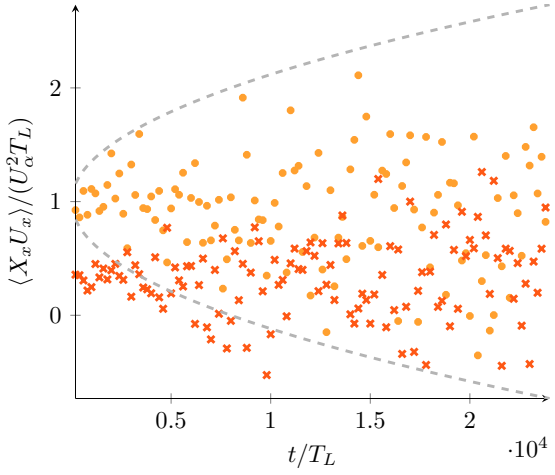
(C) $\langle XU \rangle$ using the proposed method.



(D) $\langle XX \rangle$ using an anticipating method.



(E) $\langle UU \rangle$ using an anticipating method.



(F) $\langle XU \rangle$ using an anticipating method.

Figure 9: Evolution of $\langle XX \rangle$, $\langle UU \rangle$, $\langle UX \rangle$ as a function of the dimensionless time in the diffusive limit case with a time step $\Delta t = 200 T_L$. Two spatial refinements are considered: $\times$ (resp. $\bullet$) corresponds to simulations with a cell size $\Delta x = U_\alpha \Delta t / 20$ (resp. $2.5\ U_\alpha \Delta t$). On the left part, Comparison between the analytical solution (black line) and numerical results obtained with an anticipating method. On the right part, errors obtained with the new algorithm. The dashed lines correspond to the envelop for the 99% confidence interval (analytical formula).

5.2.1 System considered: a laminar cylindrical Couette flow

Physical parameters The case studied here is a laminar cylindrical Couette flow. It consists of a fluid flow between two cylinders: the inner cylinder has a radius $r_{in} = 1\,\mathrm{m}$, rotating at a given angular velocity equal to $\omega_\theta(r_{in}) = 1\,\mathrm{s}^{-1}$, while the outer cylinder has a radius $r_{out} = 2\,\mathrm{m}$ and remains at rest. This means that the fluid is contained within the annulus of thickness $\delta r = 1\,\mathrm{m}$ that separates the two cylinders. The kinematic viscosity of the fluid is set to $1\,\mathrm{m}^2\mathrm{s}^{-1}$ so that the Reynolds number based on those quantities is equal to 1 (i.e., well within the laminar regime).

In this configuration, the fluid is flowing in a cylindrical motion. As a result, the velocity field is unidirectional and, written with the cylindrical coordinate $\mathbf{e}_\theta$, is given by the analytical solution:

$$U = U_\theta(r) = \langle U_\theta \rangle(r) = \frac{\omega_\theta(r_{in})}{\frac{r_{in}}{r_{out}} - \frac{r_{out}}{r_{in}}} \left(\frac{r}{r_{out}} - \frac{r_{out}}{r} \right) \tag{30}$$

where r is the distance from the rotating centre.

Numerical parameters The simulations are carried out using the CFD software code_saturne to solve the Navier–Stokes equation on a 360° polyhedral mesh: it comprises 360 cells in the azimuth direction (i.e. the azimuth discretization angle is $\Delta\theta = 1°$) and 21 in the radial one. The simulations are run using a constant time step, whose value is taken between 0.055 s and 200 s. This range of values has been chosen to assess the accuracy of the algorithm with respect to the time step, especially at large time steps where the average particle displacement is much greater than the grid size.

The tracking of fluid particles within this laminar cylindrical Couette flow is obtained by applying the algorithm described in Section 4. The Lagrangian integral time T_L has been set to 0 s to force a laminar flow (i.e., all stochastic integrals are equal to zero).

5.2.2 Accuracy of numerical results

The new algorithm is here validated by checking that, first, the trajectory of a single particle follows a purely cylindrical motion and, second, that the initial particle concentration along the radial direction is conserved in time. In the following, all results are plotted using a dimensionless time defined as $t^+ = t\,\omega_\theta(r_{in})/(\Delta\theta)$. This means that the dimensionless time step Δt^+ measures the ratio between the average particle displacement and the grid size (in the azimuthal direction) near the inner wall.

Results on a single particle trajectory The motion of a fluid particle in a laminar cylindrical Couette flow is expected to follow a purely circular trajectory. This means that each particle remains at a constant distance r from the rotating centre throughout the simulation. A simulation has been run for 400 iterations using a time step of 1.024 s. This means that the average particle displacement is equal to 55 times the grid size $\Delta\theta \times r_{in}$ near the inner cylinder and to 0.5 times the grid size $\Delta\theta \times r_{out}$ near the outer one.

Figure 10 displays the trajectory of two particles: one is initially located very close to the inner cylinder while the second one is initially located in the bulk. The trajectory (left-hand plot) clearly shows that the new algorithm does lead to a circular motion. These results confirm the accuracy of the new algorithm even for large values of the time step. This is further supported by the time-evolution of the radius r (right-hand plot). In fact, the results are not distinguishable from the theoretical expectations, even for the particle initially close to the inner cylinder (which has circled the cylinder close to 60 times by the end of the simulation).

Figure 10 also displays the results obtained with the reference algorithm, which does not account for cell-to-cell integration. It can be seen that the radius increases with time all the more when the time step increases. This is due to the error made when no cell-to-cell integration is made while using large time steps. The origin of the error is illustrated in Figure 11: the spatial discretization and P_0 interpolation mean that each particle experiences a constant velocity within a cell. As a particle crosses a face, if the velocity is not updated immediately, the particle will move for the remainder of the time step with the velocity encountered in the previous cell. Since this previous fluid velocity is slightly shifted with respect to the radial direction in the current new cell, it induces an error without cell-to-cell integration which is directly proportional to the time step.

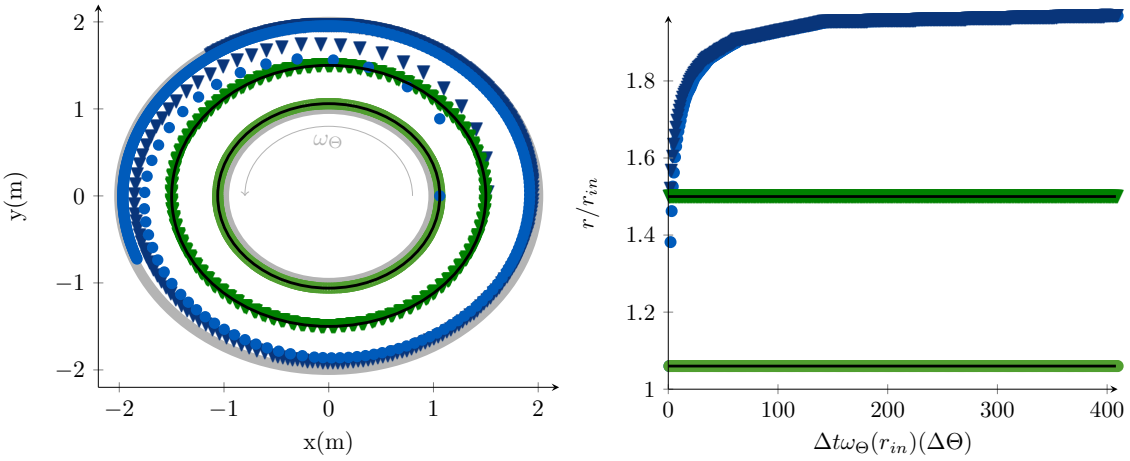


Figure 10: Trajectory of particles (left) and corresponding evolution of the radius r as a function of the dimensionless time t^+ (right) for a constant time step $\Delta t^+ = 55$. Two particles are followed: one located initially in the vicinity of the wall (•) and one in the center of the domain (▼). For each particle, the results obtained with the new algorithm (green) are in agreement with the expected trajectory (orange dotted line), while the results obtained with the reference scheme (blue) are flawed by spurious drift effects (leading to accumulation in the region near the outer cylinder).

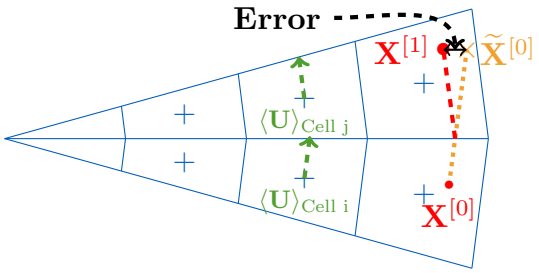


Figure 11: Scheme highlighting the origin of the numerical error in the algorithm without cell-to-cell integration. This error induces an accumulation of particles near the outer cylinder as time increases without cell-to-cell integration (see Figure 12).

Results on particle concentration We focus now our attention on simulations in which a large number of fluid particles is tracked. In that case, the statistics of interest is the particle concentration along the radial direction. In fact, since each fluid particle follows a circular orbit, the particle concentration should remain constant throughout time. With this result in mind, we verify here that the particle concentration is homogeneous in space provided that fluid particles are homogeneously distributed in space initially.

Numerical simulations have been carried out with 200,000 fluid particles initially homogeneously distributed in space. The concentration is then analyzed by computing the particle number concentration in each of the 21 cells along the radial direction. Figure 12 displays the evolution of the concentration, which has been normalized with the initial concentration, as a function of the distance r. It can be seen that, for a dimensionless time step equal to $\Delta t^+ \simeq 5.5$, the concentration obtained remains constant in all the domain throughout the whole simulated time. Meanwhile, the results obtained with the reference algorithm (i.e., without cell-to-cell integration) show that particles tend to accumulate in the outer region, leading to an increasing concentration in the outer region with time (the stationary state at longer times will consist in having all particles located at the outer cylinder).

To further assess the accuracy of the new algorithm, the error between the numerical value and the theoretical value of the number concentration has been computed within each of the 21 regions. The average error over the whole domain and simulation time (here 41 s) is displayed in Figure 13 as a function of the dimensionless time step Δt^+. It confirms that the new algorithm provides accurate results regardless of the time step used, i.e., both when the average displacement is smaller or greater than the grid size. The small (constant) error with the new algorithm comes from the statistical noise due to Monte Carlo methods. Meanwhile, the results obtained without cell-to-cell integration clearly show an increasing numerical error

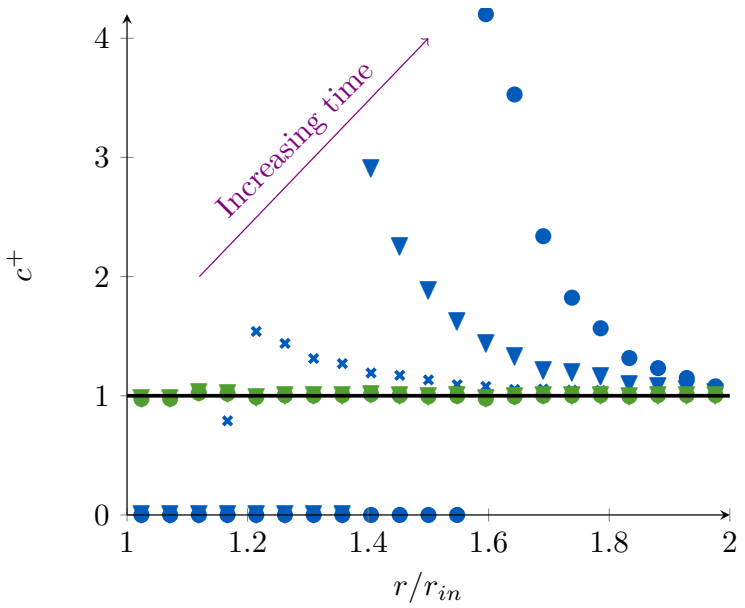


Figure 12: Evolution of the dimensionless concentration c^+ (concentration normalized by the initial concentration) as a function of the radius r. Results are displayed for various simulation times in the transitional regime: t^+=275 (×), t^+=1100 (▼) and t^+=2200 (•). The time step is constant such that $\Delta t^+ \simeq 5.5$. The results obtained with the new algorithm (green) are in agreement with the analytical results (yellow) while the results obtained with the reference trajectory (blue) lead to an accumulation toward the outer cylinder which increases with time (arrow).

with increasing time steps.

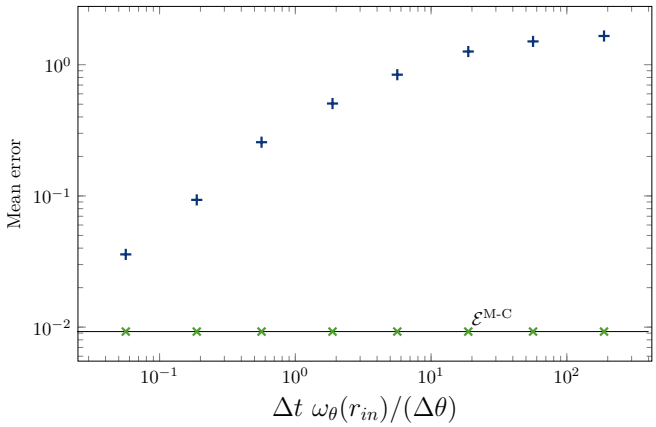


Figure 13: Convergence of the mean error on particle concentration computed at time t =41 s as a function of the time step, using the reference algorithm (+) and the new cell-to-cell algorithm (×). These results confirm that the reference algorithm induces a spurious drift (the error increases with increasing time steps) contrary to the new algorithm which remains accurate regardless of the time step (the constant non-zero error is due to the statistical noise inherent to Monte Carlo methods).

6 Conclusions and perspectives

In this paper, we have developed a new cell-to-cell algorithm for particle tracking in the context of hybrid approaches that couple Lagrangian stochastic methods with mean-field ones. The cell-to-cell integration consists in dividing the time step into a number of sub-iterations, each one corresponding to the motion of a particle within one cell. This means that the motion of a particle is stopped each time it crosses a face and leaves a cell, at which point the mean fields are updated to compute particle displacement over the remainder of the time step. This decomposition allows naturally to account for changes in the mean fields every time a particle enters a new cell. Given the stochastic nature of the model, an additional constraint has to be satisfied: no anticipation should be made when estimating the residence time in each cell. Indeed, a naive formulation would consist in first predicting particle displacements over a time step and then try to deduce the residence times in the cells that have been crossed. Yet, this would yield estimations of these residence times that become functions of the future of the Wiener process driving the diffusion term, in direct violation of the very definition of the stochastic integral in the Itô sense. To avoid this pitfall, and the resulting spurious numerical drift and diffusion values it would entail, careful estimations based only on values at the beginning of each sub-time steps must be made. In the present algorithm, this is achieved by introducing the notion of a virtual partner whose motion is governed by mean conditional increments and which is used to provide free-of-statistical-bias of the residence times in the different cells being crossed during one time step.

Drawing on these two notions (cell-to-cell integration and non-anticipation), the new algorithm is built on a three-step process. The first step consists in computing the deterministic estimation of the particle position at the end of the time step. Then, the trajectory algorithm based on a free-flight assumption is applied to determine if the particle leaves/remains in the cell. If it remains in the cell, the position and velocity of the particle at the end of the time step are computed. Otherwise, the trajectory algorithm provides information on the exit time and exit location. The whole three-step process is repeated again but starting from the last known exit location and for the remainder of the time step. This process is repeated until the particle reaches its final destination at the end of the time step Δt (i.e. it does remain in the cell).

A first analysis consists in checking that the new algorithm remains exact for constant mean fields. In the more general case of non-uniform flows, since the residence times are now only approximated, the time integration scheme cannot be regarded as exact but the overall scheme is guaranteed to respect the non-anticipating constraint and is shown to provide considerable improvements over the current algorithm for particle tracking in non-homogeneous flows. This is further demonstrated by considering numerical predictions in two test cases: particle dispersion in a uniform mean flow and particle dynamics in a non-uniform flow. These two cases have confirmed the improved accuracy of the new algorithm, even when large time steps are used.

At this point, it is worth emphasizing that, although we have considered fluid particles and a simple Langevin model, the same method can be directly applied to dispersed turbulent two-phase flows and, more generally, to similar stochastic dynamical models which require to track particles in complex geometries and meshes. Note that in the examples presented in Section 5 one concerns a laminar flow and the second one a high-Reynolds-number turbulent flow. However, the present methodology (i.e. the time-splitting algorithm) is not limited to these situations and can be applied to any flows regardless of the Reynolds number, provided that an adequate dynamical model and its corresponding numerical scheme are applied. Furthermore, this new algorithm paves the way toward the development of refined algorithms for such hybrid approaches that couple Lagrangian methods with mean-field approaches. In this work, we have applied a cell-to-cell integration every time a particle enters a new cell. On the other hand, this time-splitting scheme could be applied only when mean fields differ significantly from one cell to another. Hence, a refined algorithm could be devised using test functions to quantify variations in the mean-field quantities along particle trajectories and apply this time-splitting method only when variations are larger than a given threshold.

A Details of the particle tracking algorithm for 3-D unstructured mesh

This appendix presents the trajectory algorithm that is used in the present study. This algorithm is able to track the motion of particles even in 3-D fully unstructured meshes with warped faces. The tracking algorithm is first described in A.1. It includes the description of the original tracking algorithm with the detection of face-crossing events in Section A.1.1. Then, the algorithm is extended in A.1.2 to compute the location and exit time when a particle crosses a face. Last, the algorithm is validated in A.2 by comparing the numerical results obtained using various 3-D unstructured meshes in a simple non-uniform flow.

A.1 Principle of the neighbor search algorithm

The algorithm is based on a successive neighbor search. This means that the cell inside which a particle currently resides is determined by browsing through the neighboring cells. Such algorithms require three pieces of information: (a) the origin particle location $\mathbf{X}_O = \mathbf{X}^n$, (b) the corresponding cell inside which it was initially and (c) the destination particle location $\mathbf{X}_D = \mathbf{X}^{n+1}$ (as depicted in Figure 4). The principle is then to determine if the particle leaves the current cell assuming a free-flight motion between point $\mathbf{X}_O$ and $\mathbf{X}_D$. This is performed by:

1. computing which faces of the current cell are intersected by the line $(\mathbf{X}_O\mathbf{X}_D)$;
2. checking if the intersection belongs to the straight-line vector $\mathbf{X}_O\mathbf{X}_D = \mathbf{X}_D - \mathbf{X}_O$.

The key issue is then to have a robust method to detect the intersection between a displacement vector and any face. It is of prime importance to prevent any particle from being permanently lost in the computational domain. For that reason, the method uses Boolean elementary tests which are reproducible from one cell to another so that it can handle pathological cases such as when the vector $\mathbf{X}_O\mathbf{X}_D$ crosses a face through one of its edges (to the machine precision).

A.1.1 Method to detect face-crossing events through warped-faces

The method to detect face-crossing events is based on the decomposition of each face into a set of triangular sub-faces (see also Figure 17). Each triangular sub-face is built using one of the oriented edges of the face (formed by two consecutive vertices $\mathbf{X}_i$ and $\mathbf{X}_j$) and the center of gravity of the face $\mathbf{X}_f$. This decomposition of faces ensures that each triangular sub-face treated is planar, hence making the method tractable even for warped meshes (i.e. with faces whose vertices do not belong to the same plane).

Once a face is decomposed into a set of triangular sub-faces, the intersection between a vector and each planar sub-face is detected using a kind of Möller–Trumblemore algorithm [26]. The principle is to project the vertices of each sub-face in the oriented plane orthogonal to the displacement vector $\mathbf{X}_O\mathbf{X}_D$ passing through $\mathbf{X}_O$ noted $(\mathbf{X}_O, \mathbf{X}_O\mathbf{X}_D^{\perp})$. Then, to compute if line $(\mathbf{X}_O\mathbf{X}_D)$ crosses a sub-face, we simply need to check if the point $\mathbf{X}_O$ belongs to the triangle formed by the projection of the sub-face on this plane (as displayed in Figure 16, where the superscript $(.)^{\dagger}$ corresponds to the projection in the plane perpendicular to the displacement). For that purpose, we resort here to a series of three elementary Boolean tests, each one allowing to verify if the point $\mathbf{X}_O$ is located on the "proper side" of a projected edge.

Elementary Boolean tests For each of the three edges forming a projected sub-face (namely $\mathbf{X}_f^{\dagger}\mathbf{X}_i^{\dagger}$, $\mathbf{X}_f^{\dagger}\mathbf{X}_j^{\dagger}$ and $\mathbf{X}_i^{\dagger}\mathbf{X}_j^{\dagger}$), we have to verify whether the point $\mathbf{X}_O$ is on the proper side of the projected edge. To that extent, we resort here to simple logical tests. For the sake of clarity, let's consider the case of an oriented edge connecting two points $\mathbf{X}_\alpha\mathbf{X}_\beta$ (where $_\alpha$ and $_\beta$ are the indexes of two vertexes of a sub-face) on the projection plane. In that case, the logical test $\mathcal{L}_{\alpha,\beta}^{\text{edge}}$ reads:

$$\mathcal{L}_{\alpha,\beta}^{\text{edge}} = \begin{cases} true & \text{if } (\mathbf{X}_\alpha\mathbf{X}_\beta \wedge \mathbf{X}_\alpha\mathbf{X}_O)\cdot\mathbf{X}_O\mathbf{X}_D > 0. \\ false & \text{otherwise.} \end{cases} \tag{31}$$

As displayed in Figure 14, this elementary Boolean test provides information on whether a point $\mathbf{X}_O$ is in the half plane on the left (true) or on the right (false) of the projected line $(\mathbf{X}_\alpha^\dagger \mathbf{X}_\beta^\dagger)$. It is worth noticing that the inequality in this test being strict, the behavior on the oriented line is asymmetric. Indeed, if point $\mathbf{X}_O$ belongs to this line at the machine precision, test $\mathcal{L}_{\alpha,\beta}^{\text{edge}}$ returns $false$. In other words, we arbitrarily consider that the line $(\mathbf{X}_\alpha^\dagger \mathbf{X}_\beta^\dagger)$ belongs to the closed right half plane and not to the open left one. As a result, the Boolean test $\mathcal{L}_{\alpha,\beta}^{\text{edge}} \bigcup \text{not}(\mathcal{L}_{\alpha,\beta}^{\text{edge}})$ is a partition of the domain. This means that the computation of $\mathcal{L}_{\alpha,\beta}^{\text{edge}}$ is reproducible for a given displacement vector $\mathbf{X}_O\mathbf{X}_D$ for two faces sharing the same edge[1] provided that the ordering of the vertices α and β is fixed. In the present work, we have imposed to sort the vertices in the following order: first $\mathbf{X}_f$, then $\mathbf{X}_i$ and finally $\mathbf{X}_j$ (with $i < j$).

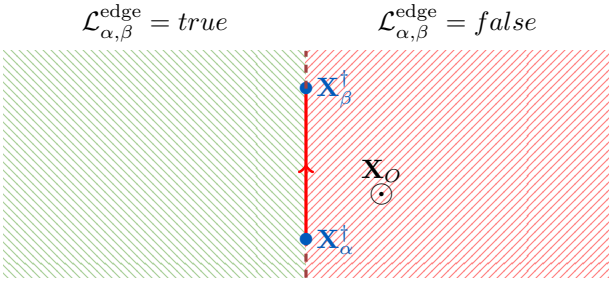


Figure 14: Sketch illustrating how the elementary boolean test $\mathcal{L}_{\alpha,\beta}^{\text{edge}}$ works: the point ($\mathbf{X}_0$ can either be located on the left-hand side or on the right-hand side of the oriented line $(\mathbf{X}_\alpha^\dagger \mathbf{X}_\beta^\dagger)$. Here, the figure is seen from above the plane $(\mathbf{X}_0, \mathbf{X}_O\mathbf{X}_D^\perp)$, meaning that the elementary Boolean test is *true* on the left-hand side of the figure. Note that the line belongs to the closed half-plane (i.e., $false$ in red color).

Figure 14 also shows that this elementary Boolean test is not enough to determine if a point $\mathbf{X}_O$ belongs to the projected triangle sub-face. In fact, a point can be located either on the right side or on the left side of the line depending on the orientation of the vector $\mathbf{X}_i^\dagger \mathbf{X}_j^\dagger$ but also on whether the point is above or below the plane. While the orientation of the vector is now fixed thanks to the sorted vertices introduced in the previous paragraph, we have to introduce the notion of plane orientation to deal with the second issue. For that purpose, we rely on the orientation of the sub-face and we introduce an additional logical test to know if the displacement vector $\mathbf{X}_O\mathbf{X}_D$ is aligned with the oriented sub-face. It returns true if they are aligned and false if they are not. This condition reads:

$$\mathcal{A}_{f,i,j}^{\text{face}} = \begin{cases} true & \text{if } (\mathbf{X}_f\mathbf{X}_i \wedge \mathbf{X}_f\mathbf{X}_j) \cdot \mathbf{X}_O\mathbf{X}_D > 0. \\ false & \text{otherwise.} \end{cases} \tag{32}$$

By combining both logical tests $\left(\mathcal{L}_{i,j}^{\text{edge}} \equiv \mathcal{A}_{f,i,j}^{\text{face}}\right)$ (where $\equiv$ means that both Boolean tests have the same value), we are able to determine on which side of the line $\mathbf{X}_i^\dagger \mathbf{X}_j^\dagger$ the particle lies with respect to the face orientation.

Combined elementary Boolean tests To determine if the intersection point $\mathbf{X}_i$ is within the projected face, we have then to combine these elementary logical tests together. As displayed in Figure 15, the intersection point belongs to the projected face if three conditions are met. These conditions depend on the orientation of the sub-face with respect to the displacement vector:

- When the displacement vector is aligned with the sub-face orientation (i.e., $\mathcal{A}_{f,i,j}^{\text{face}} = true$), the point is located to the left of each of the projected edges provided that we follow the sorted vertices (namely $\mathbf{X}_f^\dagger \mathbf{X}_i^\dagger$, $\mathbf{X}_i^\dagger \mathbf{X}_j^\dagger$ and $\mathbf{X}_j^\dagger \mathbf{X}_f^\dagger$). This means that the following three conditions have to be met: first, $\mathcal{L}_{f,i}^{\text{edge}} = true$; second, $\mathcal{L}_{f,j}^{\text{edge}} = false$ (due to its reverse orientation); third, $\mathcal{L}_{i,j}^{\text{edge}} = true$. This case is displayed in the LHS of Figure 15.
- When the displacement vector is not aligned with the sub-face orientation (i.e., $\mathcal{A}_{f,i,j}^{\text{face}} = false$), the point is located to the right of each of the projected edges provided that we follow the sorted vertices

[1] This is the case for instance for a sub-face seen from one cell or from the neighboring one.

(namely $\mathbf{X}_f^\dagger\mathbf{X}_i^\dagger$, $\mathbf{X}_i^\dagger\mathbf{X}_j^\dagger$ and $\mathbf{X}_j^\dagger\mathbf{X}_f^\dagger$). This means that the following three conditions have to be met: first, $\mathcal{L}^{\text{edge}}_{f,i} = false$; second, $\mathcal{L}^{\text{edge}}_{f,j} = true$ (due to its reverse orientation); third, $\mathcal{L}^{\text{edge}}_{i,j} = false$. This case is displayed in the RHS of Figure 15.

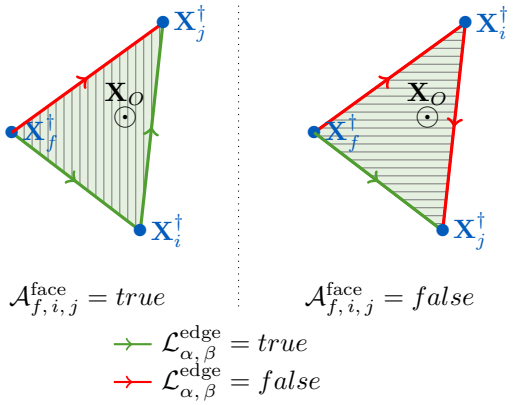


Figure 15: Sketch of the Boolean test for alignment $\mathcal{A}^{\text{face}}_{f,i,j}$. It determines if the displacement vector $\mathbf{X}_O\mathbf{X}_D$ is aligned with the oriented sub-face $\mathbf{X}_f\mathbf{X}_i\mathbf{X}_j$ (and returns *true* in that case). The sketch shows the two possible cases: on the left-hand side, the displacement vector is aligned with the sub-face orientation; on the right-hand side, they are not aligned. In each case, the point $\mathbf{X}_O$ lies within the projected sub-face if it is located on the proper side of all oriented edges (namely $\mathbf{X}_f\mathbf{X}_i$, $\mathbf{X}_f\mathbf{X}_j$ and $\mathbf{X}_i\mathbf{X}_j$, with $i < j$). The logical tests $\mathcal{L}^{\text{edge}}_{\alpha,\beta} = true$ are displayed according to their result (green = *true* and red=*false*).

To sum it up, the intersection point is inside a given sub-face if the following condition is respected.

$$\begin{aligned} & \left(\mathcal{L}^{\text{edge}}_{i,j} \equiv \mathcal{A}^{\text{face}}_{f,i,j}\right) \\ \text{and} \quad & \left(\mathcal{L}^{\text{edge}}_{f,i} \equiv \mathcal{A}^{\text{face}}_{f,i,j}\right) \\ \text{and} \quad & \left(\text{not}(\mathcal{L}^{\text{edge}}_{f,j}) \equiv \mathcal{A}^{\text{face}}_{f,i,j}\right) = true. \end{aligned} \tag{33}$$

where the symbol $\equiv$ corresponds to the operator "equivalent" (i.e., it is true if the two Boolean variables have the same value). This test is made for each sub-face. The algorithm is illustrated in Figure 16 (where the intersection point is on the right hand face).

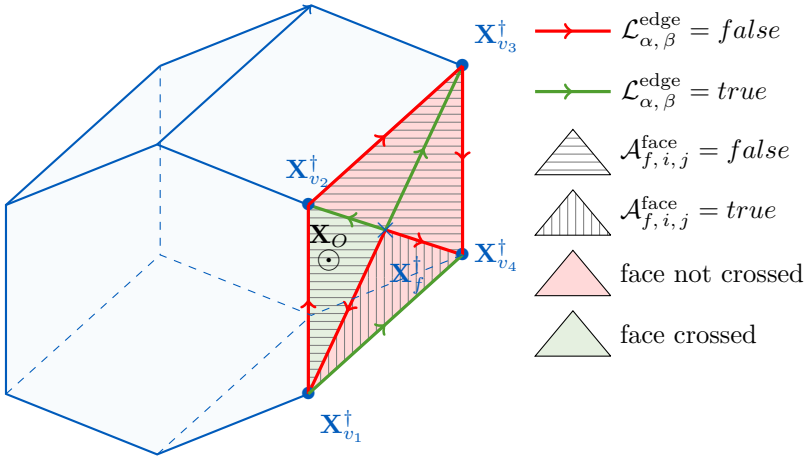


Figure 16: Sketch illustrating how the tracking algorithm determines if a particle displacement from $\mathbf{X}_O$ to $\mathbf{X}_D$ crosses a face. Here, the particle exits the cell through the face on the right. As a result, when the edges of this face are projected on the plane normal to the displacement vector (here with superscripts †), the various logical tests $\mathcal{L}^{\text{edge}}_{\alpha,\beta}$ and $\mathcal{A}^{\text{face}}_{f,i,j}$ confirm that one of the triangular sub-face is detected as an exit face (green color) while the three other sub-faces are not (red color).

In the case of warped faces, it is possible for a line $(\mathbf{X}_O\mathbf{X}_D)$ to cross the same face several times. This is depicted in Figure 17. In that case, the algorithm monitors the number of times that the line $(\mathbf{X}_O\mathbf{X}_D)$ crossed the face. If this number is even, it means that the particle remains in the current cell (it has left and reentered the cell). If this number is odd, it means that the particle leaves the cell through this face.

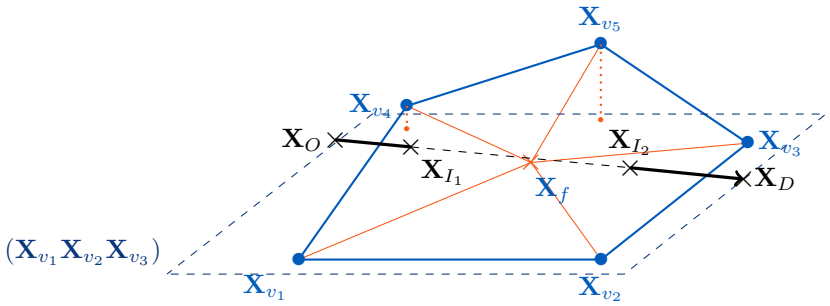


Figure 17: Sketch showing a particle displacement from $\mathbf{X}_O$ to $\mathbf{X}_D$ going through a warped face: it can be seen that the particle exits the cell through $\mathbf{X}_{I_1}$ (which belongs to the sub-face $(\mathbf{X}_f, \mathbf{X}_{v_1}, \mathbf{X}_{v_4})$ and re-enters the cell through $\mathbf{X}_{I_2}$ (which belongs to the sub-face $(\mathbf{X}_f, \mathbf{X}_{v_2}, \mathbf{X}_{v_3})$. This is naturally handled by the present algorithm which browse through all sub-faces and then counts the number of times a sub-face is crossed: a pair number means that it stays in the current cell while an odd number means that it exits the current cell.

A.1.2 Method to estimate the intersection time and position

As mentioned in Section 4.2.1, the new algorithm not only requires information on the cell containing the particle but also on the intersection time and location. This means that the trajectory algorithm described previously has to be extended to provide this information.

Having determined that the line $(\mathbf{X}_O\mathbf{X}_D)$ does cross a sub-face, the relative time $\theta = t_{\text{cross}}/\Delta t$ necessary to reach this sub-face can be estimated using the free-flight assumption. It gives:

$$\theta = \frac{\mathbf{X}_O\mathbf{X}_f \cdot (\mathbf{X}_f\mathbf{X}_i \wedge \mathbf{X}_f\mathbf{X}_j)}{\mathbf{X}_O\mathbf{X}_D \cdot (\mathbf{X}_f\mathbf{X}_i \wedge \mathbf{X}_f\mathbf{X}_j)}. \tag{34}$$

This equation is well-posed since we apply it only when we have previously determined that the line actually crosses the face. In fact, the value of θ actually provides additional information. When θ is negative, it means that the intersection point is an entrance point for the oriented axis $(\mathbf{X}_O\mathbf{X}_D)$. When θ is positive, it means that the intersection point is an exit point. The number of sub-faces through which the oriented axis $(\mathbf{X}_O\mathbf{X}_D)$ enters (n_{in}) and leaves (n_{out}) is then counted. We then check if the particle is indeed in the correct cell thanks to these numbers. The particle is in the correct cell if the number of sub-faces through which the line $(\mathbf{X}_O\mathbf{X}_D)$ enters in the cell equals to the number of sub-faces through which it exits the cell and if both of these numbers are not zero (i.e., $n_{in} = n_{out} > 0$).

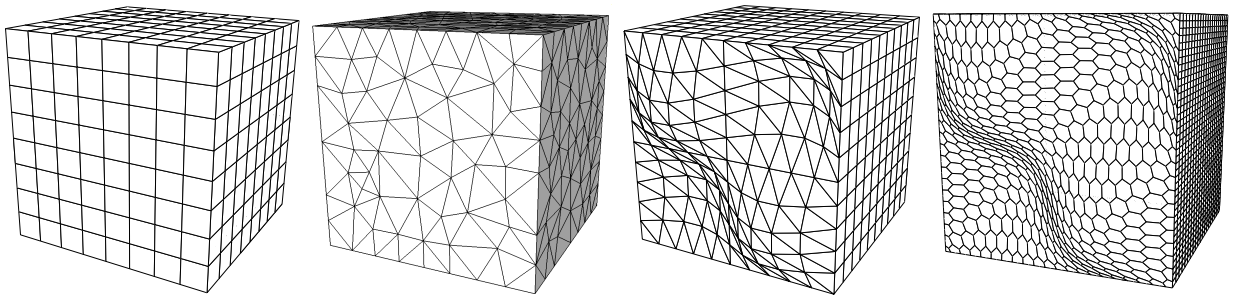
If the value of θ defined in Eq. (34) is in the interval $\theta \in [0, 1[$, the particle does actually cross a face during the time step. The position of the particle at the intersection is then simply given using a simple linear interpolation (this linear interpolation is coherent with the assumption of free-flight motion):

$$\mathbf{X}_I = \mathbf{X}_O + \theta \times \mathbf{X}_O\mathbf{X}_D. \tag{35}$$

When a particle leaves a cell after crossing one face several times (as in wrapped faces), the exit time is considered equal to the largest value in the range $[0, 1[$ (i.e., the last exit time).

A.2 Validation on 3-D unstructured meshes

The trajectory algorithm has been tested using various meshes obtained from the FVCA6 benchmark test cases [7]. These meshes were selected to be representative of a range of different meshes, going from a



(A) Hexa mesh. (B) Tetra mesh. (C) PrT mesh. (D) PrG mesh.

Figure 18: Type of mesh used in the present case, which span a range of possible configurations (from simple Cartesian mesh without wrapped faces to highly distorted meshes with wrapped faces.

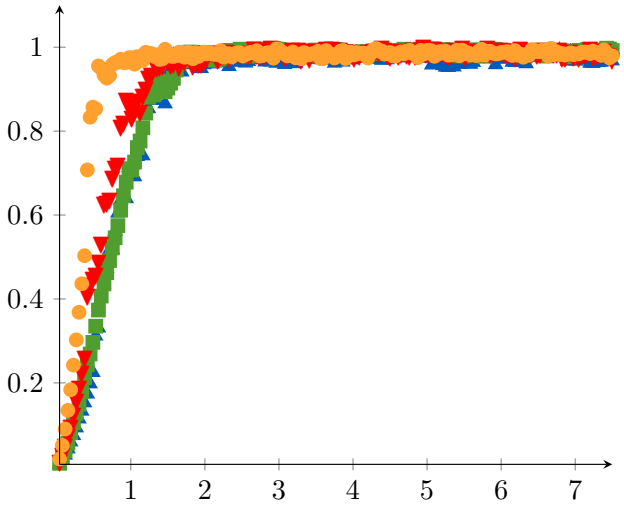


Figure 19: Maximum dimensionless distance between the particles and their center of gravity using the Hexa mesh (■), the Tetra mesh (▲), the PrT mesh (▼) and the PrG mesh (●. All maximum dimensionless distances are smaller or equal to 1, meaning that the particles stay within the physical domain.)

regular Cartesian mesh to a highly distorted mesh with different refinements. The four meshes used here are displayed in Figure 18.

The case considered for validation actually corresponds to the uniform flow described in Section 5.1.1. It consists in a point source dispersion within homogeneous isotropic turbulence. To ensure that no particle is lost, the distance between the particles and the cell center (point source) is tracked. In order to have a representative quantity independent of the mesh, the quantity followed is the dimensionless distance d^*. It is defined is defined as $d^* = \frac{\|\mathbf{X}_c - \mathbf{X}_p\|}{\max\limits_{X \in \Omega_c} \|\mathbf{X}_c - \mathbf{X}\|}$ with $\mathbf{X}_c$ the center of gravity of the cell, $\mathbf{X}_p$ the position of the particle and Ω_c the domain defined by the cell c. When particles are properly tracked, this distance is always smaller than 1. However, if one of the face crossed by a particle would be missed, the particle would be permanently lost since it could continue its motion without bound. [2] In such cases, the distance from the cell center could diverge and become much greater than 1.

Results obtained with various meshes are compared using a time scale made dimensionless using the Lagrangian time scale $t^* = t/T_L$ (which is constant for all meshes considered). The results are displayed in Figure 19: we can see that, with 100 000 particles dispersed initially from the point source, the maximum distance to the cell center converges toward 1 but remains always smaller than unity. This proves that the current algorithm is tractable even for 3-D unstructured meshes.

As in Section 5.1, we can also analyze the results obtained for the different second order moments. We focus here on verifying that the results are consistent regardless of the mesh used using a given time step (the role of the time step has been detailed in Section 5.1). The results are displayed in Figure 20: it can be

[2] If the current cell associated to a particle is not correct, the algorithm would not detect other face-crossing events.

seen that all numerical results match the analytical values regardless of the mesh used in such cases. This confirms the accuracy of the algorithm even when 3-D unstructured meshes are used. At this stage, it is also worth noting that 1-D simulation provide the same results as 3-D ones since each of the three directions can be treated independently of the other ones (this is actually a typical characteristic of homogeneous isotropic turbulence).

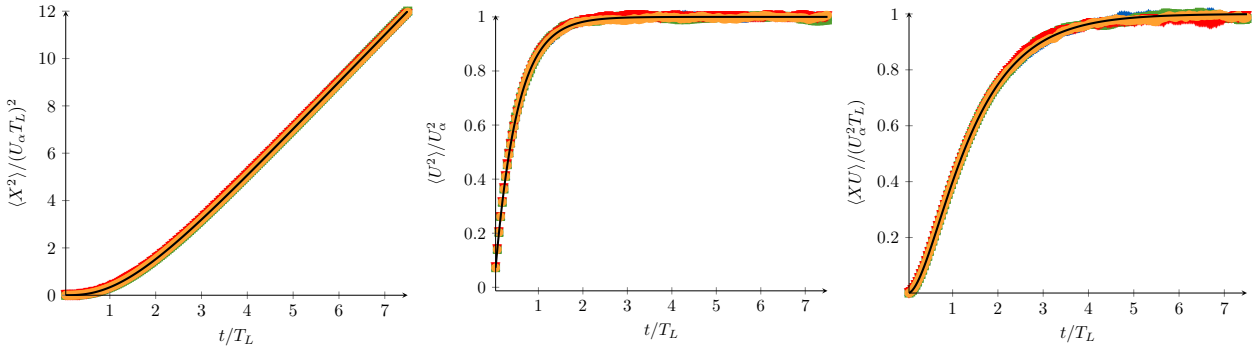


Figure 20: Evolution of $\langle XX\rangle$, $\langle UU\rangle$,$\langle XU\rangle$ for the point source dispersion in the ballistic limit case. Comparison between the analytical solution (black line) and numerical results obtained with the new algorithm for the various meshes. The numerical results are all in agreement with the analytical solution regardless of the mesh used.

Acknowledgements

G. Balvet has received a financial support by ANRT through the EDF-CIFRE contract number 2020/1387. The authors acknowledge the infrastructures at EDF R&D and the CEREA laboratory for providing access to computational resources.

References

[1] F. Archambeau, N. Méchitoua, and M. Sakiz. Code saturne: A finite volume code for the computation of turbulent incompressible flows-Industrial applications. *International Journal on Finite Volumes*, 1(1), 2004.

[2] M. L. Bahlali, C. Henry, and B. Carissimo. On the well-mixed condition and consistency issues in hybrid Eulerian/Lagrangian stochastic models of dispersion. *Boundary-Layer Meteorology*, 174(2):275–296, 2020.

[3] C. E. Brennen and C. E. Brennen. *Fundamentals of multiphase flow*. Cambridge university press, 2005.

[4] S. Chibbaro and J.-P. Minier. A note on the consistency of hybrid Eulerian/Lagrangian approach to multiphase flows. *International Journal of Multiphase Flow*, 37(3):293–297, 2011.

[5] R. Clift, J. R. Grace, and M. E. Weber. *Bubbles, drops, and particles*. Dover Civil and Mechanical Engineering Series. Dover Publications, 2005.

[6] T. D. Dreeben and S. B. Pope. Wall-function treatment in pdf methods for turbulent flows. *Physics of Fluids*, 9(9):2692–2703, 1997.

[7] R. Eymard, G. Henry, R. Herbin, F. Hubert, R. Klöfkorn, and G. Manzini. 3D benchmark on discretization schemes for anisotropic diffusion problems on general grids. In *Finite Volumes for Complex Applications VI Problems & Perspectives*, pages 895–930. Springer, 2011.

[8] C. W. Gardiner. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985.

[9] D. Haworth and S. Pope. A pdf modeling study of self-similar turbulent free shear flows. *The Physics of fluids*, 30(4):1026–1044, 1987.

[10] D. C. Haworth. Progress in probability density function methods for turbulent reacting flows. *Progress in Energy and combustion Science*, 36(2):168–259, 2010.

[11] D. C. Haworth and S. B. Pope. A generalized Langevin model for turbulent flows. *The Physics of fluids*, 29(2):387–405, 1986.

[12] R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. crc Press, 1988.

[13] A. Innocenti, R. O. Fox, and S. Chibbaro. A Lagrangian probability-density-function model for turbulent particle-laden channel flow in the dense regime. *Physics of Fluids*, 33(5):053308, 2021.

[14] A. Innocenti, N. Mordant, N. Stelzenmuller, and S. Chibbaro. Lagrangian stochastic modelling of acceleration in turbulent wall-bounded flows. *Journal of Fluid Mechanics*, 892, 2020.

[15] R. Löhner and J. Ambrosiano. A vectorized particle tracer for unstructured grids. *Journal of Computational Physics*, 91(1):22–31, 1990.

[16] J.-P. Minier. On Lagrangian stochastic methods for turbulent polydisperse two-phase reactive flows. *Progress in Energy and Combustion Science*, 50:1–62, 2015.

[17] J.-P. Minier. Statistical descriptions of polydisperse turbulent two-phase flows. *Physics Reports*, 665:1–122, 2016.

[18] J.-P. Minier. A methodology to devise consistent probability density function models for particle dynamics in turbulent dispersed two-phase flows. *Physics of Fluids*, 33(2):023312, 2021.

[19] J.-P. Minier, R. Cao, and S. Pope. Comment on the article "An effective particle tracing scheme on structured/unstructured grids in hybrid finite volume/PDF Monte Carlo methods" by Li and Modest. *Journal of Computational Physics*, 186(1):356–358, 2003.

[20] J.-P. Minier, S. Chibbaro, and S. B. Pope. Guidelines for the formulation of Lagrangian stochastic models for particle simulations of single-phase and dispersed two-phase turbulent flows. *Physics of Fluids*, 26(11):113303, 2014.

[21] J.-P. Minier and E. Peirano. The pdf approach to turbulent polydispersed two-phase flows. *Physics reports*, 352(1-3):1–214, 2001.

[22] J.-P. Minier, E. Peirano, and S. Chibbaro. Weak first-and second-order numerical schemes for stochastic differential equations appearing in Lagrangian two-phase flow modeling. *Monte Carlo Methods and Applications*, 9(2):93–133, 2003.

[23] J.-P. Minier, E. Peirano, and S. Chibbaro. PDF model based on Langevin equation for polydispersed two-phase flows applied to a bluff-body gas-solid flow. *Physics of fluids*, 16(7):2419–2431, 2004.

[24] J.-P. Minier and J. Pozorski. Derivation of a PDF model for turbulent flows based on principles from statistical physics. *Physics of Fluids*, 9(6):1748–1753, 1997.

[25] J.-P. Minier and J. Pozorski. Wall-boundary conditions in probability density function methods and application to a turbulent channel flow. *Physics of Fluids*, 11(9):2632–2644, 1999.

[26] T. Möller and B. Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.

[27] M. Muradoglu and S. Pope. Local time-stepping algorithm for solving probability density function turbulence model equations. *AIAA journal*, 40(9):1755–1763, 2002.

[28] H. Öttinger. *Stochastic Processes in Polymeric Fluids: Tools and Examples for Developing Simulation Algorithms*. Springer Berlin Heidelberg, 1996.

[29] E. Peirano, S. Chibbaro, J. Pozorski, and J.-P. Minier. Mean-field/PDF numerical approach for polydispersed turbulent two-phase flows. *Progress in energy and combustion science*, 32(3):315–371, 2006.

[30] E. Peirano and J.-P. Minier. Probabilistic formalism and hierarchy of models for polydispersed turbulent two-phase flows. *Physical Review E*, 65(4):046301, 2002.

[31] S. Pope. Application of the velocity-dissipation probability density function model to inhomogeneous turbulent flows. *Physics of Fluids A: Fluid Dynamics*, 3(8):1947–1957, 1991.

[32] S. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[33] S. Pope and Y. Chen. The velocity-dissipation probability density function model for turbulent flows. *Physics of Fluids A: Fluid Dynamics*, 2(8):1437–1449, 1990.

[34] S. B. Pope. Lagrangian PDF methods for turbulent flows. *Annual review of fluid mechanics*, 26(1):23–63, 1994.

[35] S. B. Pope. On the relationship between stochastic Lagrangian models of turbulence and second-moment closures. *Physics of Fluids*, 6(2):973–985, 1994.

[36] P. P. Popov, R. McDermott, and S. B. Pope. An accurate time advancement algorithm for particle tracking. *Journal of Computational Physics*, 227(20):8792–8806, 2008.

[37] J. Pozorski and J.-P. Minier. On the Lagrangian turbulent dispersion models based on the Langevin equation. *International Journal of Multiphase Flow*, 24(6):913–945, 1998.

[38] K. K. Sabelfeld and N. A. Simonov. *Random fields and Stochastic Lagrangian models: analysis and applications in turbulence and porous media*. Walter de Gruyter, 2012.

[39] D. P. Schmidt and C. Rutland. A new droplet collision algorithm. *Journal of Computational Physics*, 164(1):62–80, 2000.

[40] H. Sigurgeirsson, A. Stuart, and W.-L. Wan. Algorithms for particle-field simulations with collisions. *Journal of Computational Physics*, 172(2):766–807, 2001.

[41] S. Subramaniam and D. Haworth. A probability density function method for turbulent mixing and combustion on three-dimensional unstructured deforming meshes. *International Journal of Engine Research*, 1(2):171–190, 2000.

[42] J. Xu and S. Pope. Assessment of numerical accuracy of PDF/Monte Carlo methods for turbulent reacting flows. *Journal of Computational Physics*, 152(1):192–230, 1999.