AIChE

New Quadrature-Based Moment Method for the Mixing of Inert Polydisperse Fluidized Powders in Commercial CFD Codes

Luca Mazzei
Dept. of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK

Daniele L. Marchisio
Dept. di Scienza dei Materiali e Ingegneria Chimica, Politecnico di Torino, 10129, Torino, Italy

Paola Lettieri
Dept. of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK

DOI 10.1002/aic.13714
Published online January 17, 2012 in Wiley Online Library (wileyonlinelibrary.com).

To describe the behavior of polydisperse multiphase systems in an Eulerian framework, we solved the population balance equation (PBE), letting it account only for particle size dependencies. To integrate the PBE within a commercial computational fluid dynamics code, we formulated and implemented a novel version of the quadrature method of moments (QMOM). This no longer assumes that the particles move with the same velocity, allowing the latter to be size-dependent. To verify and test the model, we simulated the mixing of inert polydisperse fluidized suspensions initially segregated, validating the results experimentally. Because the accuracy of QMOM increases with the number of moments tracked, we ran three classes of simulations, preserving the first four, six, and eight integer moments of the particle density function. We found that in some cases the numerics corrupts the higher-order moments and a corrective algorithm, designed to restore the validity of the moment set, has to be implemented. © 2012 American Institute of Chemical Engineers *AIChE J,* 58: 3054–3069, 2012
Keywords: multiphase flows, Fluidization, quadrature method of moments, population balance, moment corruption

Introduction

Several industrial processes, such as catalytic polymerization, combustion, and gasification, involve fluidized bed reactors. These are attractive because they maximize the contact area between the phases and guarantee excellent heat and mass transfer. Even so, developing, innovating, and scaling up these processes is still quite challenging, because the dynamics and reactive behavior of fluidized suspensions are extremely difficult to predict and control. The complexity originates from the many physical and chemical phenomena that occur concurrently: chemical reactions take place, which usually affect the properties of the particles; in addition, these can aggregate or break into subelements, whereas others can form through nucleation. The end-product quality strongly depends on all these competing phenomena, which in turn are influenced by the suspension fluid dynamics and, indirectly, by the reactor internals, geometry, and size.

To design these units, process engineers have resorted for many years to experimental correlations and pilot plants. However, since these correlations are valid only for the specific units investigated, they cannot help engineers to innovate or improve design and performance; pilot plants, conversely are expensive and time-consuming, not always leading to adequate scale up.

Correspondence concerning this article should be addressed to L. Mazzei at l.mazzei@ucl.ac.uk.

© 2012 American Institute of Chemical Engineers

Thanks to the availability of high-speed computer processors, computational fluid dynamics (CFD) plays nowadays a key role in understanding the behavior of multiphase systems and in particular fluidized beds. The improvement in accuracy of recent fluid dynamic models[1–5] has substantially increased the interest of industry in this technique; nevertheless, since many limitations in the predictive capabilities of such models still exist, much theoretical research is required to turn CFD into a fully reliable design tool. One of the assumptions which restricts even the most advanced models is the particles having constant and equal size.[6–13] As just pointed out, in industrial processes there exists a particle size distribution (PSD), whose changes in time and space reflect the course of the very physical and chemical phenomena characterizing the processes. These changes in PSDs are associated with the possible occurrence of segregation phenomena, which result into uneven distribution of the particles within the bed. Depending on the application at hand, segregation may be beneficial or detrimental,[14–17] but in either case being able to predict its extent and dynamics is key to properly design and operate fluidized bed reactors.

To partially overcome this limitation, research groups have extended to polydisperse suspensions models originally developed for monodisperse. This approach still hinges on the constant-size assumption, but now two or more particle classes differing in size are accounted for, so that powders can segregate.[18–27] Even so, variations in size are not allowed for, whereas in reality particles can shrink, aggregate, break, and nucleate, their size distribution varying

continuously in time and space. Predicting this evolution, which depends upon the local conditions wherein the system operates, is essential for a reliable description of the suspension behavior, but requires a more powerful modeling strategy.

To account for size-changing phenomena, which characterize the physics and chemistry of the process at hand, and describe how the PSD evolves locally within the reactor, we need to solve, along with, or in place of, the averaged transport equations of conservation of mass, linear momentum and possibly energy, a population balance equation (PBE). Doing so, however, is not trivial, because the dimensionality of this equation depends on the application and on the strategy that the modeler wishes to use (e.g., on how many internal coordinates he uses to characterize the state of the particles). Hence, PBEs are not necessarily three-dimensional (3-D) and cannot be easily integrated within customary CFD codes. In the context of multiphase flows, not so many research groups have used this modeling approach. Olmos et al.[28] simulated bubble columns considering 10 different size classes to represent the bubbles, but solved only the dynamical equation for the mixture. The bubbles consequently shared the same velocity. Using similar strategies, other groups have simulated gas–liquid systems.[29–31] Dense fluid–solid systems, conversely, in which the phases strongly interact and move with different velocities, have been investigated much less.[32–34]

Various techniques can solve PBEs numerically; for a comprehensive review we refer to Ramkrishna. [35] Here, we focus on the so-called method of moments. Frequently engineers do not really need to know the particle density function, which describes how the population of elements is distributed locally over the properties of interest, but are only interested in some integral properties of the latter. Such properties, called moments, may be important because they control the product quality or because they are easy to measure and monitor. The idea behind the method of moments is to derive transport equations for the moments of interest by integrating out the internal coordinates from the PBE.[36] The method is attractive, because the transport equations that govern the moments are 3-D and the number of moments to be tracked is small; however, the transport equations are unclosed, because for any given set of moments that the modeler wishes to consider, the equations normally involve also higher-order moments external to the set.[37,38]

The quadrature method of moments (QMOM), which approximates the particle density function using a quadrature formula, overcomes this problem; turning integrals of the density function into summations, the formula eliminates the problem of closure.[39,40] To compute the quadrature nodes and weights, QMOM forces them to agree with a set of independent lower-order moments[41] that the model tracks by integrating their transport equations. From this set, QMOM then determines the finite-mode representation of the density function.

For monovariate distributions, that is, distributions with only one internal coordinate, to back-calculate the quadrature nodes and weights from the moments of the density function we can adopt the product-difference (PD) algorithm of Gordon,[42] which requires finding the eigenvalues of a real symmetrical tridiagonal matrix, or the algorithm of Wheeler.[43] Nevertheless, these algorithms cannot be applied when a higher number of internal coordinates is present. The quadrature approximation must then be determined using multivariate inversion algorithms, such as Brute-Force methods,[44] Tensor-Product methods,[45] or Conditional QMOM.[46] Even if some of these algorithms are very efficient, in this work we let the PBE account only for size dependencies; under this hypothesis, particles with the same size move with the same velocity, the latter being excluded from the set of internal coordinates, and the density function is monovariate. We then solve the PBE with the averaged dynamical equations of multiphase flows, adopting a hybrid approach.

In the present work, we develop and implement a new version of QMOM into the multifluid model of the commercial CFD code Fluent. There are two important novelties: (1) the model is based on a volume, and not on a number, density function, so that it deals with volume fractions instead of number densities, and (2) the particles no longer share the same velocity, so that they can freely mix and segregate. The method is quite general and can treat any type of particulate process, but in what follows we verify and validate it on a simple process in which the particles neither react nor agglomerate nor break. The PSD changes solely because the powders mix. This is a relatively simple problem, but its very simplicity is key to test the method, understand it better and highlight possible issues or limitations. We believe that before tackling more complex problems, involving continuous and discontinuous changes in particle size, this analysis is necessary.

The article is thus structured. First, we introduce the system investigated. Next, we describe the experimental methodology and findings. We then present the mathematical model and the numerics, reporting the predictions of the simulations and showing how these compare with the experimental data. As we shall see, when QMOM is solved with spatial discretization schemes that use higher-order numerical schemes or when it tracks a sufficiently high number of moments (eight in this work), some moments corrupt, this leading to poor results or generating instabilities that eventually make the simulations crash altogether. We thus discuss the problem of moment corruption and present a few strategies that may be able to overcome it. One of these, reported by Wright,[47] is implementing a corrective algorithm that replaces invalid moment sets with valid ones in the cells where moments corrupt. To conclude the article, we describe and discuss this method, assessing its potentials and limitations.

Goal of this Work

We intend to describe the mixing of two inert polydisperse fluidized powders initially segregated and test our new QMOM model. The physical system that we are going to investigate is a packed bed constituted of two superposed layers of polydisperse particles of equal density. Referred to as powders A and B, respectively, lower and upper layers differ only in PSD, the one on top having greater mean particle size. To thoroughly mix the powders, we feed fluid at a superficial velocity much greater than both minimum fluidization velocities u_A and u_B. The resulting powder, referred to as powder C, has a PSD that combines the two original ones. Adopting the QMOM, we intend to predict the new PSD and see how it compares with the experimental one.

Experimental

A description of the experimental apparatus, shown in Figure 1 along with its schematic representation, and a detailed discussion of the experimental results are reported in our previous work.[34] Here, let us just remind that the

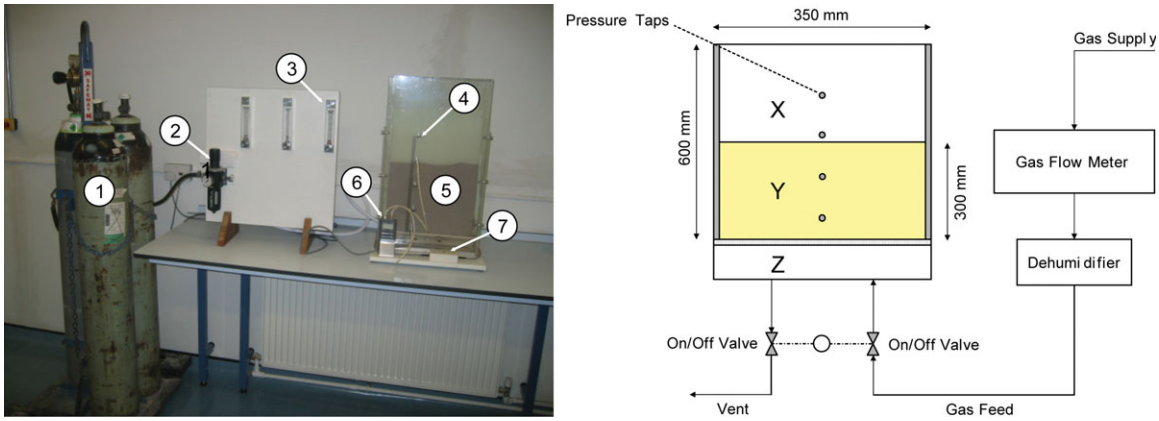

Figure 1. Photograph and schematic representation of the experimental apparatus. (1) Nitrogen tanks, (2) oil filter, (3) flow meters, (4) pressure taps, (5) fluidized bed, (6) electronic manometer, and (7) on/off valves control switch, (X) freeboard, (Y) fluidized bed, and (Z) windbox.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

powders are ballotini particles with density of 2500 kg/m^3. Figure 2 reports their density and cumulative PSDs found by sieving. For powders A and B, the Sauter average diameters,[48] coefficients of variation[36] and minimum fluidization velocities are equal to 88 μm, 273 μm, 0.16, 0.20, 1.00 cm/s and 6.40 cm/s, respectively.

When fluidized, the powders mix almost perfectly, the PSDs in the bed being nearly identical everywhere. Figures 3A,B report the PSDs averaged over the most significant bed layers: the lowest, which lies on the distributor, and the highest, which separates the bed from the freeboard. The new PSDs are identical and seem to be obtained by juxtaposing the two original distributions reported in Figures 2A,C; this indicates excellent mixing.

Multiphase Fluid Dynamic Model

The particle population that we consider is characterized by diameter and velocity; so, there are two internal coordinates, one scalar and one vectorial, and the internal state space is 4-D. To describe the population of particles, we introduce a volume density function (VDF); denoted by f_v, this is defined so that $f_v(s,\boldsymbol{v},\boldsymbol{x},t)\ ds\ d\boldsymbol{v}\ d\boldsymbol{x}$ represents the expected volume of particles contained at time t in the physical volume $d\boldsymbol{x}$ around $\boldsymbol{x}$ with size s in the range ds and velocity $\boldsymbol{v}$ in the range $d\boldsymbol{v}$.

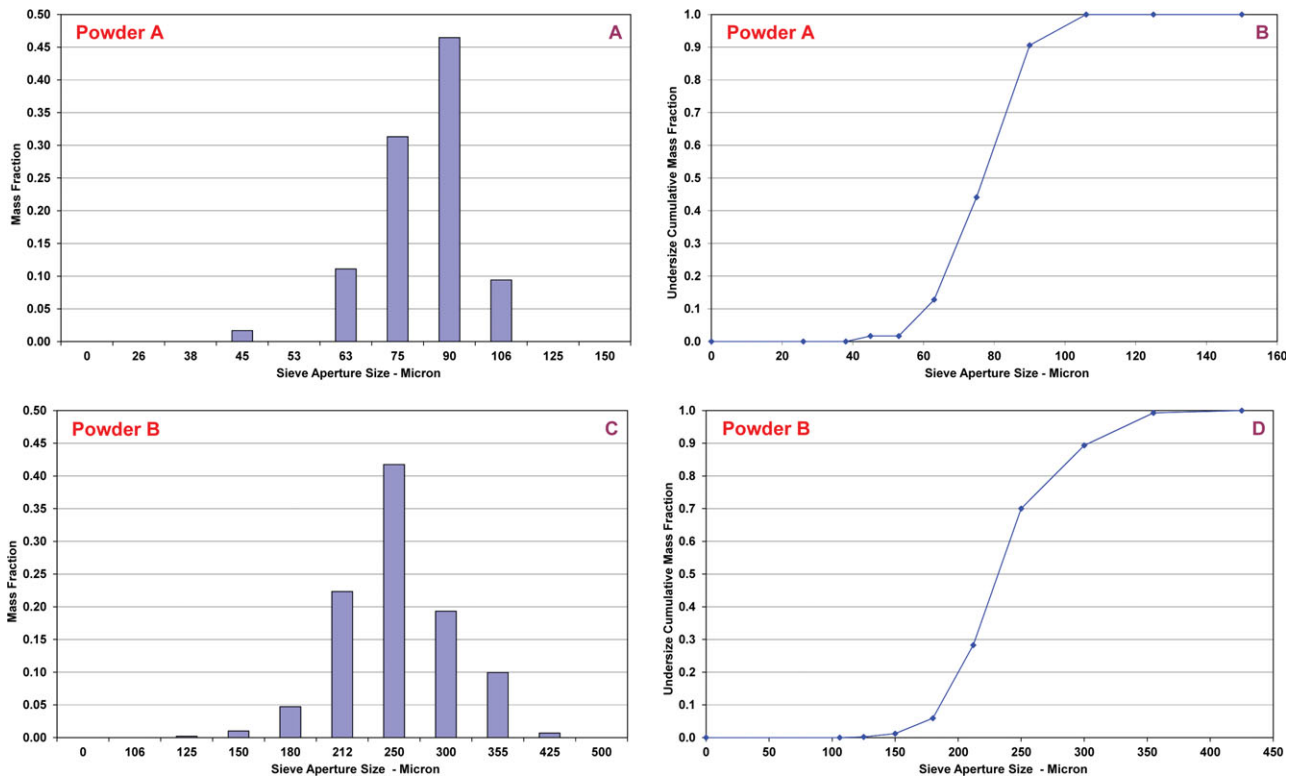

Figure 2. Experimental normal and cumulative PSDs for powders (A) and (B).

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

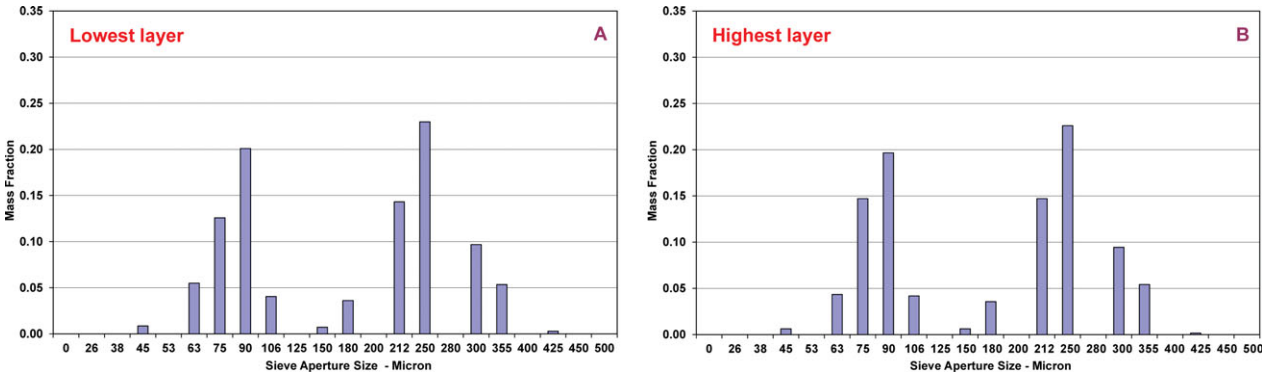

Figure 3. Experimental PSDs in the top (near the freeboard) and bottom (near the distributor plate) layers of the bed after collapse.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

We denote the domains of variation of s and $\boldsymbol{v}$ by $\Omega_s \equiv \mathbb{R}^+$ and $\Omega_v \equiv \mathbb{R}^3$, respectively. The reason for preferring a volume to a number density function is that the former deals directly with volume fractions, a more usual choice when one treats fluidized systems.

To determine f_v we should solve a PBE written in the 4-D internal state space mentioned earlier. Because neither heterogeneous reactions nor particle attrition occur, s does not vary continuously and the particles have zero velocity in size space; so, the PBE reads

$$\frac{\partial f_v}{\partial t} + \nabla_{\boldsymbol{x}} \cdot (f_v\, \boldsymbol{v}) + \nabla_v \cdot (f_v\, \dot{\boldsymbol{v}}) = \frac{\partial_e f_v}{\partial t} \tag{0.1}$$

where $\nabla_x \cdot$ and $\nabla_v \cdot$ are divergence operators in the physical and velocity spaces, respectively, $\dot{\boldsymbol{v}}$ is the particle acceleration and $\partial_e f_v/\partial t$ is a source term that accounts for discontinuous jumps in the particle state space. Instead of solving Eq. 0.1 directly, we approximate the VDF using a quadrature formula, which expresses f_v as a summation of v Dirac delta functions

$$f_v(s,\boldsymbol{v},\boldsymbol{x},t) \approx \sum_{r=1}^{v} \phi_r(\boldsymbol{x},t)\; \delta[s - s_r(\boldsymbol{x},t)]\delta[\boldsymbol{v} - \boldsymbol{v}_r(\boldsymbol{x},t)] \tag{0.2}$$

where $s_r(\boldsymbol{x},t)$ and $\boldsymbol{v}_r(\boldsymbol{x},t)$ are the rth quadrature nodes and $\phi_r(\boldsymbol{x},t)$ is the rth quadrature weight. Eq. 0.2 states that the particle population is represented by v classes, the rth of which having volume fraction $\phi_r(\boldsymbol{x},t)$, diameter $s_r(\boldsymbol{x},t)$ and velocity $\boldsymbol{v}_r(\boldsymbol{x},t)$. Therefore, the problem reduces to predicting the evolution in time and space of these $3v$ functions.

This choice of internal coordinates – particle size and velocity – results in a multivariate density function. As previously mentioned, although multivariate inversion algorithms exist, in this work we first reduce the dimensionality of the problem before going on to solve it. Being the velocity a vector, this is the coordinate that we should eliminate to turn the VDF into a monovariate distribution. In the following sections, we first do this and then develop our new version of the QMOM model.

Reduction of the VDF dimensionality

To reduce the dimensionality of the VDF, we integrate out the coordinate $\boldsymbol{v}$ from Eq. 0.1; this results in the reduced PBE

$$\frac{\partial \bar{f}_v}{\partial t} + \nabla_x \cdot (\bar{f}_v\, \langle \boldsymbol{v}|s\rangle) = \frac{\partial_e \bar{f}_v}{\partial t} \tag{0.3}$$

where, by definition, it is

$$\bar{f}_v \equiv \int_{\Omega_v} f_v\, d\boldsymbol{v}; \quad \bar{f}_v\, \langle \boldsymbol{v}|s\rangle \equiv \int_{\Omega_v} f_v\, \boldsymbol{v}\, d\boldsymbol{v}; \quad \frac{\partial_e \bar{f}_v}{\partial t} \equiv \int_{\Omega_v} \frac{\partial_e f_v}{\partial t}\, d\boldsymbol{v} \tag{0.4}$$

In this equation, $\langle \boldsymbol{v}|s\rangle(s,\boldsymbol{x},t)$ is the mean particle velocity conditioned on the particle size s. Note that being $\langle \boldsymbol{v}|s\rangle$ size-dependent, the reduced PBE features no diffusive flux in physical space. This is because particles with different size are convected with different velocity. Spatial diffusion would arise if we replaced $\langle \boldsymbol{v}|s\rangle$ with the mean velocity of the whole particle population, which would be averaged over s.[49] Using Eq. 0.2 to approximate the VDF, we find

$$\bar{f}_v(s,\boldsymbol{x},t) \approx \sum_{r=1}^{v} \phi_r(\boldsymbol{x},t)\; \delta[s - s_r(\boldsymbol{x},t)] \quad \text{and}$$
$$\langle \boldsymbol{v}|s\rangle[s_r(\boldsymbol{x},t),\boldsymbol{x},t] = \boldsymbol{v}_r(\boldsymbol{x},t) \tag{0.5}$$

Hence, as expected, the particles belonging to the size class s_r are advected with the velocity $\boldsymbol{v}_r(\boldsymbol{x},t)$, this coinciding with their conditional velocity $\langle \boldsymbol{v}|s\rangle(s_r,\boldsymbol{x},t)$. Note that, as particles neither aggregate nor break, the size-dependent source term $\partial_e \bar{f}_v/\partial t$ vanishes, reducing Eq. 0.3 to

$$\frac{\partial \bar{f}_v}{\partial t} + \nabla_x \cdot (\bar{f}_v\, \langle \boldsymbol{v}|s\rangle) = 0 \tag{0.6}$$

This new PBE governs the evolution of the monovariate function $\bar{f}_v(s,\boldsymbol{x},t)$. We can use it to find the $2v$ functions $\phi_r(\boldsymbol{x},t)$ and $s_r(\boldsymbol{x},t)$, but the equation can no longer provide any information about the velocities $\boldsymbol{v}_r(\boldsymbol{x},t)$. To compute these, we resort to the averaged dynamical equations of multiphase flows (refer to the section *Multifluid dynamical* and *pseudointernal energy equations*). With this hybrid approach, the particle velocity is no longer an internal coordinate and the VDF becomes monovariate, but additional equations are necessary in addition to the PBE, this being left only with the task of governing the PSD evolution.

QMOM transport equations

Our objective is determining the $2v$ scalar functions $\phi_r(\boldsymbol{x},t)$ and $s_r(\boldsymbol{x},t)$, because these would tell us how the PSD evolves in time and space. To this end, our strategy is

developing transport equations that govern the evolution of a set of $2v$ independent moments of the VDF and back-calculating from this set the quadrature nodes and weights, which in general entails solving a nonlinear algebraic system. We remind that, given a function $\varphi(s,\boldsymbol{x},t)$, the integral transform

$$\varphi(s,\boldsymbol{x},t) \rightarrow \mathcal{M}_k(\varphi)\ (\boldsymbol{x},t) \equiv \int_{\Omega_s} \varphi(s,\boldsymbol{x},t)\ s^k\ ds \tag{0.7}$$

defines the moment of order k with respect to the internal coordinate s of $\varphi(s,\boldsymbol{x},t)$. Applying this transform to Eq. 0.6 yields

$$\frac{\partial \mathcal{M}_k}{\partial t} + \nabla_x \cdot (\mathcal{M}_k\ \hat{\boldsymbol{v}}_k) = 0 \tag{0.8}$$

where $\mathcal{M}_k(\boldsymbol{x},t)$ is the kth order moment of $\bar{f_v}(s,\boldsymbol{x},t)$ and $\widehat{\boldsymbol{v}}_k(\boldsymbol{x},t)$ is the velocity with which this moment is advected; this is defined so that

$$\mathcal{M}_k(\boldsymbol{x},t)\ \hat{\boldsymbol{v}}_k(\boldsymbol{x},t) \equiv \int_{\Omega_s} \bar{f_v}(s,\boldsymbol{x},t)\ \langle \boldsymbol{v}|s\rangle(s,\boldsymbol{x},t)\ s^k\ ds \tag{0.9}$$

The quadrature approximation allows us to relate this velocity to the quadrature nodes and weights $\phi_r(\boldsymbol{x},t)$ and $s_r(\boldsymbol{x},t)$ and to the velocities $\boldsymbol{v}_r(\boldsymbol{x},t)$ with which these variables are advected; introducing the relations (0.5) into the defining expression of the moment velocity $\widehat{\boldsymbol{v}}_k(\boldsymbol{x},\boldsymbol{t})$ yields

$$\hat{\boldsymbol{v}}_k(\boldsymbol{x},t) = \sum_{r=1}^{v} p_{kr}(\boldsymbol{x},t)\ \boldsymbol{v}_r(\boldsymbol{x},t) \quad \text{where } p_{kr}(\boldsymbol{x},\boldsymbol{t}) \equiv \frac{\phi_{\boldsymbol{r}}(\boldsymbol{x},\boldsymbol{t})\ s_{\boldsymbol{r}}^{\boldsymbol{k}}(\boldsymbol{x},\boldsymbol{t})}{\mathcal{M}_{\boldsymbol{k}}(\boldsymbol{x},\boldsymbol{t})} \tag{0.10}$$

As we can see, each moment moves with a different velocity, which is a linear combination of the velocities $\boldsymbol{v}_r(\boldsymbol{x},t)$. The transport Eq. 0.8 governs the evolution of the moment of order k of the VDF, allowing us to determine its value in each point of the computational domain as time goes by. Assuming we know the values of $2v$ independent moments in a generic point $\boldsymbol{x}$ at time t, we can then back-calculate the quadrature nodes and weights corresponding to this set of moments. To do this, we need to solve the nonlinear algebraic system made up of the $2v$ equations

$$\mathcal{M}_k(\boldsymbol{x},t) = \sum_{r=1}^{v} \phi_r(\boldsymbol{x},t)\ s_r^k(\boldsymbol{x},t) \tag{0.11}$$

Here, we can choose any set of $2v$ values of k. The algebraic system and the VDF representation, however, depend on the set that we select, because different moments preserve different properties of the distribution. Among the many possible choices, one is particularly accurate: if we preserve the first $2v$ integer moments, $\phi_r(\boldsymbol{x},t)$ and $s_r(\boldsymbol{x},t)$ fulfill the quadrature condition

$$\int_{\Omega_s} \bar{f_v}(s,\boldsymbol{x},t)\ \pi_k(s)\ ds = \sum_{r=1}^{v} \phi_r(\boldsymbol{x},t)\ \pi_k[s_r(\boldsymbol{x},t)] \tag{0.12}$$

for any polynomial $\pi_k(s)$ of degree k with $0 \le k \le 2v-1$ and k integer. Hence, this particular choice of nodes and weights renders Eq. 0.5 a Gaussian quadrature: with v nodes, the approximation reaches an accuracy of order $2v-1$ instead of $v-1$, which is the order of accuracy that a non-Gaussian quadrature formula yields (for more details, we refer to Refs. 38 and 50). Thus, preserving the first $2v$ integer moments of the VDF is the most convenient option from the standpoint of mathematical accuracy. Also, when the quadrature is Gaussian, we can solve the nonlinear algebraic system and determine weights and nodes very efficiently by adopting the PD algorithm of Gordon.[42] For these reasons, in this study we selected this particular set of moments. This choice makes physical sense as well, because the lower-order moments of the distribution relate to important properties of the PSD. For example, $\mathcal{M}_0$ represents the overall solid volume fraction, whereas $\mathcal{M}_1/\mathcal{M}_0$ the volume-averaged particle size. Other important properties of the distribution such as the variance and the skewness, which represent respectively its broadness and its shape, can be calculated from the moments of order two and three.[36,38] These simple examples prove that knowing the first four moments already suffices to solve most problems of engineering interest concerning fluid-solid flows. Should more properties of the distribution be needed, however, more moments can be tracked.

Let us conclude by summarizing the main steps of the method: (1) we track the first $2v$ integer moments of the VDF, choosing $k = 0,1,2,\ldots,2v-1$, where v is the number of quadrature nodes; (2) integrating the transport Eq. 0.8, we compute how the moments evolve in time and space; (3) in any point $\boldsymbol{x}$ of the computational domain and for any time t of interest, we finally solve the nonlinear algebraic system (0.11) using the PD algorithm of Gordon[42] and back-calculate the nodes and weights of the corresponding VDF finite-mode representation.

A final consideration is in order. The moment transport Eq. 0.8 features no diffusive flux because each moment is convected with its own velocity. Similarly to what we said about the PBE, in these equations diffusion would arise if we replaced the velocities $\widehat{\boldsymbol{v}}_k(\boldsymbol{x},t)$ with a mean velocity shared by all moments. If we used this approach, diffusion would appear in the moment transport equations, and we would have only one average dynamical equation to solve. With our approach, conversely, there is no diffusion, but we need to determine the velocity field of each moment of the distribution.

We should point out, nonetheless, that the finite-volume scheme that the CFD code uses to discretize the equations of change (refer to the section *Numerical schemes* and *implementation techniques*) generates numerical diffusion. Hence, the moment transport equation that the code really solves is

$$\frac{\partial \mathcal{M}_k}{\partial t} + \nabla_x \cdot (\mathcal{M}_k\ \hat{v}_k) - \nabla_x \cdot (\mathcal{D}_n\ \nabla_x\ \mathcal{M}_k) = 0 \tag{0.13}$$

where $\mathcal{D}_n$ is a numerical diffusivity that depends on the discretization scheme and on the computational grid. The diffusive flux cannot be eliminated, for numerical diffusion is always present when one integrates purely convective equations with CFD codes. A classical example is given by the multifluid equations of continuity for monodisperse fluidized suspensions, where numerical diffusion smoothes out the spatial volume fraction profiles of the fluid and solid phases.

We can estimate the value of $\mathcal{D}_n$ employing the relation $\mathcal{D}_n \sim uL_c$, where u is the velocity at which the property is convected and L_c is the length of a computational cell. This relation is valid only for first-order upwind discretization schemes, which we indeed used in most of the simulations. Taking as characteristic velocity 0.10 m/s, a value that has

the same order of magnitude as the gas superficial velocity, and L_c equal to 10 mm, we obtain a diffusion coefficient of order of magnitude equal to 10^{-3} m^2/s.

The diffusive term $\nabla_{\boldsymbol{x}} \cdot (\mathcal{D}_{\boldsymbol{n}} \nabla_{\boldsymbol{x}} \mathcal{M}_{\boldsymbol{k}})$, as we well-know, alter the results. Notwithstanding, since the PBE and the moments are linear in the VDF, and since mixing is a linear process, the mixed moments and in turn the mixed VDF are correctly estimated. There is no way around this problem, but numerical diffusion can be reduced if one uses fine computational grids and higher-order discretization schemes. Unfortunately, the latter are less stable than lower-order schemes and in consequence might compromise the numerical stability of the simulations. Furthermore, as we shall discuss later on, higher-order discretization schemes are liable to corrupt higher-order moments, leading to unphysical values of quadrature nodes and eventually making the simulations crash. A tradeoff must therefore be accepted.

Multifluid dynamical and pseudointernal energy equations

We assume that the dynamical equations for the fluid and particle phases (the latter really representing the quadrature nodes) are the customary multifluid equations of multiphase systems, obtained by mathematical averaging.[5] For details we refer to the literature. We have a dynamical equation for the fluid and for each quadrature classes; the first reads

$$\rho_e \left[\frac{\partial}{\partial t} (\varepsilon \, \boldsymbol{u}_e) + \nabla_x \cdot (\varepsilon \, \boldsymbol{u}_e \, \boldsymbol{u}_e) \right] = \nabla_x \cdot \boldsymbol{S}_e - \sum_{r=1}^{v} n_r \boldsymbol{f}_r + \varepsilon \, \rho_e \, \boldsymbol{g} \tag{0.14}$$

where ρ_e and ε are its density and volume fraction, respectively, $\boldsymbol{u}_e$ its averaged velocity and $\boldsymbol{S}_e$ its effective stress tensor. Moreover, n_r is the number density of particle phase r and $\boldsymbol{f}_r$ is the force exerted by the fluid on a single particle of the rth phase. Finally, $\boldsymbol{g}$ is the gravitational field. We do not need a transport equation for ε, because $\varepsilon = 1-\phi$, where ϕ is the sum of all the quadrature weights. The dynamical equation for the rth quadrature class reads

$$\rho_s \left[\frac{\partial}{\partial t} (\phi_r \, \boldsymbol{v}_r) + \nabla_x \cdot (\phi_r \, \boldsymbol{v}_r \, \boldsymbol{v}_r) \right] = \nabla_x \cdot \boldsymbol{S}_r + n_r \boldsymbol{f}_r + \sum_{k=1}^{v} n_r \boldsymbol{f}_{rk} + \phi_r \, \rho_s \, \boldsymbol{g} \tag{0.15}$$

where ρ_s is the solid density (which is the same for all the classes), $\boldsymbol{S}_r$ is the effective stress tensor of phase r and $\boldsymbol{f}_{rk}$ is the force exerted by phase k on a single particle of phase r.

In the equation above, the effective stress $\boldsymbol{S}_r$ accounts for collisions between alike particles, whereas the particle–particle interaction force $\boldsymbol{f}_{rk}$ accounts for collisions between particles of different sizes. Both terms are functions of the granular temperatures of the quadrature classes involved.[51] To find the granular temperature for the rth quadrature class, we solved the following pseudointernal energy balance equation

$$\rho_s \left[\frac{\partial}{\partial t} (\phi_r \, U_r) + \nabla_x \cdot (\phi_r \, U_r \, \boldsymbol{v}_r) \right] = -\nabla_x \cdot \boldsymbol{q}_r + \boldsymbol{S}_r : \nabla_x \boldsymbol{v}_r + G_r^d - S_r^v - S_r^c \tag{0.16}$$

Here, $U_r(\boldsymbol{x},t) \equiv 3\Theta_r(\boldsymbol{x},t)/2$ is the pseudointernal energy, $\Theta_r(\boldsymbol{x},t)$ is the granular temperature and $\boldsymbol{q}_r(\boldsymbol{x},t)$ is the pseudothermal heat flux. The above equation differs from the usual internal energy balance equation because of a sink term $S_r^c(\boldsymbol{x},t)$ representing losses of pseudointernal energy caused by inelastic collisions, a source term $G_r^d(\boldsymbol{x},t)$ representing the generation of particle velocity fluctuations by fluctuating fluid-particle forces, and a sink term $S_r^v(\boldsymbol{x},t)$ representing their dampening by the viscous resistance to particle motion. For the constitutive equations adopted to express the unclosed terms in Eqs 0.14, 0.15 and 0.16, among which we find the effective stress tensors, the fluid–particle interaction forces (consisting of buoyancy and drag forces) and the particle–particle interaction forces, we refer the reader to Mazzei et al.[52] These are standard closure relations for dense fluidized suspensions.

Boundary and initial conditions

The computational grid (uniform, with square cells of 5 mm side) is 2-D; hence, front and back wall effects were neglected. On the left and right walls, we used no-slip boundary conditions. At the bottom of the bed, the inlet gas velocity was set to 15 cm/s. At the domain upper boundary, the pressure was set to 10^5 Pa. On all boundaries, the moment fluxes were set to zero.

To assign the initial conditions, we need to know the values of $2v$ independent moments everywhere within the computational domain. In its initial state, the bed is fixed and made up of two superposed layers; these are 15 mm high, and together occupy half of the vessel. As we know the experimental PSDs in the two layers (refer to Figures 2A,C), we can compute the moments $\mathcal{M}_k(\boldsymbol{x}, t_0)$; being the powders well mixed, in each layer the moments do not depend on $\boldsymbol{x}$ and it is

$$\mathcal{M}_k \approx (1-\varepsilon) \sum_{i=1}^{m} \left[\frac{\sigma_i^{k+1} - \sigma_{i-1}^{k+1}}{(k+1)\,(\sigma_i - \sigma_{i-1})} \right] \omega(\sigma_{i-1}, \sigma_i) \tag{0.17}$$

where m is the number of sieves used (10 in our case), σ_i is the aperture of the ith sieve and $\omega(\sigma_{i-1},\sigma_i)$ is the mass fraction of powder in the size range (σ_{i-1},σ_i). Eq. 0.17 tells us that $\mathcal{M}_k$ is a function of ε; this is because, whereas the PSD refers to solid mass fractions on a void-free basis, the VDF accounts for voids and provides volume densities, that is, volumes of solid per unit volume of physical space.

Table 1 reports the experimental values of VDF moments, nodes and weights for quadratures with two, three and four nodes. First, we computed the moments using Eq. 0.17 and setting $\varepsilon = 0.400$; then, we calculated nodes and weights using the PD algorithm of Gorden.[42] We used the values of the moments to initialize the QMOM transport Eqs. 0.8.

Numerical Schemes and Implementation Techniques

To run the simulations, we used the commercial CFD code Fluent. We implemented the governing and constitutive equations in the multifluid model of the package, which is based on an Eulerian description of the dynamics, using user-defined functions and subroutines. Simulations ran with a quadrature approximation of order two (*i.e.*, $v = 2$) required tracking the evolution of the first four integer moments of the VDF and defining three phases in the multifluid model: one gas and two particle phases. Similarly, when three and four nodes were used (i.e., $v = 3$ and $v = 4$, respectively), the first six and eight integer moments of the

Table 1. Values of the VDF Moments and of the Quadrature Nodes and Weights Obtained from the Experimental PSDs Reported in Figures 2A,C Assuming a Void Fraction of 0.400

Moments of the Volume Density Function				
Powder	$\mathcal{M}_0\,[-]$	$\mathcal{M}_1\,[\mu\mathrm{m}]$	$\mathcal{M}_2\,[\mu\mathrm{m}^2]$	$\mathcal{M}_3\,[\mu\mathrm{m}^3]$
A	0.600	5.45×10^1	5.06×10^3	4.82×10^5
B	0.600	1.70×10^2	4.98×10^4	1.52×10^7
Powder	$\mathcal{M}_4\,[\mu\mathrm{m}^4]$	$\mathcal{M}_5\,[\mu\mathrm{m}^5]$	$\mathcal{M}_6\,[\mu\mathrm{m}^6]$	$\mathcal{M}_7\,[\mu\mathrm{m}^7]$
A	4.67×10^7	4.61×10^9	4.63×10^{11}	4.73×10^{13}
B	4.80×10^9	1.57×10^{12}	5.32×10^{14}	1.86×10^{17}
Quadrature Nodes and Weights for a Two-Node Quadrature Formula				
Powder	$s_1\,[\mu\mathrm{m}]$	$\phi_1\,[-]$	$s_2\,[\mu\mathrm{m}]$	$\phi_2\,[-]$
A	75	0.262	103	0.338
B	240	0.380	355	0.220
Quadrature Nodes and Weights for a Three-Node Quadrature Formula				
Powder	$s_1\,[\mu\mathrm{m}]$	$\phi_1\,[-]$	$s_2\,[\mu\mathrm{m}]$	$\phi_2\,[-]$
A	61	0.069	89	0.400
B	208	0.143	287	0.376
Powder	$s_3\,[\mu\mathrm{m}]$	$\phi_3\,[-]$	–	–
A	112	0.131	–	–
B	395	0.081	–	–
Quadrature Nodes and Weights for a Four-Node Quadrature Formula				
Powder	$s_1\,[\mu\mathrm{m}]$	$\phi_1\,[-]$	$s_2\,[\mu\mathrm{m}]$	$\phi_2\,[-]$
A	52	0.018	77	0.214
B	181	0.043	252	0.322
Powder	$s_3\,[\mu\mathrm{m}]$	$\phi_3\,[-]$	$s_4\,[\mu\mathrm{m}]$	$\phi_4\,[-]$
A	98	0.318	119	0.050
B	328	0.194	413	0.041

distribution were respectively tracked, and four and five phases were respectively defined. As pointed out, balance equations for linear momentum and pseudointernal energy were solved for each phase.

As we computed the quadrature weights (that is, the volume fractions of the particle phases) from the moments by using the PD algorithm of Gordom,[42] we disabled the equations of mass conservation and passed the volume fractions to the main CFD solver and to the user-defined subroutines (for instance, those that implement the drag force closure and that determine the moment convective fluxes) through user-defined memories and define-property functions (for details, we refer to the code manual).

We could not treat the VDF moments directly as user-defined scalars, adding their transport equations to the default equations of the numerical code. This is because Fluent associates user-defined scalars either with a specific phase (the fluid or any granular phase) or with the mixture of all phases. In the first case, the equation solved by the code is

$$\frac{\partial}{\partial t}(\phi_k\,\mathcal{M}_k) + \nabla_x \cdot (\phi_k\,\mathcal{M}_k\,\boldsymbol{v}_k) = 0 \tag{0.18}$$

where, being constant, the solid density ρ_s does not appear. In the second case, when $\mathcal{M}_k$ is associated with the mixture, the equation solved by the code instead is

$$\frac{\partial}{\partial t}(\rho_m\,\mathcal{M}_k) + \nabla_x \cdot (\rho_m\,\mathcal{M}_k\,\boldsymbol{v}_m) = 0 \tag{0.19}$$

where ρ_m and $\boldsymbol{v}_m$ are defined so that

$$\rho_m \equiv \varepsilon\,\rho_e + \rho_s \sum_{r=1}^{v} \phi_r; \quad \rho_m\,\mathbf{v}_m \equiv \varepsilon\,\rho_e\,\mathbf{u}_e + \rho_s \sum_{r=1}^{v} \phi_r\,\mathbf{v}_r \tag{0.20}$$

Both equations differ from Eq. 0.8. To overcome this problem, we first modified the velocity field in Eq. 0.18, replacing the velocity $\boldsymbol{v}_k$ of the k th quadrature class with the velocity $\widehat{\boldsymbol{v}}_k$ of the k th moment of the distribution. To do this, we used a user-defined function (that is, a routine that the modeler writes and runs along with the CFD simulation) available in Fluent and called defined-uds-flux. In addition, we used as defined scalar the ratio $\mathcal{M}_k/\phi_k$, so that the volume fraction cancels out and the transport equation reduces to the correct one, which is to say, Eq. 0.8.

We implemented the PD algorithm (described in detail in Ref. 50), the closures for the velocities of the moments (that is, Eq. 0.10) and those for the fluid–particle interaction forces reported in Ref. 52 using additional user-defined functions called define-adjust and define-exchange-property. We did not have to implement the other constitutive equations mentioned in the previous sections, because they are available in Fluent as default. We used the pressure-based solver, which is recommended for low-speed incompressible flows. To convert scalar differential equations into algebraic equations which can be solved numerically, the code adopts a finite-volume discretization scheme. Part of the simulations were run using the first-order upwind spatial discretization scheme, where cell-face quantities are determined by assuming that the cell-center values of any field variable represent cell-averages that hold throughout the entire cells; thus, face quantities are identical to cell quantities, and are set equal to the cell-center values in the upstream cells (relative to the velocity direction). Some simulations were also run using second-order upwind schemes that, as we shall see, significantly affect their stability. The temporal discretization is first-order accurate and implicit. At every time step, we used a maximum of 150 iterations to calculate all the flow variables. Setting the tolerance to 10^{-5}, we saw the simulation converge within the iteration limit. We fixed the time step to 10^{-3} s, because shorter times steps gave equal results. Finally, we used under-relaxation factors of 0.20 for all the variables.

The Numerical Corruption of Higher-Order Advected Moments

As mentioned, the moments of a distribution represent some important physical properties of the underlying population of particles. For this reason, they have to satisfy some mathematical constraints. For instance, the positiveness of the density function over the phase space of the internal coordinate implies that the moment of order zero must be positive (note, however, that the positiveness of this moment

does not guarantee that the distribution is non-negative). Additionally, there are other simple, intuitive rules, such as that the moment of order zero of a VDF has to be smaller than one, for $\mathcal{M}_0$ represents the overall volume fraction of solid. Also, since $\mathcal{M}_1/\mathcal{M}_0$ is the volume-averaged value of the particle size, which is bounded between zero and infinity, the moment of order one (as well as all the other higher-order moments) must be positive. Another important property of the VDF is its standard deviation σ^2. As Randolph and Larson[36] report, in terms of moments it is

$$\sigma^2 = \frac{\mathcal{M}_2}{\mathcal{M}_0} - \left(\frac{\mathcal{M}_1}{\mathcal{M}_0}\right)^2 \tag{0.21}$$

The standard deviation of a monodisperse distribution is zero, while is positive for polydisperse distributions. Accordingly, it has to be $\mathcal{M}_2 \geq \mathcal{M}_1^2/\mathcal{M}_0$. If one (or more) of these conditions are not respected, the set of moments is invalid, because no VDF can generate them. A moment set corresponding to a physical VDF is instead said to be valid.

In general, to verify that a moment set is valid, we must ensure that the Hankel–Hadamard determinants[53] are all non-negative

$$\begin{vmatrix} \mathcal{M}_a & \mathcal{M}_{a+1} & \dots & \mathcal{M}_{a+b} \\ \mathcal{M}_{a+1} & \mathcal{M}_{a+2} & \dots & \mathcal{M}_{a+b+1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{M}_{a+b} & \mathcal{M}_{a+b+1} & \dots & \mathcal{M}_{a+b+b} \end{vmatrix} \geq 0 \tag{0.22}$$

for $a = 0$, 1 and $b \geq 0$. For the first four moments, the validity condition is fulfilled if the natural logarithm of $\mathcal{M}_k$ vs k is a convex function. The reader can verify this by building a difference table as explained by Petitti et al. [54]. Note that for $a = 0$ and $b = 1$ the Hankel-Hadamard determinant is $\mathcal{M}_0\,\mathcal{M}_2 - \mathcal{M}_1^2$, the validity condition thus reducing to $\mathcal{M}_2 \geq \mathcal{M}_1^2/\mathcal{M}_0$.

Because mixing is a linear process and because the PBE and the moment transform (0.7) are linear in the VDF, the moments of a powder obtained by blending together two or more powders whose initial sets of moments are valid should remain valid. Nevertheless, the moment transport equations are integrated with a finite-volume numerical code, which makes discretization errors. As Wright[47] clearly reports, most of the problems are caused by the approximation of the convective term, in particular with higher-order discretization schemes, which can turn a valid set of moments into an invalid one. This phenomenon is called moment corruption and poses a serious threat to our simulations, because when an invalid set of moments is fed to the PD algorithm, this yields negative nodes and leads to numerical instabilities.

The reason why in finite-volume numerical codes convection corrupts valid sets of moments is that these codes are designed to transport in physical space independent variables, without having to preserve relations among them. But moments are not independent scalars, since they have to satisfy the constraints posed by Eq. 0.22. So, even if the code transports each individual moment with sufficient accuracy, the (possibly small) advection errors are enough to substantially alter the relationships among the moments of the entire set, making the latter invalid. For more details we refer to Wright.[47]

Results and discussion

As described in Mazzei et al,[34] in the experiments we first of all fluidized the powder until it reached a pseudostationary state and then froze the bed by cutting off the gas supply. Finally, we divided the collapsed bed in layers and by sieving them we measured the PSDs. These were identical, for the powders had well mixed. From the PSDs reported in Figure 3, we calculated the moments using Eq. 0.17 and then applied the PD algorithm to compute the nodes and weights of the quadrature. We remind that, while the VDF accounts for the presence of the interstitial fluid, the PSD is a property of the powder and refers to void-free volume fractions. Consequently, to determine the quadrature weights from the experimental PSDs we had to assign a value to the void fraction; we chose the reference value of $\varepsilon = 0.400$, for this is roughly the one found experimentally in the collapsed bed. Table 2 reports the experimental VDF moments and the nodes and weights of the quadratures.

Table 2. Values of the VDF moments and of the quadrature nodes and weights obtained from the experimental PSDs reported in Figures 3 assuming a void fraction of 0.400

Moments of the Volume Density Function			
$\mathcal{M}_0$ [−]	$\mathcal{M}_1$ [μm]	$\mathcal{M}_2$ [μm^2]	$\mathcal{M}_3$ [μm^3]
0.600	1.16×10^2	2.82×10^4	7.84×10^6
$\mathcal{M}_4$ [μm^4]	$\mathcal{M}_5$ [μm^5]	$\mathcal{M}_6$ [μm^6]	$\mathcal{M}_7$ [μm^7]
2.34×10^9	7.32×10^{11}	2.37×10^{14}	7.95×10^{16}
Quadrature Nodes and Weights for a Two-Node Quadrature Formula			
s_1 [μm]	ϕ_1 [−]	s_2 [μm]	ϕ_2 [−]
105	0.332	304	0.268
Quadrature Nodes and Weights for a Three-Node Quadrature Formula			
s_1 [μm]	ϕ_1 [−]	s_2 [μm]	ϕ_2 [−]
88	0.258	244	0.259
s_3 [μm]	ϕ_3 [−]	–	–
360	0.083	–	–
Quadrature Nodes and Weights for a Four-Node Quadrature Formula			
s_1 [μm]	ϕ_1 [−]	s_2 [μm]	ϕ_2 [−]
84	0.226	174	0.106
s_3 [μm]	ϕ_3 [−]	s_4 [μm]	ϕ_4 [−]
275	0.222	379	0.046

If we wanted to simulate exactly the same procedure used in the experiments, we should first simulate the fluidization phase and then the collapse phase. This, however, is unnecessary and might even be detrimental. Numerically, we can easily calculate the VDFs and the PSDs when the bed is still fluidized; to this purpose, we just have to divide the bed in layers and from the numerical profiles of the VDF moments determine their average values in each layer and then the corresponding average values of the quadrature nodes and weights. There is another reason as well for which it is better to calculate the VDFs while the bed is still

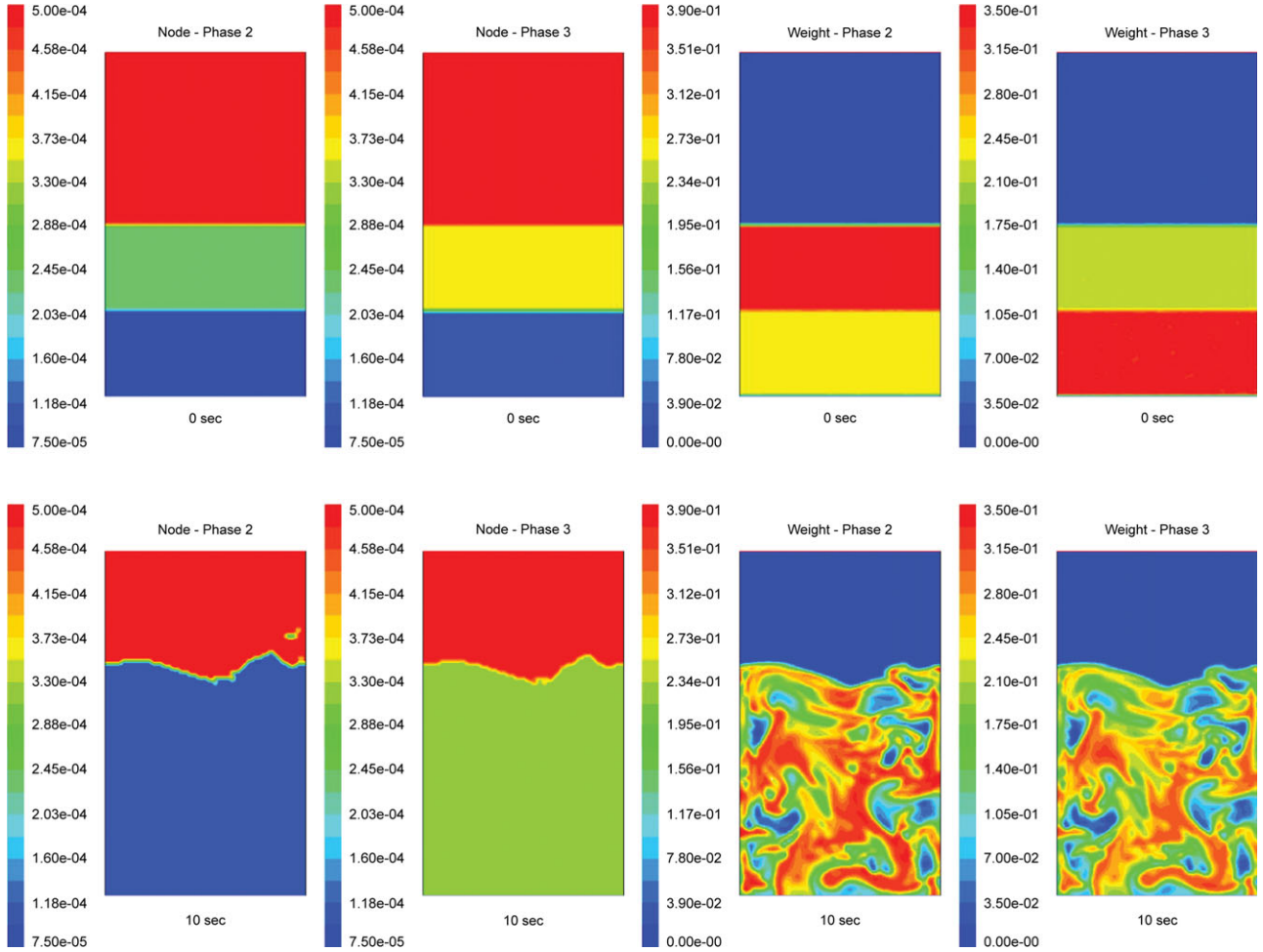

Figure 4. Numerical profiles, for a two-node quadrature formula, of nodes and weights within the fluidized bed at the start of the simulation and in pseudostationary state.

Phases 2 and 3 refer respectively to the first and second quadrature classes. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

fluidized. As reported by Mazzei,[52] freezing the bed is detrimental, because while the experimental collapse is instantaneous, and the bed conserves its PSDs, the simulated collapse is not and permits the bigger particles to segregate toward the bottom of the vessel, altering the original segregation profile. For these reasons, we simulated only the fluidization phase, calculating the VDFs in pseudostationary conditions.

As said, in the simulations we used two-node, three-node and four-node quadrature formulas. Many modeling and numerical parameters were tested and changed, in particular the effect on the final predictions of the closures for the fluid–particle and particle–particle interaction forces and of the strategies to evaluate the granular temperatures were investigated. We found that only the particle–particle interaction term seemed to influence significantly the final predictions. In fact, when this term is neglected, the expansion of the bed is much higher than the experimental value and a certain degree of segregation, with small particles floating and accumulating above the large particles, is present. The different closures examined for the fluid–particle force (namely, those reported in Wen and Yu,[55] Gidaspow[1] and Mazzei and Lettieri[56]) had little effect on the behavior of the simulations, both in terms of quality of the predictions and overall stability. The strategies to find the granular temperatures, that is, solving the differential Eq. 0.16 or their algebraic approximations (refer, for instance, to Ref. 57), did not significantly affect the predictions either, at least for the operating conditions investigated in this work.

Let us first consider the results obtained with the two-node and the three-node quadrature approximations by using first-order upwind discretization schemes. At the high fluid flux used the bed dynamics is very fast; the volume fraction profiles are not uniform, for the system operates in the bubbling regime, but vigorous mixing takes place continuously. We observed this both in the experiments and in the simulations. Figure 4 reports the profiles of the quadrature nodes and weights, for a two-quadrature approximation, at the start of the simulation and in pseudostationary conditions. Phases 2 and 3 represent the first and second quadrature classes respectively, the second corresponding to the greater particle size. As we see, while at the beginning of the simulation the upper and lower portions of the bed are clearly visible, from the different values of the nodes and weights, at the end of the simulation the bed is well mixed. This is particularly evident from the profiles of the quadrature nodes at the end of the simulation, which are completely flat. This shows that the model is capable of describing the very good mixing observed experimentally and the resulting uniformity of the PSD throughout the fluidized bed.

Table 3. Experimental and Numerical Values, for a Two-Node Quadrature Formula, of Nodes and Weights Back-Calculated from the VDF Moments Using the PD Algorithm and Assuming a Void Fraction of 0.400

	s_1 [μm]	ϕ_1 [$-$]	s_2 [μm]	ϕ_2 [$-$]
Top Layer				
EXP	105	0.332	304	0.268
CFD	104	0.330	318	0.270
Bottom Layer				
EXP	105	0.332	304	0.268
CFD	104	0.331	318	0.269

Table 4. Experimental and Numerical Values, for a Three-Node Quadrature Formula, of Nodes and Weights Back-Calculated from the VDF Moments Using the PD Algorithm and Assuming a Void Fraction of 0.400

	s_1 [μm]	ϕ_1 [$-$]	s_2 [μm]	ϕ_2 [$-$]	s_3 [μm]	ϕ_3 [$-$]
Top Layer						
EXP	88	0.258	244	0.259	360	0.083
CFD	90	0.107	259	0.386	378	0.106
Bottom Layer						
EXP	88	0.258	244	0.259	360	0.083
CFD	90	0.107	259	0.386	378	0.106

As mentioned, the profiles of the quadrature nodes are flat, that is to say, the nodes are equal everywhere within the bed, their values being $s_1 = 104$ μm and $s_2 = 318$ μm. These results agree well with the experimental values of $s_1 = 105$ μm and $s_2 = 304$ μm reported in Table 2, showing a very small error in the prediction of the evolution of the VDF. For the weights, we report the averaged values in the top and bottom layers, for these are the most representative. In the top layer, we found $\phi_1 = 0.330$ and $\phi_2 = 0.270$, while in the bottom layer $\phi_1 = 0.331$ and $\phi_2 = 0.269$. Also these results agree well with the experimental values of $\phi_1 = 0.332$ and $\phi_2 = 0.268$ reported in Table 2. This is summarized in Table 3, in which we compare the numerical and experimental findings. Figure 5 helps to visualize the results, reporting the two-node representations of the experimental and computational PSDs in the top and bottom layers of the bed and showing in the background the experimental PSD that we obtained by sieving. An additional element that supports the correctness of the approach comes from the comparison of the averaged values of the experimentally measured and computed moments of the VDF. Comparison of these moments resulted in an error for $\mathcal{M}_0$ smaller than 1% and from 5 to 10% for $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$. We shall come back to the possible sources of these errors when discussing the results of the simulations based on the three-node quadrature.

Also, the simulations based on the three-node quadrature formula ran smoothly. Starting from the initial packed condition, the bed expanded, under the action of the fluid, eventually reaching a global height in good agreement with what expected. In this instance as well the profiles of the quadrature nodes are flat, the bed being uniformly mixed and the PSDs the same in every point of the computational domain. Table 4 shows the pseudostationary values of quadrature nodes and weights in the top and bottom layers of the bed, comparing experimental and numerical findings. As we can see, little difference is found between the two layers, which indicates that the simulated fluidized bed well mixes. The nodes are predicted well, the maximum deviation between their experimental and numerical values being about 6%. However, the finite-mode representation of the VDF appears not to be exact, since the quadrature weights are affected by an error larger than for the two-node quadrature. Experimental evidence tells us that the lowest nodes, $s_1 = 88$ μm and $s_2 = 244$ μm, share the same weight, each one accounting for about 42% of the powder mass; the greatest, in contrast, has less importance, accounting for about 14% of the powder mass. The simulation, conversely, ascribes roughly the same weight to the lowest and greatest nodes, $s_1 = 90$ and $s_3 = 378$ μm, each one accounting for about 17% of the powder mass, and letting the intermediate node dominate, this accounting for about 64% of the powder mass. Figure 6 reports the three-node representations of the experimental and computational PSDs in the top and bottom layers of the bed, with in the background the experimental PSD measured by sieving. This larger error in the value of the weights is also confirmed by the errors committed on the tracked moments. Also in this case, we evaluate the error in terms of the difference of the experimentally measured and computed averaged moments of the VDF. The moment of order zero is again affected by an error smaller than 1%, while the error found for the higher-order moments with order from one to five ranges from 15 to 30%.

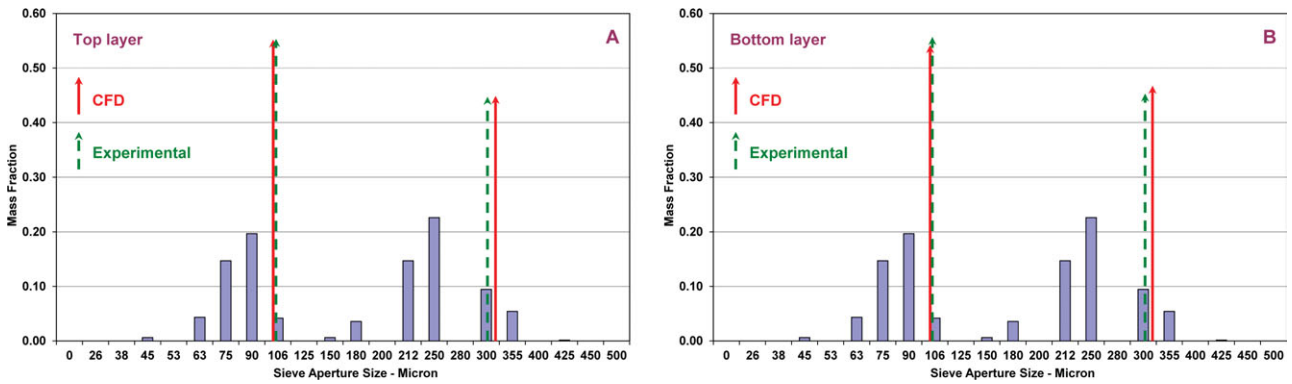

Figure 5. Two-node representations of the experimental and computational PSDs of powder C in the top and bottom layers of the bed.

The arrows symbolize the Dirac delta functions of the quadrature formulas, their positions and heights indicating respectively the quadrature nodes and weights on a void-free basis. For reference, the figure also reports the experimental PSD obtained by sieving. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

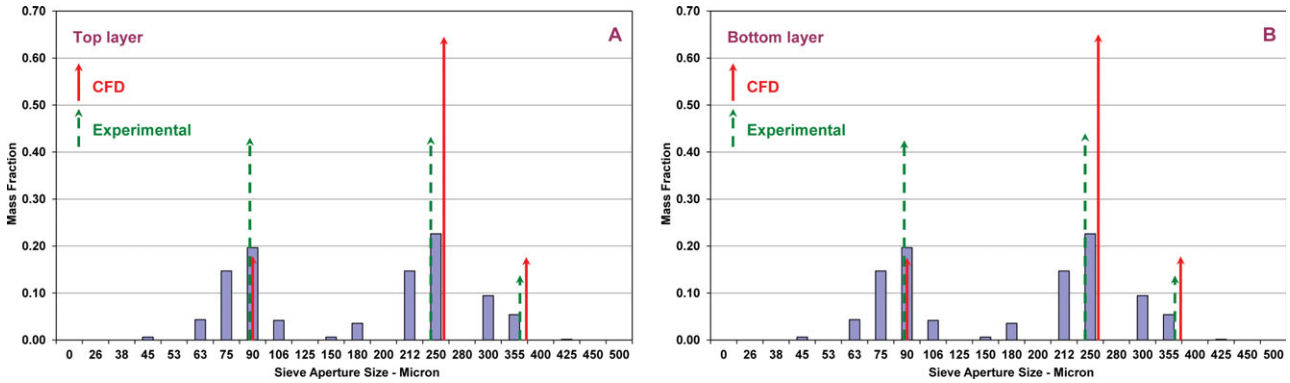

Figure 6. Three-node representations of the experimental and computational PSDs of powder C in the top and bottom layers of the bed.

The arrows symbolize the Dirac delta functions of the quadrature formulas, their positions and heights indicating respectively the quadrature nodes and weights on a void-free basis. For reference, the figure also reports the experimental PSD obtained by sieving. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Many are the possible sources of error. The powder being fluidized has a broad PSD, with diameters going from 45 to 425 μm, as shown in Figure 3. The superficial fluid velocity is sufficiently high to fully mix the fluid bed, being in particular much higher than the minimum fluidization velocity of the smallest particles. These, in consequence, tend to be dragged by the fluid in the freeboard of the bed and then fall back again in the dense bed. To reduce the computational time, in the simulations we considered a vessel whose height is twice the initial height of the resting powder. This height might not be sufficient to allow the small particles to fall back in the dense bed; if this happens, these escape the computational domain and are irreversibly lost. However, the contribution of this effect on the error is likely to be small, since this would be reflected on $\mathcal{M}_0$. This quantity, which represents the total particle volume density (and therefore its integral throughout the bed represents the total volume of solid) is predicted with a very small error, which is furthermore very similar to the error we found when using the two-node quadrature.

Other possible sources of error are likely related to the spatial and temporal discretizations; these could be drastically reduced by using higher-order discretization schemes (which, however, corrupt the moments, as we shall see below) or by using finer grids and smaller time steps. A few tests performed with a time step 10 times smaller and a computational grid four times finer did not alter the observed behavior and confirmed the grid-independency of our results. In fact, with this much finer temporal and spatial discretization, some very limited improvement was detected at the expense of an unacceptably large computational time.

Although one might expect the opposite behavior, the overall error being larger with a three-node, instead of a two-node, quadrature is probably due to the fact that, since for this application higher-order moments are more sensitive to the numerical errors than lower-order moments, the resulting three-quadrature, which uses higher-order moments, badly approximates the real VDF, when compared to the quality of the approximation given by the two-node quadrature. This irregular behavior of the error that one observes when using QMOM with quadrature approximations of increasing order has been also highlighted by Grosch et al.[58]

The simulations based on the four-node quadrature did not run as smoothly as the previous ones. This is because after a few (275) time steps the PD algorithm yielded some negative (and thus unphysical) values of the nodes, this resulting into numerical instabilities that crashed the simulation. As the voidage contour plot in Figure 7 illustrates, this happens when the bed has just begun to expand.

To confirm the nature of these instabilities, we implemented in Fluent, using define-adjust user-defined functions, an algorithm capable of checking the validity of moment sets. The algorithm first checks that the function generated by the natural logarithm of the moment set $\mathcal{M}_k$ vs. k is a convex function; then it also checks the positiveness of the Hankel–Hadamard determinants, but only for moments of order k greater than or equal to four. If both checks are successful, the set is valid and can be fed to the PD algorithm; otherwise, it is invalid and this is flagged by check variables.

Figure 7 illustrates this procedure for detecting moment corruption. The flag variable ''overall-check'' is equal to three in the cells where no solid is present, to one if the moment set is valid and to zero if it is not valid. The variable ''check-45'' is equal to zero only if the moment set is invalid because the determinant of the Hankel–Hadamard matrix of order three is negative, this meaning that the moments of order four and five are inconsistent with the others. Finally, the variable ''check-67'' examines the correctness of the moments of order six and seven through the positivity of the determinant of the Hankel–Hadamard matrix of order four. As we can observe in Figure 7, the moments that do not pass the validity checks are located where sharp node gradients develop; in particular, the moments begin to corrupt at the boundary separating the two powders. Another location where this problem often arises is the upper surface of the bed, where sharp node gradients are also present; this can be observed in the last snapshot, taken after 0.275 s.

By implementing this detection algorithm, running several simulations and changing various parameters (including the size of the grid cells, the size of the time step and the discretization scheme), we were able to confirm the key role played by the discretization scheme used. Our results show that when using the two–and three-node quadrature approximations (or in other words when transporting the first four

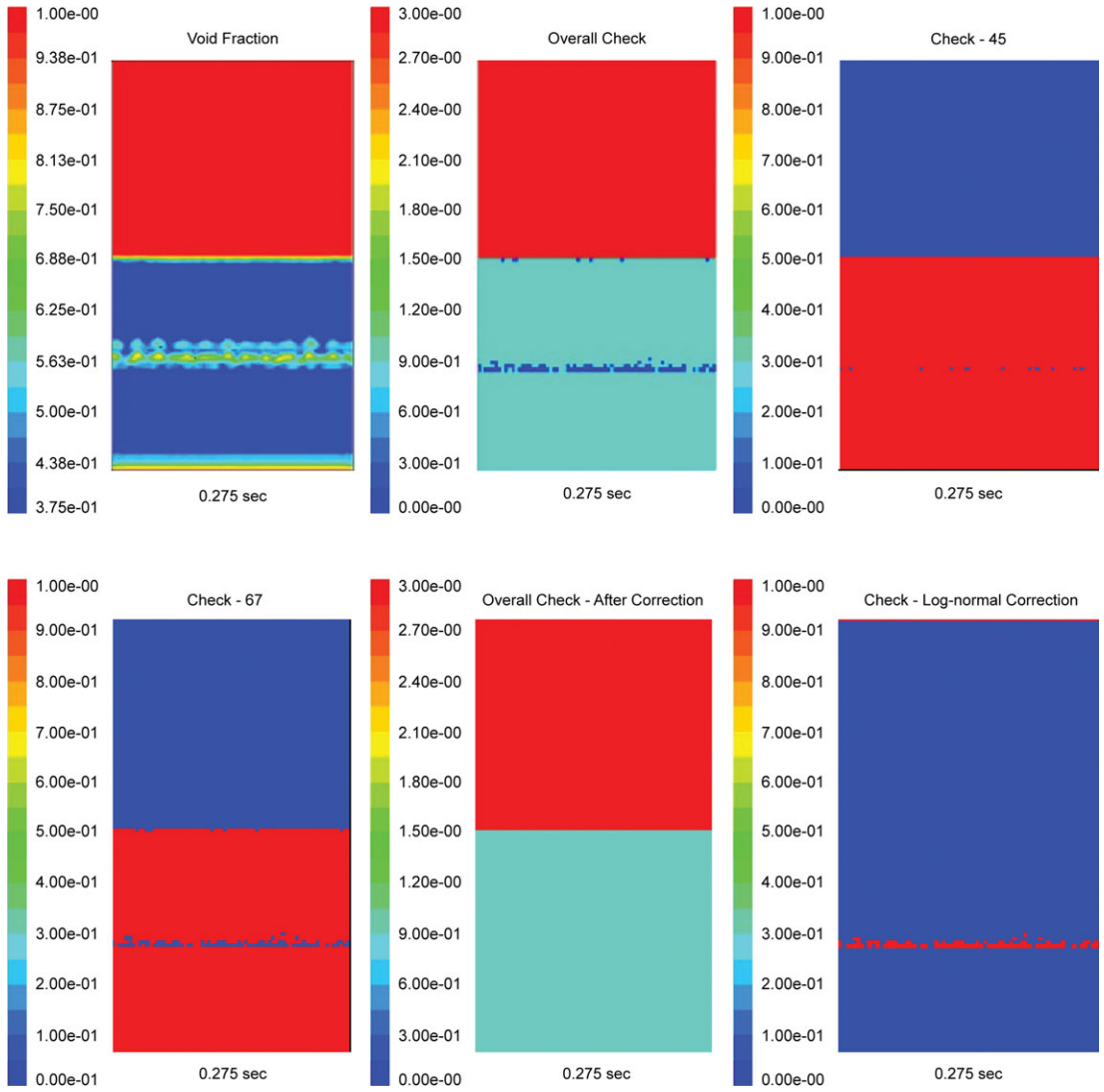

Figure 7. Numerical profiles of the void fraction and of the moment corruption check variables for the simulation based on the four-node quadrature formula.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

and six moments), and when using the first-order upwind discretization scheme, QMOM is always stable. Numerical diffusion takes place, accelerating the mixing of the powders, but does not significantly alter the solution. The simple interpolation used in first-order upwind schemes preserves the validity of the moment set when only four or six moments are tracked. But if a number of moments greater than six is tracked, then higher-order moments corrupt, even with the first-order upwind discretization scheme. This issue does remain important even when working in double precision (as we did in the present work) and when improving the convergence criteria and/or increasing the maximum number of iterations per time step.

Our results also indicate that the introduction of higher-order discretization schemes (such as the simple second-order upwind) corrupt all the moments leading to instabilities and the crash of the simulations. Let us therefore discuss the numerical strategies that modelers can adopt to overcome the corruption problem. Here, we report two that are simple to implement in commercial codes and could be effective. The first is using the direct quadrature method of moments (DQMOM), instead of QMOM, whereas the second is replacing invalid sets of moments with valid ones in the computational cells where the problem arises.

QMOM and DQMOM both approximate density functions with quadrature formulas; the methods differ in how they calculate the nodes and weights of this formula. As pointed out, forcing these to agree with a set of independent lower-order moments, QMOM tracks the moments by integrating their transport equations and back-calculates nodes and weights; conversely, DQMOM tracks directly the latter, integrating transport equations that govern their evolution. In both models, the number of transport equations that one needs to solve is the same: if a quadrature formula with v classes is adopted, QMOM requires $2v$ equations for the moments, whereas DQMOM requires v equations for the quadrature nodes and v equations for the quadrature weights. However, to back-calculate the latter from the moments, QMOM also needs to run the PD algorithm in each computational cell at each time step. DQMOM needs this additional computation only once, at

the beginning of the simulation when one has to initialize the quadrature nodes and weights (at $t = 0$ one knows the PSD of the system and therefore knows the values of the moments and not of the quadrature nodes and weights). However, at each time step the source terms of DQMOM (zero in our simple test case) must be calculated by solving a linear system, which when is singular (i.e., two identical nodes) cannot be inverted. We also point out that for multivariate systems the PD algorithm can no longer be used and back-calculating the quadrature nodes requires the use of other algorithms. Another important aspect is that DQMOM does not corrupt the moments of the distribution. This is because the quadrature nodes and weights can be advected as independent scalars and always result in valid moment sets. For details, we refer to Marchisio and Fox[40] and Wright.[47]

DQMOM, however, also presents two major disadvantages, as Mazzei et al,[34] have recently reported. In the presence of numerical diffusion, the transport equations governing nodes and weights feature not only diffusive terms but also source terms that relate to the node spatial gradients and the coefficient of numerical diffusion. As this coefficient is unknown and (in nonuniform computational grids) varies from cell to cell, estimating correctly the source terms is quite difficult. This undermines the method, since if the source terms (which the modeler has to implement) do not match sufficiently well the diffusive fluxes (which numerical diffusion generates), the nodes and weights are wrongly predicted; in other words, the moment set is valid, but the values of the nodes and weights – even if physically possible – are incorrect. Moreover, DQMOM (for its derivation) requires the continuity (in time and space) of the weight and node functions, making the description of problems with discontinuities (such as the mixing problem investigated here) very difficult to solve. For details, we refer to Mazzei et al.[34] This explains why in this work we opted for an alternative numerical strategy, which we now briefly discuss.

A strategy that may overcome the problem of moment corruption in QMOM is replacing invalid moment sets, where and as soon as they appear, with similar valid ones. The objective is eliminating the invalid sets in the cells where they are detected, solving the problem locally before it can spread to the rest of the domain and corrupt all the simulation. An algorithm of this kind was developed by McGraw.[59] It is an iterative minimization method that identifies and corrects the moment of index k that after adjustment maximizes the smoothness of the function $\ln \mathcal{M}_k$.

This is done by calculating a difference table of the function $\ln \mathcal{M}_k$ vs. k up to the third order. A function is convex if the second-order difference is positive. As already mentioned, the convexity of $\ln \mathcal{M}_k$ vs. k is a necessary condition for a moment set to be valid. If the difference table is constructed and the second-order difference contains negative elements, then the set is invalid. One way to cure the invalidity would be to change one of the moments, to transform the second-order difference vector into a positive sequence of numbers. However, this method would not be able to quantify how positive these numbers should be, since any positive sequence of numbers would correspond to a valid moment set. Therefore, a more stringent condition is used: that of the smallest third-order difference vector. In fact, the distribution resulting in a null third-order difference vector is the log-normal distribution, which results in a parabola for the function $\ln \mathcal{M}_k$ vs. k. By changing one of the moments (the one that has to be changed the least) the algorithm tries to minimize the third-order difference vector, transforming the distribution into one that is as close to a log-normal as possible. The algorithm normally converges in few iterations, defining the index of the moment to be corrected and restoring the moment set. One can use this method when tracking sets of six or more moments. For four moments, and for the other cases where this method fails, another approach must be used. The simplest and most effective consists in replacing the moments with those of a log-normal distribution which shares with the original one only some correct moments. Because the corruption usually affects higher-order moments, one can normally find the parameters of the log-normal distribution from the lower-order moments $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_2$, whose physical meaning, as stressed, is particularly important; then, using the log-normal distribution, one can calculate the other moments. For details, see McGraw[59] and Petitti et al.[54] As recently suggested by this last research group, when using higher-order discretization schemes or when tracking the evolution of higher-order moments (or in other words when using quadrature approximations with $v \geq 3$), one should always run the corrective algorithm to detect moment corruption and immediately restore the corrupted moments.

To overcome the problem of moment corruption, we implemented within Fluent the correction algorithm of McGraw,[59] adapted to the specific QMOM version used in this work, by resorting to a define-adjust user-defined function. The correction operates as follows. When the moment set is found to be invalid then the correction procedure of McGraw is applied. As mentioned, this is based on the simple idea of correcting the one moment which needs to be changed the least to restore the convexity of the function $\ln \mathcal{M}_k$ vs. k. If this first correction is unsuccessful then a second one is used. This replaces the corrupted moment set with that of a log-normal distribution that shares with the original set $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_2$. Figure 7 reports the outcome of the correction. As we can see, the variable "Overall Check-After Correction" is equal to one in all the computational cells, this meaning that the validity of the moment set has been restored everywhere. In most cells, this was done by the first correction algorithm, but in some the second one acted (where the flag variable "Check-Log-normal Correction" is equal to one).

Although this algorithm has been successfully used in many cases, it did not solve the stability problem in ours. We should stress, however, that the correction algorithm has been applied so far only to multiphase systems simulated with QMOM with the assumption that the dispersed phase moves with one single velocity. This is therefore the first time that the algorithm is tested with multiple solid velocities. We tested it in all the cases were corruption was detected, that is, when we used the second-order upwind scheme and/or tracked eight moments. Unfortunately, the algorithm failed to overcome the problem, for it only delays divergence. This is illustrated in Figure 8 through one of the quadrature nodes, which is far easier to interpret than a higher-order moment. After some time, we observe that in some cells the node becomes negative; these cells are colored in white and are located where sharp node gradients develop. Without correction the simulation would stop now, whereas, thank to the corrective algorithm, it proceeds a bit further. But the algorithm fails to prevent the rapid

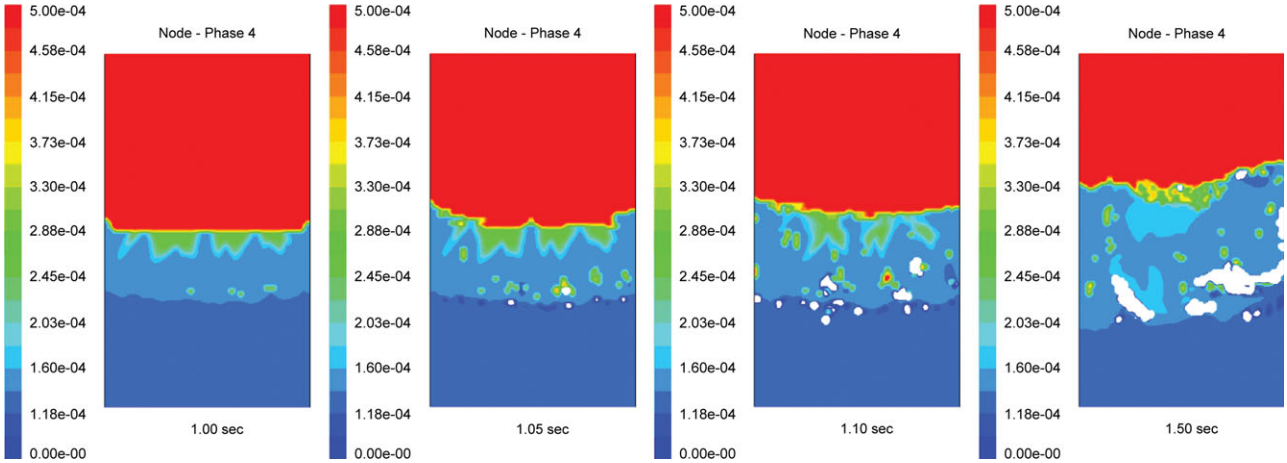

Figure 8. Numerical profiles, for a four-node quadrature formula, of the third node within the fluidized bed.

Owing to moment corruption, in some computational cells (appearing in white) the node becomes negative. Then corruption spreads to neighboring cells. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

spreading of corruption to larger regions of the bed, and eventually, half second later, the simulation crashes. As already mentioned, we found the same outcome in other tests in which we used finer computational grids, one with cells of 2.5-mm edge and another with cells of 1.0-mm edge, and smaller time steps of about 10^{-4} s (in some cases we had to use smaller computational domains and/or shorter real simulations times). The first simulation ran for about four real-time seconds and then crashed, whereas the second ran only for about three real-time seconds. In this instance, the corrective algorithm prevented the nodes from becoming negative, but not from taking on unphysical values; in particular, in some cells we found values lower than ten microns, which are impossible because the initial PSDs did not include particles with such a small size. We therefore concluded that the solution is indeed grid-independent and that smaller cells do not assist the numerics. Also note that this strategy increases the computational times so much to be impractical. Simulating 10 real-time seconds with cells of 5.0-mm edge takes about 10 days; reducing the edge to 1.0 mm increases the simulation running time 25-folds, bringing it in the region of eight months (of course, parallel processing alleviates the issue). This is why we had to use a smaller domain for the finest grid.

As already mentioned, one possible reason for the failure of the correction algorithm is that it is coupled here – for the first time – with a new version of QMOM. The correction algorithm has worked well in QMOM implementations where the dispersed phase is assumed to move with the fluid in the low Stokes number limit[60] or where all the particles of the disperse phases share the same velocity,[54] which is different from that of the fluid. Instead, in our implementation each moment is convected with its own velocity, the method using a quadrature approximation with v nodes and coupling it with a multifluid model with v disperse phases. This QMOM implementation transports only pure moments (with respect to particle size and velocity), decoupling the evolution of the two internal coordinates. Even if this strategy was successful in some cases,[61] in some others it was shown to be inadequate.[62] We shall consider other solutions, such as different formulation of spatial discretization schemes and/or transport of mixed moments, in our future work.

Conclusions

In this work, we presented a new formulation of the QMOM. There are two novelties: (1) the model is based on a volume, rather than on a number, density function, so that it deals directly with volume fractions instead of number densities; (2) the moments, and in turn the quadrature classes, no longer share the same velocity, so that particles are free to mix and segregate. To test the model, we considered the simplest case possible: inert powders initially segregated that mix in physical space. Hence, particles do not nucleate, agglomerate, break, react, and wear, being allowed only to freely move in the bed. We selected this system because its simplicity allows to test the method, understand it better and highlight possible issues or limitations, a necessary analysis before one can tackle tougher problems.

We know that theoretically the higher the order of the quadrature formula, the better its accuracy. So, to model the mixing dynamics, we considered two-, three-, and four-node quadratures, assessing their numerical performances and comparing their predictions with experimental data. When we used a first-order spatial discretization scheme, the two-node quadrature gave very good results, predicting nodes and weights accurately (1% being the maximum deviation); the three-node quadrature predicted the nodes well (with 6% maximum deviation) but the weights poorly (with 60% being the maximum deviation); finally, the four-node quadrature corrupted the higher-order moments, crashing the simulation. Higher-order spatial discretization schemes exacerbated the problem. This numerical issue, described by McGraw[59] and Wright,[47] arises from the convective terms in the moment transport equations: each moment is correctly advected, but the relations among them, which make their set valid, are not preserved. This leads to unphysical node values that eventually crash the simulation.

There are various strategies to overcome this problem. In this work, we used a corrective algorithm developed by McGraw[59] that replaces invalid moment sets, in the cells

where they appear, with similar valid ones, attempting to solve the problem locally before it can spread to the rest of the domain and corrupt the entire simulation. Unfortunately, this procedure was not successful, which indicates that the problem of moment corruption needs to be investigated further. One possible strategy to be investigated and implemented in the future is based on the idea of calculating the moment flux by decomposing it into nodes and weights, and then use higher-order schemes only for the weights. This will be the subject of our future work.

Literature Cited

1. Gidaspow D. *Multiphase Flow and Fluidization*. London: Academic Press, 1994.
2. Enwald H, Peirano E, Almstedt AE. Eulerian two-phase flow theory applied to fluidization. *Int J Multiphase Flow*. 1996;22:21–66.
3. Drew DA, Passman SL. *Theory of Multicomponent Fluids*. Applied Mathematical Sciences. New York: Springer, 1998.
4. Jackson R. *The Dynamics of Fluidized Particles*. Cambridge Monographs on Mechanics. Cambridge: Cambridge University Press, 2000.
5. Lettieri P, Mazzei L. Challenges and issues on the CFD modeling of fluidized beds: a review. *J Comput Multiphase Flows*. 2009;1:83–131.
6. Sinclair JL, Jackson R. Gas-particle flow in a vertical pipe with particle-particle interactions. *AIChE J*. 1989;35:1473–1486.
7. Ding J, Gidaspow D. A bubbling fluidization model using kinetic theory of granular flow. *AIChE J*. 1990;36:523–538.
8. Kuipers J, van Duin K, van Beckum F, van Swaaij W. Computer simulation of the hydrodynamics of a two-dimensional gas-fluidized bed. *Comput Chem Eng*. 1993;17:839–858.
9. Hrenya CM, Sinclair JL. Effects of particle-phase turbulence in gas-solid flows. *AIChE J*. 1997;43:853–869.
10. Pain CC, Mansoorzadeh S, de Oliveira CRE. A study of bubbling and slugging fluidised beds using the two-fluid granular temperature model. *Int J Multiphase Flow*. 2001;27:527–551
11. Lettieri P, Cammarata L, Michale G, Yates JG. CFD simulations of gas-fluidized beds using alternative Eulerian-Eulerian modelling approaches. *IJCRE*. 2003;1:1–21.
12. Lettieri P, Saccone G, Cammarata L. Predicting the transition from bubbling to slugging fluidization using computational fluid dynamics. *Chem Eng Res Des*. 2004;82:939–944.
13. Mazzei L, Lettieri P. CFD simulations of expanding/contracting homogeneous fluidized beds and their transition to bubbling. *Chem Eng Sci*. 2008;63:5831–5847.
14. Nienow AW, Chiba T. *Fluidization of dissimilar materials*. In: Davidson JF, Clift R, Harrison D, editors. London: Academic Press. 1985.
15. Dolgunin VN, Ukolov AA. Segregation model of particle rapid gravity flow. *Powder Technol*. 1995;83:95–103.
16. Khang DY, Lee HH. Particle size distribution in fluidization beds for catalytic polymerization. *Chem Eng Sci*. 1997;52:421–431.
17. Olivieri G, Marzocchella A, Salatino P. A fluid-bed continuous classifier of polydisperse granular solids. *J Taiwan Inst Chem Eng*. 2009;40:638–644.
18. Mathiesen V, Solberg T, Hjertager BH. An experimental and computational study of multiphase flow behavior in a circulating fluidized bed. *Int J Multiphase Flow*. 2000;26:387–419.
19. Mathiesen V, Solberg T, Hjertager BH. Predictions of gas/particle flow with an Eulerian model including a realistic particle size distribution. *Powder Technol*. 2000;112:34–45.
20. van Wachem BGM, Schouten JC, van den Bleek CM, Krishna R, Sinclair JL. CFD modeling of gasfluidized beds with a bimodal particle mixture. *AIChE J*. 2001;47:1292–1302.
21. Wirsum M, Fett F, Iwanowa N, Lukjanow G. Particle mixing in bubbling fluidized beds of binary particle systems. *Powder Technol*. 2001;120:63–69.
22. Huilin L, Yurong H, Gidaspow D, Lidan Y, Yukun Q. Size segregation of binary mixture of solids in bubbling fluidized beds. *Powder Technol*. 2003;134:86–97.
23. Gera D, Syamlal M, O'Brien TJ. Hydrodynamics of particle segregation in fluidized beds. *Int J Multiphase Flow*. 2004;30:419–428.
24. Cooper S, Coronella CJ. CFD simulations of particle mixing in a binary fluidized bed. *Powder Technol*. 2005;151:27–36.
25. Sun Q, Lu H, Liu W, He Y, Yang L, Gidaspow D. Simulation and experiment of segregating/mixing of rice husk-sand mixture in a bubbling fluidized bed. *Fuel*. 2005;84:1739–1748.
26. Huilin L, Yunhua Z, Ding J, Gidaspow D, Wei L. Investigation of mixing/segregation of mixture particles in gas-solid fluidized beds. *Chem Eng Sci*. 2007;62:301–317.
27. Owoyemi O, Mazzei L, Lettieri P. CFD modeling of binary-fluidized suspensions and investigation of role of particle-particle drag on mixing and segregation. *AIChE J*. 2007;53:1924–1940.
28. Olmos E, Gentric C, Vial C, Wild G, Midoux N. Numerical simulation of multiphase flow in bubble column reactors. Influence of bubble coalescence and break-up. *Chem Eng Sci*. 2001;56:6359–6365.
29. Lehr F, Mewes D. A transport equation for the interfacial area density applied to bubble columns. *Chem Eng Sci*. 2001;56:1159–1166.
30. Buwa VV, Ranade VV. Dynamics of gas-liquid flow in a rectangular bubble column: experiments and single/multi-group CFD simulation. *Chem Eng Sci*. 2002;57:4715–4736.
31. Venneker BCH, Derksen JJ, van den Akker HEA. Population balance modeling of aerated stirred vessels based on CFD. *AIChE J*. 2002;48:673–685.
32. Fan R, Marchisio DL, Fox RO. Application of the direct quadrature method of moments to polydisperse gas-solid fluidized beds. *Powder Technol*. 2004;139:7–20.
33. Fan R, Fox RO. Segregation in polydisperse fluidized beds: validation of a multi-fluid model. *Chem Eng Sci*. 2008;63:272–285.
34. Mazzei L, Marchisio DL, Lettieri P. Direct quadrature method of moments for the mixing of inert polydisperse fluidized powders and the role of numerical diffusion. *Ind Eng Chem Res*. 2010;49:5141–5152.
35. Ramkrishna D. *Population Balances*. London: Academic Press, 2000.
36. Randolph AD, Larson MA. *Theory of Particulate Processes*. London: Academic Press, 1971.
37. Hulburt HM, Katz S. Some problems in particle technology. *Chem Eng Sci*. 1964;19:555–574.
38. Marchisio DL, Fox RO. *Multiphase reacting flows: modelling and simulation*. CISM courses and lectures No. 492. New York: Springer, 2007.
39. Fox RO. *Computational Models for Turbulent Reacting Flows*. Cambridge: Cambridge University Press, 2003.
40. Marchisio DL, Fox RO. Solution of population balance equations using the direct quadrature method of moments. *J Aerosol Sci*. 2005;36:43–73.
41. McGraw R. Description of aerosol dynamics by the quadrature method of moments. *Aerosol Sci Technol*. 1997;27:255–265.
42. Gordon RG. Error bounds in equilibrium statistical mechanics. *J Math Phys*. 1968;9:655–663.
43. Wheeler JC. Modified moments and Gaussian quadratures. *Rocky Mt Math*. 1974;4:287–296.
44. Wright DL, McGrow R, Rosner DE. Bivariate extension of the quadrature method of moments for modeling simultaneous coagulation and sintering of particle populations. *J Colloid Interphase Sci*. 2001;236:242–251.
45. Fox RO. Higher-order quadrature-based moment methods for kinetic equations. *J Comput Phys*. 2009;228:7771–7791.
46. Yuan C, Fox RO. Conditional quadrature method of moments for kinetic equations. *J Comput Phys*. 2011;230:8216–8246.
47. Wright DL. Numerical advection of moments of the particle size distribution in Eulerian models. *J Aerosol Sci*. 2007;38:352–369.
48. Geldart D. Gas Fluidization Technology. New York: Wiley, 1973.
49. Curtiss CF, Bird RB. Multicomponent diffusion in polymeric liquids. *Proc Natl Acad Sci. USA* 1996;93:7440–7445.
50. Mazzei L.Eulerian modelling and computational fluid dynamics simulation of mono and polydisperse fluidized suspensions. Ph.D. Dissertation. Department of Chemical Engineering, University College London, 2008.
51. Lu H, Gidaspow D. Hydrodynamics of binary fluidization in a riser: CFD simulation using two granular temperatures. *Chem Eng Sci*. 2003;58:3777–3792.
52. Mazzei L, Casillo A, Lettieri P, Salatino P. CFD simulations of segregating fluidized bidisperse mixtures of particles differing in size. *Chem Eng J*. 2010;156:432–445.
53. Shohat JA, Tamarkin JD. *The Problem of Moments*. Providence: American Mathematical Society, 1943.
54. Petitti M, Nasuti A, Marchisio DL, Vanni M, Baldi G, Mancini N, Podenzani F. Bubble size distribution modeling in stirred gas-liquid reactors with QMOM augmented by a new correction algorithm. *AIChE J*. 2010;56:36–53.

55. Wen CY, Yu YH. Mechanics of fluidization. *Chem Eng Prog Symp Ser*. 1966;62:100–111.
56. Mazzei L, Lettieri P. A drag force closure for uniformly-dispersed fluidized suspensions. *Chem Eng Sci*. 2007;62:6129–6142.
57. Syamlal M, Rogers WA, O'Brien TJ. MFIX Documentation and Theory Guide. DOE/METC94/1004, NTIS/DE94000087. Electronically available from: http://www.mfix.org. 1993.
58. Grosch R, Briesen H, Marquardt W, Wulkow M. Generalization and numerical investigation of QMOM. *AIChE. J*. 2007;53:207–227.
59. McGraw R. Correcting moment sequences for errors associated with advective transport. http://www. ecd.bnl.gov/pubs/momentcorrection_mcgraw2006.pdf, 2006.
60. Di Pasquale N, Marchisio DL, Barresi AA. Validation of a comprehensive model for precipitation in solvent-displacement processes. *AIChE J*. Submitted.
61. Buffo A, Vanni M, Marchisio DL. Multidimensional population balance model for the simulation of turbulent gas-liquid systems in stirred tank reactors. *Chem Eng Sci*. doi:10.1016/j.ces.2011.04.042.
62. Vikas V, Yuan C, Wang ZJ, Fox RO. Modeling of bubble-column flows with quadrature-based moment methods. *Chem Eng Sci*. 2011;66:3058–3070.

Manuscript received May 20, 2011, and revision received Oct. 5, 2011.