PERGAMON

International Journal of Multiphase Flow 25 (1999) 813–825

International Journal of Multiphase Flow

www.elsevier.com/locate/ijmulflow

An improved particle-locating algorithm for Eulerian-Lagrangian computations of two-phase flows in general coordinates

Q. Zhou *, M.A. Leschziner

Department of Mechanical Engineering, UMIST, P.O. BOX 88, Manchester, U.K.

Received 10 October 1997; received in revised form 17 July 1998

Abstract

A new scheme is proposed for tracking the position of particles within a fixed mesh used to compute the motion of a continuous carrier fluid. The principal output of the scheme is the identification of the grid cell within which the particle resides at any time during the tracking of its trajectory. This information is required for the solution of the particle-motion equation and for modelling the exchange processes between the particle and surrounding fluid. The new method is explained in detail, and its performance is contrasted, by way of two examples, against a recently proposed highly efficient scheme. It is shown that the new method is about 30% faster than that scheme, held to be the most efficient algorithm documented in the literature. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Particles; Locating; Tracking; Computational modelling; Eulerian/Lagrangian algorithm

1. Introduction

A key element in any hybrid Eulerian/Lagrangian calculation procedure for two-phase flow is an algorithm by which particles (or droplets) are tracked in time and space within a fixed aero/hydrodynamic field. This algorithm involves the solution of well-established ODEs for particle motion and position along the particle's trajectory, with the instantaneous carrier velocity being extracted from the Eulerian NS solution—in conjunction with a dispersion

* Corresponding author.

PII: S0301-9322(98)00045-7

model, in the case of RANS schemes. An important element of the algorithm is the determination of the particle's position within the discrete mesh of nodes at which properties of the continuous carrier phase are stored, and an interpolation of these properties to the particle's position, which are required to solve the equations governing the particle's kinematic, dynamic and thermal state. In a fully interactive, conservative procedure—say, in a finite volume scheme in which volume-averaged mass, momentum and heat sources arising from inter-phase exchange must be determined—a supplementary task is to search for the finite volume within which the particle and the carrier exchange the relevant properties.

If a structured Cartesian grids is adopted for the simulation of the carrier flow, the particle-locating problem can be greatly simplified by exploiting the fact that faces of control volumes are parallel to the axes of the Cartesian coordinates. Unfortunately, these simplifications do not extend to general systems of body-fitted curvilinear coordinates, which are now widely used for complex geometries in industrial applications.

Chen (1997) has recently proposed an efficient particle-search algorithm for body-fitted curvilinear coordinates which, in contrast to the scheme by Guzman et al. (1995), can also accommodate particles moving by more than one Eulerian grid cell within one Lagrangian particle-tracking time step. Chen's algorithm is much more efficient than a scheme by Oliveria (1992) in which an iterative numerical solution is employed to determine the index of the cell containing the particle by use of trilinear isoparametric functions. Also, in comparison with the 'circular-search' approach proposed by Frank and Schulze (1994), Chen's algorithm offers a more efficient, self-adjusting mechanism for searching the new Eulerian cell in the event of a particle changing cells during a Lagrangian tracking time step. Valentine and Decker (1995) have proposed an interesting route to extending the simple particle-locating algorithm applicable to Cartesian coordinates to curved coordinates by computing the particle trajectory in terms of its contravariant velocity components. However, as observed by Chen, this practice requires additional metric calculations as well as extra computer memory, both of which adversely affect the method's efficiency.

In specific terms, any particle-locating algorithm must answer two questions. First, is a point (or particle location) inside or outside any particular Eulerian cell? Second, in which control volume does the particle reside if it has moved out of its previous cell within a Lagrangian tracking time step? Since a particle-locating calculation has to be performed for every particle after every tracking time step, an efficient algorithm is essential if computations involving thousands of particles and thousands of time steps are to be undertaken at economically tolerable costs. Hence, any improvement in the efficiency of existing approaches can have an important impact on the size and complexity of two-phase-flow problems which can be computed within given resource constraints. This is an especially important consideration when the two-phase-flow scheme involves the use of resource-intensive anisotropy-resolving turbulence and dispersion models based on second-moment closure, such as those devised and used by the present authors (Zhou and Leschziner, 1996, 1997). It is the recent publication of Chen's scheme that motivates the presentation of our independently developed particle-locating algorithm which is simpler and more efficient than Chen's, as well as being applicable to any coordinate system.

2. Particle-search algorithm

Within an Eulerian carrier scheme, the carrier-velocity vector is available at spatially discrete nodal locations. The particle-locating algorithm is required to evaluate the velocity (and any other pertinent fluid or flow properties) at the particle location. Once this information is available, the particle-motion and location equations can be solved by one of many standard numerical solution procedures for ODEs, such as the 2nd-order modified Euler method, the 4th-order Runge–Kutta method (Lambert, 1973) or exponential-Lagrangian tracking schemes (Barton, 1996).

Fig. 1 depicts the motion of a particle within a tracking time step δt along an arbitrary trajectory of a particle across a general mesh of quadrilateral cells. The task is one of obtaining the carrier properties at the particle locations $[x(t), y(t)]$ and $[x(t + \delta t), y(t + \delta t)]$ to solve the particle-motion equations. The arbitrary shape of the Eulerian cells makes it almost impossible to extend the one-to-one look-up table or the analytic expressions used in algorithms for Cartesian coordinates, unless the contravariant velocity components of the particle are computed, as proposed by Valentine and Decker (1995). Most particle-locating algorithms take the (I, J) indices of the cell containing the particle at the old time level as the reference and then search for the particle in cells around the reference control volume following a forward step δt. Clearly, this approach is much more efficient than an indiscriminate search applied to the whole computational domain. Such a search is occasionally unavoidable, however—for example, at the very beginning of the particle-tracking process whenever new particles are being introduced—and it is important, therefore, to have an efficient general search algorithm in addition to a scheme which determines whether a particle is inside or outside a particular cell.

The present scheme is a simpler and more efficient alternative to that of Chen, both in terms of determining whether a particle is inside or outside a cell and of searching of the new cell containing the particle. For the sake of clarity, the principles of the algorithm are presented here in two-dimensional (2D) space, but the scheme can be readily extended to three-dimensional (3D) or unstructured-grid arrangements. Once the principles of the scheme have

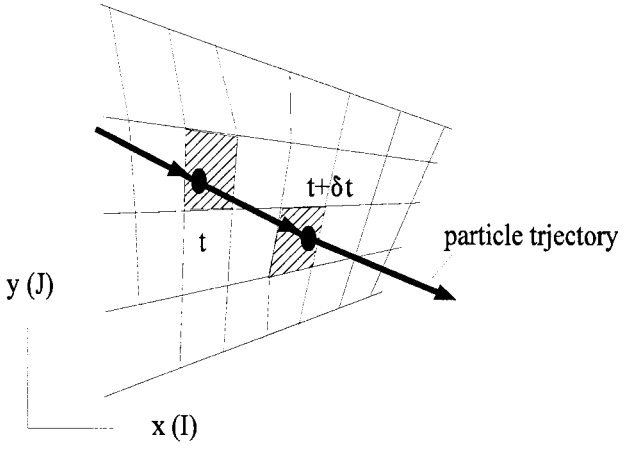

Fig. 1. Particle locations at times t and $t + \delta t$ in general structured two-dimensional mesh.

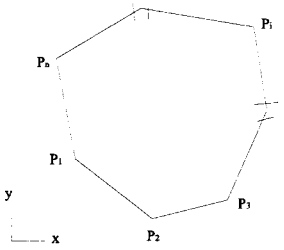

Fig. 2. An arbitrary polygon in Eulerian mesh with n vertices $\{P_i \equiv (x_i, y_i) | i = 1, 2, \ldots, n\}$.

been outlined for arbitrary volumes, it will be simplified for quadrilateral cells, facilitating a direct comparison with Chen's scheme. Section 4 will discuss aspects of the algorithm's extension to 3D.

Fig. 2 shows an arbitrary 2D polygon defined by the Cartesian coordinates of its n (>2) vertices $\{P_i \equiv (x_i, y_i) | i = 1, 2, \ldots, n\}$ here ordered in anti-clockwise manner. Such a description of a convex polygon is called an 'oriented convex set of vertices'.

The n boundary edges of the Eulerian cell are the line segments:

$$L_i(x, y) \equiv (x_{i+1} - x_i)(y - y_i) - (y_{i+1} - y_i)(x - x_i), \tag{1}$$

where $i = 1, 2, \ldots, n$ and addition in the subscripts is modulo n (that is, $n + j \equiv j$ for $1 \leqslant j \leqslant n$).

The analytic representation of a given line segment, say the one joining P_i to P_{i+1} for some i, is represented in the form (1) in order to take advantage of an interesting and useful property of this formulation. If one imagines oneself astride on the line looking from P_i towards P_{i+1} then the positive side of the line is to the left and the negative side to the right. If the vertices of a polygon are oriented anti-clockwise, then the inside of the polygon is defined by:

$$\{(x, y) \mid L_i(x, y) > 0 \quad \text{for all } i, \quad 1 \leq i \leq n\}, \tag{2}$$

a point on the boundary is given by:

$$\{(x, y) \mid L_i(x, y) \geq 0 \text{ for all } i, \quad 1 \leq i \leq n, \\ \text{with } L_i(x, y) = 0 \text{ for at least one } i\} \tag{3}$$

and the outside of the polygon is defined by:

$$\{(x, y) \mid L_i(x, y) < 0 \text{ for all } i, \quad 1 \leq i \leq n\}. \tag{4}$$

This inside/outside technique also yields an important indicator for the next search if the particle being tracked is moving out of the reference cell. For example, when the reference cell is P (Fig. 3) with vertices $\{P_i \equiv (x_i, y_i) | i = 1, 2, \ldots, n\}$, the result $L_k(x, y) < 0$, with k being the first index encountered, indicates that the particle is outside of P. Then the adjacent cell Q sharing the common boundary with P at

$$L_k(x, y) \equiv 0 \tag{5}$$

should be the cell examined first to determine whether the particle has entered that cell.

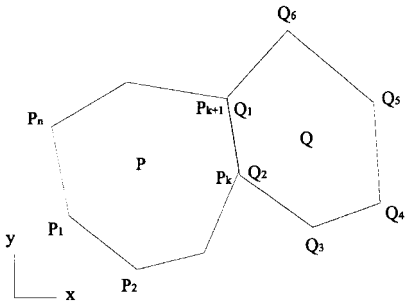

Fig. 3. Cells used to derive the searching direction indicator $L_k(x, y) < 0$ for cell P.

Alternatively, Q can be declared as the new reference cell. If the particle is found to be outside Q as well, a new reference cell may be found as was done with cell P. The procedure is repeated until the newly defined reference cell is eventually found to contain the particle. This practice not only ensures that the next cell destined for searching is known, but it also prevents unproductive searches along the ith edge ($k < i \leq n$) of the current reference cell, thus enhancing efficiency.

3. Comparison with Chen's scheme

To contrast the present algorithm with that of Chen, both are applied to the quadrilateral cell shown in Fig. 4. For clarity of comparison, Chen's notation is used. The reference cell consists of four vertices given in Cartesian form as $A(x_a, y_a)$, $B(x_b, y_b)$, $C(x_c, y_c)$ and $D(x_d, y_d)$.

In Chen's algorithm, a reference point, $O(x_o, y_o)$ is introduced at the cell's centroid as an aid to a check of whether a particle, located at $P(x_p, y_p)$, is within the quadrangle $ABCD$. The centroid is defined by the average of the coordinates at the four vertices:

$$x_o = \frac{x_a + x_b + x_c + x_d}{4}; \quad y_o = \frac{y_a + y_b + y_c + y_d}{4}. \tag{6}$$

The task of judging whether P is inside of $ABCD$ is then divided into four sub-checks for each edge of the quadrangle ($ABCD$) to determine whether P lies on the correct side of these edges. Consider the west edge AB as an example. In order to check whether P is on the correct side

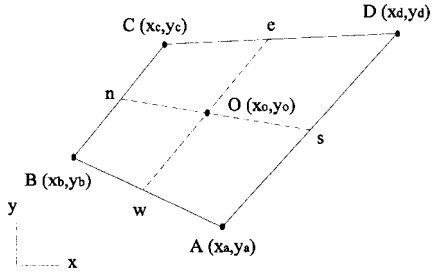

Fig. 4. General quadrilateral reference cell.

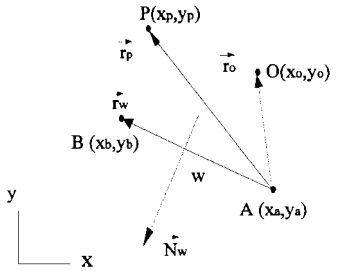

Fig. 5. Chen's auxiliary vectors for the west edge AB of quadrangle $ABCD$.

of edge AB, Chen introduces three auxiliary vectors, as shown in Fig. 5, namely:

$$\begin{aligned}\mathbf{r}_w &= \mathbf{AB} = (x_b - x_a)\mathbf{i} + (y_b - y_a)\mathbf{j}\\ \mathbf{r}_o &= \mathbf{AO} = (x_o - x_a)\mathbf{i} + (y_o - y_a)\mathbf{j}\\ \mathbf{r}_p &= \mathbf{AP} = (x_p - x_a)\mathbf{i} + (y_p - y_a)\mathbf{j},\end{aligned} \tag{7}$$

where $\mathbf{i}$ and $\mathbf{j}$ are the Cartesian unit vectors in the x- and y-directions, respectively. A fourth vector is then defined as the normal to vector $\mathbf{r}_w$:

$$\mathbf{N}_w = -(y_b - y_a)\mathbf{i} + (x_b - x_a)\mathbf{j}. \tag{8}$$

This vector is used to calculate a scalar 'test variable' Ω_w:

$$\Omega_w = (\mathbf{r}_o \cdot \mathbf{N}_w)(\mathbf{r}_p \cdot \mathbf{N}_w), \tag{9}$$

when the particle and the cell centroid are on the same side of edge AB, Ω_w is positive, otherwise, it is negative.

To make use of the above property of the test variable Ω_w, a unit integer variable I_w is introduced, which depends on the sign of Ω_w:

$$I_w = \mathrm{SIGN}(1, \Omega_w) \equiv \begin{cases} 1 & \Omega_w \geq 0 \\ -1 & \Omega_w < 0 \end{cases}, \tag{10}$$

where SIGN can be viewed as the intrinsic FORTRAN sign function. Analogous unit integer variables I_e, J_s and J_n may be derived, respectively, for the east, south and north edges of the cell $ABCD$.

Within a structured-grid system in which indices I and J define cell vertices, as shown in Fig. 6, the search direction for a particle that is found to be outside the current (old) cell is determined by calculating increments ΔI and ΔJ which take the search from the old reference cell (I_{old}, J_{old}) to the new cell (I_{new}, J_{new}). These increments are derived in Chen's scheme as follows:

$$\begin{aligned} I_{new} &= I_{old} + \Delta I \equiv I_{old} + \frac{1 - I_e}{2} - \frac{1 - I_w}{2}\\ J_{new} &= J_{old} + \Delta J \equiv J_{old} + \frac{1 - J_n}{2} - \frac{1 - J_s}{2}, \end{aligned} \tag{11}$$

where divisions in the above equations are modulo 2 or the integer operations in FORTRAN.

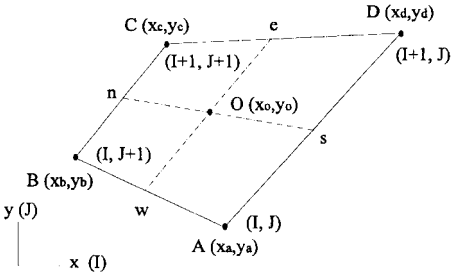

Fig. 6. (*I*, *J*) indexing of the four vertices of a quadrilateral cell *ABCD* in a structured grid system.

Attention is directed next to the present scheme. We note first that the boundary-edge formulation (1) (see (2)–(4)) has the same characteristics as Chen's 'test variables' Ω_b (b = w, e, n, s). Closer examination of our boundary-edge equations reveals that L_i is the combined product of three vectors:

$$L_i = (\mathbf{r}_i \times \mathbf{r}_p) \cdot \mathbf{k}, \tag{12}$$

where $\mathbf{r}_i = \pm \mathbf{r}_b$ $\{b$ = w, e, s, n$\}$. The positive sign of $\mathbf{r}_b$ indicates those boundary-edge vectors which relate to anti-clockwise ordering (i.e. looking along the $\mathbf{r}_i$ direction at the starting point of boundary vector $\mathbf{r}_i$, the reference control volume is on the left-hand side); otherwise, the negative sign applies. Vector $\mathbf{r}_p$ has its origin at one of the end points of vector $\mathbf{r}_i$ and terminates at point P. Unit vector $\mathbf{k}$ is normal to the 2D cell, pointing towards the reader (consistent with a right-hand system). Taking the west edge in Figs. 4 and 5 as an example, if P is on the same side of the reference cell, we find that the cross-product of vector $\mathbf{r}_i(= -\mathbf{r}_w)$ and vector $\mathbf{r}_p$ results in a vector pointing towards the reader; therefore, a positive value is obtained from the formulation of $L_i(i$ = w) above. If, on the other hand, P is on the opposite side of the edge, the vector $\mathbf{r}_i \times \mathbf{r}_p$ will point away from the reader and $L_i(i$ = w) will have a negative value. This demonstrates that L_i indeed has the same characteristics as Ω in Chen's scheme.

The improvement in computational efficiency achieved by the present algorithm can be conveyed, qualitatively, by juxtaposing the full expressions of Chen's 'test variable' Ω and the analogous parameter L for the west edge of the cell in Fig. 4:

$$\begin{aligned}\Omega_w &= (\mathbf{r}_o \cdot \mathbf{N}_w)(\mathbf{r}_p \cdot \mathbf{N}_w) \\ &= [-(x_o - x_a)(y_b - y_a) + (y_o - y_a)(x_b - x_a)][-(x_p - x_a)(y_b - y_a) + (y_p - y_a)(x_b - x_a)]\end{aligned} \tag{13}$$

$$\begin{aligned}L_w &= (-\mathbf{r}_w \times \mathbf{r}_p) \cdot \mathbf{k} \\ &= (x_a - x_b)(y_p - y_a) - (y_a - y_b)(x_p - x_a).\end{aligned} \tag{14}$$

Hence, the present algorithm requires neither the auxiliary vector $\mathbf{r}_o$ nor the edge-normal vector $\mathbf{N}_w$ to determine the location P relative to the west face, and this results in a significant improvement in efficiency. It will be recognized that the unconditional use of an anti-clockwise arrangement in Chen's scheme results in the vector $\mathbf{N}_w$ pointing out of the reference cell, as shown in Fig. 5, in which case the 'test variable' can be simplified along the same lines as our

inside/outside parameter L, and this obviates the need to introduce the vector $\mathbf{r}_o$. The unit integer variable indicating whether P lies on the correct side of the edge can be expressed in the present algorithm as:

$$I_w = \mathrm{SIGN}(1, L_w) \equiv \begin{cases} 1 & L_w \geq 0 \\ -1 & L_w < 0 \end{cases}. \tag{15}$$

The benefit of the anti-clockwise arrangement does not end with the simplification of the above 'test variable' calculation. It also facilitates the search process when a particle has left the reference cell. Since the new cell always shares a common edge with the previous reference cell, there is no need to calculate the inside/outside parameter again for that edge. Instead, the parameter for the new cell can be obtained by simply changing the sign of the parameter calculated for the previous cell. This represents, typically, a 25% saving in computational cost for quadrilateral 2D cells.

Note that as soon as an unit integers in a list (say, J_s, I_e, J_n and I_w) is determined to be negative, the calculation of the remaining unit integers of the same cell is redundant, which further improves the algorithm's efficiency. It is also important to point out here that the search efficiency does not only depend on the efficiency of the inside/outside check, but is a sensitive function of the number of cells that need to be visited to locate the cell in which the particle resides.

Taking the above two factors into account, we propose that a measure of search efficiency may be the number of unit integer variables (or edges) the method has to calculate to locate the cell that contains the particle. If we assume that the probabilities of a particle leaving the reference cell through the south, east, north and west sides are all 1:4 then the average number of edges that need to be checked to determine whether a particle has moved out of the cell is 2. To find out whether a particle is inside or outside the next reference cell, the three edges which are not shared with previous cell need to be checked. Assuming, again, equal probabilities for the particles traversing the three sides into the next cell, the average number of cells which need to be checked in the new and any subsequent cell is 1.5.

In Chen's scheme, four edges must be considered in order to compute both ΔI and ΔJ for every cell during any one searching step. Therefore, for those cases where only the adjacent cell is entered by the particle during one tracking time step (i.e. either ΔI or ΔJ is zero and neither is greater than 1), $8(=4+4)$ edges must be computed, while in our algorithm, an average of $5(=2+3)$ edges need to be calculated. For the case where both ΔI and ΔJ are 1, the improved algorithm needs to compute an average of $6.5(=2+1.5+3)$ unit integer variables, while in Chen's scheme, the number of edges to be calculated is either $8(=4+4)$ or $12(=4+4+4)$. Hence, the present algorithm is substantially more efficient in terms of both the checking and searching processes.

4. Extension to three-dimensional space

Although the new algorithm is presented in the context of a structured 2D quadrilateral grid, it can be applied readily within a 3D structured or unstructured grid system. In case of 3D flow, every control volume consists of a larger set of surfaces which separate neighbouring

control volumes, and each surface is composed of a set of vertices. If the vertices comprising a particular surface under consideration are in the same plane, no further sub-division of the surface is needed; otherwise, the surface needs to be sub-divided until all the vertices in sub-surfaces are in the same plane. Hence, the task of determining whether a particle is inside or outside a given cell consists of identifying whether the particle is located on the correct side of a set of plane surfaces which make up the control volume. By using anti-clockwise ordering of the vertices, which define the plane surface, and the right-hand system, as indicated in Fig. 7, the L parameter for surface ACB of the control volume ABCD is as follows:

$$L_{\mathrm{acb}} \equiv (\mathbf{r}_{\mathrm{AC}} \times \mathbf{r}_{\mathrm{CB}}) \cdot \mathbf{r}_{\mathrm{BP}}, \tag{16}$$

where P is the position of the particle under consideration and vertices A, B, C and D form the 3D cell. Vector $\mathbf{r}_{\mathrm{MN}}$ is that directed from point M to point N (M or N = A, B, C, P, ...).

5. Verification

This section compares the performance of the present algorithm with that of Chen by reference to two specific examples. The first example is one that has been considered by Chen and thus offers a convenient means of a direct comparison. Given the condition shown in Fig. 8, the task is to find the index of the cell containing a particle at $t + \delta t$ after it has moved from its original reference cell at time t. For convenience, this cell is identified by the same index (i, j) of its south-west vertex. The first unit integer encountered in the reference cell (i, j) is $J_{\mathrm{s}} = -1$, which indicates that the new reference cell is $(i, j-1)$. For $(i, j-1)$, the second integer is $I_{\mathrm{e}} = -1$, which suggests the new reference cell to be $(i + 1, j-1)$. Then a further check for $(i + 1, j-1)$ indicates its second unit integer to be $I_{\mathrm{e}} = -1$, which leads to the updated reference cell $(i + 2, j-1)$. After checking all the three unit integers J_{s}, I_{e} and J_{n} (there is no need to check I_{w}), the method confirms that it is the cell $(i + 2, j-1)$ that contains the particle at $t + \delta t$. The total number of unit integers computed during this trial-and-error searching process is $8(= 1 + 2 + 2 + 3)$. The search path is identified by the dotted line in Fig. 8. With Chen's scheme, the search path is indicated by the broken line. Following the first

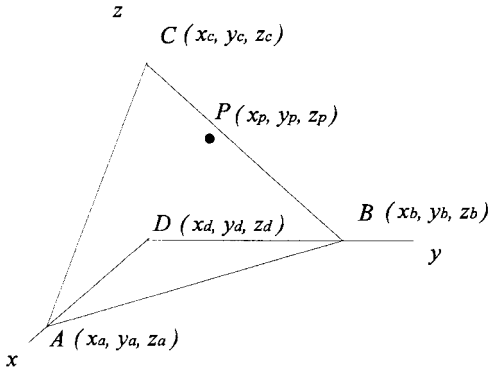

Fig. 7. Three-dimensional cell ABCD used for deriving particle position P.

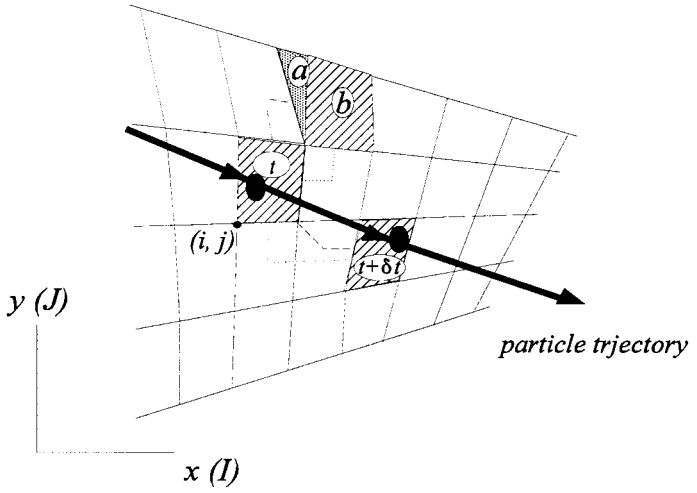

Fig. 8. Searching path for the particle moved from t to $t + \delta t$.

check, the reference cell moves from (i, j) to $(i + 1, j-1)$ and then to $(i + 2, j-1)$. Therefore, the total number of unit integers calculated is $12(=3 \times 4)$. Hence, in this example, a saving of 33% is achieved in the computational effort.

Assume, as a variation to the forgoing example, that at time $t + \delta t$, the particle has moved from the cell (i, j) to $(i + 1, j + 1)$ consisting of the shaded/hatched areas (a) and (b), as indicated in Fig. 8. If the present algorithm is used, the searching path will be from (i, j) to $(i + 1, j)$ and then to $(i + 1, j + 1)$. The total number of unit integer variables calculated is $8(=2 + 3 + 3)$. If Chen's scheme is used, there are two possibilities. First, if the particle lies in area (a), the search path is from (i, j) to $(i, j + 1)$ and then to $(i + 1, j + 1)$. Second, if the particle is in area (b), the search path progresses directly from (i, j) to $(i + 1, j + 1)$. Therefore, there are, respectively, $12(=3 \times 4)$ or $8(=2 \times 4)$ calculations of unit integers required. Hence, at worst, no saving arises, while a 33% saving may arise again.

Ultimately, the value of a particle-locating scheme is dictated by the CPU effort it demands. This will be addressed in the second application. Here, a particle is tracked within a carrier-flow executing solid-body rotation with the angular velocity, ω, fixed at unity. The same problem has been studied by Barton (1996) and Ruetsch and Meilburg (1993), and its attractiveness lies partly in the availability of an analytic solution for the particle's outward spiralling motion associated with its inertia. The particle is released at $(x, y) = (0, 1)$ and has the velocity components $(u, v) = (1, 0)$. The fluid velocity is known at any given position (x, y) in the flow field and can be described as $(\omega y, -\omega x)$. The exact solution for the particle path as a function of time is an exponentially increasing spiral whose shape depends on the particle's relaxation time τ (see appendix of Barton, 1996). The analytical solution circumvents the numerical computation of the particle-motion equations, and the CPU costs associated with the particle-locating scheme can, therefore, be studied in isolation.

Fig. 9 shows the analytical solutions of three different particle trajectories for the respective values of relaxation times $\tau \equiv 4/3\ \rho_p D_p/\rho_f C_D U_r = 100.0$ [s], 1.0 [s] and 0.01 [s] within the time

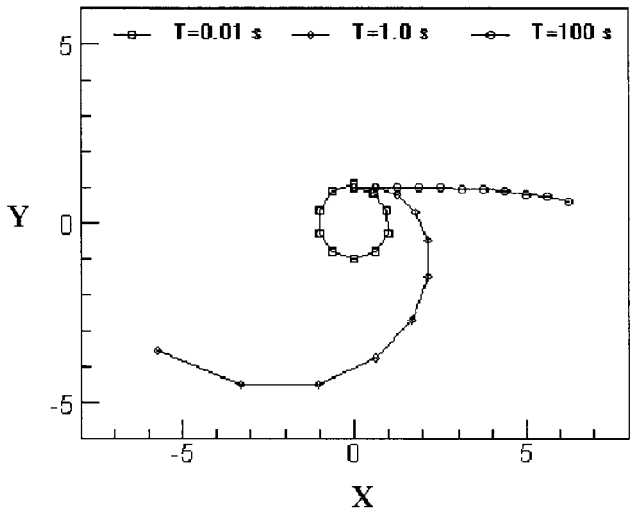

Fig. 9. Trajectories of three particles with different relaxation times within solid-body rotation between $t = 0$ and $t = 2\pi$ s.

range $0 < t < 2\pi$ [s]. The time interval can be divided into any given time steps (10 steps in the present test) and the (x, y) positions of the particle at every time step can be obtained directly from the analytical solution. These positions are identified in Fig. 9 by the open symbols.

To test the efficiency of the algorithm, a grid must be super-imposed upon the flow field in the figure. Hence, the CPU time used for identifying all the cells which contain the particle at the marked trajectory positions as the particle moves along the trajectory is recorded and used as a measure of efficiency.

Two types of structured grid, one Cartesian and the other intentionally distorted, both shown in Fig. 10, have been used. For each mesh type, three densities, 20×20, 40×40 and 60×60, have been examined. Here attention is restricted to structured grids merely to enable comparison with Chen's scheme. Both Chen's scheme and the present method give identical results for the cell indices associated with the particle positions at different time steps for all three grid densities, and results for the 20×20 and 40×40 Cartesian grids are listed in Table 1. Particular attention is drawn to the fact that, in the case of the finer mesh, successive particle positions repeatedly change by more than one cell distance, which makes the searching task more taxing than that in which the particle moves by less than one cell distance in any one time step. Full agreement between the two schemes is, of course, an essential result and merely verifies that both tracking algorithms are correctly implemented. Corresponding agreement has also been achieved for the distorted grid of Fig. 10.

A comparison of computational efficiency is provided in Table 2 for the distorted grid in Fig. 10. The CPU time used for identifying the 33 marked positions (including the initial starting positions) along the trajectories in Fig. 9 is too low to be recorded reliably. Therefore, the computation of the particle searching and locating operations was repeated 100 times on a PC (166 MHz Pentium with 64 Mb RAM), and the resulting CPU time is that recorded in Table 2.

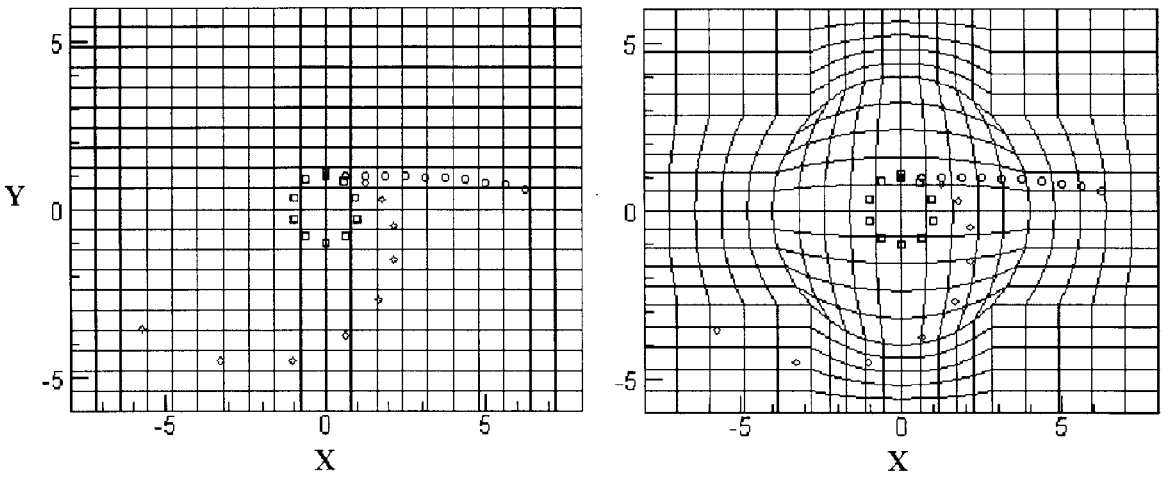

Fig. 10. Grid topology used for testing particle-tracking algorithm.

As seen, the present method is significantly more economical than Chen's scheme. The reduction in computational effort is around 30%, a value consistent with the levels given previously. There is also a tendency for the savings to increase slightly with increasing grid density, both as a consequence of the relative decrease in computational overheads not directly associated with the search and the fact that the present scheme is particularly efficient when successive particle positions span more than one cell, a condition arising here with increasing frequency as the grid is refined.

Table 1
Cell indices corresponding to particle location at different time levels, Cartesian grid

τ	0.01 (s)		1.0 (s)		100.0 (s)		Time step:
Grid	20×20	40×40	20×20	40×40	20×20	40×40	2π (s)
Grid indices (i, j) at associated time step	(10,12)	(20,24)	(10,12)	(20,24)	(10,12)	(20,24)	0.0
	(11,12)	(22,23)	(11,12)	(22,24)	(11,12)	(22,24)	0.1
	(12,11)	(23,22)	(12,12)	(24,23)	(12,12)	(24,24)	0.2
	(12,10)	(23,19)	(13,11)	(25,21)	(13,12)	(25,24)	0.3
	(11,9)	(22,18)	(13,10)	(26,19)	(14,12)	(27,24)	0.4
	(11,9)	(21,17)	(13,8)	(26,15)	(14,12)	(28,24)	0.5
	(10,9)	(19,18)	(13,6)	(25,11)	(15,12)	(30,24)	0.6
	(9,10)	(18,19)	(11,4)	(22,8)	(16,12)	(31,23)	0.7
	(9,11)	(18,22)	(9,3)	(18,5)	(17,12)	(33,23)	0.8
	(10,12)	(19,23)	(6,3)	(12,5)	(18,12)	(35,23)	0.9
	(10,12)	(20,24)	(3,5)	(6,9)	(18,11)	(36,22)	1.0

Table 2
CPU time of particle-locating algorithms for distorted grid in Fig. 10

	CPU time used for 3300 search operations (ms)		
Scheme used	20×20 grid	40×40 grid	60×60 grid
Chen's scheme	220	330	490
The present scheme	160	220	320

6. Concluding remarks

Chen's particle-locating algorithm has greatly improved the efficiency of searching and locating particles in Eulerian–Lagrangian computations relative to earlier methods. The present scheme, developed independently, rests on a different search methodology and is competitive with Chen's scheme, in terms of both CPU time and simplicity. The method applies to 2D as well as 3D conditions, and imposes no restrictions on the mesh topology. It is currently employed by the authors in the context of computations of particle dispersion in turbulent flows in conjunction with Reynolds-stress modelling and an anisotropy-sensitized particle-dispersion model.

References

Barton, I.E., 1996. Exponential–Lagrangian tracking schemes applied to Stokes law. Journal of Fluid Mechanics 118, 85–89.

Chen, X-Q., 1997. Efficient particle tracking algorithm for two-phase flows in geometries using curvilinear coordinates. Numerical Heat Transfer 32A (4), 387–405.

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

Guzman, M.M., Fletcher, C.A.J., Behnia, M., 1995. Gas particle flows about a cobra probe with purging. Computers and Fluids 2 (2), 121–134.

Lambert, J.D., 1973. Computational Methods in Ordinary Differential Equations. Wiley, New York.

Oliveira, P.J. 1995. Computer modelling of multiphase flow and application to T-junction. Ph.D. thesis. Department of Mechanical Engineering. Imperial College, London, UK.

Ruetsch, G.R., Meilburg, E., 1993. On the motion of small spherical bubbles in two-dimensional vortical flows. Physics of Fluids A 5, 2326–2341.

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., 1996. Modelling particle dispersion in anisotropic turbulence. Computational Fluid Dynamics '96 Proceedings of the 3rd European Computational Fluid Dynamics Conference (ECCOMAS), Wiley, pp. 577–583.

Zhou, Q., Leschziner, M.A., 1997. Modelling particle dispersion in turbulent recirculating flow with an anisotropy-resolving scheme. ASME, FEDSM97-3643.