PERGAMON

International Journal of Multiphase Flow 28 (2002) 1565–1580

International Journal of Multiphase Flow

www.elsevier.com/locate/ijmulflow

An efficient particle-locating algorithm for application in arbitrary 2D and 3D grids

R. Chordá, J.A. Blasco *, N. Fueyo

Fluid Mechanics Group-CPS, Calle María de Luna 3, 50018 Zaragoza, Spain

Received 14 October 2001; received in revised form 3 June 2002

Abstract

Several alternative techniques have been proposed in the literature to efficiently locate particles within unstructured grids. The present paper reviews two recently published, particle-locating algorithms, and introduces a new approach which improves the performance of the previous methods. The proposed algorithm is valid for arbitrary two-dimensional (2D) or three-dimensional (3D) grids, it is simple to implement, and results in fairly small CPU-time requirements. Furthermore, the directed-search feature of the present method offers shorter search paths and allows the detection of walls during the tracking of the particle trajectory. The performance of the proposed particle-locating strategy is compared with existing ones, and is evaluated on two tests, namely a 2D/3D random particle-locating test and a 2D Lagrangian simulation of a premixed turbulent flame.

Keywords: Particles; Locating; Algorithm; Lagrangian; Unstructured grid; Joint composition PDF; Monte Carlo; Turbulent flame

1. Introduction

The numerical simulation of fluid flow by means of Lagrangian techniques typically involves the tracking of the convected particles within the spatial domain of the problem. In the case of structured cartesian grids, the particle-locating problem can be readily solved. However, this does not hold for the general case of unstructured grids, which are nowadays widely used for complex geometries, such as those encountered in industrial applications.

Several algorithms have been recently proposed aimed at solving this problem (Chen, 1997; Guzman et al., 1995; Oliveira et al., 1997; Frank and Schulze, 1994; Valentine and Decker, 1995).

* Corresponding author.

0301-9322/02/$ - see front matter
PII: S0301-9322(02)00045-9

Nevertheless, the majority of the particle-locating algorithms are quite elaborated, and thus they turn out to be quite complex to implement and to yield a poor CPU-time performance. Furthermore, some particle-locating approaches are only valid for certain applications, such as two-dimensional (2D) grids or limited Eulerian cell displacements within a given Lagrangian time step.

Among the most prominent particle-locating methods are the ones proposed by Zhou and Leschziner (1999) (ZL) and Chen and Pereira (1999) (CP).

The most interesting feature of the ZL algorithm is that it can be implemented very straightforwardly, and requires very small CPU times (Chen and Pereira, 1999). However, due to its simplicity, the ZL location algorithm traverses through a larger number of cells than strictly required, and can even get trapped in an infinite circular search around the particle cell, as will be shown in the present work.

In contrast, the CP algorithm stands as a more robust, though more time-consuming, particle-locating technique. The direct-search feature of the CP algorithm offers shorter search paths and it can also be quite beneficial for the detection of internal walls and for an efficient implementation of particle reflections at walls (''particle bouncing'').

The three-dimensional (3D) extensions of the available particle-locating algorithms present important deficiencies. As a result, there is a need for improved, 3D particle-locating algorithms.

The present work introduces a new particle-locating technique which adopts the best properties of the ZL and CP algorithms. It is a directed search, as the CP technique, but its implementation is as simple as the ZL one, which results in lower CPU-time requirements. Furthermore, the 3D extension of the proposed algorithm is rather simple and allows fast location of particles in complex 3D grids.

2. Particle-search algorithms

This section presents a brief summary of two recently published particle-tracking algorithms, and discusses the virtues and drawbacks of each method.

2.1. Zhou–Leschziner algorithm (particle-to-the-left)

The algorithm introduced by Zhou and Leschziner (1999) defines an arbitrary 2D convex polygon **P** by giving the Cartesian coordinates (x, y) of its n vertices ordered anticlockwise (see Fig. 1):

$$\mathbf{P} \equiv \{P_i = (x_i, y_i) | i = 1, 2, \ldots, n\} \tag{1}$$

The ZL approach suggests a simple, yet effective procedure to find out whether a particle lies within a given cell: move along the cell faces (i.e., the segments that join two consecutive cell vertices) anticlockwise and check if the particle lies to the left of *all* the cell faces. If this is the case, the particle is within the cell. This is exemplified in Fig. 1.

The *particle-to-the-left* condition, which will be referred to as P2L, can be checked for each cell face, as suggested by the ZL algorithm, by looking at the z component of the cross product between the face vector P_iP_{i+1} and the particle vector $P_iX(t + \Delta t)$, where $X(t + \Delta t)$ is the particle position to be located (see Fig. 2):

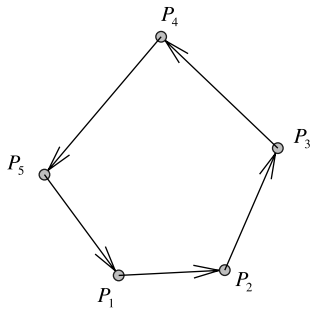

Fig. 1. Definition of a polygon by its vertices ordered anticlockwise.

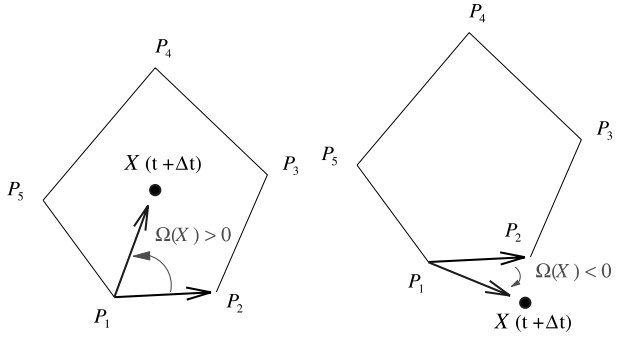

Fig. 2. Examples of the P2L test.

$$\Omega_i \equiv (x_{i+1} - x_i)(y(t + \Delta t) - y_i) - (x(t + \Delta t) - x_i)(y_{i+1} - y_i) \tag{2}$$

- $\Omega_i > 0$ indicates that the point P_i is on the left-hand side of the cell face.
- $\Omega_i < 0$ indicates that the point is on the right-hand side of the face.
- $\Omega_i = 0$ indicates that the point is on the face.

The P2L test Ω_i is computed for all the faces of a cell until one face yields a negative value. Then, the ZL algorithm moves to the cell sharing the face for which the P2L test failed (this face is drawn in Fig. 3 with thick lines). The particle tracking ends when the algorithm reaches a cell where all the P2L tests yield a positive value (see Fig. 3), which indicates that the particle is in that cell.

2.2. Chen–Pereira algorithm (directed search)

The technique introduced by Chen and Pereira (1999) suggests a particle-locating method quite different from the ZL algorithm. The CP approach performs a *directed* search of the particle by travelling through the cells crossed by the particle trajectory $X(t)X(t + \Delta t)$ which joins the initial and final particle position (see Fig. 4).

The criterion employed by the CP algorithm to choose the next cell is to find the cell face through which the particle trajectory exits the current cell. (The *exit faces* are shown in Fig. 4 with thick lines).

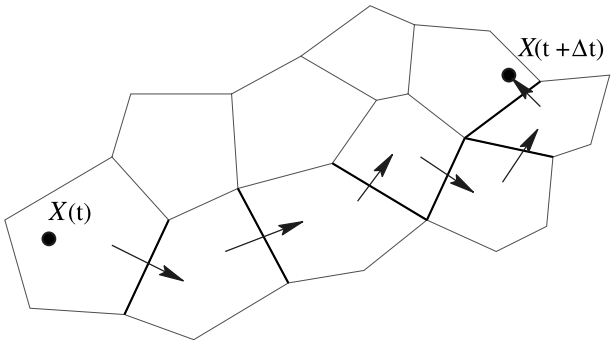

Fig. 3. Example of a typical path followed by the ZL algorithm (P2L).

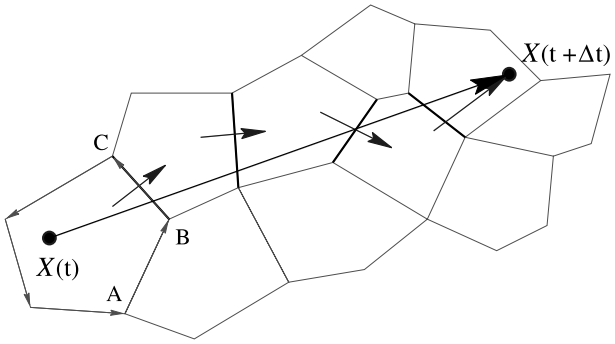

Fig. 4. Example of a typical path followed by a directed-search algorithm, such as the CP or the proposed algorithm.

The CP technique (Chen and Pereira, 1999) can be summarized in the following steps:

1. Perform the P2L test for all the cell faces of a given cell.
2. If all the P2L tests are passed, then the particle lies within the current cell.
3. If not, locate the face through which the particle trajectory exits the current cell:
   (a) Select the cell faces which have failed the P2L test (faces AB and BC in Fig. 4).
   (b) Compute the intersection between the particle trajectory ($X(t)X(t+\Delta t)$) and the selected faces.
   (c) Choose the face (face BC in Fig. 4) whose intersection lies within the limits of the face (i.e., an *internal intersection*; see Chen and Pereira (1999) for details).
4. Move to the neighbouring cell which shares the face containing the exit trajectory and go to step 1.

2.3. Comparison of the approaches

The two particle-locating methods presented in the previous sections have quite distinctive characteristics. This section highlights the pros and cons of each method, and concludes with some desired features of an improved, particle-locating algorithm.

One of the major advantages of the ZL method is that it can be implemented very straightforwardly, and has shown best CPU-time performance (Chen and Pereira, 1999) than the more-elaborated CP algorithm. However, there are some drawbacks associated to the ZL method which are addressed next.

Firstly, the ZL method can fail to locate a particle due to a *shadowing effect* of the P2L approach. An example of such a special (but not improbable) situation is shown in Fig. 5. The P2L test leads to a circular search of the particle around the target cell which would, not only fail to find the particle, but would also hang the simulation in an infinite loop.

Secondly, the extension of the ZL method to 3D grids seems complex due to the possible existence of non-planar cell surfaces in 3D grids.

And, finally, the ZL method traverses a larger number of cells (compared to other methods, such as CP) during the course of the particle location. This fact has been reported in Chen and Pereira (1999) to result in a CPU-time penalty for the numerical simulation of certain fluid mechanics problems, such as strongly coupled two-phase flows.

The CP method emerges as more robust alternative to the ZL approach, providing smaller search paths. However, the CP method also suffers from some important disadvantages, such as a more complex implementation, larger CPU times in a general particle-locating context, and a not-too-clear extension to 3D grids.

It would be desirable to design a new particle-locating algorithm which shared the advantages of the previous methods (easy implementation and robustness) while maintaining other important features such as small CPU-time requirements and potential extension to 3D grids. The next section presents a new particle-locating algorithm aimed at fulfilling these goals.

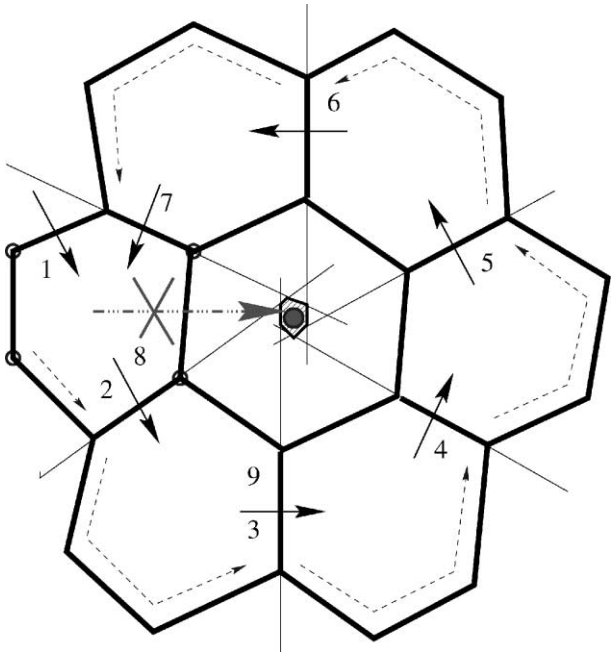

Fig. 5. *Shadowy effect* of the ZL algorithm.

3. Proposed algorithm

The largest computational burden of the CP approach lies in the determination of the face containing the exit trajectory (step 3 of CP algorithm), which requires the calculation of several line intersections. In the present work, a more efficient procedure to locate the *exit face* is presented. Besides, the CP algorithm has been slightly reformulated in order to improve its performance.

In essence, the new algorithm uses the economical P2L test (Eq. (2)) to identify the exit face of the particle trajectory, rather than having to calculate line intersections.

The detection of face–trajectory intersection is illustrated in Fig. 6. This figure shows how the z component of the cross product of the vectors $X(t)P_i$ and $X(t)X(t+\Delta t)$ can be used to detect the face–trajectory intersection. The expression of the z component of vertex i is:

$$L_i \equiv (x_i - x(t))(y(t+\Delta t) - y(t)) - (x(t+\Delta t) - x(t))(y_i - y(t)) \tag{3}$$

Fig. 6 shows that $L > 0$ when the particle trajectory lies to the left of a given vertex (e.g., vertex $i+1$ in Fig. 6), and $L < 0$ otherwise (e.g., vertex i). For this reason, the computation of the value of L_i will be termed as *trajectory-to-the-left* (T2L) test. Fig. 6 also reveals that if two consecutive vertices have opposite signs of T2L, then the particle trajectory crosses the face connecting such vertices. The procedure just outlined will be employed in the proposed algorithm for the detection of face–trajectory intersections

The proposed algorithm has the following steps:

1. Check if the particle trajectory $X(t)X(t+\Delta t)$ crosses one face of the current cell. This is done by applying the T2L test (Eq. (3)) to the two face vertices and comparing the sign of the T2L test. In cell I of Fig. 7, the crossing faces are BC and FA.
2. If it does, check the P2L test on that face. If the particle lies to the right of the face (P2L < 0), then we have found the appropriate (i.e., exit) crossing face (this is the case of cell I, face BC in Fig. 7). Exit the loop and move to the neighbouring cell that shares that face.
3. Move to the next face and go to step 1.

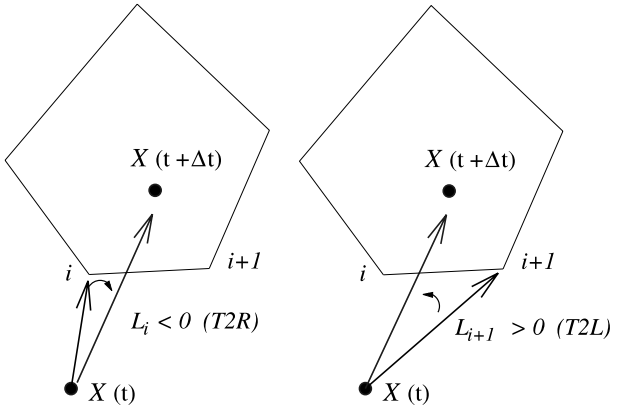

Fig. 6. Illustration of the T2L test employed for the detection of the face–trajectory intersections.

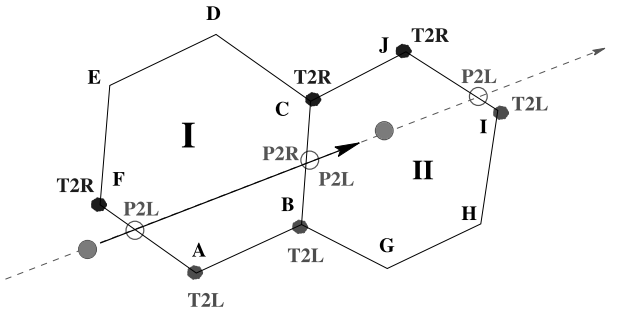

Fig. 7. Example of the proposed particle-locating algorithm.

4. If the loop over all the cell faces is finished without fulfilling step 2 (i.e., P2L > 0 for all the crossing faces, such as faces IJ and CB of cell II, in Fig. 7), then the particle lies within the current cell.

The face–trajectory intersection would be impossible to compute in the event of having a particle trajectory crossing through a cell vertex. This would make the T2L test fail in the case of having an $L = 0$ value for i or $i + 1$.

In order to account for this (highly improbable) situation, the following modification to the face-intersection is suggested: assume that the face $i - (i + 1)$ is crossed if $L_i \cdot L_{i+1} \leqslant 0$.

4. Extension to three-dimensional grids

This section presents a brief outline of the 3D extensions of the ZL and CP algorithms proposed by their authors (Zhou and Leschziner, 1999; Chen and Pereira, 1999), and describes the extension of the proposed algorithm to 3D grids, highlighting the advantages of the present method compared to the ZL and CP ones.

4.1. ZL and CP extensions to 3D

The 3D extension of the ZL algorithm must redefine the P2L concept to 3D grids, where the cell face is no longer a line segment, but a set of vertices. The ZL approach assumes that the cell-face vertices are anticlockwise ordered when the cell face is observed from the outside of the cell (see Fig. 8). With this vertex order, the cross product vector between $P_{i+2}P_{i+1}$ and $P_{i+1}P_i$ is always directed towards the inside of the cell. Thus, the P2L extension to 3D, which will be termed as *particle-towards-the-inside*, P2I, can be written as:

$$\forall i \text{ of the face} \quad \mathrm{Sign}[(P_{i+2}P_{i+1} \times P_{i+1}P_i) \cdot P_{i+1}X(t + \Delta t)] > 0 \tag{4}$$

The undesired shadowy effect of the ZL algorithm (see Section 2.3), which is attributed to the indirect search performed by the ZL algorithm, is expected to increase for the case of 3D grids, due to the imprecision introduced by the non-planar nature of some 3D cell faces. This fact has been evaluated by the authors of this work by performing a 3D particle-locating simulation. It has been found that the ZL algorithm gets trapped in a 3D shadowy zone when a sufficiently large number of points of the spatial domain are tested.

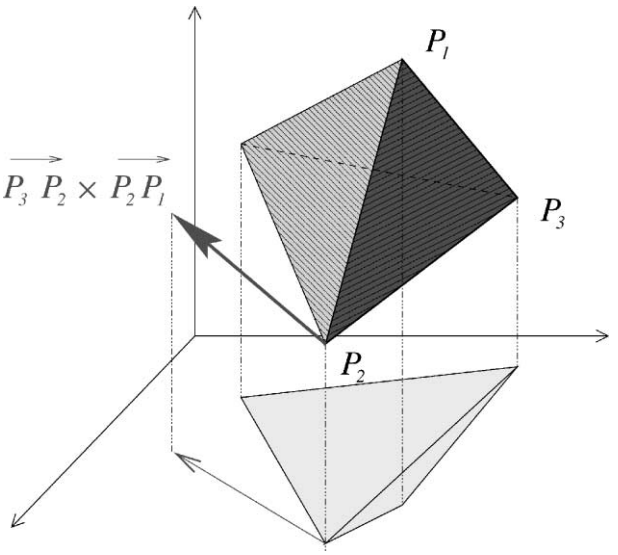

Fig. 8. Schematic of a 3D cell, showing anticlockwise order for the face vertices.

Regarding the CP algorithm, no details are given in Chen and Pereira (1999) about the extension of the CP method to 3D grids. It could be assumed that the *natural* extension of the CP approach to 3D grids would be to compute the intersection between the particle trajectory and the cell face. At this point, two important disadvantages for the CP method emerge. Firstly, due to the deformation of 3D grids in order to account for complex geometries, the cell faces are usually slightly non-planar. This fact greatly complicates the calculation of the intersection between the particle trajectory and the cell face. And secondly, the location of such intersection may turn computationally more expensive than in 2D meshes, which would further spoil the algorithm performance.

4.2. 3D extension of the proposed algorithm

The new algorithm must locate, as the CP algorithm, the face through which the particle leaves the cell when it travels along the trajectory $X(t)X(t+\Delta t)$. The proposed algorithm avoids, as for its 2D counterpart, the expensive computation of face–trajectory intersections. Instead, a set of simple algebraic expressions is evaluated.

The first step for the detection of trajectory–face intersection is to check the relative orientation of a face segment and the trajectory. This exercise is shown in Fig. 9 for the face segment P_1P_2 and the bottom cell face.

The key element for the relative orientation of P_1P_2 and the particle trajectory is the plane that contains the points P_1, P_2 and $X(t+\Delta t)$. Three different situations can be found:

(a) Trajectory towards the inside of the face ($X(t)X^a(t+\Delta t)$).
(b) Trajectory passing through the face border ($X(t)X^b(t+\Delta t)$).
(c) Trajectory towards the outside of the face ($X(t)X^c(t+\Delta t)$).

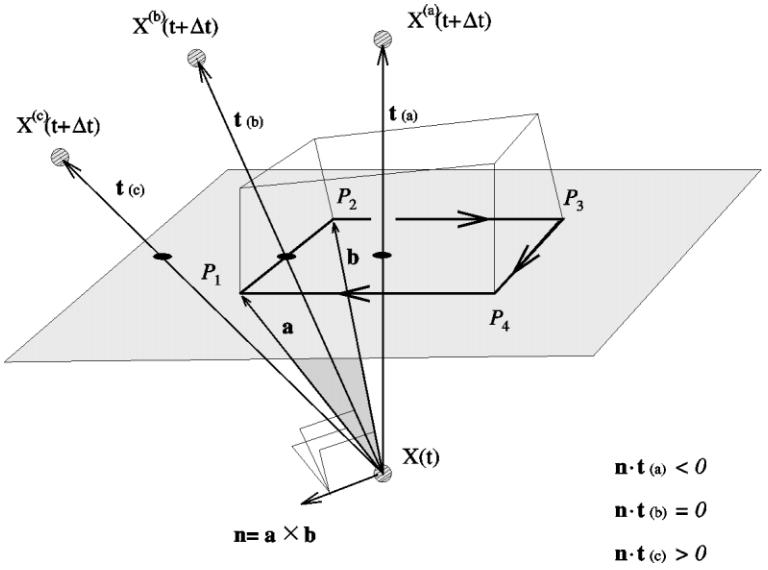

Fig. 9. Definition of vectors for the detection of 3D trajectory–face intersection (bottom cell face).

It is clear that, in a situation like the one portrayed in Fig. 9, a particle trajectory crosses a given cell if, and only if, the condition (a) is verified for all the face segments. Thus we need a mathematical expression that classifies the trajectory as type (a), (b) or (c) with respect to each face segment P_iP_{i+1}.

For that purpose, we define the following vectors:

$$\mathbf{a} = X(t)P_1 \tag{5}$$

$$\mathbf{b} = X(t)P_2 \tag{6}$$

$$\mathbf{n} = \mathbf{a} \times \mathbf{b} \tag{7}$$

$$\mathbf{t} = X(t)X(t + \Delta t) \tag{8}$$

Here, $\mathbf{n}$ is the normal vector of the plane $P_1P_2X(t)$. This vector, due to the ordering of the face vertices, always points towards the outside of the face. Hence, the three trajectory types can be identified by projecting the trajectory vector $\mathbf{t}$ on the normal vector $\mathbf{n}$:

(a) $\mathbf{n} \cdot \mathbf{t} < 0 \rightarrow$ trajectory towards inside of face.
(b) $\mathbf{n} \cdot \mathbf{t} = 0 \rightarrow$ trajectory through face border.
(c) $\mathbf{n} \cdot \mathbf{t} > 0 \rightarrow$ trajectory towards outside of face.

A similar analysis can be carried out for the *top* face of the same cell. This analysis is presented in Fig. 10. Due to the face-vertices ordering, now the normal vector $\mathbf{n}$ points towards the inside of the face. As a result, a trajectory–face intersection is found if $\mathbf{n} \cdot \mathbf{t} > 0$ for all the face segments P_iP_{i+1}.

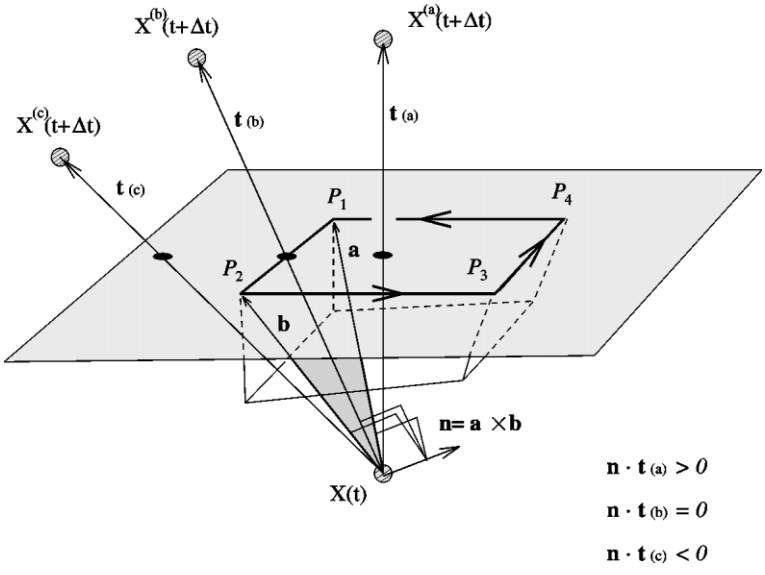

Fig. 10. Definition of vectors for the detection of 3D trajectory–face intersection (top cell face).

The method indicated above can be summarized in the following two points:

1. The particle trajectory crosses a given face if and only if:

$$\forall i \quad \mathrm{Sign}[(X(t)P_i \times X(t)P_{i+1}) \cdot t] = \mathrm{const} \tag{9}$$

2. The sign of $\mathbf{n} \cdot \mathbf{t}$ provides additional information. As the vertices are anticlockwise ordered for each face of a cell, the value of the sign tells us whether the particle leaves or enters the cell through that face (see Figs. 9 and 10):

$$\mathbf{n} \cdot \mathbf{t} > 0 \rightarrow \text{particle leaving cell} \tag{10}$$

$$\mathbf{n} \cdot \mathbf{t} < 0 \rightarrow \text{particle entering cell} \tag{11}$$

After checking the face–trajectory intersection, one must, as in the CP or ZL algorithm, verify if the final position of the particle is towards the inside the cell. This can be easily done with the P2I test (Eq. (4)).

5. Results

In order to evaluate the performance of the proposed particle-locating algorithm and to compare its performance with the ZL and CP methods, two particle-locating problems have been addressed, and their results are shown in this section.

The first test consists in a 2D unstructured grid where the initial and final particle positions are chosen at random. The process of random particle location is carried out for two types of particle

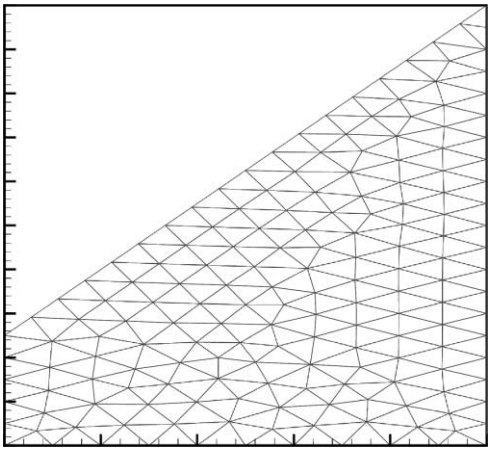

Fig. 11. 2D triangular grid employed in the random particle-location test.

paths, short and long ones, in order to compare the efficiency of the three algorithms in different situations.

The second test applies the three particle-locating techniques to a Lagrangian simulation of the convection of fluid particles in a frozen velocity field extracted from a numerical simulation of a turbulent reacting flow (Fueyo et al., 2000). The aim of this test is to assess the performance of each method in a more realistic context.

5.1. Random particle location

A 2D triangular (and thus, unstructured) mesh is set up for a domain with slanting limits. The resulting triangular grid is shown in Fig. 11 and is comprised of 289 cells.

The initial and final positions of the particles are chosen at random within the 2D grid. Two cases are considered: short and long particle strides. The short-range particle excursions include random displacements of 1–3 cells (in average), while the long-range particle ones are taken as 10-cell strides in average. The results of such an exercise are reported in Table 1 and Fig. 12.

Table 1 presents the CPU time [1] required by the three algorithms in the search of particles located at different distances from the initial position: 1, 2, 3, and 10 cells away. The total number of particles being searched is 10^7.

Table 1 and Fig. 12 show that the ZL and the proposed algorithm are almost equally fast, and are between 2 and 4 times faster than the CP method, which confirms the CPU-time results reported by Chen and Pereira (1999). Table 1 also reveals that the CPU time employed by the new algorithm is slightly smaller (3–8%) than the ZL technique. Another (already expected) conclusion that can be gathered from Table 1 and Fig. 12 is that the search time is proportional to the number of crossed cells.

[1] All the CPU times reported in this paper have been measured in an Intel, 550-MHz Pentium-III running the Linux operating system.

Table 1
CPU time and average number of crossed cells for the 2D random particle-location test

Displacement	Method	CPU time (s)	Time ratio to ZL	Number of crossed cells
1-cell	ZL	13.63	1.00	1.02
	CP	27.43	2.01	1.01
	New	12.75	0.94	1.01
2-cell	ZL	19.89	1.00	2.06
	CP	48.91	2.45	2.03
	New	18.32	0.92	2.03
3-cell	ZL	25.11	1.00	3.07
	CP	70.00	2.78	3.01
	New	23.33	0.93	3.01
10-cell	ZL	59.65	1.00	10.44
	CP	230.43	3.86	10.05
	New	57.74	0.97	10.05

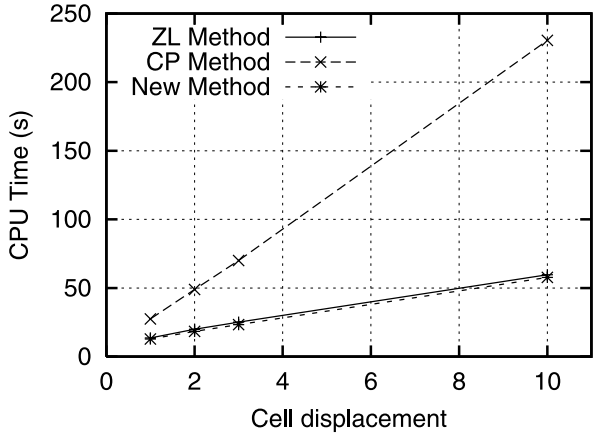

Fig. 12. CPU time of the particle-locating algorithms as a function of the crossed cells.

Regarding the number of crossed cells, the CP and the new algorithm traverse exactly the same number of cells. This fact was indeed expected since both algorithms employ the same *directed-search* scheme. On the other hand, the number of crossed cells of the ZL method is slightly greater (around 1%) than the CP/new methods. This feature can make the CP or the new algorithms a better choice for certain fluid mechanics problems.

The random particle-locating test has also been carried out in a 3D hexahedral mesh. However, due to the problems associated with the ZL and CP extensions to 3D grids (see Section 4), the present 3D test has only been performed with the proposed algorithm.

The results of the 3D location test are shown in Table 2. The 3D particle-locating algorithm is between 5 and 6 times more expensive than the 2D one.

Table 2
CPU time and average number of crossed cells for the 3D random particle-location test with the proposed algorithm

Displacement	CPU time (s)	Time ratio to 2D
1-cell	66.44	5.21
2-cell	101.93	5.56
3-cell	133.36	5.72
10-cell	356.37	6.17

5.2. Lagrangian simulation

The second test performs a Lagrangian simulation of the turbulent flow of a laboratory, methane–air burner (Nandula et al., 1996). A schematic of the setup is shown in Fig. 13. The methane–air mixture is fed into the combustion space through an annular ring surrounding a circular obstacle that acts as a bluff-body. A recirculation region is established immediately downstream of the bluff-body, in which the trapped hot products serve the purpose of anchoring the flame.

In the present Lagrangian simulation of the methane–air flame, a frozen velocity field taken from previous numerical simulations of the flame (Fueyo et al., 2000) is employed, since this suffices to test the performance of the particle-locating algorithms.

The evolution of the chemical species is given by the integration of the composition-PDF equation by means of a Lagrangian Monte Carlo method (Pope, 1985). The Lagrangian PDF particles undergo two processes: transport in physical space and evolution in compositional space.

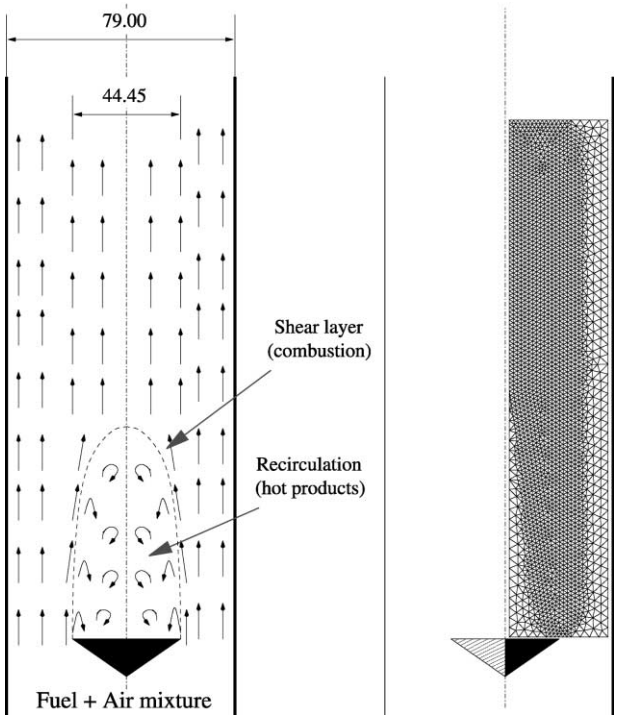

Fig. 13. 2D triangular grid employed in the Lagrangian simulation.

The convection of the PDF particles is described through the following equation:

$$\Delta x^{(n)} = V(x^{(n)})\Delta t + \Delta x_{\mathrm{rw}}^{(n)} \tag{12}$$

where $\Delta x^{(n)}$ is the change in the position of the particle n, $V(x^{(n)})$ is the mean velocity vector at the particle position, Δt is the time step, and $x_{\mathrm{rw}}^{(n)}$ is a stochastic displacement of the particle due to turbulent motion. The last term is modeled as a random walk (Pope, 1985).

The changes to the PDF particles in composition space are due to chemical reaction and molecular mixing:

$$\frac{\partial \phi_i^{(n)}}{\partial t} = \dot{\omega}_i^{(n)} + \frac{1}{\tau_{\mathrm{mix}}}(\bar{\phi}_i - \phi_i^{(n)}) \tag{13}$$

where $\phi_i^{(n)}$ and $\dot{\omega}_i^{(n)}$ are, respectively, the concentration and the rate of reaction of chemical species i and particle n, τ_{mix} is an externally supplied mixing timescale (typically $\tau_{\mathrm{mix}} = k/\epsilon$, where k and ϵ are the turbulent kinetic energy and its dissipation rate), and $\bar{\phi}_i$ is the mean value of chemical species i in the current cell. In Eq. (13), the molecular mixing term has been represented with the LMSE model (Dopazo, 1975).

A simulation step of the Lagrangian composition-PDF consists of the following parts (Topaldi and Correa, 1997):

1. Inlet of particles: An appropriate number of particles with the inlet chemical composition is placed in the domain inlet cells.
2. Particle-tracking phase: The positions of the Lagrangian particles are updated using Eq. (12). The particle-locating algorithm is then employed to find the new cell occupied by each particle. During the convection of the particles, their trajectories may be reflected on walls or symmetry axes. This situation must be efficiently dealt with by the particle-locating algorithm.
3. Chemistry-mixing phase: The composition of each Lagrangian particle is changed following Eq. (13).
4. Averaging phase: This phase computes the averages of the properties of the cells (typically: species concentrations, temperature and density).

The Lagrangian simulation of the turbulent reacting flow has been addressed with the three methods (CP, ZL and the proposed algorithm).

Table 3 presents the CPU time employed by the three particle-locating algorithms (ZL, CP and proposed one) within a 5×10^5-particle Lagrangian simulation carried out for 20 time steps. The CPU times are almost the same for the ZL and the proposed methods, as was expected from the results obtained in the previous test (random particle location). The CP method is almost 2 times

Table 3
CPU time and average number of crossed cells for the location subroutine in a Lagrangian simulation

Method	CPU time (s)	Time ratio to ZL	Number of crossed cells
ZL	33.57	1	2.282
CP	61.44	1.83	2.268
New	32.99	0.983	2.268

slower than the two other ones. The particle-locating CPU time scales with the number of crossed cells and, for the case of the proposed algorithm, the particle-locating time represents only a moderate fraction (7.5%) of the overall Lagrangian simulation-time (440 s).

6. Conclusions

The present paper has introduced and evaluated a new particle-locating algorithm valid for arbitrary 2D or 3D grids.

Firstly, two recently published, particle-locating methods (ZL and CP) have been reviewed, and some important shortcomings have been identified. Based on this analysis, an improved 2D/3D particle-locating algorithm has been described.

The performance of the proposed particle-locating method has been compared to the ZL and CP algorithms by means of two tests: a random, particle-locating test and a Lagrangian simulation of a turbulent premixed flame.

The simulations conducted show that the proposed method is slightly faster than the ZL algorithm, and between two and almost four times faster than the CP method. The unsophisticated design-principles of the proposed particle-locating algorithm makes its 2D/3D implementation quite straightforward and yields CPU times which are quite affordable for its inclusion in complex 2D/3D numerical simulations. Besides, the directed-search feature of the proposed algorithm eliminates the shadowy zones of the ZL algorithm, optimizes the number of crossed cells, and allows the detection of walls during the tracking of the particle trajectory, thus making the proposed algorithm a better choice for a great variety of fluid mechanics applications.

Acknowledgements

This work has been partially supported by the European Union under project number GRD1-1999-10325 (CFD4C).

References

Chen, X.-Q., 1997. An efficient particle-tracking algorithm for two-phase flows in geometries using curvilinear coordinates. Numerical Heat Transfer 31, 387–405.

Chen, X.-Q., Pereira, J.C.F., 1999. A new particle-locating method accounting for source distribution and particle-field interpolation for hybrid modeling of strongly coupled two-phase flows in arbitrary coordinates. Numerical Heat Transfer, Part B 35, 41–63.

Dopazo, C., 1975. Probability Density Function approach for a turbulent axisymetric heated jet centerline evolution. Physics of Fluids 18, 397.

Frank, T., Schulze, I., 1994. Numerical simulation of gas-droplet flow around a nozzle in a cylindrical chamber using a Lagrangian model based on a multigrid Navier–Stokes solver. Numerical Methods for Multiphase Flows 185, 93–107.

Fueyo, N., Vicente, W., Blasco, J.A., Dopazo, C., 2000. Stochastic simulation of NO formation in lean premixed methane flames. Combustion Science and Technology 153, 295–311.

Guzman, M.M., Fletcher, C.A.J., Behnia, M., 1995. Gas particle flows about a cobra probe with purging. Computers & Fluids 24, 121–134.

Nandula, S.P., Pitz, E.W., Barlow, R.S., Fiechtner, G.J., 1996. Rayleigh-Ramman-LIF measurements in a turbulent lean premixed combustor. In 34th Aerospace Sciences Meeting Exhibit, Reno, NV, January 15–18, 1996.

Oliveira, P.J., Gosman, A.D., Issa, R.I., 1997. A method for particle location and field interpolaion on complex, three-dimensional computational meshes. Advances in Engineering Software 28, 607–614.

Pope, S.B., 1985. PDF methods for turbulent reactive flow. Progress in Energy and Combustion Science 11, 119–192.

Topaldi, A.K., Correa, S.M., 1997. Monte Carlo probability density function method for gas turbine combustor flowfield predictions. Journal of Propulsion and Power 13 (2), 218–225.

Valentine, J.R., Decker, R.A., 1995. A Lagrangian–Eulerian scheme for flow around an airfoil in rain. International Journal of Multiphase Flow 21, 639–648.

Zhou, Q., Leschziner, M.A., 1999. An improved particle-locating algorithm for Eulerian–Lagrangian computations of two-phase flows in general coordinates. International Journal of Multiphase Flow 25, 813–825.