New spatial decomposition method for accurate, mesh-independent agglomeration predictions in particle-laden flows

Kerlyns Martínez Rodríguez[a,*], Mireille Bossy[a], Radu Maftei[a], Seyedafshin Shekarforush[a], Christophe Henry[a]

[a] *Université Côte d'Azur, Inria (CaliSto Team), Sophia-Antipolis, France*

Abstract

This article presents a new data-driven spatial decomposition algorithm that allows the splitting of a domain containing point particles into elementary cells, each cell containing a spatially-uniform distribution of particles. For that purpose, the algorithm relies on the use of statistical information for the spatial distribution of particles and then extracts an optimal spatial decomposition. After evaluating the convergence and accuracy of the algorithm on homogeneous and inhomogeneous cases, this optimal spatial decomposition is applied to study the case of particle agglomeration. Indeed, in CFD context, recent developments on numerical simulations of particle agglomeration in complex and turbulent flows increasingly resort to Euler-Lagrange approaches. These methods are coupled with population balance equation (PBE)-like algorithms to compute agglomeration inside each cell of the Eulerian mesh. One of the key issues with such approaches is related to the spatially-uniform condition, i.e. agglomeration should be computed on a set of particles that are uniformly distributed locally in each cell. Yet, CFD simulations in realistic industrial/environmental cases often involve non-homogeneous concentrations of particles (due to local injection or accumulation in specific regions). We show that more accurate and mesh-independent predictions of particle agglomeration are made possible by the application of this new data-driven spatial decomposition algorithm.

Keywords: Spatial decomposition, Uniformity criteria, Agglomeration, Multiphase flow, Population Balance Equation, Turbulence, Clustering

1. Introduction

1.1. General context: particle agglomeration

Agglomeration (also called aggregation, flocculation, coalescence or coagulation) refers to the process that leads to the assembly of particles or molecules dispersed in a fluid. This process is very common in nature with implications in geo-morphology (role of floc sedimentation in river delta formation [1]), meteorology (formation of clouds [2] or droplet growth due to turbulent collisions [3, 4]), and astrophysics (growth of dust aggregates in planet formation [5]). Agglomeration also plays a role in a number of industrial applications, such as in waste-water treatment facilities (separation of solids through settling of aggregates [6] or flotation processes [7]), in combustion systems (agglomeration in fluidized beds [8] and growth of soot particles [9]) or in oil/sludge refining (use of polymer conditioning to control agglomeration [10]). Agglomeration also has implications in medical fields (protein aggregation being linked to some degenerative diseases [11]) as well as in the food industry (dry milk powders [12]). From this brief overview, it appears that particles can take various forms (molecules, polymers, solids, droplets, bubbles, etc.) and that agglomeration involves a wide range of temporal and spatial scales (from molecular scales with proteins up to astrophysical scales with planetoids). According to the IUPAC [13], agglomeration of solid particles is the process of contact and

*Corresponding author
Email address: kerlyns.martinez-rodriguez@inria.fr (Kerlyns Martínez Rodríguez)

adhesion whereby dispersed solid particles are held together by weak physical interactions ultimately leading to phase separation by the formation of precipitates of larger than colloidal size – i.e. solid particles with sizes ranging from a few nanometres up to a few micrometres [14].

Following this definition, we focus here on the prediction of agglomeration of solid colloids in suspension in complex turbulent flows.

1.2. Existing models for particle agglomeration

Significant efforts have been devoted recently to the simulation of particle-laden flows. This led to the development of a wide variety of models (interested readers are referred to existing reviews for more details [15]). In the following, the main approaches are briefly summarised and organised with respect to their level of description of particle agglomeration (from microscopic to macroscopic approaches):

a. N-particle tracking (or individual particle tracking) with collision detection;
   As implied by its name, this is a microscopic approach with a very fine level of description that consists in tracking simultaneously a large number of particles carried by a flow and detecting collisions between particles. For that purpose, the equations of particle motion are solved explicitly considering all forces and torques acting on/between particles. Such Lagrangian tracking algorithms are often coupled with fine-scale simulations of the fluid, such as direct numerical simulation (DNS) of turbulent flows where all the scales of turbulence are fully resolved. Particle collisions are then monitored using specific detection algorithms based either on an overlap criterion (i.e. the two spherical particles lie on top of each other [16] at the beginning/end of the time step) or on a geometric criterion valid in the ballistic regime (i.e. particles follow straight lines during the time step) [17].
   Such approaches allow the number of agglomeration events to be obtained directly as a result of the simulation. Besides, they provide detailed information on local properties of agglomeration (e.g. shape of agglomerates, spatial/temporal correlations between collisions) [18]. However, due to their high computational costs and to the fine level of description required, their use is currently limited to idealized cases or specific small-scale applied cases.
b. One-particle tracking with given collision frequencies.
   More recently, new methods have emerged based on the coupling between Eulerian simulations of turbulent flows and one-particle pdf models for the particle phase (interested readers are referred to existing reviews on the topic, e.g. [19, 20, 21, 22]). The key idea in such simulations is to have a separate treatment of the transport step and of the collision step over each small time step, leading to time-splitting algorithms. This means that particles are moved through the domain without interacting with each other in the first step (transport without collision) while particles interact with each other without changing their positions in the second step (collision without transport). This separate treatment of the 'transport' and 'collision' steps relies on a kind of Bird-like algorithm (interested readers are referred to the first DSMC methods of the early 60s [23, 24, 25]). Regarding the 'transport step', the key difference with N-particle tracking approaches is that one-point pdf methods have a 'coarser' level of description. The equations of particle motion are indeed expressed using effective or modelled forces, i.e. particles do not interact with each other but rather with a mean-field obtained from the whole set of particles by statistical averaging (more details can be found in [19]). This approach is thus consistent with the turbulence models that are used for the fluid phase, such as Reynolds-Averaged Navier Stokes models (RANS) which provide information on mean-field quantities. It should be noted here that, in order to avoid the high computational costs associated with the tracking of all real particles, the notion of parcel is often used (one 'numerical' parcel is a statistical representation of a large number of real particles).
   The treatment of the 'collision step' brings out new difficulties since a consistent model for the description of particle agglomeration has to be chosen. Indeed, the stochastic nature of particle motion prevents the direct calculation of particle agglomeration with standard techniques used in N-particle tracking approaches (e.g. overlap or geometric criteria). Two types of detection models for particle agglomeration have been used. First, probabilistic collision algorithms are based on the evaluation of the probability of collision between a pair of particles, depending on a number of parameters (initial and final separation distance, diffusion coefficient) [26, 27]. Second, mesh-based algorithms consist in treating the collision

between all pairs of parcels within a given cell [28, 29]. The probability of collision between a pair of parcels during a time step is then derived from geometric considerations (using the size of both particles and the relative velocity between them). This algorithm has been later refined to treat collision only between a subset of candidates, which are randomly selected among the possible particle pairs within each cell [30]. Similar formulations have been used recently replacing the probability of collisions between pairs of particles/parcels with the discretised PBE formula [31].

c. Population Balance Equations (PBE):
PBE are macroscopic approaches that describe the evolution in time of mean-field quantities. More precisely, they are derived from the approach initiated by Smoluchowski [32], which models changes in the number density of a population of particles due to agglomeration within a control volume $\mathbb{V}_C$. As particles agglomerate, the distribution of sizes changes according to the following integro-partial-differential equation [33, 34]:

$$\frac{\partial n(\mathbb{v},t)}{\partial t} = \frac{1}{2}\int_0^{\mathbb{v}} \alpha(\mathbb{v}-\mathbb{w},\mathbb{w})\beta(\mathbb{v}-\mathbb{w},\mathbb{w})n(\mathbb{v}-\mathbb{w},t)n(\mathbb{w},t)d\mathbb{w} - \int_0^{\infty} \alpha(\mathbb{v},\mathbb{w})\beta(\mathbb{v},\mathbb{w})n(\mathbb{v})n(\mathbb{w})d\mathbb{w}, \quad (1)$$

where $n(\mathbb{v},t)$ is the number density of particles of volume $\mathbb{v}$ at time t, α is the collision efficiency and β the collision frequency kernel between particles of volume $\mathbb{v}$ and $\mathbb{w}$. The first term on the r.h.s. of Eq. (1) corresponds to the formation of new aggregates of volume $\mathbb{v}$ produced by the collision between pairs of smaller aggregates (such that the sum of their volumes equals $\mathbb{v}$) while the second term on the r.h.s. accounts for the loss of particles of volume $\mathbb{v}$ due to their agglomeration with other particles. The factor $\frac{1}{2}$ in the first term is required to avoid counting twice each collision.
PBE approaches allow to perform fast evaluations of the kinetics of particle agglomeration at the macroscopic level. Besides, PBE describes the time evolution of mean-field quantities and, as such, is compatible with the level of description used in standard CFD (Computational Fluid Dynamics) simulations, which resort to turbulence models describing mean-field fluid quantities. For these reasons, PBE formulations are widely used in CFD simulations where the fluid is described using Eulerian or MOM (method-of-moment) approaches [35, 34]. However, PBE formulations suffer from several limitations that have been well identified in the literature (interested readers are referred to [34, 36] for more details).

In this paper, we adopt the viewpoint of Lagrangian one-point pdf methods for the tracking of parcels coupled to an Eulerian simulation of the fluid phase (using RANS turbulence models) and using a mesh-based PBE-inspired algorithm for the detection and treatment of particle agglomeration. Nevertheless, some of the conclusions drawn are also applicable to other methods that rely on PBE formulations for particle agglomeration.

1.3. Shortcomings of current models

The methods relying on the use of a mesh-based PBE-inspired algorithm for the detection of particle collisions suffer from several limitations. In particular, several of these shortcomings are related to the use of PBE formulations (interested readers are referred to [34, 36] for more details), among which:

a. Only binary collisions are considered.
b. There is no statistical correlation between individual particles of volume $\mathbb{v}$ and $\mathbb{w}$ at their respective positions, so that the number of collision events can be written as the product $\beta(\mathbb{v}-\mathbb{w},\mathbb{w})n(\mathbb{v}-\mathbb{w},t)n(\mathbb{w},t)$.
c. The population of particles is uniformly distributed in each spatial region considered (e.g. cells in a mesh) so that no external coordinates are involved in the 'collision step'. Note that this assumption is sometimes referred to as a 'well-mixed condition' [34] but, since it is widely used in the CFD community for stochastic models of turbulent diffusion/dispersion with a different meaning, we will refer to it as spatially-uniform condition within the rest of the paper to avoid confusion.
d. The velocity field acting on particles is constant in each spatial region so that no external coordinates are involved in the 'collision step'.

e. Information on the collision efficiency α and on the collision frequency kernel β are required. Approximations for these two quantities are available in simple cases (e.g. purely Brownian-induced, shear-induced or gravity-induced agglomeration). However, such approximations are missing in cases representative of realistic situations (implying a mixing of turbulence, Brownian motion and gravity). This is due to the intricate coupling between the particle dynamics and physico-chemical interactions, which does not favour the splitting of the transport step and the collision step (both in PBE formulations and in one-particle pdf approaches).

The present paper is mainly motivated by refining the accuracy of the computations of particle agglomeration using such approaches by addressing specifically the issue *c.* related to the spatially-uniform condition. In practice, this spatially-uniform condition has to be fulfilled locally for each computational cell, meaning that the particles have to be uniformly distributed in each cell where the method is applied (this is actually a necessary but not sufficient condition). However, numerical parcels tracked within complex flows are not necessarily uniformly distributed in the domain: particles can indeed be injected locally in the system, leading to significant gradients of particle concentration at the system scale which can translate in non-uniform concentration at the cell scale (see for instance spray dryer injection in [29]). As depicted in Fig. 1, such non-uniform spatial distributions of particles can induce severe mis-estimations of the number of agglomeration events generated (see also [37]). It should be noted that this issue is common to other Euler-Lagrange approaches relying on PBE formulations, as in combustion processes [9], meaning that the present developments can also be used in other domains/applications. Moreover, a spatial-decomposition algorithm can prove useful in any Euler-Lagrange approach that relies on two independent meshes (one for the fluid phase and another one for the particle phase), since it will allow the design of a mesh for the particle phase that respects some criteria allowing for more accurate and mesh-independent results.

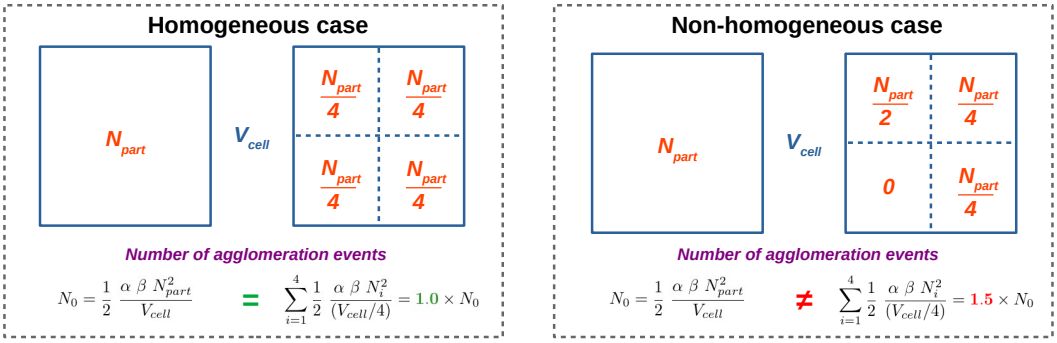


Figure 1: Sketch showing the impact of non-uniform distributions on the number of agglomeration events estimated either in one single cell or in four smaller ones.

Several attempts have been made to reduce the mesh-dependency of the results [38, 39, 40, 41, 42]. For instance, Zhang et al. [41] rely on the mesh used for the CFD simulations and reduce the mesh-dependence of O'Rourke method [28] by computing the agglomeration over a volume comprising the cell and its surrounding neighbours. The drawback of this approach lies in the use of a CFD-based mesh which has no connections to the spatial distribution of particles/parcels. Another approach consists in generating a specific mesh for the computation of particle agglomeration, as is done in [30, 38, 40] (this is called the NTC algorithm, for No-Time Counter). More precisely, in the original NTC algorithm [30], Schmidt and Rutland generate a cylindrical mesh with a randomly chosen orientation every time step and then apply their collision algorithm on this spatial decomposition. The random orientation of the mesh reduces mesh dependency by allowing the interaction between nearby particles that were in two different cells at one given time step. This has been later on improved by Hou and Schmidt [39], who introduced an adaptive collision mesh method. It consists in creating an initial cube that includes all the particles/parcels (the orientation of this cube is again randomly generated), which is then split equally in 8 half-length cubic-cells. This partitioning is applied iteratively for each existing cubic cell as long as the number of particles in a cubic cell is larger than a given

threshold. Despite providing good results (especially due to the random orientation of the initial cube that prevents static meshes), this method suffers from two limitations: first, the method blindly splits each cubic cell into eight smaller cubic cells with a criterion on the number of particles only, i.e. without taking into account the spatial distribution of particles in the cell; second, the end-iteration criterion has to be fixed a priori (note that, since it corresponds to the maximum number of particles inside a cubic cell, it implies that the results' accuracy is governed by this threshold). This method has been extended later to the case of parcels [40].

1.4. Objectives of this study

Considering the mesh-dependence issues described above and the existing methods to reduce it, the aim of this study is to come up with a specific algorithm that generates a mesh accounting for the spatial repartition of particles while respecting the spatially-uniform condition required for the computation of particle collision/agglomeration. In particular, the following objectives need to be met:

(i) the algorithm has to detect local variations in the spatial distribution of a given set of particles/parcels in a volume;
(ii) the algorithm has to generate a spatial decomposition of the volume based on the spatially-uniform condition;
(iii) the algorithm should be robust regardless of the input data (set of particles in a volume);
(iv) the algorithm has to be compatible with one-point pdf approaches for particle/parcel tracking;
(v) the algorithm should not induce mesh-dependence when applied to study particle agglomeration;

Drawing on these objectives, several remarks can be made. First, objective (iii) implies that the algorithm should detect non-uniform spatial distributions using information coming from the actual data (i.e. a set of particles in a volume). As a result, the algorithm should not rely on user-defined parameters (which can be case-dependent). Second, objective (iv) suggest that the algorithm should rely on statistical information regarding the spatial distributions of particles/parcels instead of relying on information regarding the exact distance between these elements (two-point correlations are indeed not included in current one-point pdf approaches). As a result, we cannot rely here on existing clustering techniques that have been developed in the context of image analysis and signal processing [43, 44]. Several clustering techniques allow indeed to process data points to identify regions with non-uniform spatial distributions (e.g. K-means algorithm, DBCLASD algorithm, DBSCAN algorithm). Clustering is also a common issue in the context of inertial particles in turbulent flows (see for instance [45]). Yet, most of these techniques are based on the actual computation of the distance between data points and rely on a user-defined parameter (see Appendix A for an example of the use of DBSCAN in this context). Besides, these methods focus on the classification of data points by identifying clusters (defined as a group of points having enough neighbours in close proximity). As such, they are not adapted to the scope of this study which aims at constructing a spatial decomposition (objective (ii)).

1.5. Layout of the paper

Within this scope, a new data-driven spatial decomposition (D2SD) method is proposed. The paper is organised as follows. First, the new D2SD algorithm is presented in Section 2 for general purposes. Then, numerical results obtained on spatially homogeneous and inhomogeneous cases are detailed in Section 3 together with an analysis of the convergence and accuracy of the algorithm. Finally, in Section 4, the method is adapted to the case of particle agglomeration and applied to practical cases that show the accuracy of the method and the independence of the results to the mesh thus generated.

2. D2SD : A new algorithm for spatial decomposition of non-globally uniform particle distributions

This section describes a new method that detects the presence of non-uniform spatial distributions of point particles and that comes up with an optimal spatial decomposition restoring locally the uniform distribution.

In this section as well as in the next one, we focus on presenting a general spatial decomposition algorithm that can be applied to a range of situations (computation of particle agglomeration or any Euler-Lagrange approach relying on two independent meshes).

2.1. Input of the D2SD

As depicted in Fig. 2, the sole input of the model is a set of positions of point particles inside a volume (e.g. a snapshot of particle positions at a given iteration time in a CFD simulation). The method then detects the presence of spatial inhomogeneities in the particle concentration and treats them according to the aforementioned objectives (see Section 1.4). The complete algorithm is composed of three steps that we detail in the following:

step 1. apply an a-priori check of the spatially-uniform condition: returns `acceptance` or `rejection`. Our basic proposition in this paper is to use statistical uniformity tests, but step 1 could be extended to any relevant measure of uniformity;

in case step 1 returns `rejection`

step 2. apply the D2SD algorithm to detect and separate non-uniform regions;

step 3. apply an a-posteriori statistical uniformity test on the output of the D2SD algorithm to check if all locally uniform regions have been properly identified. If the uniformity test is rejected, go back to step 2 until the a-posteriori test is accepted;

2.2. Step 1: statistical uniformity test

The key idea of step 1 is to make a first a-priori statistical test to check whether the input set of particles is uniformly distributed or not. According to its size, the input sample of particle position may present irregularities (clusters, voids) even if the sample corresponds to points randomly and independently generated according to a uniform distribution.

In the literature, a vast list of uniformity tests have been proposed in the one-dimensional case (interested readers are referred to D'Agostino and Stephens [46, Chapters 6 and 8], and the references therein). Many of these tests also analyse the spacing of the sample, such as the Greenwood statistic, which under the null hypothesis is expected to have an exponential distribution. The case of uniform distribution test on domains with unspecified boundaries has also been studied. Nevertheless, when it comes to the multidimensional case, normality tests have attracted more attention, and uniformity tests are less established. Following the usual framework within the classical literature of goodness of fit, we consider a sample of n particles inside a regular box of size 1 in d-dimensions and we assume that the sample $\mathbf{X} = \{X^1, X^2, \dots, X^n\}$ in $[0,1]^d$ is independent and identically distributed (i.i.d). In addition, we set the null hypothesis:

$$H_0\text{: } \mathbf{X} = \{X^1, \dots, X^n\} \text{ are i.i.d according to the uniform distribution } \mathcal{U}([0,1]^d). \tag{2}$$

In this context, i.i.d. uniformity distribution is tested with the series of five following tests:

S1-a-b-c The symmetric, the centred and the star discrepancy tests;

S1-d The Henze-Zirkler normality test for the – also i.i.d – sample $\left\{\Phi^{-1}(X^1), \Phi^{-1}(X^2), \dots, \Phi^{-1}(X^n)\right\}$, based on an inverse transform sampling intuition, with Φ being the cumulative distribution function of a standard normal random variable;

S1-e The Pearson statistic to test the independence of the position of the particles.

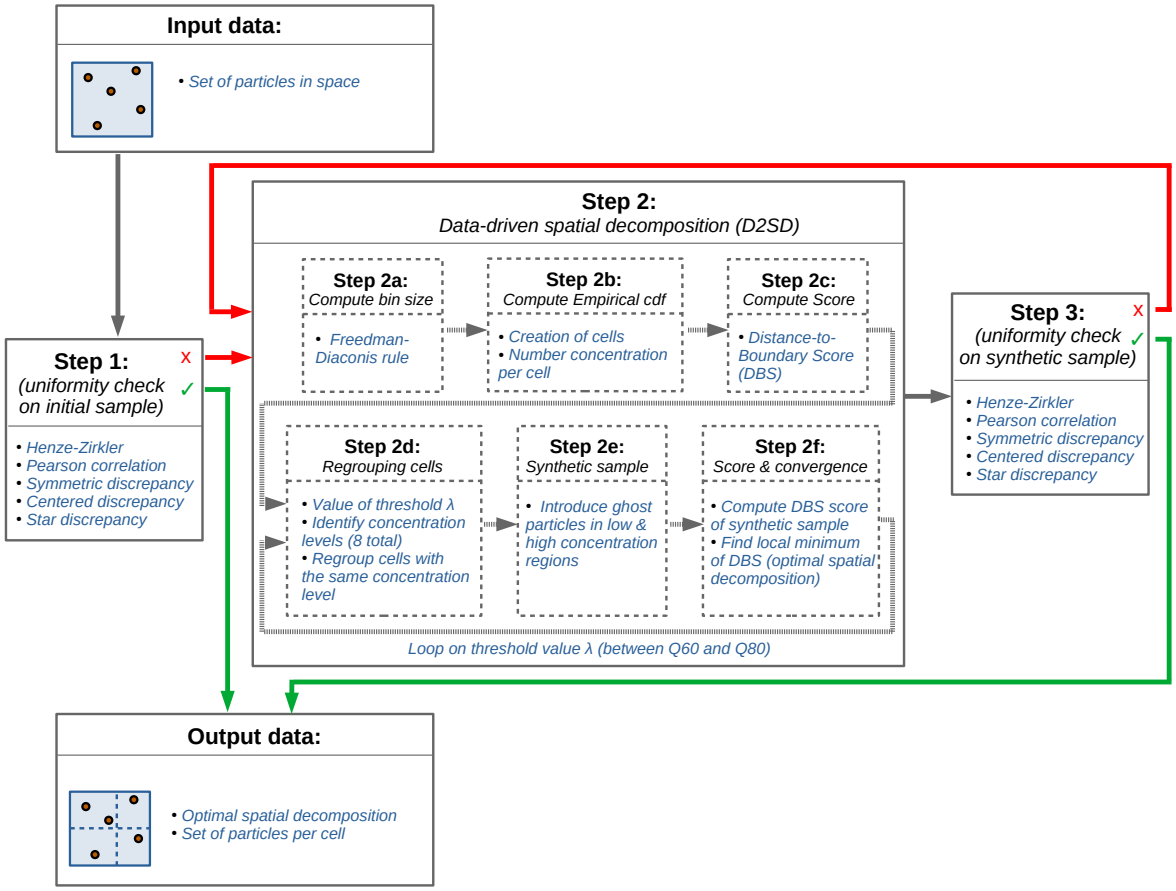


Figure 2: Sketch summarising the three steps of the algorithm

Discrepancy tests. In quasi-Monte Carlo methods for numerical computation of multiple integrals, a discrepancy criterion is used for measuring if a given set of points is uniformly scattered. In Liang et al [47], new statistics are proposed to test the uniformity of a random sample, of size n and dimension $d \geq 2$, in the unit hyper-cube. These statistics are based on the following discrepancy measure $\mathcal{D}$ of a set of points $\mathbf{X}$. Given a positive constant κ and μ, a function in the space $\{f : \ f' \in L^{\infty}[0,1] \text{ and } \int_0^1 f(x)dx = 0\}$,

$$\begin{aligned}\mathcal{D}(\mathbf{X})^2 =& M^d - \frac{2}{n}\sum_{k=1}^{n}\prod_{m=1}^{d}\left(M + \kappa^2\mu(X_m^k)\right) \\ &+ \frac{1}{n^2}\sum_{k,l=1}^{n}\prod_{m=1}^{d}\left[\!\left[M + \kappa^2\left(\mu(X_m^k) + \mu(X_m^l) + \tfrac{1}{2}B_2([\![X_m^k - X_m^l]\!]) + B_1(X_m^k)B_1(X_m^l)\right)\right]\!\right],\end{aligned} \tag{3}$$

where $[\![x]\!]$ stands for the decimal part of x (i.e. $[\![x]\!] = x - \lfloor x \rfloor$), the functions B_1 and B_2 are the first and second order Bernoulli polynomials, respectively and

$$M = 1 + \kappa^2 \int_0^1 \left(\mu'(x)\right)^2 dx. \tag{4}$$

The choice of parameters κ and μ in (3) determines the type of discrepancy we are interested in testing. Following [47, pp. 4-5], three tests are considered here:

– The symmetric discrepancy, with $\kappa = 2$ and $\mu(x) = -\frac{1}{2}B_2(x)$.

– The centred discrepancy, with $\kappa = 1$ and $\mu(x) = -\frac{1}{2}B_2(\llbracket x - \frac{1}{2} \rrbracket)$.

– The star discrepancy, with $\kappa = 1$ and $\mu(x) = \frac{1}{6} - \frac{x^2}{2}$. Note that the star discrepancy test is in fact related to the Kolmogorov-Smirnov statistic.

The discrepancy measure $\mathcal{D}$ can be rewritten as

$$\mathcal{D}(\mathbf{X})^2 = M^d - 2U_1 + \frac{n-1}{n}U_2 + \frac{1}{n^2}\sum_{k=1}^{n} g_2(X^k), \tag{5}$$

with $U_1 = \frac{1}{n}\sum_{k=1}^{n} g_1(X^k)$ and $U_2 = \frac{2}{n(n-1)}\sum_{k<l}^{n} h(X^k, X^l)$, for some bounded functions g_1, h and g_2 depending on κ, M and μ (see proof of Theorem 2.1 in [47]). As is proven in [47, Theorem 2.3], under the null hypothesis H_0 in (2), we have the following convergence in distribution toward the normal law:

$$\begin{aligned}\sqrt{n}\begin{pmatrix} U_1 - M^d \\ U_2 - M^d \end{pmatrix} \longrightarrow & \mathcal{N}_2(0, \xi\Sigma), \text{ as } n \longrightarrow \infty, \\ & \text{where } \xi = (M^2 + \kappa^4 \textstyle\int_0^1 \mu^2(x)dx)^d - M^{2d}, \text{ and } \Sigma = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}.\end{aligned} \tag{6}$$

From this convergence, the following statistic is proposed (see [47, Corollary 2.5]):

$$A_n := \sqrt{n}\frac{U_1 - M^d + 2(U_2 - M^d)}{5\sqrt{\xi}}, \tag{7}$$

which, under H_0, converges in distribution to a standard 1D-normal random variable, of cumulative distribution function $x \mapsto \Phi(x)$. Then, by evaluating A_n, we can reject the null hypothesis H_0 if its value is too large with respect to the quantiles associated with a standard normal random variable. More precisely, we introduce a confidence level ε such that if $\texttt{p-value} = 1 - \Phi(|A_n|) \leq \varepsilon$, then the discrepancy test rejects the uniformity of the sample $\mathbf{X}$.

Normality test. Parallel to discrepancy measures, we use a second uniformity test based on the inverse transform sampling method. When generating random numbers from any probability distribution with cumulative distribution function F, one can consider the transformation $F^{-1}(\mathcal{U}[0,1])$. By virtue of this inverse transform sampling method, under the null hypothesis H_0, the sample of size n

$$\mathbf{Y} = \left\{(\Phi^{-1}(X_j^1), j = 1, \ldots, d), \ldots, (\Phi^{-1}(X_j^n), j = 1, \ldots, d)\right\} \text{ valued in } \mathbb{R}^d$$

comes from a normal distribution. Thus, we rewrite the null hypothesis into:

$$\widetilde{H_0}\text{: } \mathbf{Y} \text{ (of size } n) \text{ are i.i.d according to a normal distribution on } \mathbb{R}^d. \tag{8}$$

The advantage in testing the hypothesis $\widetilde{H_0}$ is that we have at our disposal a greater number of possible statistics. In this paper, we have opted for the implementation of the Henze-Zirkler test (see e.g. [48]) based on a L^2-weighted distance between the empirical characteristic function of the sample and the characteristic function of a normal random variable. Rigorously, the Henze-Zirkler test is given by:

$$HZ_{\widetilde{\beta}} := n\left(4\ \delta_{\{S_n \text{ is singular}\}} + D_{n,\widetilde{\beta}}\ \delta_{\{S_n \text{ is non-singular}\}}\right), \tag{9}$$

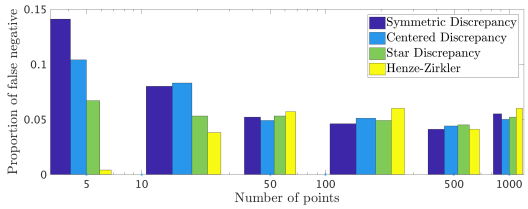
where S_n is the sample covariance of $\mathbf{Y}$, $\widetilde{\beta} = \frac{1}{\sqrt{2}}\left(\frac{n(2d+1)}{4}\right)^{1/(d+4)}$, and

$$D_{n,\widetilde{\beta}} = \frac{1}{n^2}\sum_{m,k=1}^{n}\exp\left\{-\frac{\widetilde{\beta}^2\|Y^m - Y^k\|^2}{2}\right\} + (1+2\widetilde{\beta}^2)^{-d/2} - \frac{2(1+\widetilde{\beta}^2)^{-d/2}}{n}\sum_{m=1}^{n}\exp\left\{-\frac{\widetilde{\beta}^2\|Y^m\|^2}{2(1+\widetilde{\beta}^2)}\right\},$$

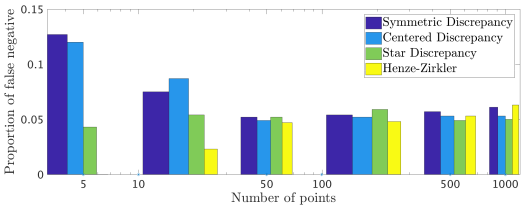
for $Y^i := S_n^{-1/2}\left(\Phi^{-1}(X^i) - \overline{\Phi^{-1}(X)}\right)$ in $\mathbb{R}^d$, with $(i = 1, 2, \ldots, n)$ and $\overline{Z}$ denotes the empirical mean of the random sample $\{Z^i;\ i = 1, 2, \ldots, n\}$.

Under the null hypothesis, the statistic $HZ_{\widetilde{\beta}}$ has approximately a log-normal distribution. Thus, we evaluate $\texttt{p-value} = 1 - \Phi(\log HZ_{\widetilde{\beta}})$ and compare against the level of confidence ε. When $\texttt{p-value} < \varepsilon$, the Henze-Zirkler test rejects normality.

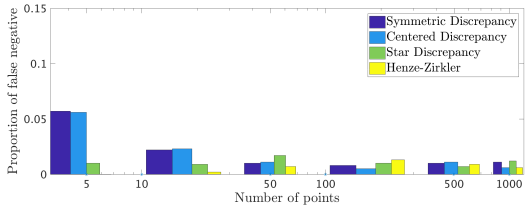
Acceptance/rejection decision. The accuracy of these tests is assessed using 1000 uniform samples generated randomly of size $n = 5, 10, 50, 100, 500, 1000$ and dimension $d = 2, 3$. The results of these tests are shown in Fig. 3 for two different confidence levels: $\varepsilon = 0.05$ and $\varepsilon = 0.01$. As expected, it can be seen that the proportion of samples for which the uniformity test has been rejected remains very low. Yet, despite a good accuracy of these tests, it should be noted that the accuracy decreases as the size of the sample becomes too small. The statistics described above were also tested with non-uniform samples with similar rejection levels (figures not shown here).



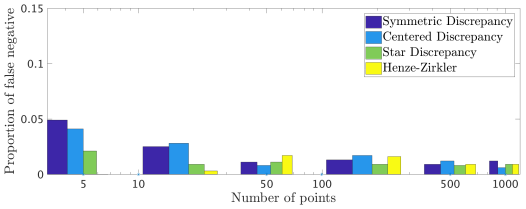
(a) Proportion of false negative in 2D, $\varepsilon = 0.05$



(b) Proportion of false negative in 3D, $\varepsilon = 0.05$



(c) Proportion of false negative in 2D, $\varepsilon = 0.01$



(d) Proportion of false negative in 3D, $\varepsilon = 0.01$

Figure 3: Proportion of uniformity tests rejected for 1000 uniform samples generated randomly with size $n = 5, 10, 50, 100, 500, 1000$ in 2D (left) and 3D (right) cases, using statistic A_n in (7), based on discrepancy measures, and Henze-Zirkler test $HZ_{\widetilde{\beta}}$ in (9).

Considering the results obtained in these numerical experiments, we proceed to apply a majority voting criterion when implementing the statistics A_n (for the three discrepancies discussed above) and the Henze-Zirkler statistic $\log HZ_{\widetilde{\beta}}$, verifying the independence assumption by using the well-known Pearson correlation coefficient (see e.g. [49]). In this context, we reject the null hypothesis H_0 if two (or more) tests reject the uniformity of the sample.

2.3. Step 2: spatial decomposition (D2SD)

In the case where the uniformity test is rejected in step 1, we proceed to split the space according to a partitioning so that each sub-domain contains a given level of particle concentration. This spatial decomposition should distinguish between areas of high concentration (where there is clustering) and areas of low concentration (almost free of particles). As a result, the concentration in each sub-domain is expected to fulfill locally the spatially-uniform condition. For that purpose, the D2SD algorithm consists of the following six steps (see also Fig. 2):

S2-a Computation of the bin size

S2-b Computation of the empirical pdf

S2-c Computation of the score of the initial sample

S2-d Detection and merging of bins according to concentration level.

S2-e Creation of a synthetic sample (with ghost particles)

S2-f Computation of the score of the synthetic sample and convergence criterion

Steps S2-d & S2-e & S2-f are repeated until an optimal regrouping of 'bin-cells' is obtained. In the following, more details about each step are provided.

Step 2-a: Bin size. The bases of the proposed algorithm are established in the analysis of the probability density function (pdf) associated with the sample. The pdf provides information about how the particles are scattered and how the local concentration is within the domain. Besides, the use of the sample pdf instead of exact distances between particles is in line with objectives (iii) and (iv) described in Section 1.4. A suitable approximation of the pdf is needed, and the most natural way to construct this approximation is to consider histograms. In this context, the choice of an approximately optimal bin size is required.

Among the contemplated techniques to select the width of the bins of an histogram, the Freedman-Diaconis rule [50] has the best performance in our case. This rule is characterized by minimizing the difference between the area under the empirical pdf and the analytic pdf using the statistical properties of the observations. Following the Freedman and Diaconis rule [50], we define the bin size for the jth-coordinate as follows:

$$h_j(\mathbf{X}) = 2\frac{\texttt{iqr}(\mathbf{X}_j)}{\sqrt[3]{n}}, \tag{10}$$

where $\texttt{iqr}(\cdot)$ is the interquartile range (the distance between the 75th and 25th percentiles) of a random sample, which corresponds to a measure of statistical dispersion.

Step 2-b: Empirical pdf. Following the choice of a proper bin size in Step 2-a, the empirical pdf is computed. This results in a d-dimensional tensor, where each entry corresponds to the number density in each 'bin-cell' formed using this binning procedure (i.e. the number of observations in the bin normalized by n times the volume of the bin).

Step 2-c: Score of the initial sample. We introduce a mathematical measure to quantify how far the data is from a uniform sample. This measure will be called hereafter score function and is used as a convergence criteria for the iterative processes in Steps 2-d, 2-e and 2-f. In the one-dimensional case, an intuitive choice for the score function is the to use the L^1-distance between two cumulative distribution functions. As a first possible extension in the multi-dimensional case, we consider the sum of the projected distance on each of the d directions: denoting $\mathbf{X}_j$ the vector of the jth-coordinate of the d-dimensional random sample $\mathbf{X}$, we define the projected score:

$$d_{L^1}(\mathbf{X}, \mathcal{U}([0,1]^d)) := \sum_{j=1}^{d} \int_0^1 |\bar{F}_{n,\mathbf{X}_j}(x) - F_{\mathcal{U}([0,1])}(x)|dx, \tag{11}$$

where $F_{\mathcal{U}([0,1])}$ is the uniform cumulative distribution function (cdf), and $\bar{F}_{n,\mathbf{X}_j}$ is the empirical cdf associated with $\mathbf{X}_j$:

$$\bar{F}_{n,\mathbf{X}_j}(x) = \frac{1}{n}\sum_{k=1}^{n} \mathbf{1}_{\{X_j^k < x\}}, \qquad \text{for } 1 \le j \le d.$$

However, the accuracy of the projected score degrades quickly when the number of dimensions d increases, due to the loss of information about how the coordinates are related to each other.

As an alternative, we propose to quantify the uniformity of the observations considering the distance to the boundary of the domain and its application to testing multivariate uniformity (see Berrendero, Cuevas

and Vásquez-Grande [51]). In [51], the authors proposed a new test of uniformity based on the relative depth. For an initial random sample $\{X^1, \dots, X^n\}$ within a domain $\mathcal{D}$ of radius $R = \max\{d(x, \partial\mathcal{D}); x \in \mathcal{D}\}$, the relative depth is defined as the one-dimensional sample $\mathbf{Z} = \{Z^1, \dots, Z^n\}$ in $\mathbb{R}$,

$$Z^i = \frac{d(X^i, \partial\mathcal{D})}{R},$$

with $d(X^i, \partial\mathcal{D})$ denoting the Eulerian distance between the point X^i in $\mathcal{D}$ and the boundary of the domain $\partial\mathcal{D}$. Following the notation in [51], we define the distance-to-boundary score as:

$$d_b(\mathbf{X}, \mathcal{U}) := \int_0^1 |G_{n,\mathbf{Z}}(z) - H_{\partial\mathcal{D},\mathcal{U}}(z)| dz, \tag{12}$$

where $G_{n,\mathbf{Z}}$ stands for the empirical cdf of the relative depth and $H_{\partial\mathcal{D},\mathcal{U}}$ is the cdf of the relative depth associated with a uniform sample. In the case of a cube $\mathcal{D} = \Pi_{1\leq i\leq d}[a_i, b_i]$, it takes the following form (see [51, Theorem 1]):

$$H_{\partial\mathcal{D},\mathcal{U}}(z) = 1 - \prod_{i=1}^{d}(1 - \frac{2R}{b_i - a_i}z), \text{ for } 0 \leq z \leq 1,$$

where $R = \min_{1\leq i\leq d}\left\{\frac{b_i - a_i}{2}\right\}$.

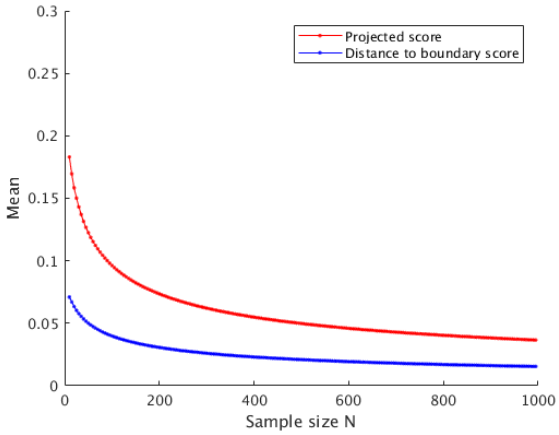
To analyse the behaviour of the scores defined in (11) and (12), 1000 samples of size $n = 10, 20, \dots, 1000$ of uniform distributed points in the cube $[0, 1]^{2,3}$ have been generated, in order to compute the mean and variance of the resulting scores in terms of n. Figure 4 shows the results of this numerical experiment: the decrease of the mean score values in terms of n is a typical $\mathcal{O}(1/\sqrt{n})$ decay, although the accuracy of the score is tied with the dimension of the observations. Nevertheless, the standard deviation of the projected score is unsurprisingly higher and limits the use of this measure to large values of n. In the implementation of the scores, we use a trapezoid method to approximate the integrals in (11) and (12).

In line with the previous comparison, in the coming numerical analysis, we define the score on the dataset to be the d_b score. This choice is well adapted to the geometry of the considered domain (a 2D-3D cube), but can be adapted with a combination of several other distances according to the use-cases.

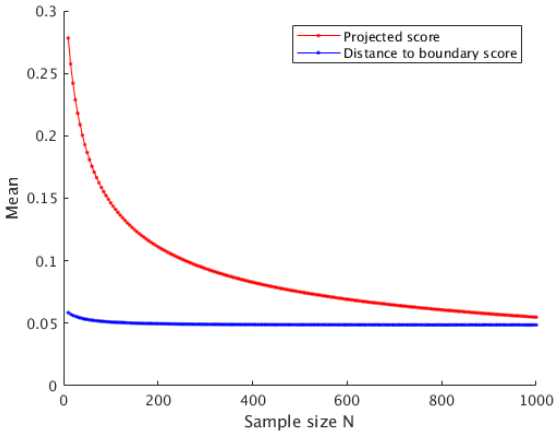
Step 2-d: Regrouping 'bin-cells'. To obtain an optimal mesh decomposition, the next step consists in putting together 'bin-cells' identified as having the same level of particle concentration. This step is performed to reduce the statistical noise encountered when dealing with cells having a small amount of particles. To that end, we have to introduce different levels of the particle presence density and classify the cells with respect to these levels. This issue rises to the following difficulty: even for a simple classification of cells according to three levels (i.e. low-, ambient- and high-concentrations), what are the thresholds to separate each level? A threshold can hardly be defined a priori since it depends on the actual distribution of the pdf-values. For that reason, we have opted for the percentile indicator as a measure of dispersion.

To identify high-concentration regions, we introduce a parameter λ, referred hereafter as the threshold, that aims at separating ambient uniform areas from high-concentration areas. Following objective (iii), we avoid the use of arbitrary parameters as inputs by making use of the information on the observations itself. We introduce the 1D-distribution of the pdf-values, called reduced-pdf. Then, starting from an initial threshold λ fixed at the 60th percentile $\lambda = Q_{60}$, an iterative procedure computes the optimal value of λ that respects the criterion defined in the coming Step 2-f. Given the threshold λ, we further split the high concentration regions into five different sub-categories according to the distribution of the truncated-pdf values on $[\lambda, \max \text{pdf}]$. The five pdf levels are defined with the four renormalised percentiles $\widetilde{Q}_{20}$, $\widetilde{Q}_{40}$, $\widetilde{Q}_{60}$, $\widetilde{Q}_{80}$ (when they are distinguishable) related to the truncated-pdf values in the interval $[\lambda, \max \text{pdf}]$.

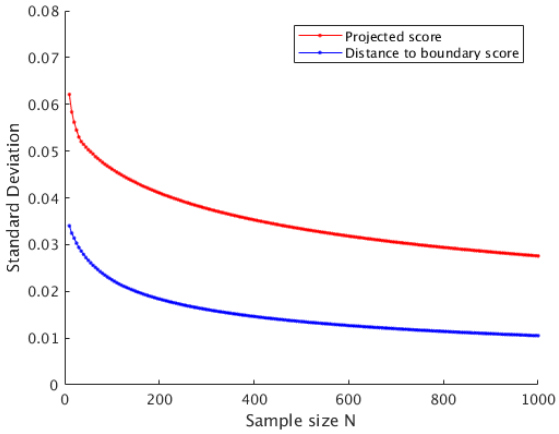
For the low concentration areas, we have defined three main levels: empty bins mark void areas; bins with pdf values below the 20th percentile Q_{20} mark the low concentration areas; bins with pdf values between Q_{20} and λ mark mean (ambient) concentration. The choice of the 20th percentile Q_{20} as the low concentration threshold has been made to simplify the algorithm. However, as an improvement of the algorithm, we



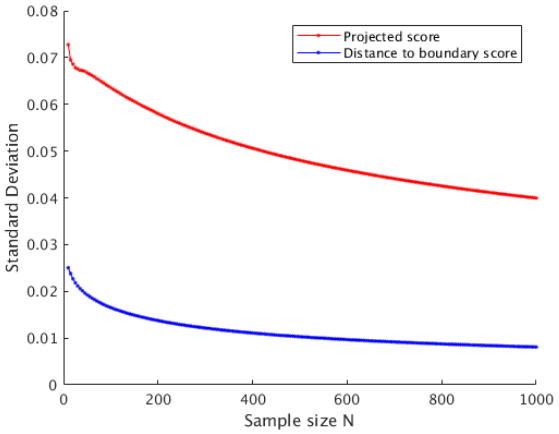
(a) Mean value of the distance to boundary score (blue) and projected score (red) in 2D



(b) Mean value of the distance to boundary score (blue) and projected score (red) in 3D



(c) Variance of the distance to boundary score (blue) and projected score (red) in 2D



(d) Variance of the distance to boundary score (blue) and projected score (red) in 3D

Figure 4: Mean and variance of the projected score (in red) in Equation (11) and distance to boundary score (in blue) in Equation (12) for 1000 uniform samples of size $n = 10, 20, \ldots, 1000$. Dimension $d = 2$ (left figures) and $d = 3$ (right figures).

suggest implementing a search for a lower threshold similar to the one proposed for the search for an upper threshold λ.

Using these pre-defined pdf levels for high and low concentration regions (here up to eight levels), the algorithm marks each bin in the domain according to its particle concentration (computed from the empirical pdf). In this way, we distinguish regions by labelling their concentration, which represents a great advantage over most of the algorithms in the existing literature. In practice, these labels are represented by a set of eight binary d-dimensional arrays of size $\otimes_{1\leq i\leq d}$`Nb_bins(i)`, where `Nb_bins(i)` denotes the number of bins in the ith-dimension. Then, using the edges of the pdf bins and the binary arrays, we construct a regular domain decomposition.

Step 2-e: synthetic sample (ghost particles). Coming up with a domain decomposition at the end of Step 2-d, the next step is to assess whether this decomposition is the optimal one. For that purpose, we rely on the score evaluated in Step 2-c as a convergence criterion.

To re-evaluate the score of the domain decomposition obtained after Step 2-d, the particles within 'bin-cells' identified as low- or high-particle concentration regions have to be replaced by ghost particles. This is to ensure that the uniformity tests (see Step 3) performed at the end of the D2SD algorithm provide accurate information on how far the remaining particles are from a uniform set. The aim of the ghost particles is to provide synthetic observations $\mathbf{X}^{\lambda}$. This $\mathbf{X}^{\lambda}$ is generated as follows: we remove point-data from those cells with detected anomalies and fill those cells with ghost points. These ghosts are uniformly distributed in each concerned bin. Their number inside a cell is determined using the cell volume matching the ambient concentration evaluated globally.

Step 2-f: score and λ^*-stopping criterion. The idea behind the choice of the optimal threshold λ^* (and thus optimal domain decomposition) is to vary the value of λ within the set `threshold_range`, and then apply the following principle: the further the score is from an ideal score, the worse the mesh will be. With this approach, the selection of the optimal decomposition starts with the computation of an ideal score

$$S_{\texttt{ideal}} = \langle d_b(\mathbf{U},\mathcal{U})\rangle,$$

where $\mathbf{U}$ is a random sample of size n uniformly distributed, and $\langle\cdot\rangle$ stands for the expectation under the law of the sample.

Then, we compute the score using the synthetic sample $\mathbf{X}^{\lambda}$ and compare this value to the ideal score, which represents a target score that the final distribution of particles $\mathbf{X}^{\lambda^*}_{\texttt{final}}$ is expected to approximate. By means of the ideal score, the optimal threshold λ^* is defined as:

$$\lambda^* = \texttt{argmin}_{\lambda} \frac{|S_{\texttt{ideal}} - \langle S(\lambda)\rangle_{\text{ghosts}}|}{S_{\texttt{ideal}}}, \text{ for } \lambda \text{ in } \texttt{threshold\_range}, \quad \text{where } S(\lambda) = d_b(\mathbf{X}^{\lambda},\mathcal{U}) \tag{13}$$

The bracket $\langle S(\lambda)\rangle_{\text{ghosts}}$ means that we average the score over several samples of the synthetic observations.

Numerically speaking, this means that Step 2-d, Step 2-e and Step 2-f are applied iteratively, one by one, starting from $\lambda = Q_{60}$ till $\lambda = Q_{80}$ until the optimal threshold λ^* is found.

2.4. Step 3: uniformity test

Once an optimal spatial decomposition is identified by the D2SD algorithm, a uniformity test is implemented on $\mathbf{X}^{\lambda^*}_{\texttt{final}}$ to verify the optimality of the constructed domain. The same uniformity tests as those described in Step 1 are applied here. If uniformity tests are rejected, it means that some clusters or voids were missed in the process. In that case, the D2SD algorithm is applied once more until we get a uniform $\mathbf{X}^{\lambda^*}_{\texttt{final}}$ sample. Nonetheless, convergence is expected to be attained fast due to the re-adjustment of the model parameters to the observations.

For simplicity, when the D2SD algorithm needs to be applied iteratively to attain uniformity, the bin size is kept constant and equal to the one obtained for the first iteration of the algorithm: this means that Step 2-a is applied only once and that the subsequent iterations of the D2SD algorithm start at Step 2-b (see also Fig. 2). Then, at each iteration, we update the location of the bins that have been used in previous iterations in order to avoid their repetition. A more accurate way to use the update of the information during subsequent iterations is to recompute the size of the bins. However, by doing this, the bin size might change at each iteration and one should then pay attention to the fact that the bins of each iteration may overlap with each other.

3. Numerical analysis of D2SD in 2D cases: convergence and accuracy

In this section, we present a series of numerical experiments to assess the accuracy and performance of the proposed algorithm to detect non-uniform distributions of particles. The application to the case of particle agglomeration is done later in Section 4. The idea here is to apply the D2SD algorithm to a number of cases that are representative of both uniform and non-uniform particle distributions (e.g. void regions,

clusters). These cases are generated by varying the level of the non-uniformity as well as the number of particles in the domain considered, allowing to assess the convergence and accuracy of the D2SD algorithm.

For visualization reasons, the experiments presented below are focused on two-dimensional cases. Nevertheless, the algorithm has been tested in similar three-dimensional cases, where very similar results were obtained (not shown here). These tests were performed using an implementation of the D2SD algorithm within Matlab R2016b.

3.1. Illustration of the steps of the algorithm

We first illustrate how the algorithm is working on a specific case. For that purpose, a highly non-uniform 2D case is given as input data. Figure 5 illustrates how the various steps of the algorithm introduced in Section 2 work. It can be seen that the initial sample results in a rejected uniformity test in Step 1 (all three discrepancy tests are rejected), thus triggering the start of the D2SD algorithm. To enrich the coming convergence analysis, we quantify the Step 1 result with the help of the initial distance-to-boundary score applied to the input dataset: $S_0 = d_b(\mathbf{X}, \mathcal{U})$.

In the present case, we obtain an initial distance-to-boundary score of 0.1584546, from which we compute an initial error $|S_{\texttt{ideal}} - S_0|/S_{\texttt{ideal}}$. After computing the bin size (Step 2-a), the spatial decomposition is applied iteratively for several values of λ until the optimal decomposition is found for $\lambda^* = 0.99624$ (see Step 2-f showing the evolution of the score with the threshold). The new data set generated with ghost particles inside low/high concentration regions (Step 2-e) is then accepted by the uniformity test in Step 3 (the five tests are accepted). The optimal spatial decomposition together with the colouring of particles with respect to this decomposition is shown in the output data.

3.2. Accuracy of the method

Having detailed how the algorithm works on a practical 2D case, we now focus on assessing the accuracy of the method regarding the computation of λ^* on four different samples: uniform samples, slightly non-uniform samples, highly non-uniform samples and symmetric samples. These samples have been chosen to be representative of a vast range of possible uniform and non-uniform data.

- Uniform case

  As we mentioned in Section 2, step 1 tests the uniformity of the data. Some tests can result in false negatives, that is, the test can reject the uniformity of a sample that comes from a uniform distribution. These false negatives will trigger the D2SD algorithm even though the set of particles is close to a uniform one. In order to assess the accuracy of the method in the case of a uniform random sample, we force here the D2SD algorithm to be applied to the set of data regardless of the result of the uniformity test, and then observe the behaviour of the algorithm in the uniform case.

  The results are displayed in Figure 6. As seen in the top left panel, we start from a uniform random sample $\{X^1, \dots, X^n\}$ of size $n = 10000$, in the box $[0, 1]^2$. In this case, the uniformity was accepted with an initial distance-to-boundary score $S_0 = 0.0015931$ (see Step 1 in the top center panel). The empirical reduced-cdf visible in the top right panel is typical of a uniform case and displays the threshold used for the spatial decomposition. The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 1.15$ and the corresponding new particle data $\mathtt{X}^{\lambda^*}_{\text{final}}$ (with ghost particles) can be seen in the bottom right panel. The modified sample being accepted as uniform, the optimal spatial decomposition is shown in the bottom left panel: it consists in 21×21 bins, with 0 empty bins and 17% under-concentrated bins (based on the percentile Q_{20}). Table 1 summarises the resulting score for each threshold in the set `threshold_range` (see Step 2-c).

  This example shows that, if the uniformity-test is rejected for a uniform distribution, this algorithm is able to detect slight variations in the spatial repartition of uniformly distributed particles and isolate these regions to be treated accordingly.

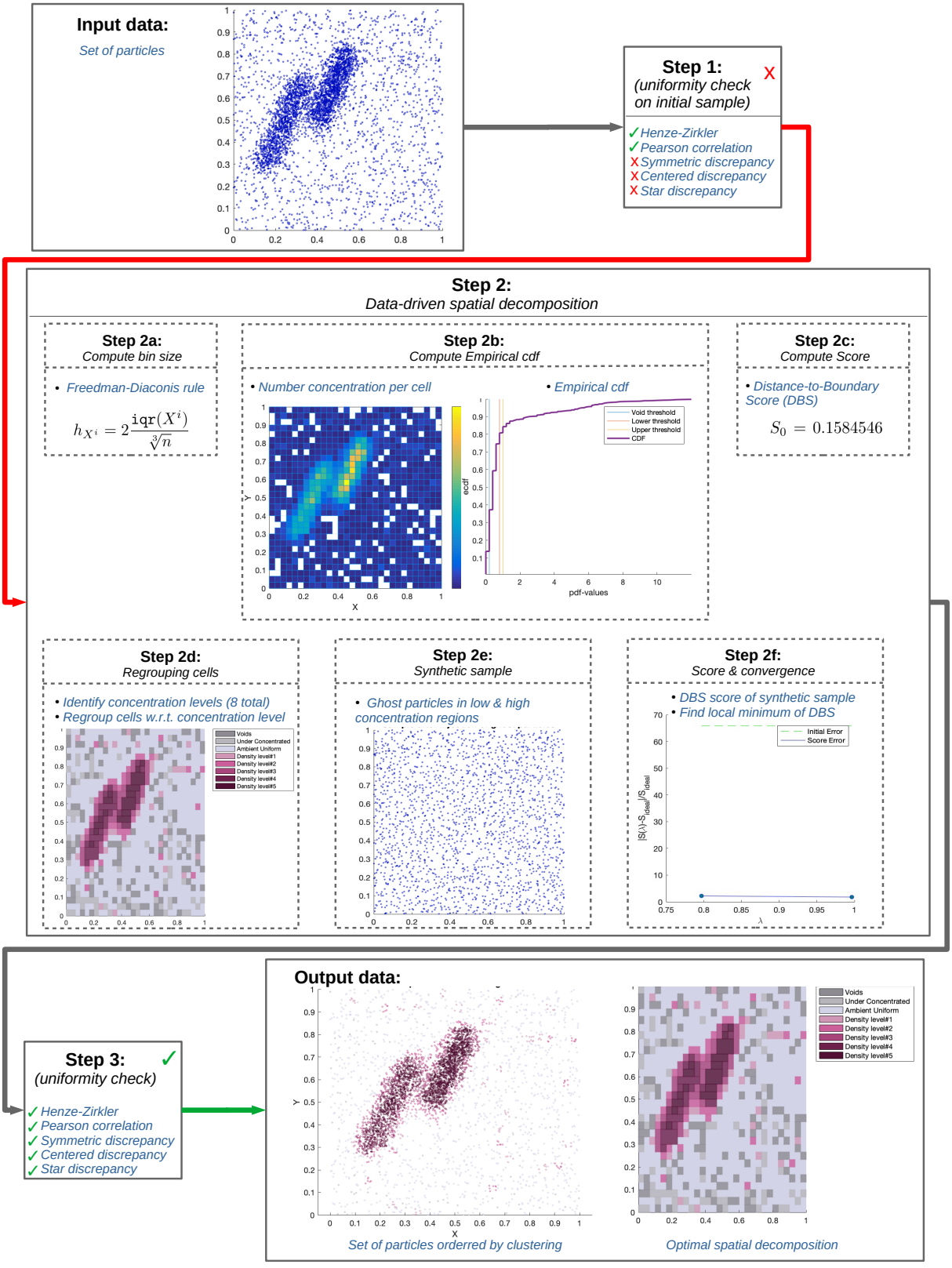


Figure 5: Analysis of the various steps of the algorithm on the case of a highly non-uniform 2D sample

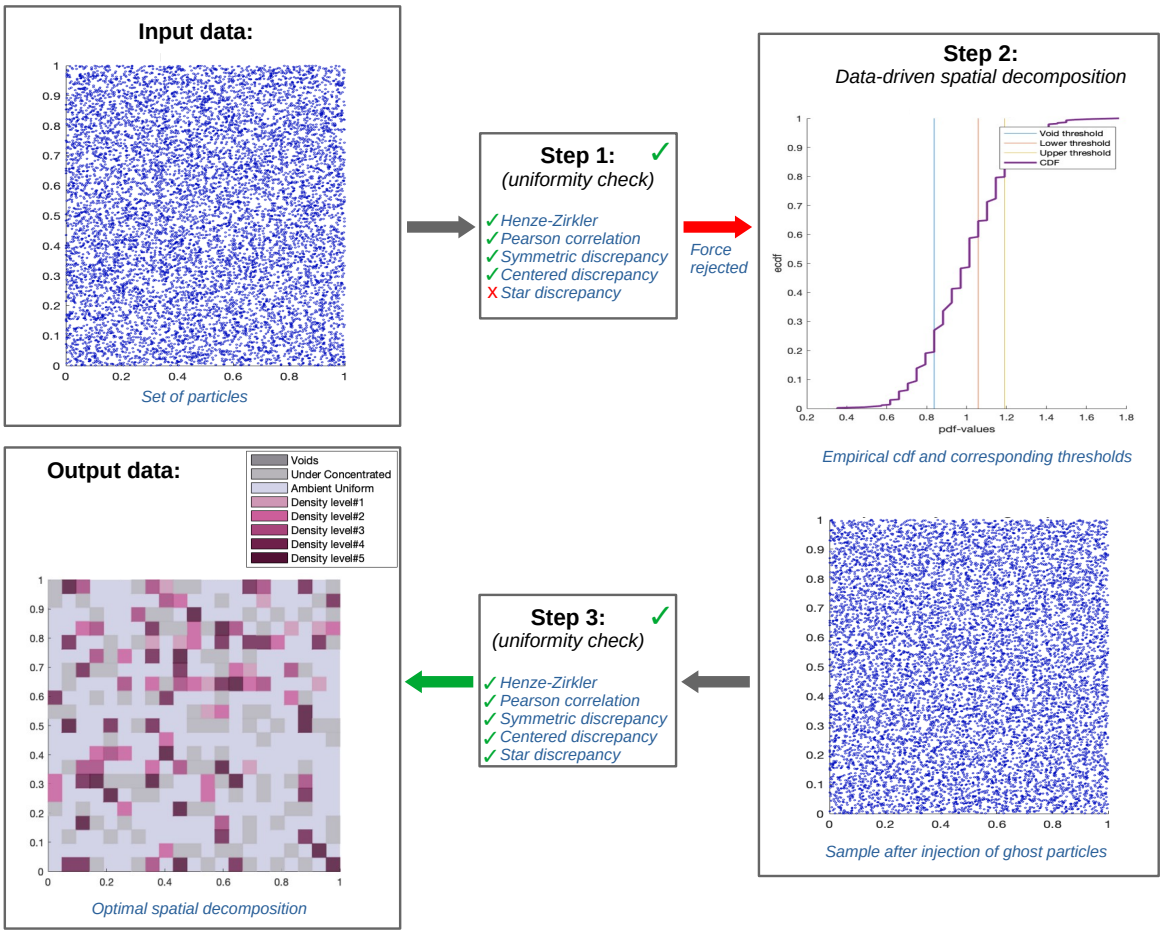


Figure 6: **Uniform case**. Processing elements and post-processing results for a uniform random sample $\mathbf{X}$ of size $n = 10000$.

Threshold λ	**Fraction $\langle S(\lambda)\rangle_{\texttt{ghosts}}/S_{\texttt{ideal}}$**	**Fraction of highly-concentrated cells (in %)**
1.06	0.40	31.75
1.10	0.37	27.89
1.15	0.46	20.18
1.19	0.43	18.14

Table 1: **Uniform case**. Distance-to-boundary scores and number of detected clusters for `threshold_range` $= \{1.06, 1.10, 1.15, 1.19\}$, with an initial score $S_0 = 0.0015931$ and ideal score $S_{\texttt{ideal}} = 0.0027556$ (giving an initial fraction of 0.58).

- Slightly non-uniform case

  The second numerical experiment treats the case where the random sample $\{X^1, \dots, X^n\}$ has a slightly non-uniform distribution. More precisely, 85% of the particles correspond to a uniform random sample and the remaining 15% corresponds to two Gaussian clusters with variance $0.05Id_{2\times 2}$ and $0.08Id_{2\times 2}$, respectively.

  The results are displayed in Figure 7. As seen in the top left panel, we start from a nearly uniform random sample $\{X^1, \dots, X^n\}$ of size $n = 2000$ in the box $[0, 1]^2$, composed of a uniform sample and two additional Gaussian distributions. In this case, the uniformity test was rejected (see Step 1 in the top center panel) with an initial distance-to-boundary score $S_0 = 0.0390917$. The empirical reduced-cdf visible in the top right panel reveals the presence of clustering (steeper slope). The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 1.27$ and the corresponding new particle data $\mathtt{X}^{\lambda^*}_{\text{final}}$ (with ghost particles) can be seen in the bottom right panel. The modified sample being accepted as uniform, the optimal spatial decomposition is shown in the bottom left panel: it consists in 14×14 bins, with 0 empty bins and 13% under-concentrated bins (based on the percentile Q_{20}). Table 2 summarises the resulting score for each threshold in the set `threshold_range`.

  This case shows that the D2SD algorithm is able to identify two regions with slightly higher concentrations (corresponding to the two Gaussian clusters introduced). Moreover the distance-to-boundary score is reduced by one order of magnitude when applying the D2SD algorithm, confirming that the optimal spatial decomposition respects the spatially-uniform condition much better than the initial spatial distribution.

Threshold λ	**Fraction $\langle S(\lambda)\rangle_{\texttt{ghosts}}/S_{\texttt{ideal}}$**	**Fraction of highly-concentrated cells (in %)**
1.08	0.49	25.00
1.18	0.65	18.88
1.27	0.90	15.31

Table 2: **Slightly non-uniform case**. Distance-to-boundary scores and number of detected clusters for `threshold_range` $= \{1.08, 1.18, 1.27\}$, with an initial score $S_0 = 0.0390917$ and ideal score $S_{\texttt{ideal}} = 0.0054152$ (giving an initial fraction of 7.22).

- Highly non-uniform case

  The third case corresponds to the highly non-uniform case seen in Figure 5.

  The results are displayed in Figure 8. As seen in the top left panel, we start from a highly non-uniform random sample $\{X^1, \dots, X^n\} \subset \mathbb{R}^2$ of size $n = 5000$, composed of a uniform sample (38% of the particles) and a large cluster (containing 62% of the particles). In this case, the uniformity test was rejected with an initial distance-to-boundary score $S_0 = 0.1584546$ (see Step 1 in the top center panel). The empirical reduced-cdf visible in the top right panel reveals the presence of severe clustering (very steep slope) and a high proportion of void regions. The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 1.00$ and the corresponding new particle data $\mathtt{X}^{\lambda^*}_{\text{final}}$ (with ghost particles) can be seen in the bottom right panel. The modified sample being accepted as uniform, the optimal spatial decomposition is shown in the bottom left panel: it consists in 30×30 bins, with 96 empty bins

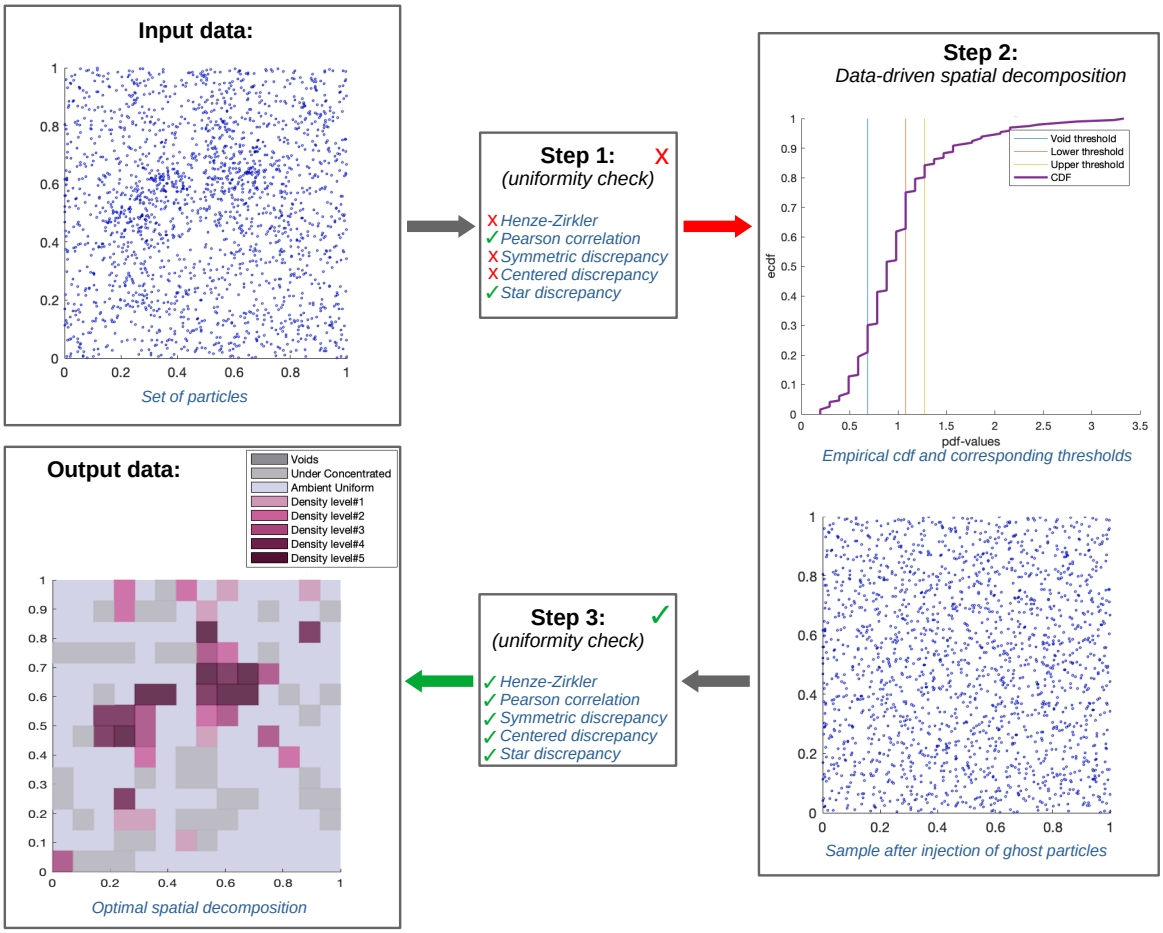


Figure 7: **Slightly non-uniform case**. Processing elements and post-processing results for a slightly non-uniform random sample $\mathbf{X}$ of size $n = 2000$ where the 15% of the sample is non-uniform.

and 55 under-concentrated bins (based on the percentile Q_{20}). Table 3 summarises the resulting score for each threshold in the set `threshold_range`.

This case shows that the D2SD algorithm is able to identify large clustering region with very high concentrations (corresponding to the cluster introduced). Moreover, the distance-to-boundary score is reduced by nearly two orders of magnitude when applying the D2SD algorithm, confirming that the optimal spatial decomposition respects the spatially-uniform condition much better than the initial spatial distribution.

Threshold λ	**Fraction** $\langle S(\lambda)\rangle_{\texttt{ghosts}}/S_{\texttt{ideal}}$	**Fraction of highly-concentrated cells (in %)**
0.80	3.23	147
1.00	2.81	108

Table 3: **Highly non-uniform case**. Distance-to-boundary scores and number of detected clusters for `threshold_range` $= \{0.80, 1.0\}$, with an initial score $S_0 = 0.1584546$ and ideal score $S_{\texttt{ideal}} = 0.0023693$ (giving an initial fraction of 66.88).

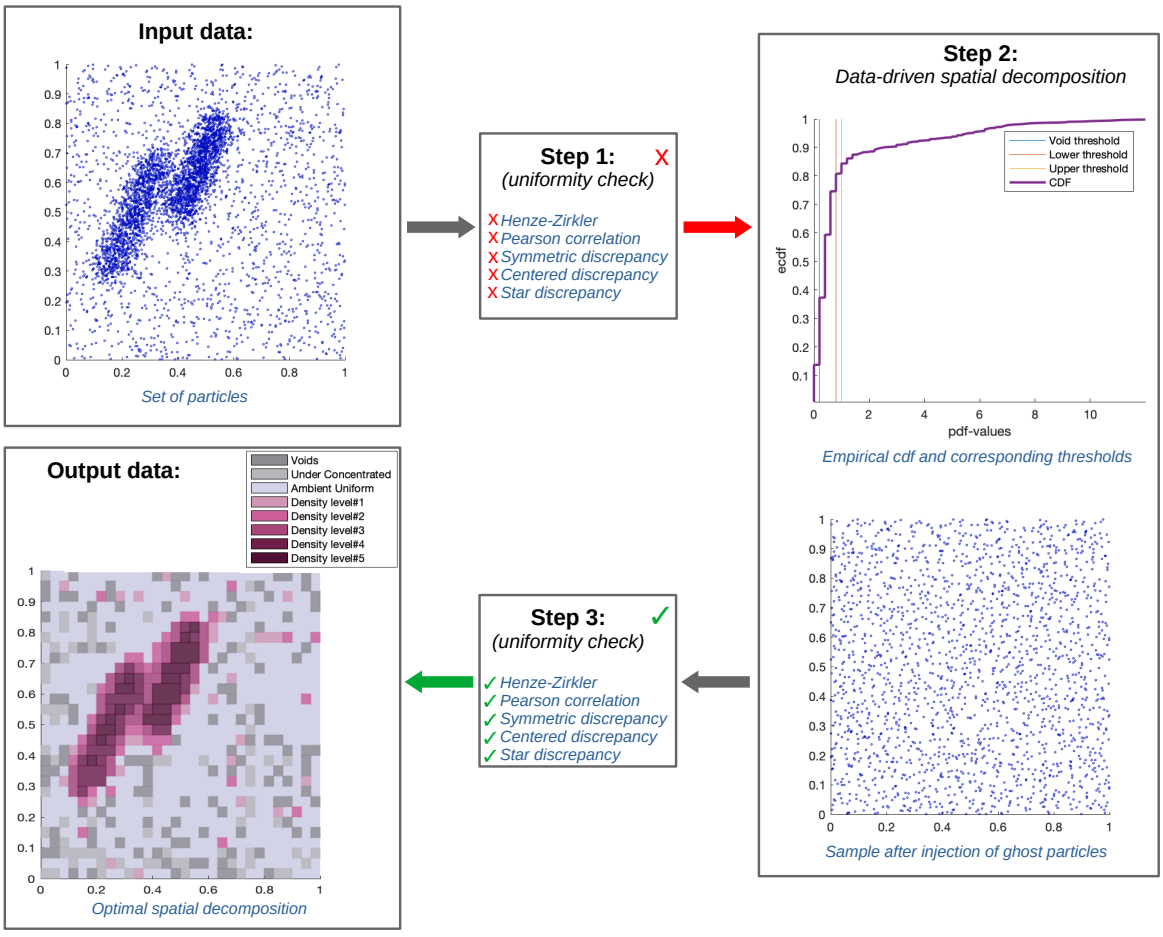


Figure 8: **Highly non-uniform case**. Processing elements and post-processing results for a highly non-uniform random sample $\mathbf{X}$ of size $n = 5000$ where the 62% of the sample is non-uniform.

- Symmetric case The fourth case corresponds to a symmetric case with void regions. The aim of this case is to highlight the advantages of using the joint probability density function when detecting symmetric patterns.

The results are displayed in Figure 9. As seen in the top left panel, we start from a highly non-uniform random sample $\{X^1, \dots, X^n\}$ of size $n = 5000$, composed of a uniform sample in the boundary of the box $[0, 1]^2$, forming a frame. In this case, the uniformity test was rejected with an initial distance-to-boundary score $S_0 = 0.0780929$ (see Step 1 in the top center panel). The empirical reduced-cdf visible in the top right panel reveals the presence of a high proportion of void regions (more than 25%) and some clustering. The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 1.53$ and the corresponding new particle data $\mathtt{X}^{\lambda^*}_{\text{final}}$ (with ghost particles) can be seen in the middle right panel. It appears that the D2SD algorithm is able to detect the void region but that the current binning left some void regions untouched. Yet, the uniformity test applied on the modified set of data is rejected, which triggers another loop in the D2SD algorithm (see middle center panel). During the second iteration of the D2SD algorithm, the empirical reduced-cdf visible in the middle left panel reveals the presence of some void regions and a few clustering regions (high slopes). The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 1.26$ and the corresponding new particle data $\mathtt{X}^{\lambda^*}_{\text{final}}$ (with ghost particles) can be seen in the bottom left panel. After this second iteration, the D2SD algorithm clearly generated a more uniform set of data. The modified data being accepted as uniform, the optimal spatial decomposition is shown in the bottom right panel: it consists of 12×12 bins, with 1 empty bins (in the center) and 6 under-concentrated bins (based on the percentile Q_{20}). Table 4 summarises the resulting score for each threshold in the set $\texttt{threshold\_range} = \{1.53, 1.56, 1.58, 1.61, 1.64, 1.67, 1.70\}$, for the first iteration, and $\texttt{threshold\_range} = \{1.26, 1.28, 1.31, 1.34, 1.39\}$ for the second iteration.

This case shows that the D2SD algorithm is able to identify the low-concentration region introduced in the center of the domain. Moreover, the distance-to-boundary score is reduced by nearly one order of magnitude when applying the D2SD algorithm, confirming that the optimal spatial decomposition respects the spatially-uniform condition much better than the initial spatial distribution.

First Iteration		
Threshold λ	**Fraction $\langle S(\lambda)\rangle_{\texttt{ghosts}}/S_{\texttt{ideal}}$**	**Fraction of highly-concentrated cells (in %)**
1.53	6.32	23.61
1.56	6.43	22.22
1.58	6.65	19.44
1.61	6.74	15.97
1.64	6.74	16.67
1.67	6.80	14.58
1.70	6.98	13.19
Second Iteration		
Threshold λ	**Fraction $\langle S(\lambda)\rangle_{\texttt{ghosts}}/S_{\texttt{ideal}}$**	**Fraction of highly-concentrated cells (in %)**
1.26	1.13	15.97
1.28	1.49	12.50
1.31	1.72	9.03
1.34	2.21	4.17
1.39	2.40	2.08

Table 4: **Symmetric non-uniform case**. Distance-to-boundary scores and number of detected clusters for $\texttt{threshold\_range} = \{1.53, 1.56, 1.58, 1.61, 1.64, 1.67, 1.70\}$ in the first iteration and $\texttt{threshold\_range} = \{1.26, 1.28, 1.31, 1.34, 1.39\}$, with an initial score $S_0 = 0.0780929$ and ideal score $S_{\texttt{ideal}} = 0.0037523$ (giving an initial ratio of 20.81).

3.3. Precision and convergence of the method

We detail here the convergence of the method with respect to the number of particles. To assess this convergence, we consider the four types of samples (uniform, slightly non-uniform, highly non-uniform and symmetric non-uniform) with different particle sizes ($n = \{500, 1000, 1500, 2000, \dots, 5000\}$) and we analyse the result accuracy with respect to the number of particles n. In order to obtain an accurate mean score (and diminish the randomness of this measure), we consider a Monte Carlo approximation of $\langle S(\lambda)\rangle_{\text{ghosts}}$, using $M = 1000$ i.i.d ghost samples (see Appendix B for more details).

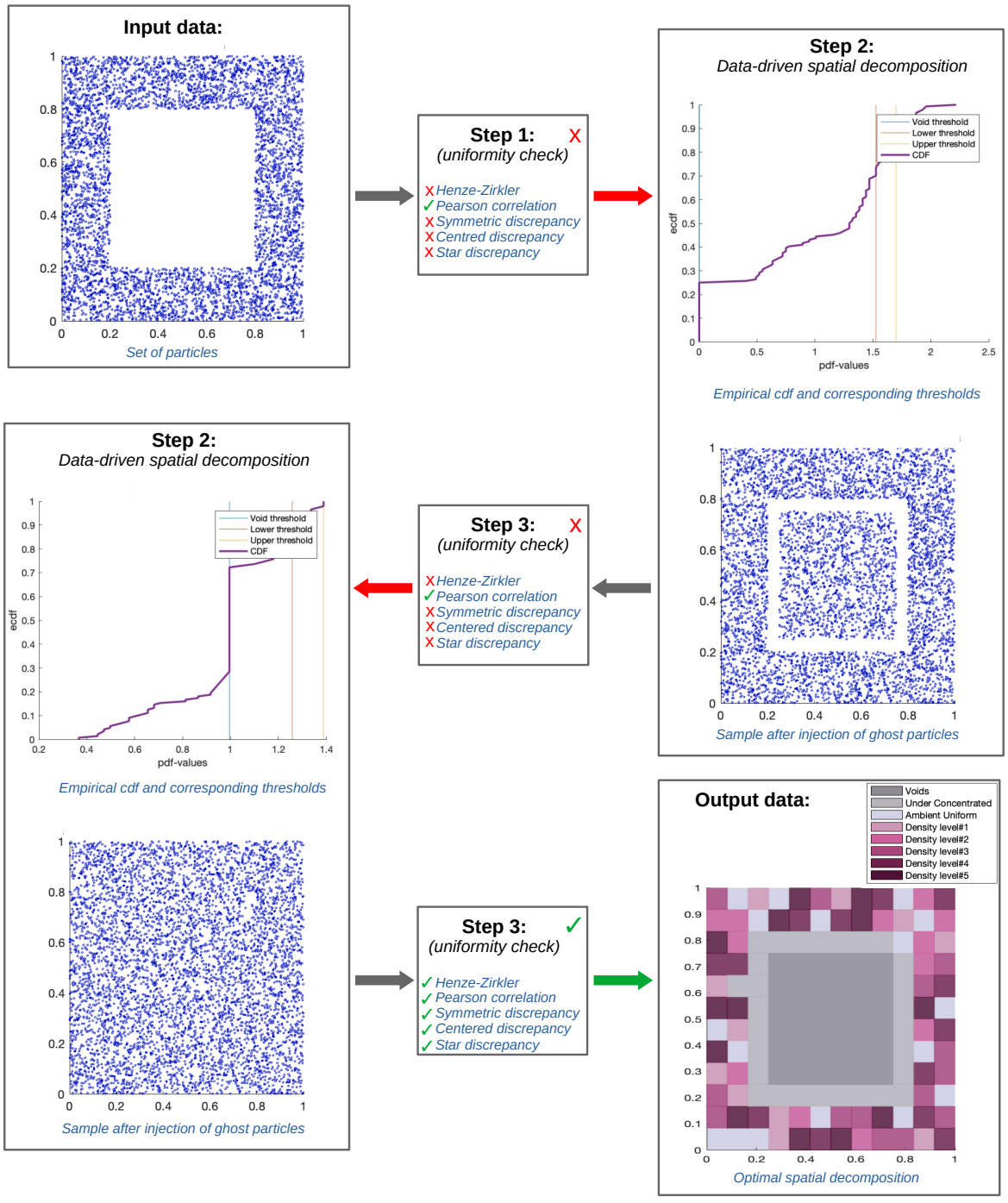


Figure 9: **Symmetric non-uniform case**. Processing elements and post-processing results for a symmetric non-uniform random sample X of size $n = 5000$ with frame shape.

The results are compiled in Table 5, which provides information on n, the initial score of the sample S_0, the optimal threshold λ^* with its corresponding score $\langle S(\lambda^*)\rangle_{\text{ghosts}}$, the total number of bins and the number of high-concentrated regions (referred to as clusters), under-concentrated regions and void regions associated with the optimal mesh. The numerical experiments shown in this section were programmed in Matlab and tested on a laptop (Intel Core i7, 2.50GHz). In Table 5, we illustrate the evolution of the computational time for a sample of size n by re-scaling with the CPU time of the initial sample $\texttt{CPU}(n_0)$. Several main conclusions can be drawn:

- First, it is sufficient to apply the current algorithm one time to accomplish the required accuracy (i.e. the uniformity check in Step 3 is accepted) except in the last case. The rejection of the uniformity check in the case of a symmetric non-uniform sample is due to the proportion of void regions present in the domain. For this reason, in Table 5, we include a column with the number of iterations of the method necessary to accept the uniformity of $\mathtt{X}^{\lambda^*}_{\text{final}}$ in the symmetric non-uniform case.
- Second, the D2SD algorithm converges relatively quickly towards a case close to the ideal one as the number of particles is increased. This convergence of the score with respect to the number of data points is shown in Fig. 10 for the uniform, slightly non-uniform and highly non-uniform cases. The reason for omitting the symmetric case in Fig. 10 is that the proportion of ghost particles introduced in $\mathtt{X}^{\lambda^*}_{\text{final}}$ corresponds to more than 50% of the initial data. As a result, although we can observe a decreasing trend in the score for the Symmetric case, the randomness has a strong impact in the visualization. This effect is expected to be less and less important as the size of the data increases.
- Third, due to the randomness of both the data and the algorithm, the analysis of the computational complexity requires further studies. Indeed, the scaled CPU times provided in Table 5 exhibit the dependency of the computational complexity of the algorithm with respect to the sample distribution. Nevertheless, after optimizing properly the implementation, we expect the computational complexity to be of the order of n.

Uniform case								
n	**N. bins**	λ^*	$S_0\times10^3$	$\langle S(\lambda^*)\rangle_{\text{ghosts}}\times10^3$	**Fraction of clusters (in %)**	**Fraction of under concentrated cells (in %)**	**Fraction of void regions (in %)**	**Scaled time** $\frac{\texttt{CPU}(n)}{\texttt{CPU}(n_0)}$
500	7×8	1.27	15.00	8.88	17.86	12.5	0.00	1.00
1000	9×9	1.35	6.50	6.31	19.75	14.81	0.00	1.06
1500	11×11	1.04	7.34	4.12	27.27	15.70	0.00	1.48
2000	12×12	1.22	5.50	3.06	17.36	13.88	0.00	1.89
2500	13×13	1.21	4.07	2.88	18.34	13.61	0.00	2.67
3000	14×14	1.24	3.20	2.71	15.31	15.31	0.00	5.80
3500	15×15	1.20	4.15	3.02	17.33	14.67	0.00	6.40
4000	15×16	1.26	5.26	2.42	23.33	16.67	0.00	11.56
4500	16×16	1.25	3.28	2.11	20.70	16.80	0.00	10.52
5000	17×17	1.21	3.16	2.38	18.69	15.92	0.00	9.09
Slightly non-uniform case								
n	**N. bins**	λ^*	$S_0\times10^2$	$\langle S(\lambda^*)\rangle_{\text{ghosts}}\times10^3$	**Fraction of clusters (in %)**	**Fraction of under concentrated cells (in %)**	**Fraction of void regions (in %)**	**Scaled time** $\frac{\texttt{CPU}(n)}{\texttt{CPU}(n_0)}$
500	8×9	1.15	3.50	10.40	18.06	12.50	1.39	1.00
1000	11×11	1.21	3.80	4.97	16.35	14.05	0.00	1.09
1500	12×13	1.04	4.14	3.15	25.00	16.03	0.00	1.47
2000	14×14	1.27	3.99	4.02	15.31	12.76	0.00	1.67
2500	15×15	1.17	4.15	2.96	18.22	13.33	0.00	1.91
3000	16×16	1.11	4.39	2.51	25.00	16.02	0.00	6.35
3500	16×18	1.23	4.50	3.12	16.67	14.58	0.00	8.42

4000	17×18	1.22	3.89	2.23	15.03	15.36	0.00	9.55
4500	18×19	1.21	4.33	2.17	15.50	15.50	0.00	12.80
5000	18×20	1.00	4.29	2.40	24.17	14.72	0.00	13.33
Highly non-uniform case								
n	**N. bins**	λ^*	$S_0 \times 10$	$\langle S(\lambda^*)\rangle_{\text{ghosts}} \times 10^3$	**Fraction of clusters (in %)**	**Fraction of under concentrated cells (in %)**	**Fraction of void regions (in %)**	**CPU time (in s)**
500	12×14	1.34	1.30	9.55	12.50	0.00	17.26	1.00
1000	16×18	1.15	1.33	7.23	11.46	0.00	14.58	1.20
1500	19×20	1.01	1.27	5.92	12.11	1.84	13.16	1.27
2000	21×22	1.15	1.32	4.88	12.99	4.98	11.25	1.70
2500	22×24	0.63	1.33	4.29	20.45	6.06	9.28	2.39
3000	25×25	1.04	1.27	3.26	12.32	10.24	8.00	2.78
3500	26×27	0.60	1.29	3.99	20.23	8.26	8.55	3.65
4000	27×28	0.75	1.36	2.88	14.68	7.28	8.33	4.00
4500	28×29	0.54	1.31	2.52	22.41	9.73	8.00	4.35
5000	28×30	0.84	1.33	2.53	13.81	12.26	6.55	4.27
Symmetric non-uniform case								
n	**N. bins**	λ^*	$S_0 \times 10^2$	$\langle S(\lambda^*)\rangle_{\text{ghosts}} \times 10^3$	**N. of iterations**			
500	5×5	1.80	7.79	7.23	1			
1000	7×7	$\{1.52, 1.28\}$	7.72	$\{42.00,\ 8.87\}$	2			
1500	8×8	$\{1.49, 1.15\}$	7.95	$\{21.60,\ 6.45\}$	2			
2000	9×9	1.53	7.71	7.95	1			
2500	9×9	1.62	7.70	9.15	1			
3000	10×10	1.76	7.95	2.55	1			
3500	11×11	$\{1.55, 1.42\}$	7.94	$\{40.30,\ 6.65\}$	2			
4000	11×11	$\{1.60, 1.41, 1.19\}$	7.84	$\{39.00,\ 8.99,\ 3.87\}$	3			
4500	12×12	$\{1.60, 1.26\}$	7.96	$\{27.10,\ 3.90\}$	2			
5000	12×12	$\{1.52, 1.25\}$	7.80	$\{2.37,\ 4.7\}$	2			

Table 5: **Uniform case, Slightly non-uniform, Highly non-uniform and symmetric non-uniform case**. Optimal thresholds, scores and grids versus the number of particles in $n = \{100, 500, 1000, 1500, 2000, \ldots, 5000\}$.

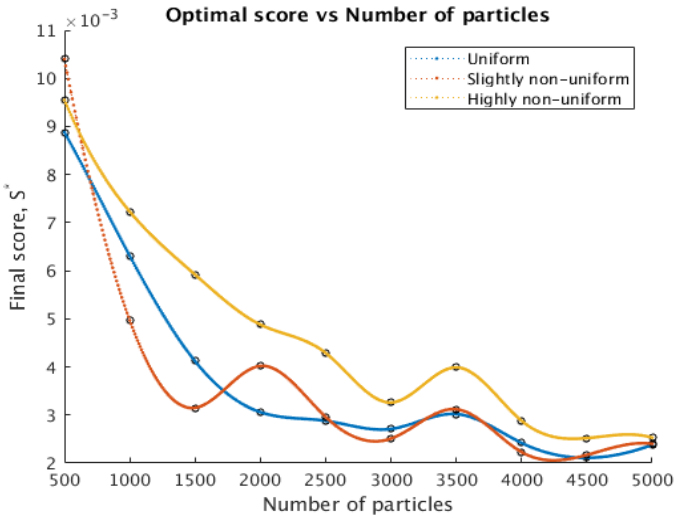


Figure 10: **Convergence of the optimal score** $\langle S(\lambda^*)\rangle_{\text{ghosts}}$. Cases: uniform, slightly non-uniform and highly non-uniform.

4. Application to particle agglomeration

In this section, we apply the D2SD algorithm to the case of particle agglomeration. This requires adapting the D2SD algorithm so that it can be coupled with CFD simulations of particle agglomeration in realistic 3D cases. In particular, different versions of the D2SD algorithm are introduced in Section 4.1 depending on the level of accuracy and computational costs desired. The accuracy of these methods is then addressed by comparing the numerical results obtained with these agglomeration-adapted D2SD algorithms to theoretical estimations in simple cases (see Section 4.2). Finally, the agglomeration-adapted D2SD algorithms are applied to a realistic 3D case, where data on particle positions are coming from a CFD simulation of a turbulent flow in a half-pipe (see Section 4.3).

4.1. Adaptation of D2SD algorithm for particle agglomeration

Since the D2SD algorithm presented in Section 2 provides a spatial decomposition in which the spatially-uniform condition is respected in each cell, it can be easily coupled with typical one-particle tracking approaches where collisions are computed in cells with a given collision frequency β and collision efficiency α. More precisely, since agglomeration models in one-particle tracking approaches are based on a Bird-like algorithm (i.e. transport is treated first without collision and collisions are then evaluated without transport), the D2SD algorithm is to be applied, after computing the motion of particles, at each time step for which the distribution of the particles has varied sufficiently according to the previous time step. The algorithm will thus come up with an optimal spatial decomposition based on the current positions of particles that are present in the simulation. Then, assuming that the collision efficiency α is equal to unity (i.e. all collisions lead to agglomeration), the number of agglomeration events computed using the D2SD-generated mesh is given by:

$$\mathcal{A}_{estimated} = \sum_{\texttt{Cell } C \in \texttt{ Mesh}} \beta \frac{n_C^2}{V_C} \Delta t, \tag{14}$$

where n_C is the number of particles in cell C, V_C is the volume of the cell and Δt is the time step.

In the rest of the paper, several adaptations of the D2SD model for particle agglomeration are proposed. The suggested variations have been designed in order to provide a range of possible algorithms depending on the accuracy desired in CFD simulations and on the computational costs. The following adaptations are analysed:

1. Full D2SD (Step 1–Step 3): this corresponds to the whole algorithm presented in Section 2 including both uniformity checks.
2. Step 1 + Step 2: this corresponds to the D2SD algorithm coupled with the initial uniformity check.
3. Fast D2SD (Step 2 only): this corresponds to the D2SD algorithm only (no uniformity check).

It should be noted that these algorithms were programmed in Matlab R2016b and tested on an ordinary laptop (Intel Core i7, 2.50GHz). As such, the CPU times that are provided in the following are not intended to represent absolute computational times. It should rather be used as reference times to be compared together in order to measure the ratio between accuracy and computational resources involved. Moreover, the computation time of the D2SD algorithm and its derivatives depends directly on the data (which are considered always as random) and especially on the length of the threshold vector λ: as a result, computational times will display variations with each spatial distribution.

4.2. Accuracy and convergence of agglomeration-adapted D2SD to compute agglomeration events

Bearing in mind that, when dealing with numerical approximations of agglomeration processes, we can easily lose the notion of real number of agglomeration events and that agglomerations will always depend on the collision rate (which is not part of the scope of this article), we have designed two simple two-dimensional experiments to evaluate the accuracy of the agglomeration-adapted D2SD algorithm for the computation of particle agglomeration. In the first experiment, we treat the case of uniformly distributed particles. In the second experiment, the agglomeration-adapted D2SD algorithms are applied to a population of point particles with a certain spatial preference. In both cases, numerical results are compared to theoretical values/estimations.

4.2.1. Agglomeration in a uniform case

The first numerical experiment consists of particles that are uniformly distributed in space. For that purpose, we consider a family of n point particles, for n in $\mathcal{N} = \{10^p,\ p = 1, 2, 3, 4, 5\} \cup \{5 \times 10^p,\ p = 1, 2, 3, 4\}$. These particles are uniformly distributed in the square domain $[0, 1] \times [0, 1]$. The collision frequency β and the time step are assumed to be constant, such that $\beta \Delta t = 10^{-6}$. In that case, the number of agglomeration events obtained theoretically is given by:

$$\mathcal{A}_{\text{events}}(n) = n^2 \beta \Delta t.$$

Figure 11 displays both the illustration of the methods used to compute particle agglomeration and the evolution of the error made in the estimation of particle agglomeration using these methods. More precisely, we compare the results obtained with the spatial decomposition algorithms to those obtained considering a regular mesh with 2×2 or 20×20 as well as with a triangulated mesh with 20×20 elements. The comparison with such arbitrarily chosen mesh allows to highlight the interest of the D2SD approach, which allows the computation of a spatial decomposition based on the statistical information on the data points. The error plotted in Figure 11b corresponds to the absolute error for n in $\mathcal{N}$. It is computed with the formula

$$\frac{|\mathcal{A}_{\text{events}}(n) - \mathcal{A}(k, n)|}{\mathcal{A}_{\text{events}}}, \text{ for } k = 0, 1, \ldots, 5,$$

where $\mathcal{A}(k, n)$ is the numerical estimation of the number of agglomeration events, which is approximated by

$$\mathcal{A}(k, n) = \sum_{\texttt{Cell}\ C \in \texttt{Mesh(k)}} \min \left\{ \beta \frac{n_C^2}{V_C} \Delta t, \frac{n_C}{2} \right\},$$

for each mesh $k = 0, 1, \ldots, 5$ where $k = 0, 1, 2$ corresponds to the Step 2, Steps 1-2 and Steps 1-2-3 of the D2SD algorithm, respectively.

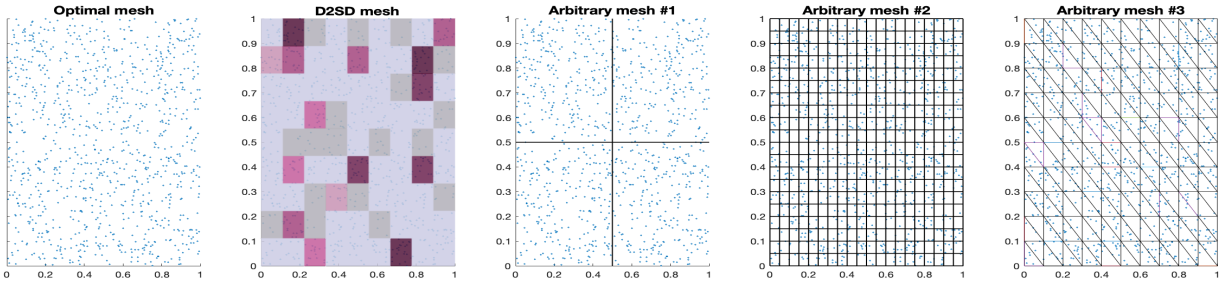
Several conclusions can be drawn from this comparison:

1. As expected in the uniform case, the results obtained with either D2SD decomposition or arbitrary meshes all converge to the theoretical value as the number of point particles is increased.
2. The arbitrary 20×20 or triangulated meshes induce a relatively high error even for samples up to 10^3 particles. This can be attributed to the bias induced by the fact that small cells contain a small number of particles and, as a result, the number of agglomeration events is more prone to fluctuations (this is all the more pronounced as the total number of particles n is low).
3. The arbitrary 2×2 mesh seems to provide the best convergence with respect to the number of particles n, which in this case is natural since it is the mesh that least divides the domain.
4. The most accurate results are obtained using the full D2SD algorithm. As visible in the inset of Figure 11, this accuracy comes at the price of a computational cost proportional to the number of particles in the domain.

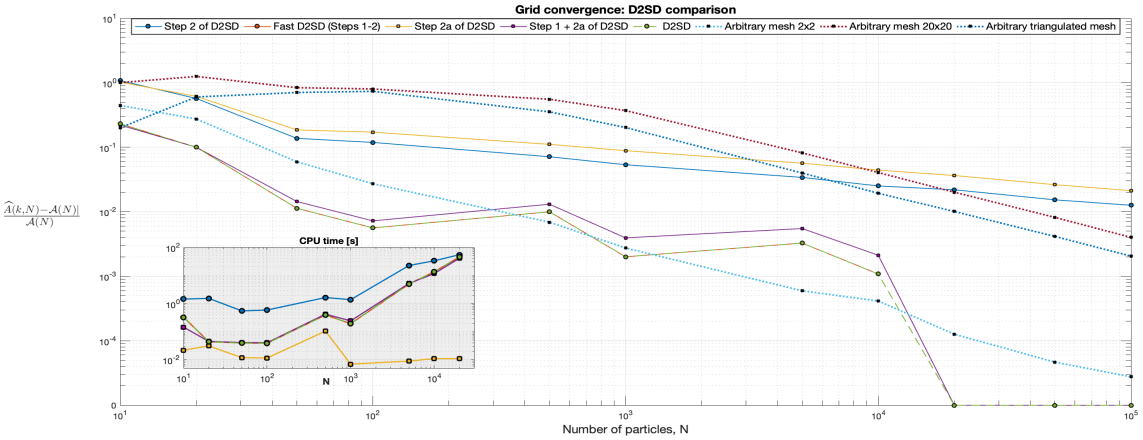
4.2.2. Agglomeration in a non-uniform case

The second numerical experiment consists in a system of n point particles, with n in $\mathcal{N}$. As illustrated in Figure 12, $n_1 = \frac{17n}{20}$ of the particles are uniformly distributed in the subdomain $[\frac{1}{4}, 1] \times [\frac{2}{3}, 1]$ while the rest of the $n_2 = \frac{3n}{20}$ particles are uniformly distributed in the remaining subdomain $[0, 1] \times [0, 1] \setminus [\frac{1}{4}, 1] \times [\frac{2}{3}, 1]$. We further assume that the collision efficiency $\alpha = 1$ and we fix the collision rate and time step such that $\beta \Delta t = 10^{-6}$. To evaluate the accuracy of the results, the expected number of agglomeration events has to be estimated theoretically. To that extent, we approximate the total number of agglomeration events as the sum of the agglomeration events in the two uniform areas. This is only an estimation since theoretical calculations are not possible in the non-uniform case (it basically neglects the events occurring at the boundaries of the sub-domains). Thus, the approximation of the number of agglomeration events is:

$$\mathcal{A}_{\text{events}}(n) \approx \min \left\{ \frac{4n_2^2}{3} \Delta t \beta, \frac{n_2}{2} \right\} + \min \left\{ 4n_1^2 \Delta t \beta, \frac{n_1}{2} \right\}.$$



(a) Spatial decomposition using D2SD (Step 2 only) versus arbitrary regular meshes of sizes 2×2, 20×20 and a triangulated mesh with 20×20 elements.



(b) Absolute error for number of agglomeration events and CPU time.

Figure 11: First experiment for convergence of the D2SD algorithm (in log-log scale): Uniform case. Results correspond to an average of 20 samples for each population size $n \times 10^{-3} = 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100$.

As in the uniform case, we compare the results obtained with the spatial decomposition algorithms to those obtained considering a regular mesh with 2×2 or 20×20 as well as with a triangulated mesh with 20×20 elements (see also Figure 13a). The comparison with such arbitrarily chosen mesh allows to highlight the interest of the D2SD approach, which allows to compute a spatial decomposition based on the statistical information on the data points. The error plotted in Figure 13b corresponds to the absolute error for n in $\mathcal{N}$.

Several conclusions can be drawn from this comparison:

1. As expected, the results obtained with the arbitrary 2×2 mesh do not converge to the theoretical estimation. This is due to the fact that the mesh used is not adapted to the spatial distribution of particles.
2. The arbitrary 20×20 or triangulated meshes induce a relatively high error when the number of particles is low (typically below $n = 10^3$). As for the uniform case, this can be attributed to the bias induced by the fact that small cells contain a small number of particles and, as a result, the number of agglomeration events is more prone to fluctuations. Moreover, the error reaches a plateau value independent of the number of particles n when n becomes large enough (which depends on the number of cells). This plateau is due to the fact that, even if the mesh is somewhat adapted to the non-uniform case, the arbitrary splitting is independent of the actual repartition of the n sample points.
3. Among the agglomeration-adapted D2SD algorithms, the full D2SD as well as the (Step 1 and Step 2) algorithms provide the most accurate results. Moreover, all results converge to the theoretical estimation

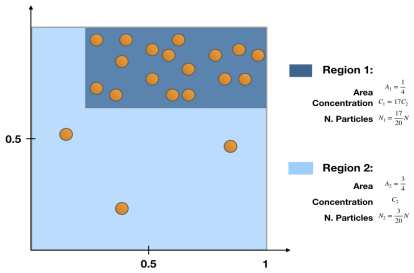
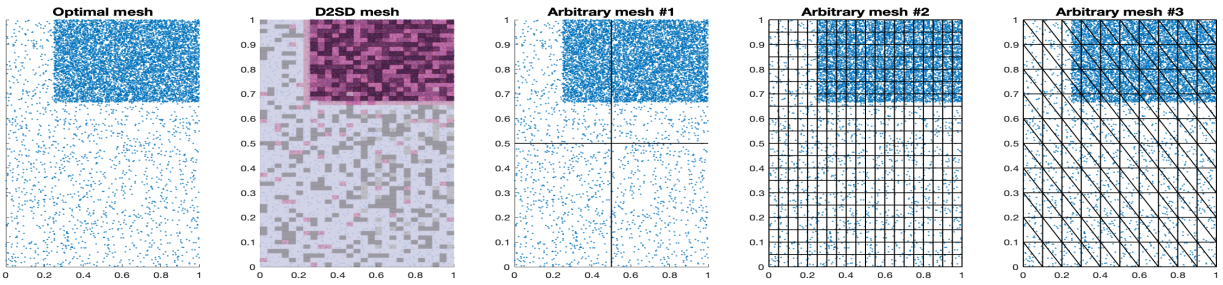
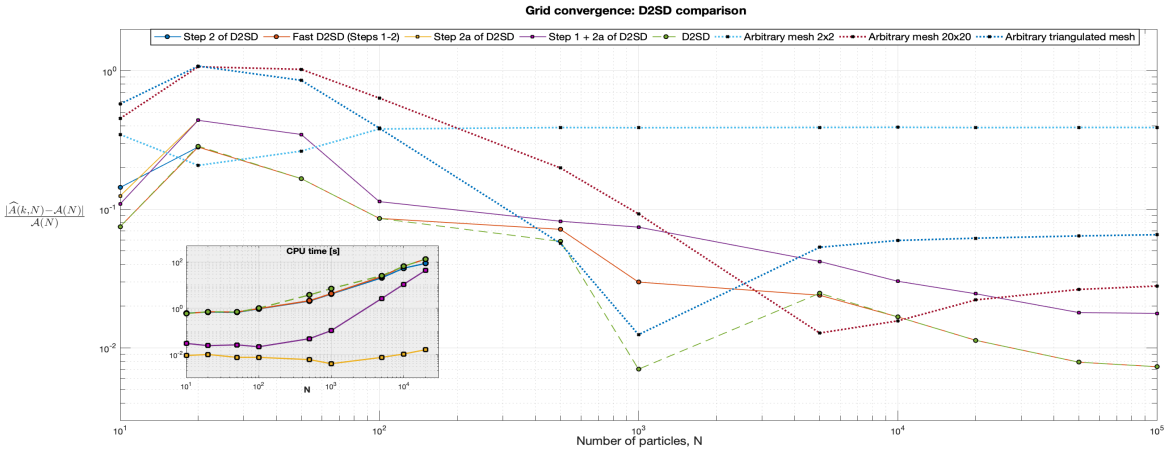


Figure 12: Sketch of the second experiment of convergence.



(a) Spatial decomposition using D2SD versus arbitrary regular meshes of sizes 2×2, 20×20 and a triangulated mesh with 20×20 elements.



(b) Absolute error for number of agglomeration events and CPU time.

Figure 13: Second experiment for convergence of the D2SD algorithm (in log-log scale): Non-uniform case. Results correspond to an average of 20 samples for each population size $n \times 10^{-3} = 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100$.

as the total number of particles increases.

At this point, it is worth mentioning that the performance of the algorithm proposed in this paper has more impact in more complex situations. However, to measure its accuracy, it is necessary to make a qualitative analysis of the agglomeration mechanisms, which will be discussed in a future article.

4.3. Application of agglomeration-adapted D2SD to a 3D sample from a CFD simulation

In the following, we apply the current method to a particle position dataset $\{X_t^1, \dots, X_t^n\}$ which has been generated from a CFD simulation obtained with *Code_Saturne* (an open-source CFD code) on the NEF cluster available at INRIA. More precisely, we focus on the case of a point-source dispersion of particles inside a turbulent flow (see also Fig. 14). Particle dispersion in a half-pipe is representative of industrial/environmental cases where particles are injected locally in space and are then dispersed by the underlying flow (e.g. point-source dispersion in atmospheric flows or droplet injection in spray nozzles). Figure 14 displays the position of particles in the half-pipe together with the particle volume fraction obtained at various cross-sections: it can be seen that particles are initially dispersed within the center of the pipe and particles then diffuse in the whole pipe section as they are transported further downstream. Results have been obtained with an Euler-Lagrange simulation (plots after 20 s of simulation with a time step of 0.01 s and 10 μm particles injected in a turbulent flow at $Re = 10^4$). The half pipe had a radius of 0.1 m and a length of 1 m. Particles are injected at each iteration of the simulation inside a small area close to the pipe centre at the inlet boundary and with various concentrations, leading to different total number of the particle sample within the domain ($n \in \{955, 1920, 3820, 9580\}$). We then apply the agglomeration-adapted D2SD algorithm directly on data coming from the CFD simulation, i.e. on a snapshot of particle positions at a given time step. This is used to illustrate the interest of such an approach on a realistic CFD case.

Dealing with non cubic geometry. Since the domain does not correspond to a cubic domain, in order to implement the discrepancy statistics, we need to transform the dataset $\mathbf{X}$, for which we propose two methods: (1) form a complete cube through the injection of ghost particles or (2) transform the positions $\mathbf{X}$ with a change of coordinates. In both cases, we normalize the observations in order to have a sub-domain of the unit cube.

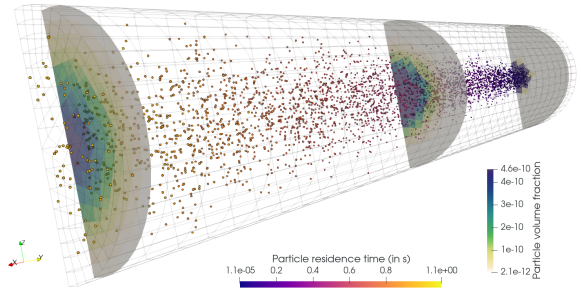


Figure 14: Point-source injection and dispersion of particles in a half-pipe obtained with a CFD simulation in *Code_Saturne* (the pipe inlet is in the background face and the pipe outlet is in the front).

The idea of injecting ghost particles is to uniformly fill the void region generated when embedding the pipe in a box (the concentration of ghost particles in this region is the same as the sample $\mathbf{X}$). This way, an extended sample $\widetilde{\mathbf{X}}$ is obtained with real and ghost particles inside a box where the D2SD algorithm can be applied. Since the ghost particles have the same concentration as $\mathbf{X}$, the rejection of the uniformity of $\widetilde{\mathbf{X}}$ will immediately imply the rejection of the uniformity of $\mathbf{X}$. The main drawback of this approach is that the algorithm is forced to process more data, thus increasing the computational cost and memory needed. Also, since we construct a grid composed of cubes, we will be – in a certain way – approximating cylindrical sections through cubic sections.

Inspired by this volume approximation, we propose a second method: a transformation based on cylindrical coordinates. For this second method, we will use the transformation from Cartesian to cylindrical coordinate $(x, y, z) \mapsto (\rho, \theta, z)$ given by

$$\begin{cases} \rho = x^2 + y^2 \\ \tan\theta = \frac{y}{x} \end{cases} \tag{15}$$

If the point (X, Y, Z) is uniformly distributed in the half-pipe, we know that the joint density of (X, Y, Z) is given by $f_{\text{quadr}}(x, y, z) = 1/\texttt{Cylinder volume} = 200/\pi$. Then, through the change of coordinates defined

in (15), the joint density of the random point $(X, Y, Z) = (P, \Theta, Z)$ is given by

$$f_{\text{cylind}}(\rho, \theta, z) = \frac{1}{2} f_{\text{carte}}(\sqrt{\rho}\cos\theta, \sqrt{\rho}\sin\theta, z) = \frac{100}{\pi} = \frac{1}{\texttt{Cube volume}}$$

for P, Θ and Z random variables with distribution $P \sim \mathcal{U}[0, (0.1)^2]$, $\Theta \sim \mathcal{U}[0, \pi]$ and $Z \sim \mathcal{U}[0, 1]$. That is, the random point (P, Θ, Z) is uniformly distributed in the cube $[0, 0.01] \times [0, \pi] \times [0, 1]$. The idea is then to apply directly the D2SD method to the transformed sample $\widetilde{\mathbf{X}} = (P(\mathbf{X}), \Theta(\mathbf{X}), Z(\mathbf{X}))$. With this, the optimal mesh will be given in cylindrical-type coordinates, so the grid will be better adapted to the data.

Analysis of results. Since the aim of this algorithm is to evaluate agglomeration of particles in complex flows, the number of agglomeration events generated with and without the current algorithm is assessed. For that purpose, we compare the total number of agglomeration events obtained using the mesh provided in the CFD simulation with *Code_Saturne* to the one obtained with the mesh generated by the D2SD method. In our reduced case, since we consider only primary particles, the total number of agglomeration events is given by:

$$\mathcal{A}_{\text{events}} = \sum_{\texttt{Cell } C \in \texttt{Mesh}} \min\left\{\beta \frac{n_C^2}{V_C}\Delta t, \frac{n_C}{2}\right\}, \tag{16}$$

where V_C corresponds to the volume of each cell C, n_C is the number of primary particles inside each cell, Δt is a time-step and β is the agglomeration kernel. In Table 6, we compute the agglomeration events corresponding to the mesh generated by *Code_Saturne* for $n \in \{955, 1920, 3820, 9580\}$ (at a fixed time t with a constant time step $\Delta t = 0.01$ and a given rate $\beta = 10^{-5}$). This computation, without the addition of D2SD, is used later on to highlight the effect of a mesh prescribed only for the flow simulation.

Number of particles	Agglomeration events
955	285
1920	697
3820	1572
9580	4330

Table 6: **Half-pipe case.** Number of agglomeration events computed with Formula (16) using the `Mesh` available in the CFD simulation with *Code_Saturne* .

In order to assess the accuracy of the method, we consider the sample $\{X_t^1, \ldots, X_t^n\}$, with $n = 3820$ simulated with *Code_Saturne* within the half-pipe domain. Since the input data is different for each method, we present here the results obtained with each method, specifying the initial score, the number of bins and the thresholds with their corresponding distance-to-boundary score:

- **Extension of data:** The results are displayed in Figure 15. As seen in the top left panel, after computing the concentration of the sample $\mathbf{X}$ and injecting the ghost particles in the exterior of the half-cylinder, we obtain an extended sample with a number of particles increased by roughly 27%. In this case, the uniformity test was rejected with an initial distance-to-boundary score $S_0 = 0.0267532$ (see Step 1 in the top center panel). The empirical reduced-cdf visible in the top right panel reveals the presence of a very high proportion of void regions (close to 60%) and some clustering. The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 6.55$ and the corresponding new particle data $\mathtt{X}_{\text{final}}^{\lambda^*}$ (with ghost particles) can be seen in the middle right panel. The modified data being accepted as uniform, the optimal spatial decomposition is shown in the bottom right panel: it consists of $43 \times 16 \times 23$ bins, with 7.4% either empty or under-concentrated bins (based on the percentile Q_{20}). The whole method took 208 seconds of computational time. It should be noted here that most of the computational time comes from the computation of the uniformity checks (which scale as n^2). Table 7 summarises the resulting score for each threshold in the set $\texttt{threshold\_range} = \{6.55, 9.82\}$.

  This case shows that the D2SD algorithm is able to identify clustering and void regions in a 3D case (with data points coming directly from a CFD simulation). Moreover, the distance-to-boundary score

is reduced by nearly two orders of magnitude when applying the D2SD algorithm, confirming that the optimal spatial decomposition respects the spatially-uniform condition much better than the initial spatial distribution. However, Fig. 15 also highlights one of the drawback of the method with an extension of the particle data: the generated mesh follows typical Cartesian coordinates and is not necessarily adapted to the cylindrical coordinates of the half-pipe here. This will be improved using the second method with a change of coordinates (described below).

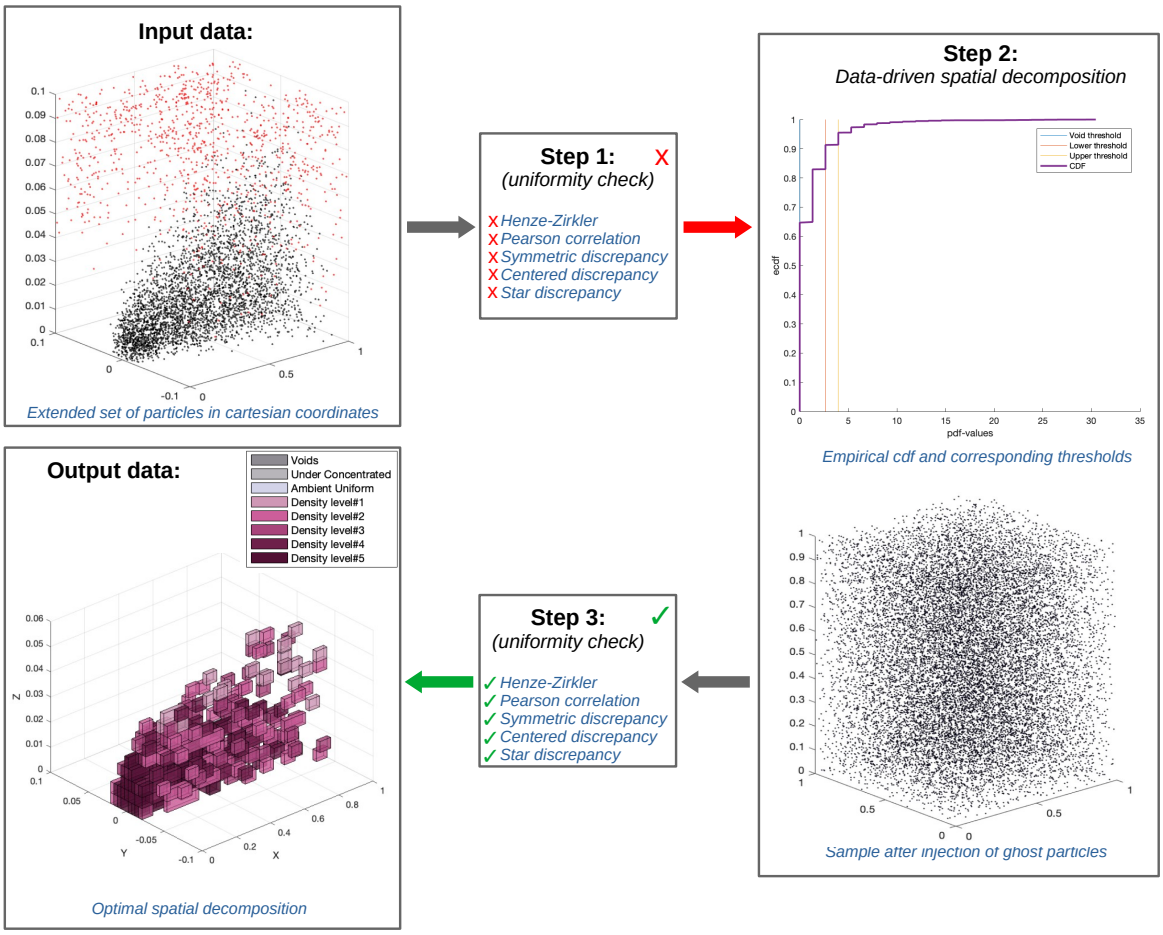


Figure 15: **Half-pipe case with extension of data method**. Processing elements and post-processing results for an initial sample $\mathbf{X}$ with $n = 3820$ fluid particles and extended with 1015 ghost particles completing the boxed domain.

- **Transformation of data:** The results are displayed in Figure 16. As seen in the top left panel, the particle data set $\mathbf{X} = \{X_t^1, \ldots, X_t^n\}$ (with a size $n = 3820$) is transformed through the change of coordinates defined in (15). In this case, the uniformity test was rejected with an initial distance-to-boundary score $S_0 = 0.1131566$ (see Step 1 in the top center panel). The empirical reduced-cdf visible in the top right panel reveals the presence of a very high proportion of void regions (close to 80%). The optimal domain decomposition is generated with the optimal threshold $\lambda^* = 6.95$ and the corresponding new particle data $\mathtt{X}_{\text{final}}^{\lambda^*}$ (with ghost particles) can be seen in the middle right panel (note that it is plotted in the Cartesian coordinates system to see the uniformity of the data). The modified data being accepted as uniform, the optimal spatial decomposition is shown in the bottom right panel: it consists of $15 \times 59 \times 15$ bins, with 7.6% empty plus under-concentrated bins (based on the percentile Q_{20}). The whole method took 38 seconds of computational time, which appears more reasonable in the

context of CFD simulations (note again that a significant time is devoted to the uniformity checks). Table 7 summarises the resulting score.

As previously, the D2SD algorithm is able to identify clustering and void regions in a 3D case (with data points coming directly from a CFD simulation) and the distance-to-boundary score is reduced by nearly two orders of magnitude when applying the D2SD algorithm. In addition, the use of a transformation of coordinates provides an optimal spatial decomposition that better suits the original volume (here a half-pipe).

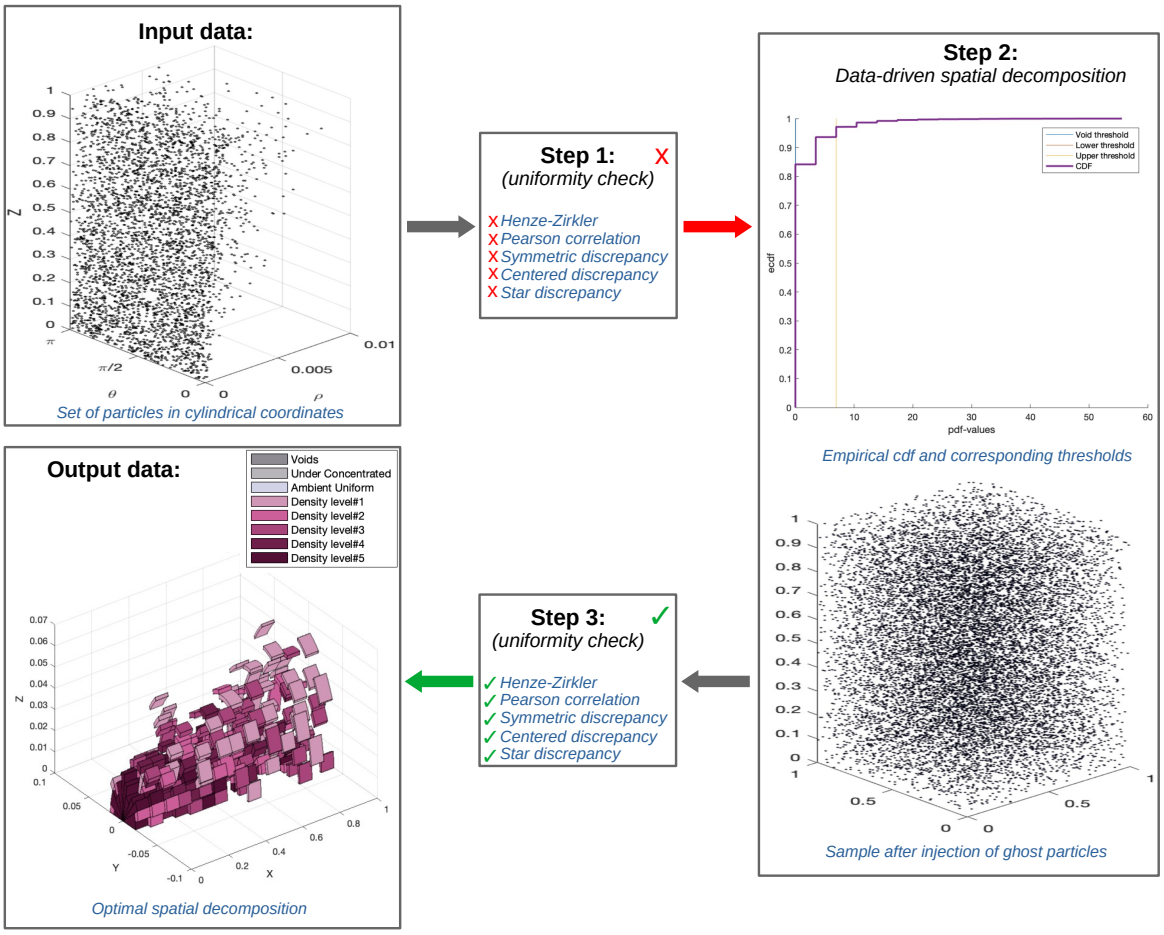


Figure 16: **Half-pipe case with transformation of data method**. Processing elements and post-processing results for an initial sample $\mathbf{X}$ with $n = 3820$ fluid particles.

Finally, we compute the number of agglomeration events generated using the mesh provided by the D2SD method in Equation (16) and compare it to the initial one obtained using the mesh from *Code_Saturne* simulation. Table 8 summarises the results obtained for various number of particles n, together with the initial score of the sample S_0, the optimal threshold λ^* with its corresponding score $\langle S(\lambda^*)\rangle_{\text{ghosts}}$, the total number of bins, the number of high-concentrated regions and the void regions associated with the optimal mesh. Additionally, the last column in Table 8 represents the ratio between the number of agglomeration events computed with D2SD and the number of agglomeration events computed for the grid used in *Code_Saturne* (see values in Table 6). In particular, it can be seen that agglomeration computed at a given time in a CFD simulation depends strongly on the mesh. The D2SD algorithm allows to compute agglomeration without mesh dependence, providing results that may differ up to 40 % from the initial results obtained with the

Half-pipe processing with extension of the data		
Threshold λ	$\langle S(\lambda)\rangle_{\mathbf{ghosts}}\times 10^4$	**Fraction of highly-concentrated cells (in %)**
6.55	6.96160	2.02
9.82	6.62342	1.57

Half-pipe processing with transformation of the data		
Threshold λ	$\langle S(\lambda)\rangle_{\mathbf{ghosts}}\times 10^3$	**Fraction of highly-concentrated cells (in %)**
6.95	3.4174	2.52

Table 7: **Half-pipe case**. Domain processed through an extension of the data (method 1) and a transformation of the data (method 2).

Half-pipe processing with extension of the data							
n	**N. bins**	λ^*	$S_0\times 10^2$	$\langle S(\lambda^*)\rangle_{\text{ghosts}}\times 10^4$	**Fraction of clusters (in %)**	**Fraction of void regions (in %)**	**Agglomeration ratio**
955	$27\times 10\times 14$	6.17	1.876	16.33	2.41	7.96	1.392982
1920	$34\times 13\times 18$	9.73	2.371	11.75	1.65	7.96	1.078910
3820	$42\times 16\times 22$	9.84	2.677	6.985	2.05	7.94	0.973919
9580	$57\times 23\times 31$	9.98	2.401	4.265	1.57	6.95	0.895381
Half-pipe processing with transformation of the data							
n	**N. bins**	λ^*	S_0	$\langle S(\lambda^*)\rangle_{\text{ghosts}}\times 10^3$	**Fraction of clusters (in %)**	**Fraction of void regions (in %)**	**Agglomeration ratio**
955	$10\times 37\times 9$	6.97	0.1031	2.341	2.91	8.05	1.340351
1920	$12\times 48\times 12$	7.20	0.1134	3.145	2.50	7.58	1.084648
3820	$15\times 59\times 15$	6.95	0.1131	3.494	2.52	7.62	0.957379
9580	$20\times 78\times 21$	6.84	0.1096	2.233	3.42	7.49	0.876905

Table 8: **Numerical results for half-pipe case**. Optimal thresholds, scores and grids versus the number of particles in $n = \{955, 1920, 3820, 9580\}$.

mesh used in the CFD simulation. It should be noted here that uniform samples are obtained after only one iteration of the D2SD algorithm.

5. Conclusion

5.1. Summary of the key findings

A new data-driven spatial decomposition (D2SD) algorithm has been described in this paper. It allows the detection of non-homogeneous concentrations in a set of particles (point data) within a regular volume. For that purpose, the D2SD algorithm relies solely on the information coming from the particle set, without requiring other input parameters. More precisely, lower and higher concentration regions are detected by computing the pdf associated with the sample of particles and by comparing the values obtained to a threshold. The optimal spatial decomposition is then obtained by iterating on several values of the threshold (across a range of possible values coming from the quantiles of the distribution) and by looking for the result giving the score closest to the ideal targeted case (using either distance-to-boundary or projection score). The D2SD algorithm has been coupled here with a-priori and a-posteriori uniformity tests to check whether the input and output data satisfy the condition of uniformly-distributed particles.

This algorithm has been tested in several 2D cases, which confirmed that the D2SD algorithm is able to detect spatial inhomogeneities such as: slightly higher concentrations of particles; high clustering regions of complex shapes; symmetric cases with void regions. Moreover, all these tests have shown that the present method converges in a few iterations toward a spatial decomposition that respects the uniformity test. Indeed,

except in the 2D symmetric case where 2 iterations of the D2SD algorithm were required, one iteration of the D2SD algorithm was enough in all other cases to obtain an accurate spatial decomposition.

The D2SD algorithm has been then adapted to be applied to study particle agglomeration. For that purpose, particle agglomeration was computed in each region identified by the D2SD algorithm. Comparison of the numerical results with theoretical values/estimations in non-uniform cases have shown the accuracy of the algorithm to evaluate the number of agglomeration events. Even when the population of particles is uniformly distributed in the domain (and thus the spatially-uniform condition is satisfied) the D2SD algorithm has proven to have better results, compared to an arbitrary choice of mesh. Several variations of the D2SD algorithm have been tested to show that, depending on the specific requirements of CFD codes, less complex and faster algorithms can be designed but at the cost of a lower accuracy of the results. Finally, the algorithm has been applied to data on particle positions coming from a realistic 3D CFD simulation: it has been shown that the agglomeration can be significantly over- or under-estimated using the mesh used for CFD simulations. This highlights the robustness and accuracy of the algorithm and the need to apply it when computing particle agglomeration in complex 3D industrial/environmental cases where the spatially-uniform condition is not necessarily respected.

Finally, although the D2SD algorithm has been designed to address the spatial uniformity hypotheses behind PBE-based methods, the proposed method can be translated to different state variables (e.g., temperature) where we only need to adapt the construction of the pdf associated with the variable of interest.

5.2. Perspectives for future work

These promising results pave the way for future developments/refinements of the current algorithm. In particular, we have identified several issues that need to be addressed:

- First, it should be noted that the D2SD algorithm has been coupled here with a-priori and a-posteriori uniformity checks. However, these two uniformity tests are not in line with the objective (iv) described in Section 1.4 since they rely on the computation of inter-particle distances. These uniformity tests were used here to demonstrate the robustness and efficiency of the D2SD algorithm. Since they have shown that the method converges very quickly to an optimal spatial decomposition, one could decide to apply recursively the algorithm 2 or 3 times without performing any uniformity test. This requires further testing of the algorithm in complex situations to identify how many times the algorithm needs to be applied to obtain results in line with objective (v). Moreover, the computation of the uniformity tests is very time-consuming and the overall method can be significantly faster if such tests are not performed. Further works are also needed to speed up the D2SD algorithm, especially to couple it with standard CFD codes, where D2SD will be applied (and thus the domain will be decomposed accordingly) when a certain criterion, responding to how different the distribution is between one time step and another, is met.

- The D2SD algorithm has been successfully applied to regular geometries (2D or 3D boxes or cylinders) but is not easily applicable to complex geometries. This is mainly due to the computation of the score (either through distance-to-boundary or projection methods) as well as to the computation of the pdf, which is based on a regular domain decomposition. To apply the method to a more general geometry, new score measurements are thus needed. This can be done relying on the existing theoretical literature on the distance-to-boundary or developing dedicated scores from the existing literature on distance on measure space. Besides, as was the case for the 3D data from the half-pipe, transformations (e.g. using cylindrical coordinates) are required to use a regular domain decomposition that fits the initial domain. However, such transformations need to conserve the uniformity of the initial distribution (otherwise the D2SD algorithm will not be applicable). Since this extra condition on the transformation of the data can be quite restrictive, the method based on the extension of the data to regular geometries (where the D2SD algorithm can be safely applied) seems more suitable to complex geometries (the drawback being then that the mesh is not necessarily adapted to the initial geometry).

- The D2SD algorithm has been applied here to static situations, i.e. the data points (particle positions) were fixed and were generated by a CFD simulation. In future developments, the aim will be to couple

the D2SD algorithm with the particle dynamics in order to assess the robustness and efficiency of the algorithm when the number and location of particles evolves in time. This will require further work to improve the numerical efficiency of the approach (especially to use parallel computation) and to properly couple the approach to the chosen CFD software (and its libraries). Another approach to reduce the numerical costs associated to the creation of a new spatial decomposition is to apply the D2SD algorithm only when deemed necessary (and not every time step as in a 'brute-force' manner). More precisely, a new criterion could be added, which measures either how far the spatial distribution of the particles is from uniformity or how different the particle positions at the current time step are from those at the previous one (a criterion on their motion). Such additional criteria can easily be included in step 1 of the algorithm, for example through the DBS, and will be directly related to the context of the application. It should be noted that the use of two different meshes for the fluid phase and the particle phase requires an extrapolation between these two meshes (which is not accounted for in the computational costs described in the present paper but should be evaluated in a proper implementation within a CFD code).

- We are also planning to adapt the current algorithm to the tracking of parcels, i.e. numerical elements that actually represent a large number of real particles. For that purpose, we may rely on some of the notions introduced by Pischke et al. [40], especially the parcel diameter estimator which represents the space available per parcel.

- The present D2SD algorithm can be applied in a wide range of contexts, both in particle clustering and in different frameworks of machine learning focussed on data clustering. In particular, it can be applied to detect clusters of particles in DNS simulations coupled with particle-tracking approaches. In that case, even when particles are homogeneously distributed at the system scale, local fluctuations of particle concentrations can occur due to turbulent fluctuations and particle inertia: this is referred to as preferential concentration (see for instance [52, 53] and references therein). Such non-uniform distribution of particles in space can significantly impact the agglomeration rate (which is directly proportional to the local particle concentration in PBE formulations). The present algorithm can then be used to identify these regions with intense clustering and to help design efficient models that accurately reproduce the agglomeration and clustering statistics in turbulent flows.

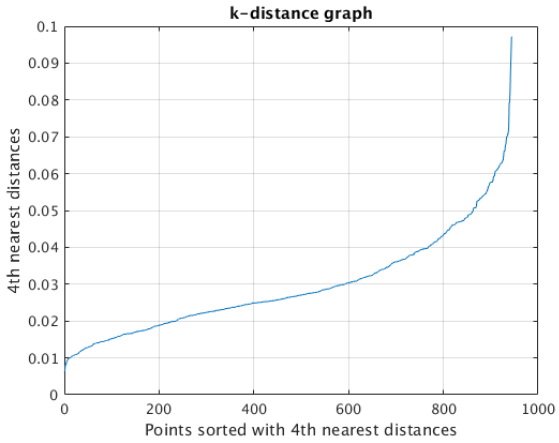
- Finally, the D2SD algorithm has also implications in the general context of Lagrangian one-particle pdf approaches. One-particle pdf approaches are indeed mean-field approaches that rely on the evaluation of mean quantities related to particles in cells (e.g. particle volume fraction or velocity). Yet, accurate statistics and converged statistical data are only possible if sufficient particles are present within each cell. In that context, the D2SD algorithm appears as a promising tool to perform such statistical evaluations on a mesh generated by the spatial decomposition technique presented here, which will be naturally adapted to the concentration of particles.

Appendix A. Quick assessment of existing clustering methods

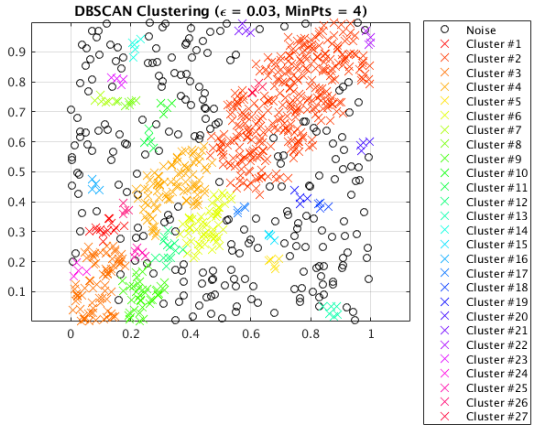
The issue of identifying regions with non-homogeneous repartition of particles has been long studied in the context of image analysis and signal processing [43, 44]. In particular, several clustering techniques exist to process data points, including: algorithms based on space partition (e.g. the K-means algorithm), algorithms based on distributions (e.g. the DBCLASD algorithm) or algorithms based on density (e.g. the DBSCAN algorithm).

One of the most common clustering algorithms is the DBSCAN algorithm (see, for instance, [54]). It consists in identifying cluster regions, i.e. points within a distance ϵ having enough neighbours (the minimum size of a cluster), and noise regions, i.e. points that lie in low-density regions. Although the results for DBSCAN can be very accurate in the identification of the clusters, the parameters used for the DBSCAN algorithm have a strong impact on the final clustering. Thus, the selection of the parameters of the model requires extra attention. Figure A.17 illustrates the sensitivity of the DBSCAN algorithm in the case of a 2D

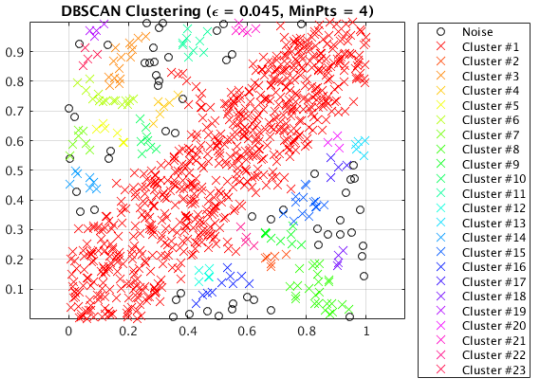
set of particles non-uniformly distributed in space (here composed of uniform particles in a box together with particles along the diagonal). More precisely, the parameters of the model have been chosen following the rule of thumb $\texttt{MinPts} = 2d$ and ϵ taken from a k-distance graph. It appears that $\epsilon = 0.045$ gives here the most accurate results for cluster identification but the clusters identified with other values of ϵ differ significantly from the expected result.



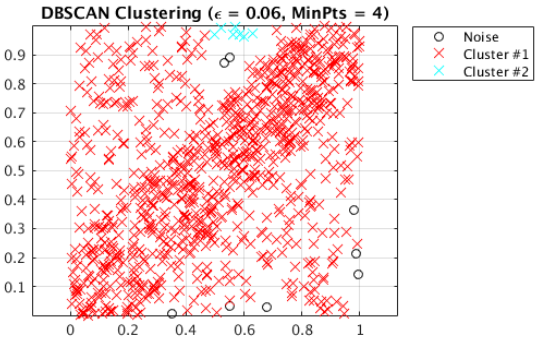
(a) k-distance graph



(b) DBSCAN clustering with $\epsilon = 0.03$, $\texttt{MinPts} = 4$.



(c) DBSCAN clustering with $\epsilon = 0.045$, $\texttt{MinPts} = 4$.



(d) DBSCAN clustering with $\epsilon = 0.06$, $\texttt{MinPts} = 4$.

Figure A.17: DBSCAN clustering algorithm on a set of particles in 2D: comparison of the results obtained with various parameters $\texttt{MinPts} = 4$ and $\epsilon = 0.03, 0.045, 0.06$. Values of ϵ are chosen from the k-distance graph where, for each data point, the distance to the 4th nearest point is displayed.

In this study, we are interested in having an algorithm that detects non-homogeneous particle density regardless of the initial set of particles. The DBSCAN algorithm is thus not appropriate here since it requires a-priori information on some parameters. Besides, these methods focus on the classification of data points, in contrast with our more general scope of constructing a separate regions of space, which separates the points having different concentrations. The computational costs of some of these algorithms (e.g. scaling with n^2) are also not compatible with CFD simulations, which require fast evaluations of the spatial distribution of

particles for each iteration in time. For these reasons, we have not opted for such clustering algorithms in the present study.

Appendix B. Detail of the D2SD algorithm

Algorithm 1: The D2SD algorithm.

Data: Input sample $\mathbf{X}$
Result: Optimal spatial decomposition : `Optimal_Mesh` with 8 labels
`reject_uniformity` = `uniformity_test`($\mathbf{X}$);
Compute `bin_size`;
while reject_uniformity **do**
  Compute $S_{\texttt{ideal}}$;
  for i in `position_of_bins` **do**
    $\texttt{app\_pdf}(i) = \frac{\#\text{observations in bin } i}{n \times \texttt{Volume\_of\_bin}}$
  end
  Compute the `reduced-pdf`;
  Compute the percentiles Q_{20}, Q_{60} and Q_{80};
  `threshold_range` = `unique(sort(app_pdf`$(Q_{60} <$ `app_pdf` & `app_pdf` $\leq Q_{80})))$;
  for λ in `threshold_range` **do**

$$\texttt{Cluster\_detected} = \begin{cases} 1 & \text{if } \texttt{app\_pdf} \geq \lambda \\ 0 & \text{otherwise.} \end{cases}$$

    Deselect bins selected in previous iterations;
    `Detected_anomalies`$\{1{:}5\}$ = `classify_clusters_by_concentrations(Cluster_detected)`;

$$\texttt{Detected\_anomalies}\{6\} = \begin{cases} 1 & \text{if } 0 < \texttt{app\_pdf} \leq Q_{20} \\ 0 & \text{otherwise.} \end{cases}$$

$$\texttt{Detected\_anomalies}\{7\} = \begin{cases} 1 & \text{if } \texttt{app\_pdf} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

    `Mesh`(λ) = `construct_mesh(Detected_anomalies)`;
    Compute concentration of ambient uniform bins, $c_{\texttt{unif}}$;
    for `Cell` in `Mesh`(λ) **do**
      `empty_cell(Cell)`;
      Inject $M = c_{\texttt{unif}} \times$ `Volume(Cell)` uniform ghosts in `Cell`;
    end
    Compute score $S(\lambda)$;
  end
  Select optimal threshold: $\lambda^* = \texttt{argmin}_\lambda \frac{|S_{\texttt{ideal}} - S(\lambda)|}{S_{\texttt{ideal}}}$, for λ in `threshold_range`;
  Update selected bins;
  `merge_grid(Mesh(`λ^*`),Optimal_Mesh)`;
  `reject_uniformity` = `uniformity_test`($\mathbf{X}^{\lambda^*}_{\texttt{final}}$);
end
Note that the 8th label are cells with ambient concentration which are not detected as anomalies.

Once the construction of the optimal spatial decomposition is completed, we compute agglomeration in each cell following PBE based methods (see Algorithm 2 below), knowing now that well-mixed condition is locally fulfilled.

Algorithm 2: Compute agglomeration with the D2SD optimal mesh.

```
Result: Simulation of agglomeration
for j = 1, 2, ..., N do
    Data: Position of n particles {X_t^1, ..., X_t^n}
    Optimal mesh = D2SD_method(X_t^1, ..., X_t^n);
    for Cell in Optimal_Mesh do
        Compute agglomeration in Cell (applying Formula (3) in [55];
    end
end
```

Remark. *A few remarks are due on the D2SD algorithm:*

- *When we opt for the implementation of the projected score, the computation of the ideal score must be done for each threshold since its mean value is expected to vary with respect to n, and the variance of the score is higher (see Fig. 4).*
- *Due to the randomness of the ghost particles, the chosen threshold will have a random nature. This randomness can be diminished by taking average on the realization of ghosts and the computation of the final score $S(\lambda)$. The averaging will increase the computational time. Nonetheless, given that the algorithm does not require greater computational effort, the increment of the computational time is expected to be low.*
- *Since the score represents a distance between the sample distribution and the uniform distribution, it can be used as a second phase of the uniformity test. More precisely, in case the test rejects the uniformity, we evaluate the distance between the ideal score and the score of the observations. If this distance is too small, we will consider the result of the uniformity test as a false negative, and accept the uniformity. Therefore, we integrate the distance to the boundary into the battery of tests.*

Authors' contributions

The work on the D2SD algorithm was initiated by CH and the study was conceived and designed by CH and MB. SS started the development of the D2SD algorithm during his master internship under the supervision of MB and CH. KMR significantly improved the first draft of the D2SD algorithm, added the uniformity checks and carried out all the numerical simulations using the D2SD algorithm. CH and RM carried out the CFD simulation used as a 3D input data. KMR, MB and CH performed the data analysis. KMR, CH and MB drafted the manuscript. All authors read and approved the manuscript.

Data access

The data that support the findings of this study are available from the corresponding author on request.

Funding

This work was performed using computer resources from INRIA Sophia Antipolis.

Acknowledgement

We acknowledge Jérémie Bec and Jean Pierre Minier for useful and constructive discussions on the method validation. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur and Inria Sophia Antipolis - Méditerranée "NEF" computation platform for providing resources and support.

References

[1] R. H. Meade, Transport and deposition of sediments in estuaries, Geological Society of America 133 (1) (1972) 91–120. doi:https://doi.org/10.1130/MEM133-p91.

[2] K. Gotoh, Y. Fujii, A fractal dimensional analysis on the cloud shape parameters of cumulus over land, Journal of Applied Meteorology 37 (10) (1998) 1283–1292. doi:https://doi.org/10.1175/1520-0450(1998)037<1283:AFDAOT>2.0.CO;2.

[3] G. Falkovich, A. Fouxon, M. Stepanov, Acceleration of rain initiation by cloud turbulence, Nature 419 (6903) (2002) 151. doi:https://doi.org/10.1038/nature00983.

[4] R. A. Shaw, Particle-turbulence interactions in atmospheric clouds, Annual Review of Fluid Mechanics 35 (1) (2003) 183–227. doi:https://doi.org/10.1146/annurev.fluid.35.101101.161125.

[5] J. Blum, G. Wurm, The growth mechanisms of macroscopic bodies in protoplanetary disks, Annu. Rev. Astron. Astrophys. 46 (2008) 21–56. doi:https://doi.org/10.1146/annurev.astro.46.060407.145152.

[6] M.-F. Pouet, A. Grasmick, Urban wastewater treatment by electrocoagulation and flotation, Water Science and Technology 31 (3-4) (1995) 275–283. doi:https://doi.org/10.1016/0273-1223(95)00230-K.

[7] J. Rubio, M. Souza, R. Smith, Overview of flotation as a wastewater treatment technique, Minerals Engineering 15 (3) (2002) 139–155. doi:https://doi.org/10.1016/S0892-6875(01)00216-3.

[8] M. Bartels, W. Lin, J. Nijenhuis, F. Kapteijn, J. R. Van Ommen, Agglomeration in fluidized beds at high temperatures: Mechanisms, detection and prevention, Progress in Energy and Combustion Science 34 (5) (2008) 633–666. doi:https://doi.org/10.1016/j.pecs.2008.04.002.

[9] L. Gallen, A. Felden, E. Riber, B. Cuenot, Lagrangian tracking of soot particles in LES of gas turbines, Proceedings of the Combustion Institute 37 (4) (2019) 5429–5436. doi:https://doi.org/10.1016/j.proci.2018.06.013.

[10] N. Maximova, O. Dahl, Environmental implications of aggregation phenomena: current understanding, Current Opinion in Colloid & Interface Science 11 (4) (2006) 246–266. doi:https://doi.org/10.1016/j.cocis.2006.06.001.

[11] J. P. Bernacki, R. M. Murphy, Model discrimination and mechanistic interpretation of kinetic data in protein aggregation studies, Biophysical Journal 96 (7) (2009) 2871–2887. doi:https://doi.org/10.1016/j.bpj.2008.12.3903.

[12] D. Henning, R. Baer, A. Hassan, R. Dave, Major advances in concentrated and dry milk products, cheese, and milk fat-based spreads, Journal of Dairy Science 89 (4) (2006) 1179–1188. doi:https://doi.org/10.3168/jds.S0022-0302(06)72187-7.

[13] A. D. McNaught, A. Wilkinson, IUPAC: Compendium of chemical terminology, 2nd ed. (The "Gold Book"), online version (2019-) created by S. J. Chalk. Edition, Blackwell Scientific Publications, Oxford, 1997. doi:https://doi.org/10.1351/goldbook.

[14] M. Elimelech, J. Gregory, X. Jia, Particle deposition and aggregation: measurement, modelling and simulation, Butterworth-Heinemann, 2013. doi:https://doi.org/10.1016/B978-0-7506-7024-1.X5000-6.

[15] C. Henry, J.-P. Minier, G. Lefèvre, Towards a description of particulate fouling: From single particle deposition to clogging, Advances in colloid and interface science 185 (2012) 34–76. doi:https://doi.org/10.1016/j.cis.2012.10.001.

[16] M. Chen, K. Kontomaris, J. McLaughlin, Direct numerical simulation of droplet collisions in a turbulent channel flow. Part I: collision algorithm, International Journal of Multiphase Flow 24 (7) (1999) 1079–1103. doi:https://doi.org/10.1016/S0301-9322(98)00007-X.

[17] H. Sigurgeirsson, A. Stuart, W.-L. Wan, Algorithms for particle-field simulations with collisions, Journal of Computational Physics 172 (2) (2001) 766–807. doi:https://doi.org/10.1006/jcph.2001.6858.

[18] J. Bec, S. S. Ray, E. W. Saw, H. Homann, Abrupt growth of large aggregates by correlated coalescences in turbulent flow, Physical Review E 93 (3) (2016) 031102. doi:https://doi.org/10.1103/PhysRevE.93.031102.

[19] J.-P. Minier, E. Peirano, The pdf approach to turbulent polydispersed two-phase flows, Physics Reports 352 (1-3) (2001) 1–214. doi:https://doi.org/10.1016/S0370-1573(01)00011-4.

[20] J.-P. Minier, On Lagrangian stochastic methods for turbulent polydisperse two-phase reactive flows, Progress in Energy and Combustion Science 50 (2015) 1–62. doi:https://doi.org/10.1016/j.pecs.2015.02.003.

[21] J.-P. Minier, Statistical descriptions of polydisperse turbulent two-phase flows, Physics Reports 665 (2016) 1–122. doi:https://doi.org/10.1016/j.physrep.2016.10.007.

[22] S. Subramaniam, Lagrangian–Eulerian methods for multiphase flows, Progress in Energy and Combustion Science 39 (2-3) (2013) 215–245. doi:https://doi.org/10.1016/j.pecs.2012.10.003.

[23] G. Bird, Approach to translational equilibrium in a rigid sphere gas, Physics of Fluids 6 (1963) 1518–1519. doi:https://doi.org/10.1063/1.1710976.

[24] G. A. Bird, J. Brady, Molecular gas dynamics and the direct simulation of gas flows, Vol. 42, Clarendon press Oxford, 1994.

[25] F. J. Alexander, A. L. Garcia, The direct simulation Monte Carlo method, Computers in Physics 11 (6) (1997) 588–593. doi:https://doi.org/10.1063/1.168619.

[26] C. Henry, J.-P. Minier, M. Mohaupt, C. Profeta, J. Pozorski, A. Tanière, A stochastic approach for the simulation of collisions between colloidal particles at large time steps, International Journal of Multiphase Flow 61 (2014) 94–107. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2014.01.007.

[27] C. Henry, J.-P. Minier, J. Pozorski, G. Lefèvre, A new stochastic approach for the simulation of agglomeration between colloidal particles, Langmuir 29 (45) (2013) 13694–13707. doi:https://doi.org/10.1021/la403615w.

[28] P. J. O'Rourke, Collective drop effects on vaporizing liquid sprays, Tech. rep., Los Alamos National Lab., NM (USA) (1981).

[29] M. Mezhericher, A. Levy, I. Borde, Probabilistic hard-sphere model of binary particle–particle interactions in multiphase flow of spray dryers, International Journal of Multiphase Flow 43 (2012) 22–38. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2012.02.009.

[30] D. P. Schmidt, C. Rutland, A new droplet collision algorithm, Journal of Computational Physics 164 (1) (2000) 62–80. doi:https://doi.org/10.1006/jcph.2000.6568.
[31] C. Henry, K. K. Norrfors, M. Olejnik, M. Bouby, J. Luetzenkirchen, S. Wold, J.-P. Minier, A refined algorithm to simulate latex colloid agglomeration at high ionic strength, Adsorption 22 (4-6) (2016) 503–515. doi:https://doi.org/10.1007/s10450-015-9714-4.
[32] M. V. Smoluchowski, Versuch einer mathematischen Theorie der Koagulationskinetik kolloider Lösungen, Zeitschrift für Physikalische Chemie 92 (1) (1918) 129–168. doi:https://doi.org/10.1515/zpch-1918-9209.
[33] D. Ramkrishna, The status of population balances, Reviews in Chemical Engineering 3 (1) (1985) 49–95. doi:https://doi.org/10.1515/REVCE.1985.3.1.49.
[34] D. Ramkrishna, Population balances: Theory and applications to particulate systems in engineering, Elsevier, 2000.
[35] D. L. Marchisio, R. O. Fox, Solution of population balance equations using the direct quadrature method of moments, Journal of Aerosol Science 36 (1) (2005) 43–73. doi:https://doi.org/10.1016/j.jaerosci.2004.07.009.
[36] D. Ramkrishna, M. R. Singh, Population balance modeling: current status and future prospects, Annual Review of Chemical and Biomolecular Engineering 5 (2014) 123–146. doi:https://doi.org/10.1146/annurev-chembioeng-060713-040241.
[37] G. Kasper, On the coagulation rate of aerosols with spatially inhomogeneous particle concentrations, Journal of Colloid and Interface Science 102 (2) (1984) 560–562. doi:https://doi.org/10.1016/0021-9797(84)90261-3.
[38] D. P. Schmidt, C. J. Rutland, Reducing grid dependency in droplet collision modeling, J. Eng. Gas Turbines Power 126 (2) (2004) 227–233. doi:https://doi.org/10.1115/1.1564066.
[39] S. Hou, D. P. Schmidt, Adaptive collision meshing and satellite droplet formation in spray simulations, International Journal of Multiphase Flow 32 (8) (2006) 935–956. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2006.02.013.
[40] P. Pischke, D. Cordes, R. Kneer, A collision algorithm for anisotropic disperse flows based on ellipsoidal parcel representations, International Journal of Multiphase Flow 38 (1) (2012) 1–16. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2011.09.002.
[41] J. Zhang, J. Mi, H. Wang, A new mesh-independent model for droplet/particle collision, Aerosol Science and Technology 46 (6) (2012) 622–630. doi:https://doi.org/10.1080/02786826.2011.649809.
[42] P. Pischke, R. Kneer, D. P. Schmidt, A comparative validation of concepts for collision algorithms for stochastic particle tracking, Computers & Fluids 113 (2015) 77–86. doi:https://doi.org/10.1016/j.compfluid.2015.01.018.
[43] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2) (2015) 165–193. doi:https://doi.org/10.1007/s40745-015-0040-1.
[44] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, C.-T. Lin, A review of clustering techniques and developments, Neurocomputing 267 (2017) 664–681. doi:https://doi.org/10.1016/j.neucom.2017.06.053.
[45] R. Monchaux, M. Bourgoin, A. Cartellier, Analyzing preferential concentration and clustering of inertial particles in turbulence, International Journal of Multiphase Flow 40 (2012) 1–18. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2011.12.001.
[46] R. B. D'Agostino, Goodness-of-fit-techniques, Vol. 68, CRC press, 1986.
[47] J.-J. Liang, K.-T. Fang, F. Hickernell, R. Li, Testing multivariate uniformity and its applications, Mathematics of Computation 70 (233) (2001) 337–355. doi:https://doi.org/10.1090/S0025-5718-00-01203-5.
[48] N. Henze, B. Zirkler, A class of invariant consistent tests for multivariate normality, Communications in Statistics-Theory and Methods 19 (10) (1990) 3595–3617. doi:https://doi.org/10.1080/03610929008830400.
[49] R. A. Fisher, Statistical methods for research workers, in: Breakthroughs in statistics, Springer, 1992, pp. 66–70. doi:https://doi.org/10.1007/978-1-4612-4380-9_6.
[50] D. Freedman, P. Diaconis, On the histogram as a density estimator: L2 theory, Probability Theory and Related Fields 57 (4) (1981) 453–476. doi:https://doi.org/10.1007/BF01025868.
[51] J. R. Berrendero, A. Cuevas, F. Vázquez-grande, Testing multivariate uniformity: The distance-to-boundary method, Canadian Journal of Statistics 34 (4) (2006) 693–707. doi:https://doi.org/10.1002/cjs.5550340409.
[52] J. Bec, L. Biferale, M. Cencini, A. Lanotte, S. Musacchio, F. Toschi, Heavy particle concentration in turbulence at dissipative and inertial scales, Physical Review Letters 98 (8) (2007) 084502. doi:https://doi.org/10.1103/PhysRevLett.98.084502.
[53] F. Toschi, E. Bodenschatz, Lagrangian properties of particles in turbulence, Annual review of fluid mechanics 41 (2009) 375–404. doi:https://doi.org/10.1146/annurev.fluid.010908.165210.
[54] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[55] J. Kim, T. A. Kramer, Improved orthokinetic coagulation model for fractal colloids: Aggregation and breakup, Chemical Engineering Science 61 (1) (2006) 45–53. doi:https://doi.org/10.1016/j.ces.2005.01.044.