Journal of Computational Physics 228 (2009) 9001–9019

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

An improved known vicinity algorithm based on geometry test for particle localization in arbitrary grid

Ke Peng [a,*,1], Zhang Shuguang [a], Wu Jianghao [a], Yang Chunxin [b]

[a] School of Transportation Science and Engineering, Beihang University, Beijing 100191, PR China
[b] School of Aeronautical Science and Engineering, Beihang University, Beijing 100191, PR China

ARTICLE INFO

Article history:
Received 30 December 2008
Received in revised form 5 August 2009
Accepted 4 September 2009
Available online 13 September 2009

MSC:
65K10
76T10
76M28

PACS:
47.55.Ca
47.55.Kf

Keywords:
Particle localization
Eulerian–Lagrangian simulation
Known vicinity algorithm
Binary search
Unstructured grid

ABSTRACT

The known vicinity algorithm based on the geometry test for the particle localization problem in the hybrid Eulerian–Lagrangian model was extended and enhanced aiming at the connected grids with convex polygon/polyhedral cells. Such extensions were achieved by proposing novel improvements. Specifically, a new "side function", to determine the relative position of the particle and the cell, was introduced to build a more formal test process. In addition, a binary search method was developed to accelerate the particle in cell test and trajectory/face intersection test for grids consisting of arbitrary polygon/polyhedral cells. Further, the particle location problem without the known vicinity position was established and solved by special boundary treatment through considering the internal/external boundary and larger particle displacement in one single Lagrangian step. The improved algorithm was applied to the particle location problem with both two dimensional and three dimensional Eulerian grids. Additionally, the proposed algorithm was compared with the previous ones to exhibit its higher efficiency and broader application. Sample cases focusing the water impingement computation for aircraft icing were solved by adopting this algorithm assisted by the Lagrangian particle dynamics model, and the computational results were verified by the experiments.

1. Introduction

The hybrid Eulerian–Lagrangian model is widely used for the numerical simulation of two-phase flow and multiphase flow. Using this hybrid model, the mass, momentum and energy of each unit volume of the carrier phase are firstly obtained by solving the partial differential equations with finite volume/difference/element methods, while the position, mass, momentum and energy of each single particles moving in the fixed Eulerian mesh are derived using Lagrangian approach. Furthermore, the fluid properties are stored in the Eulerian mesh and the properties at the particle location must be evaluated prior to solving the ordinary differential equations of the dispersed particle. Therefore, locating the host cell containing each tracked particle, termed as particle localization problem (PLP), is important.

There are two kinds of particle localization problem: those with prior knowledge of a neighboring location, and those without. The first kind was formally defined by Haselbacher et al. [1]: "Given a grid, a particle position, and the cell which

* Corresponding author. Tel.: +86 01 82316627; fax: +86 01 82315237.
E-mail address: p.ke@buaa.edu.cn (P. Ke).
[1] The work is supported by the China Postdoctoral Science Foundation under No. 20080440012.

0021-9991/$ - see front matter Crown Copyright © 2009 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.09.003

contains that particle position, determines the cell which contains a nearby particle position." The second one could be characterized as: Given a grid and a particle position, to locate the cell containing that particle position.

Usually, the host cell can be determined quickly using the particle location and grid mapping for uniform Cartesian grids. However, this approach cannot be directly utilized in the unstructured grid. For unstructured grids, Lohner and Ambrosiano [2] listed three most promising methods for solving the particle localization problem: (a) Using a Cartesian background grid to superimpose the irregular foreground gird on a regular background grid. (b) Using tree structures to circumvent the big grid with some small, different size Cartesian grids in a hierarchy. (c) Using successive neighbor searches. They preferred to the third one and presented a particle tracer for unstructured girds with linear basis shape functions used in finite element methods. Subsequently, Lohner [3] developed a more robust search algorithm, "known vicinity algorithm", which was stemmed from the successive neighbor searcher combined with the brute force or Octree search methods in case that the successive neighbor searcher failed.

This known vicinity algorithm has been widely studied and applied to the first kind of the PLP due to its conciseness and high efficiency. Li and Modest [4] introduced an "element-to-element" search scheme, similar to the know vicinity method, to perform hybrid finite volume PDF Monte Carlo simulations, which implies a wide range of potential applications utilizing this scheme. Apte et al. [5] employed this algorithm to track the particle on unstructured hexahedral grids when developing the large eddy simulation of particle-laden, swirling flow in a coaxial-jet combustor. Petera et al. [6] applied the similar method to the tetrahedral mesh to model a dispersion of electrically charged droplets in motion inside a second immiscible liquid phase continuum in the presence of an external electric field. Widhalm et al. [7] adopted it to determine the droplet trajectories on Euler grids in order to predict the water impingement and ice accretion on the aircraft.

There are two key issues in the known vicinity algorithm: the particle in cell (PIC) test, which is used to determine a cell hosting a given particle position or not, and the neighbor selection, which is used to choose the next cell to search in case that the particle is not in current cell. Many researchers have performed great work and built a variety of useful schemes to improve the algorithm's effectiveness, robustness and application of this algorithm.

Judgments in PIC test have also been broadly studied. For example, Lohner [3] discussed the linear basis shape functions on triangle to search for cells holding a given particle, and pointed out that the elements must be split into sub-triangles or tetrahedra when the nodes are more than shape equations. Widhalm et al. [7] adopted such methods and implemented for each element type supported by their unstructured Navier–Stokes solver TAU, respectively. They then used the local coordinates deduced from shape function to interpolate the flow properties to the particle position. Westermann [8] presented three localization schemes for PIC test in two dimensional (2D) mono-block structured grids consisting of arbitrary convex four-point cells, which were based on calculation of the areas, or simplices, or an interpolation scheme. Li and Modest [4] introduced a new judgment, which was the ratio of the particle trajectory normal to a face to the normal distance between the old particle location and the face. These PIC test methods are elaborated, but complex to practically implement and heavy computational burden to reach the judgements.

To reduce the computational time, more straightforward and prominent methods [1,9–13] based on the geometry relationship test were developed for the PIC test. In detail, Chen [9], Chen and Pereira [10] found that a particle must be inside the volume when the particle and the central reference point lay at the same side of all the boundary side of an control volume, and then proposed a new PIC test approach. It is efficient for body-fitted curvilinear coordinates and can be used when the particle moved many Eulerian cells among one Lagrangian tracking time step. A similar scheme was independently developed by Zhou and Leschziner [11]. They employed a counter-clockwise definition to avoid the auxiliary interior reference point, and named it as particle-to-the-left (P2L) test to conduct the PIC test. Their method was applicable to any coordinate system. Chorda et al. [12] adopted these methods and extended to three dimensional (3D) grids. Haselbacher et al. [1] enhanced its robustness and efficiency to consider arbitrary polyhedral cells and large particle displacements, and dealt with the interaction of particles with boundaries by reflection.

As to the neighbor selection schematic in the known vicinity algorithm based on geometry relationship test, all neighbor cells were searched firstly [8,10] under the assumption of small particle displacement, such as the circular search. Nevertheless, it was inefficient for the bigger Lagrangian steps, where the number of the neighbor cells increased dramatically with bigger time step. Besides that, Lohner [3] selected the element adjacent to the face opposite the node with minimum shape function value. Apte [5] chose the successive face-neighboring cells where the distance between the centroid of a cell and the new particle location was minimized. Consequently, they used an exhaustive search to ensure that the particle can be located in case successive neighbor search failed. Coppola [14] proposed a hybrid approach to advance a particle in both the physical and the parametric space without requiring nonlinear iterations for the high-order elements to solve the shape function.

Zhou and Leschziner [11] selected the cell adjacent to the first face that failed the P2L test. Although such selection mode was very simple, it may result in worse performance under certain conditions [12]. Martin et al. [13] used the maximum positive dot product method to identify the adjacent cell, and showed its advantage by comparing it with P2L test. Chen and Pereira [10] computed the intersection of the trajectory and the faces failing the same side test, and determined the new cell by the face with internal intersecting point, which was chosen by the comparison of the distance to the initial particle position. Haselbacher et al. [1] tracked a particle along its trajectory by computing the intersections of the trajectory and the cell faces, planar or non-planar, similarly to the one designed by Chen and Pereira [10]. Chorda et al. [12] proposed trajectory-to-the-left (T2L) test to determine the intersection of the trajectory and face, and then extended it into 3D grids. This method was much efficient as the result taking advantage of P2L test and avoiding the complex intersection computation.

In summary, the known vicinity algorithm based on geometry relationship test was well studied for the first kind PLP and proven to be efficient. But it has not been applied to the second kind PLP, where the initial cell was unknown, which sometime is hard to be located. For example, in the computation of water droplet impingement in numerical simulation of aircraft icing, the droplets need to be released from some upwind positions, which are a little far away from the wall face to be considered, such as the wing or stabilizer. Usually the cell of the Eulerian grid containing the release position is unknown, although the release position can be generated.

Furthermore, two problems need further considerations. Specially, the first problem is the boundary problem. For the greater particle displacement and the longer search path, the internal boundary, such as unreachable cells or holes, and the external boundaries in grids with complicate shape, may encounter on the search path. Currently the known vicinity algorithm needs to turn to other auxiliary means, such as reflection computation [1], which is efficient but might face difficulties when the predicted trajectory parallels the normal vector of the centroid of the boundary face, or brute force [2–4], which is exhaustive and time-consuming, or Octree [2,3,13], which needs more additional work to build and store the complex data structures for the Eulerian grid. The second problem is the efficiency of PIC test and the neighbor selection, where all faces of a cell must be test currently to ensure that the particle is really in a given cell. It will be a burden when the number of the face or edge of the polygon/polyhedral cell increases.

The goal of this paper, therefore, is to extend and improve the current widely used known vicinity algorithm targeting both kinds of the particle localization problem mentioned above. The proposed algorithm was based on the geometry test and could be applied to both 2D and 3D grids. It took into consideration the initial cell and boundary problem, and developed a binary search method to speed up the PIC test or neighbor selection for grids consisting of arbitrary polygon/polyhedral cells.

The remainder of this article is arranged as follows: Section 2 describes the extensions and improvements in detail, including the basic notations and the introduction to the algorithm. Section 3 shows the implementation of this algorithm and compares it with previous ones. Section 4 demonstrates the algorithms' applications combined with the particle dynamics models in the numerical simulation of aircraft icing. The conclusions are drew in Section 5.

2. Descriptions of the new particle localization algorithm

In this study, we focused in the connected grids with convex, structured or unstructured, cells. The connected grid was defined by Bredon [15] as: "the grid is not the disjoint of two nonempty open sub-grids.". Accordingly, a particle can move from one cell to any other cells through a serial of adjacent cells in such grid without going out of the boundary. All terms involved in this section will be defined in Appendix A.

The notations including the representation of the cell and face will be introduced firstly.

2.1. Notations

In any Eulerian grid, a cell C consists of at least three faces and can be represented by its face as, $C(F) = \{F_i | i = 1, 2, \ldots, n_F,\ n_F \geqslant 3\}$, where n_F is the number of face. Any face has at least two nodes and can be denoted by nodes as $F(N) = \{N_i | i = 1, 2, \ldots, n,\ n \geqslant 2\}$, or just $F = N_1 N_2 \cdots N_n$, where n is the node number of the face, and N_i is the node of the face. Particularly, in 2D grid, the nodes number of a cell is equal to the number of its face, so the cell can be expressed by the nodes directly, i.e., $C(N) = \{N_i | i = 1, 2, \ldots, n,\ n \geqslant 3\}$.

The reverse face, denoted by F^-, represents the face which has the same nodes of F, but those nodes are sorted in reverse order. For example, the face $F = N_1 N_2$ in 2D grid has the reverse face $F^- = N_2 N_1$. The position of node N_i is expressed as (x_i, y_i) for 2D Cartesian coordinates, or (x_i, y_i, z_i) for 3D Cartesian coordinates.

The nodes of all faces or cells must be sorted in the same order, either clockwise or counter-clockwise. The nodes in 2D grid sorted in counter-clockwise are shown in Fig. 1(a)–(c); all nodes in 3D grid sorted in counter-clockwise are demonstrated in Fig. 1(d), where the normal vector of the face computed by the right-hand rule points to the cell inner. The clockwise order was defined reversely.

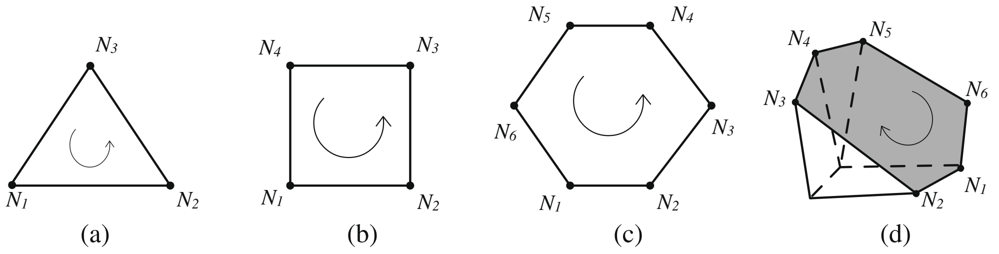


Fig. 1. Sample faces with nodes sorted counter-clockwise.

2.2. Algorithm overview

To locate the target position P' in a given Eulerian grid for both kinds of PLP, several key steps of the current improved algorithm were given followed by its detailed discussion. A flowchart (Fig. 2) depicting the current algorithm was given.

(1) Initiate the current cell C by the presetted starting cell for the first kind of PLP, or by an arbitrary guess from the current grid for the second kind; and then take its centroid P as the current position. It is possible that the guessed position P is far away from the target position P' for the second kind of PLP.
(2) Perform the PIC test to determine whether the P' is in the cell C or not. The localization process will succeed when PIC test returns true. Otherwise, the procedure turns to step (3).
(3) Conduct the TFI (trajectory/face intersection) test to find the exit face F intersecting the trajectory PP' from the faces where the PIC test in step (2) failed.
(4) Select the exit face by considering the special boundary treatment, which will be discussed in detail in Section 2.3.5. If there is any suitable face F, the procedure turns to step (5). If not, the localization process fails.
(5) Identify the cell C' sharing face F with the current cell C, reset the new cell C using the cell C'. Afterward, the process returns to step (2).

2.3. Details of the algorithm

In this section, the algorithm will be discussed in details, where the side function is the core of the PIC and TFI test.

2.3.1. Side function

A side function $\Phi(P,F)$ was defined to indicate the position relationship of the given point P to the given face F based on geometry test, which came of the P2L test [11] and was used to conduct both the PIC and TFI test. Besides, the side function has general expressions for both 2D and 3D case, which is more formal to formulate the test process. For any point lying at the same side of the face, it will result in the same side value, which is the result generated by the side function.

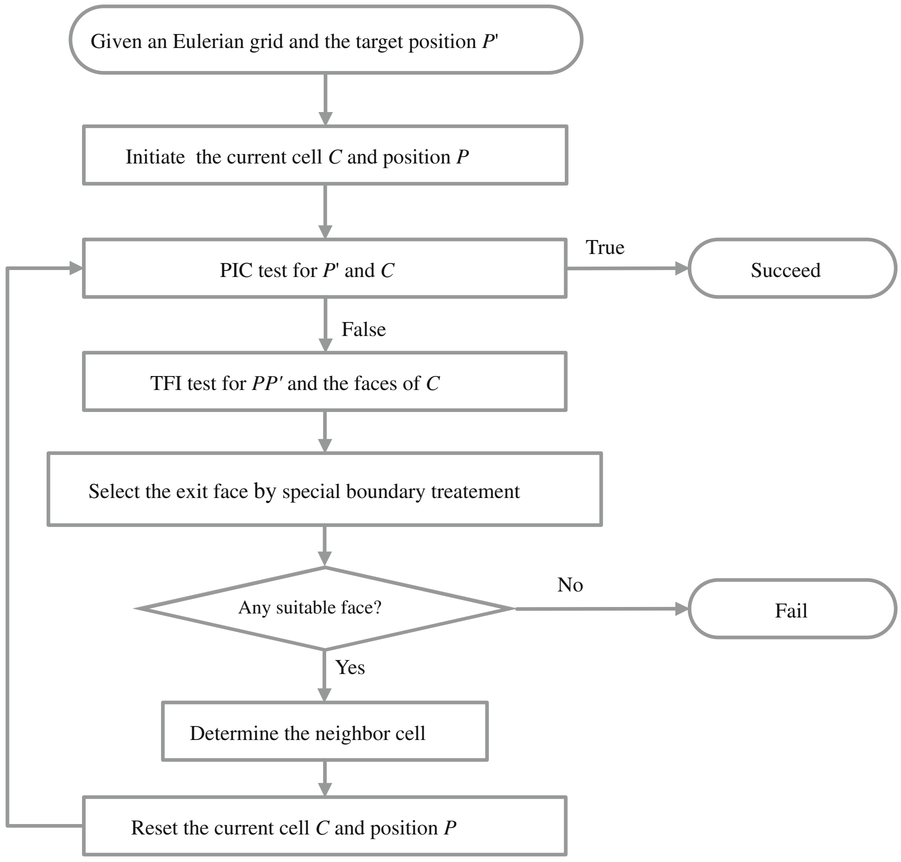

Given an Eulerian grid and the target position P'
Initiate the current cell C and position P
PIC test for P' and C
True
Succeed
False
TFI test for PP' and the faces of C
Select the exit face by special boundary treatement
Any suitable face?
No
Fail
Yes
Determine the neighbor cell
Reset the current cell C and position P

Fig. 2. Flowchart of the improved algorithm for the particle location problems.

As shown in Fig. 3, for a given face $F(N) = \{N_i | i = 1, 2, \ldots, n,\ n \geqslant 2\}$ and a given position P, the side function was defined as,

$$\Phi(P, F) = sign[\overline{N_1P} \cdot (\overline{N_1N_2} \times \overline{N_1N_3})], \tag{1}$$

where $N_1,\ N_2,\ N_3$ were the arbitrary three different nodes sorted counter-clockwise. For 2D face, the vector $\overline{N_1N_3} = (0, 0, -1)$. The $sign(a)$ was the signature function for scalar a, and defined as,

$$sign(a) = \begin{cases} +1, & a \geqslant 0, \\ -1, & a < 0. \end{cases} \tag{2}$$

Given a 2D face using Cartesian coordinate

$$F = \{N_1(x_1, y_1), N_2(x_2, y_2)\}$$

and a point $P(x, y)$ demonstrated in Fig. 3(a), the side function can be expanded as:

$$\Phi(P, F) = sign[(x_2 - x_1) \cdot (y - y_1) - (x - x_1) \cdot (y_2 - y_1)]. \tag{3}$$

Examples are shown in Fig. 4 for the face with counter-clockwise sorted nodes, where $\Phi(P, F) = \Phi(P_1, F) = +1$, and $\Phi(P_2, F) = -1$. The P and P_1 lie at one side of face F, while P_2 lies at another side.

As to the reverse face, it can be derived as follows,

$$\Phi(P, F^-) = -\Phi(P, F). \tag{4}$$

Eq. (4) is useful for the binary search method in PIC and TFI test described below.

2.3.2. PIC test

As introduced by Zhou and Leschziner [11], one particle lies in a cell only when it lies at the same side of all the faces forming that cell, that is, all side functions return the same side value. If any face fails to yield the same side value as others, the particle must be outside the corresponding cell. Previously, all faces were tested one-by-one to ensure the position is really in a cell. Such process is exhaustive and time-consuming. A binary search method will be proposed to accelerate the PIC test, where the cell faces were not tested directly. Only those consisted of pairs of diagonal nodes were tested.

2.3.2.1. Basics of the binary search method with side function. For a 2D cell, supposed a given cell $C(N) = \{N_i | i = 1, 2, \ldots, n,\ n \geqslant 3\}$ and a given position P, as displayed in Fig. 6(a) as a example. If there are three nodes $N_i,\ N_{j_1},\ N_{j_2},\ i < j_1 < j_2,\ i,\ j_1,\ j_2 \in [1, n]$, and $\Phi(P, N_iN_{j_1}) = \Phi(P, N_{j_2}N_i) = +1$. The conclusion can be reached that the position P lies in the region between face $N_iN_{j_1}$ and $N_{j_2}N_i$. Thus, there will be two possible side value for any node $N_k,\ k \in (j_1, j_2)$:

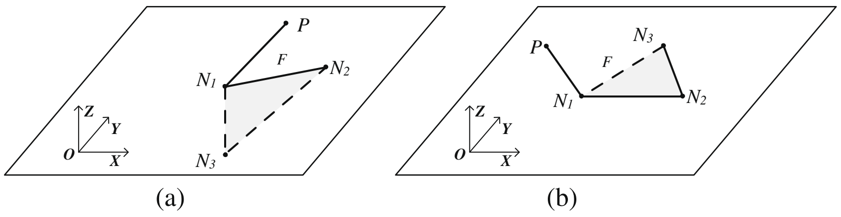


Fig. 3. Demonstration of the side function definition: (a) 2D and (b) 3D.

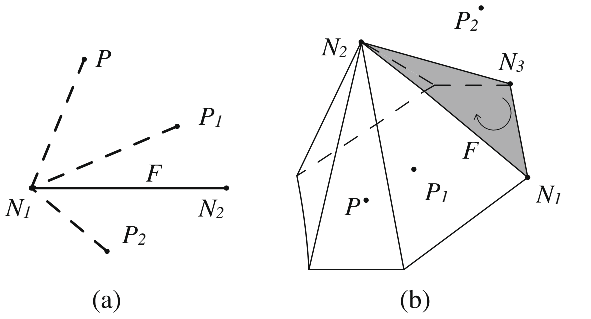


Fig. 4. Demonstrations of the side function application: (a) 2D and (b) 3D.

Case (a): $\Phi(P, N_kN_i) = +1$. Therefore, it can be concluded without further computations that $\Phi(P, N_tN_i) = +1$ for any node N_t, $t \in (k, j_2)$.
Case (b): $\Phi(P, N_kN_i) = -1$, that is, $\Phi(P, N_iN_k) = +1$. Similarly, no more computations are needed. It is true that $\Phi(P, N_iN_{t'}) = +1$ for any node $N_{t'}$, $t' \in (j_1, k)$.

In this way, more side values can be derived directly by one test. Specially, relationships of this position with more faces could be determined directly. Hence, the number of the remainder faces which require test decrease sharply after each test.

2.3.2.2. Binary search method in PIC test. A quick binary search method for PIC test based on the above idea was developed. Taking $i = 1,\ j_1 = 2$ and $j_2 = n$, the detailed process of the binary search method was listed below. The corresponding flowchart is given in Fig. 5.

Step (1). Compute $\phi_{f12} = \Phi(P, N_1N_2)$ and $\phi_{fn1} = \Phi(P, N_nN_1)$. If $\phi_{f12} = \phi_{fn1} = +1, P$ must lie at the same side of the face N_1N_2 and N_nN_1. Then the process turns to step (2); Otherwise P must lie outside the cell C; and the output of the PIC test is false.
Step (2). Turn to step (3) if $n > 3$, or compute $\phi_{f2n} = \Phi(P, N_2N_n)$ in order to find the position relationship between P and face N_2N_n if n is less than 3. If $\phi_{f2n} = +1$, implying that P lies in the cell C. The PIC test then returns true. Or else, the PIC test outputs false.
Step (3). Select the middle node N_m, which divides the cell C into two sub-cells C_1 and C_2. Then compute $\phi_{fm1} = \Phi(P, N_mN_1)$ to test the geometry relationship of P and the face N_mN_1. If $\phi_{fm1} = +1$, the point must be outside the sub-cell C_2. Consequently, only sub-cell C_1 needs more PIC tests. Otherwise, the determined cell left C_2. Then, it is required to re-index the nodes if needed. Afterward, the process turns to step (2), where $C_1 = \{N_i | i = 1, 2, \ldots, m\},\ C_2 = \{N_i | i = 1, m+1, m+2, \ldots, n\}$

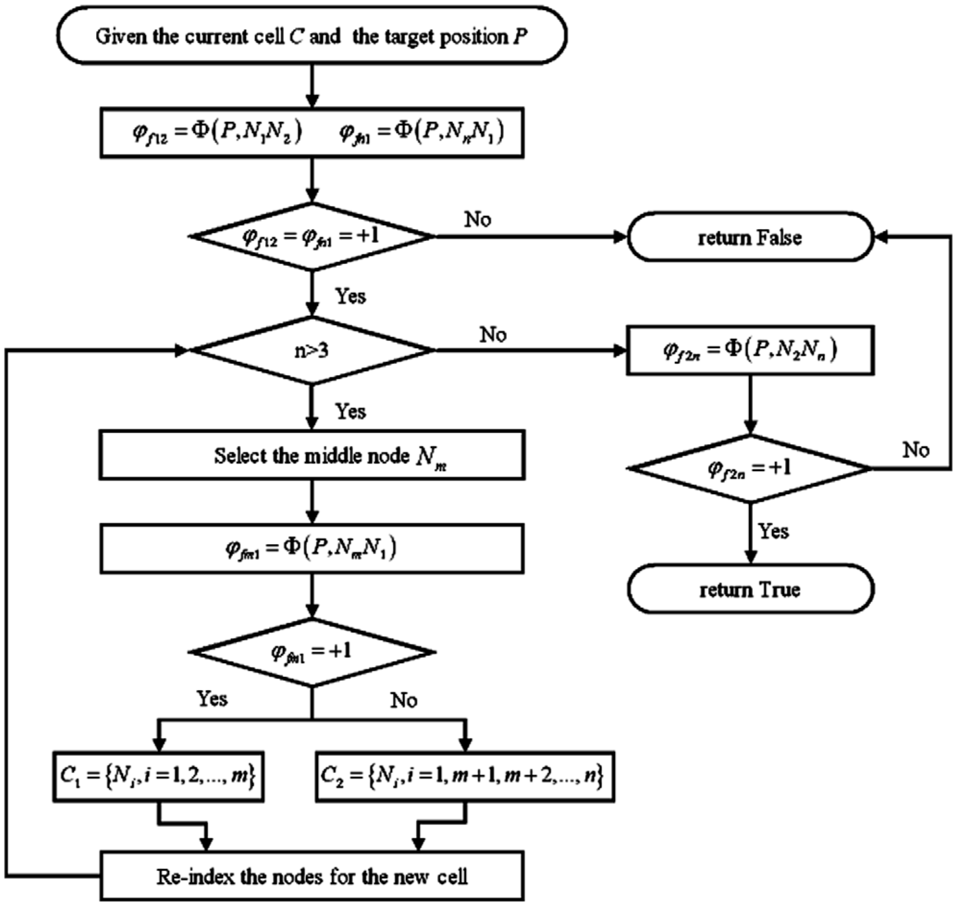

Given the current cell C and the target position P
$\varphi_{f12} = \Phi(P, N_1N_2)$ $\varphi_{fn1} = \Phi(P, N_nN_1)$
$\varphi_{f12} = \varphi_{fn1} = +1$
No
Yes
return False
n>3
No
$\varphi_{f2n} = \Phi(P, N_2N_n)$
Yes
Select the middle node N_m
$\varphi_{f2n} = +1$
No
Yes
$\varphi_{fm1} = \Phi(P, N_mN_1)$
return True
$\varphi_{fm1} = +1$
Yes
No
$C_1 = \{N_i, i = 1, 2, \ldots, m\}$
$C_2 = \{N_i, i = 1, m+1, m+2, \ldots, n\}$
Re-index the nodes for the new cell

Fig. 5. Flowchart of the PIC test with the binary search method.

$$m = \begin{cases} n/2, & n \equiv 0 \pmod 2, \\ (n \pm 1)/2, & n \equiv 1 \pmod 2. \end{cases} \tag{5}$$

An example was provided in Fig. 6(b), where only six tests are needed to finish the PIC test in order to assure the point to lie in an octagon cell, while there needs eight tests by the previous methods. For the 3D grids, the binary search becomes too complicated for PIC test. Therefore it is employed for the TFI test, which would be described in next section.

2.3.2.3. Performance analysis. Generally, the average testing time demanded by the binary search method for a polygon cell with n faces (that is, n nodes) is at the level of $\log_2 n$, while the previous exhaustive methods are at the level of n. Furthermore, the exact testing times can be calculated by

$$f(n) = f(m+1) + 1, \tag{6}$$

where $n > 4$, m is given in Eq. (5) and the initial value is $f(3) = 3$ and $f(4) = 4$.

Partial results of Eq. (6) are shown in Fig. 7 for the cells with less than 60 nodes and those more reasonable with no more than 10 nodes, where the worst condition was defined as m holding the value of $(n+1)/2$. Conversely, the best condition was defined as m with the value of $(n-1)/2$. As exhibited in Fig. 7, the binary search method is expected to be more efficient than the exhaustive ones, which will be demonstrated by more cases in Section 3. Furthermore, the binary search methods is much easy to be implemented than other efficient methods, such as Octree or split-tree, while the efficiency of the proposed algorithm is comparable to them under a real Eulerian mesh.

2.3.3. TFI test

TFI test was defined as the intersection test between the particle trajectory and the face, where the trajectory is always from the current particle position to the target position during the search process. It was applied to determine the exit face in order to select the neighbor cell after the PIC test found that the target position lay outside the current cell. Here the side

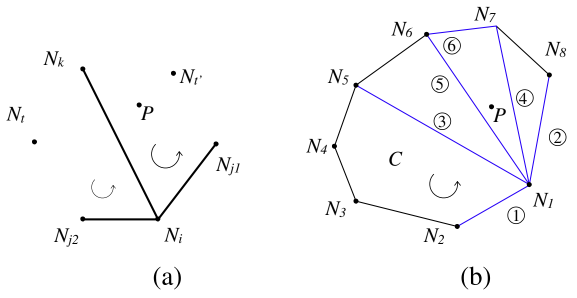


Fig. 6. Demonstrations of the PIC test with binary search method in 2D cell.

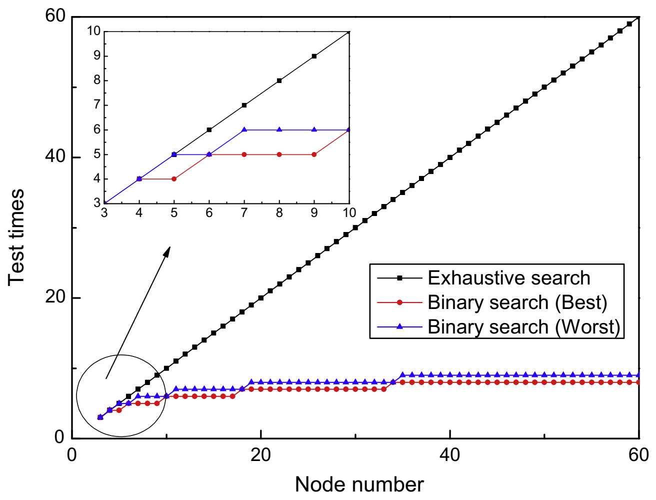


Fig. 7. Comparison of the binary search method with the exhaustive one.

function was used to judge whether lines or line and face intersected with each other, similar as the T2L test [12], to prevent the direct intersection computations. Moreover, the binary search method was integrated to speed up the TFI test for 3D grids.

2.3.3.1. Basics of the TFI test with side function. Suppose that the starting position P is inside the cell C, while the target position P' lies outside the cell C, and a face failed the PIC test of C and P' is represented by $F(N) = \{N_i | i = 1, 2, \ldots, n, \ n \geqslant 2\}$. It can be derived with the side function as,

$$\Phi(P, F) \neq \Phi(P', F). \tag{7}$$

On the other hand, any 2D face (where $n = 2$) will intersect with the trajectory PP' in case of $\Phi(N_1, PP') \neq \Phi(N_2, PP')$, and any 3D face (where $n > 2$) will cross the trajectory PP' if $\Phi(P', N_iPN_{i+1}) = \Phi(P', N_nPN_1)$ for any face $N_iPN_{i+1}, i \in (1, n-1)$.

As displayed in Fig. 8(a), $\Phi(P, F) = +1$, $\Phi(P_1, F) = \Phi(P_2, F) = -1$, thus trajectory PP_1 intersects the face because $\Phi(N_1, PP_1) \neq \Phi(N_2, PP_1)$. While PP_2 do not cross the face when $\Phi(N_1, PP_2) = \Phi(N_2, PP_2)$. In Fig. 8(b), the trajectory PP' intersects the face F, since $\Phi(P', N_1PN_2) = \Phi(P', N_2PN_3) = \Phi(P', N_3PN_1)$.

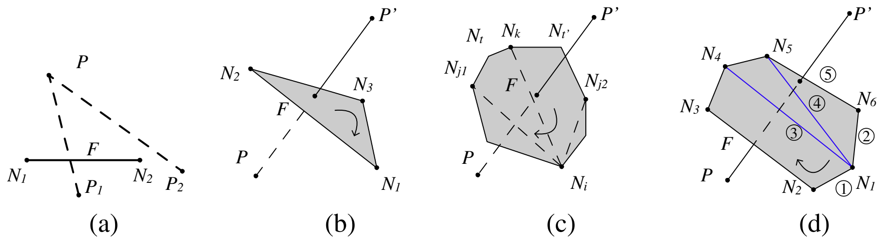
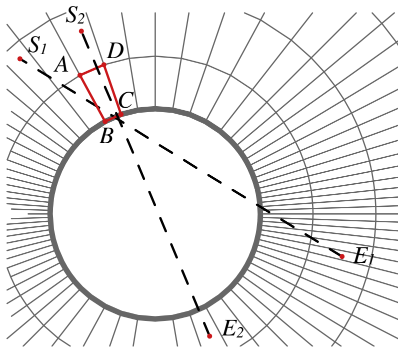
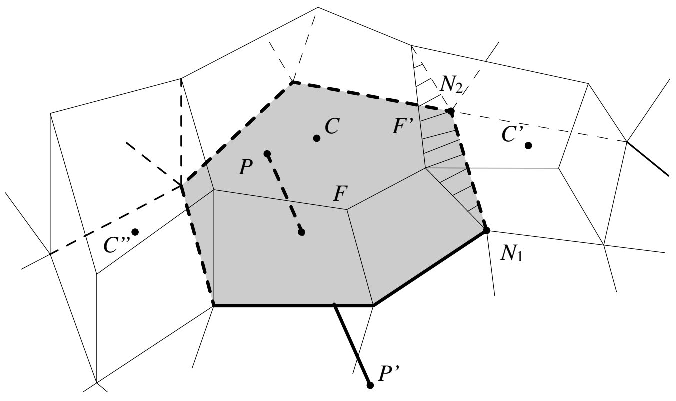


Fig. 8. Demonstrations of the TFI test application.



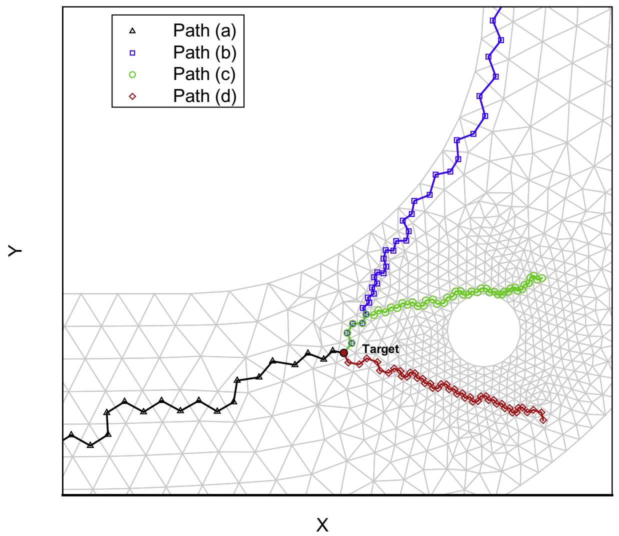
(a) 2D.



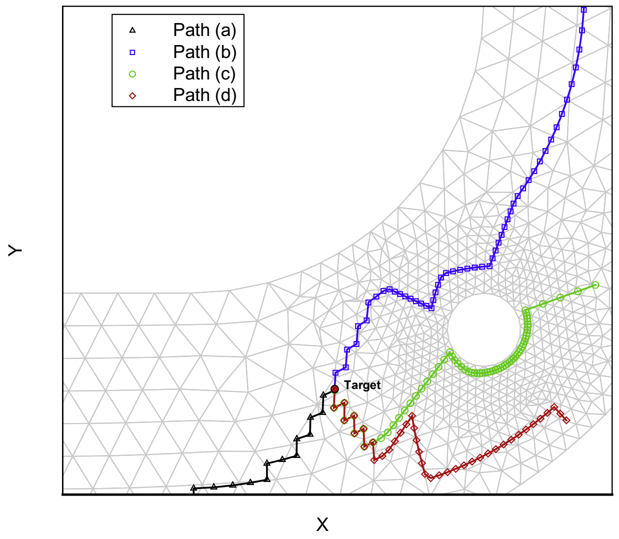
(b) 3D.

Fig. 9. Demonstrations of the special boundary treatment for face selection.

2.3.3.2. Binary search method in TFI test. Following the above descriptions, it is easy to retrieve the face intersecting the trajectory by using the exhaustive search from all faces which failed the PIC test. It is relatively simple for 2D grids, where the failed faces are limited. The exhaustive tests might be acceptable. But it will be more time-consuming for 3D grids, particularly when the failed faces contain many nodes. A quick binary search method was, hence, requested to accelerate the process.



(a) Unstructured grid.



(b) Structured grid.

Fig. 10. Search paths generated by KE.

Table 1
The comparison among the four algorithms.

	PIC	Neighbor selection			Boundary treatment
		TFI	Candidate face selection	Search method	
ZL	P2L	No	The first face failed	Exhaustive	No
CP	P2L	Intersect computation	The crossed face	Exhaustive	No
CH	P2L	T2L	The crossed face	Exhaustive	No
KE	Side function	Side function	The crossed face	Binary search	Yes

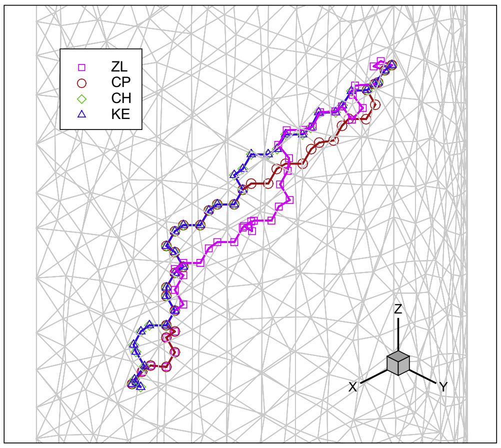
Suppose that $\Phi(P,F) \neq \Phi(P',F)$ for the given face $F = \{N_i | i = 1, 2, \ldots, n,\ n \geqslant 3\}$ and the given positions P and P'. As shown in Fig. 8(c), if there are three nodes $N_i,\ N_{j_1},\ N_{j_2},\ i < j_1 < j_2,\ i,\ j_1,\ j_2 \in (1,n)$ and $\Phi(P', N_i P N_{j1}) = \Phi(P', N_{j2} P N_i) = +1$, there will be two results from the side function computation for any node $N_k,\ k \in (j_1, j_2)$:

Case (a): $\Phi(P', N_k P N_i) = +1$. There is $\Phi(P', N_t P N_i) = +1$ for any node $N_t,\ t \in (k, j_2)$;
Case (b): $\Phi(P', N_k P N_i) = -1$, that is, $\Phi(P', N_i P N_k) = +1$. There is $\Phi(P', N_i P N_{t'}) = -1$ for any node $N_{t'},\ t' \in (j_1, k)$.

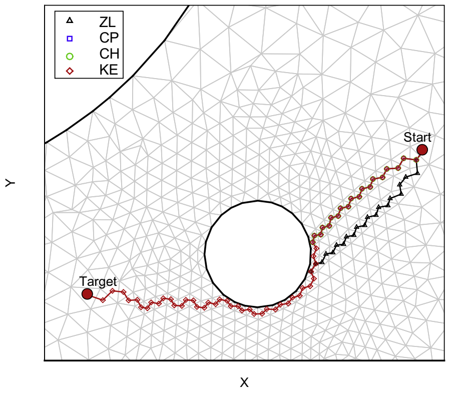
Similar as the binary search method in PIC test, more TFI test results can be derived directly by one test. In this way, one face can be divided into two sub-faces by one test, where only one sub-face need more tests and it has much less nodes than the original one. Consequently, the number of the remaining TFI tests decrease dramatically after each test.

2.3.3.3. Implementation of the TFI test with binary search method. Given the trajectory PP' and the face F with n nodes sorted counter-clockwise, as shown in Fig. 1(d), a quick binary search method to accomplish the TFI test according to the above theorem could be developed. By assigning $i = 1,\ j_1 = 2$ and $j_2 = n$, the process can be summarized as follow:

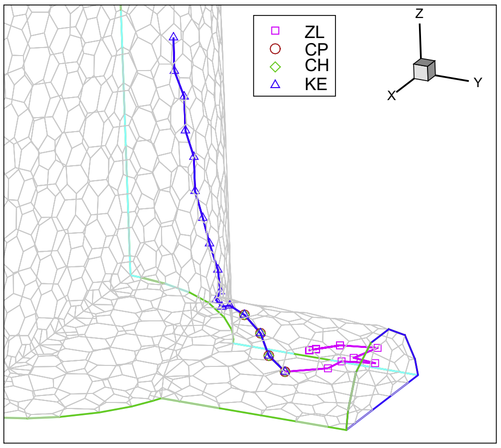
Step (1). Compute $\phi_{f12} = \Phi(P', N_1 P N_2)$ and $\phi_{fn1} = \Phi(P', N_n P N_1)$. If $\phi_{f12} = \phi_{fn1} = +1$, trajectory PP' must locate at the same side of the face $N_1 P N_2$and $N_n P N_1$. The process then turns to step (2). Or else trajectory PP' has no intersection with the face F; and TFI test returns false.
Step (2). Turn to step (3) in case of $n > 3$, or else compute $\phi_{f2n} = \Phi(P', N_2 P N_n)$ to find the relationship between trajectory PP' and face $N_2 P N_n$. If $\phi_{f2n} = +1$, the trajectory must intersect face F. The TFI test returns true, otherwise returns false.



(a) 3D tetrahedral grid.



(b) 2D triangular grid with internal bound-ary.



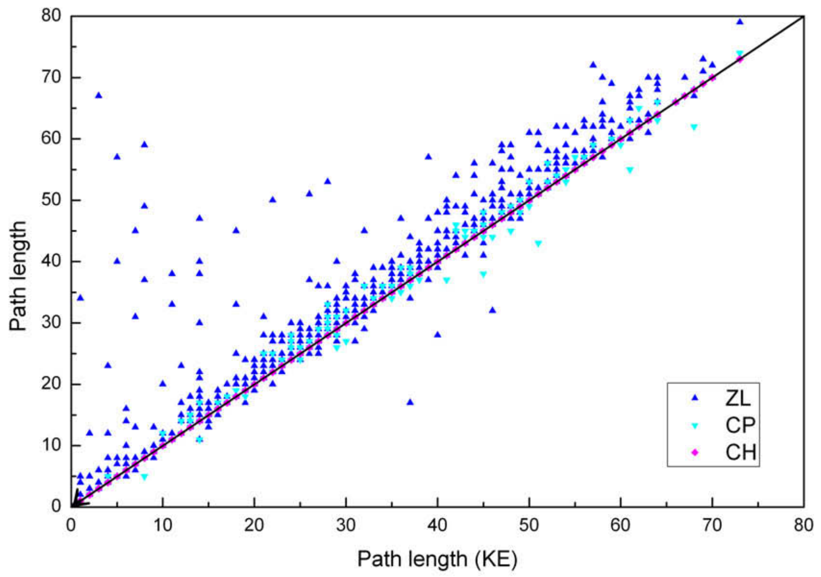
(c) 3D polyhedral grid.

Fig. 11. Search paths generated by the four algorithms.

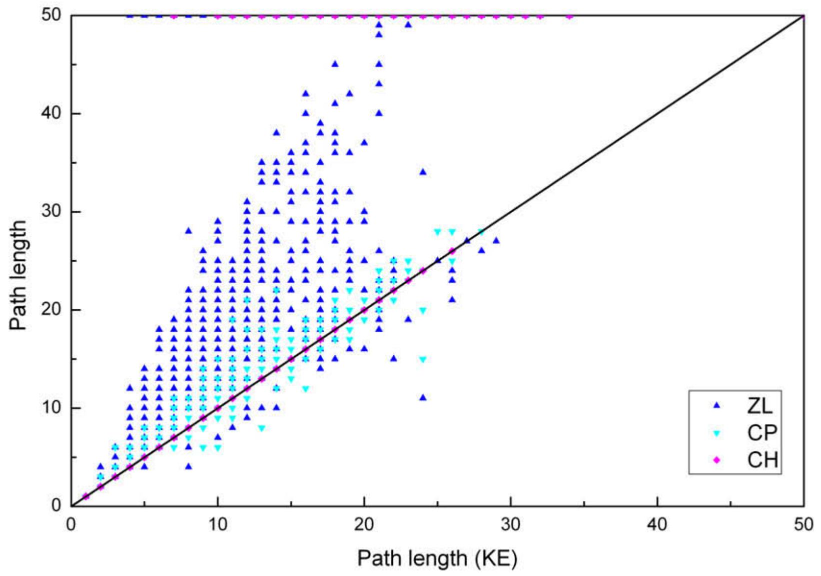
Step (3). Select the middle node N_m, where m is computed by Eq. (5), and divide the face into two sub-face F_1 and $F_2, F_1 = \{N_i,\ i = 1, 2, \ldots, m\},\ F_2 = \{N_i,\ i = 1, m+1, m+2, \ldots, n\}$. Then compute $\phi_{fm1} = \Phi(P', N_m P N_1)$ and determine the sub-face to continue TFI test. Accordingly, the nodes would be re-indexed if necessary. Finally, the process goes to step (2).

Alike the binary search method used in PIC test, this search is more efficient than the direct exhaustive search, because its average testing time is at the level of $\log_2 n$, while the direct search is at the level of n.

Here is a sample case to show the worst performance of the algorithm, which is still better than the direct exhaustive search. As shown in Fig. 8(d), the worst condition for the face of six nodes needs five side function computations (i.e., $\Phi(P', N_1 P N_2),\ \Phi(P', N_6 P N_1),\ \Phi(P', N_4 P N_1),\ \Phi(P', N_5 P N_1)$ and $\Phi(P', N_5 P N_6)$), while the direct exhaustive search needs six side function evaluations.



(a) 3D tetrahedral grid without boundary.



(b) 3D polyhedral grid with boundary.

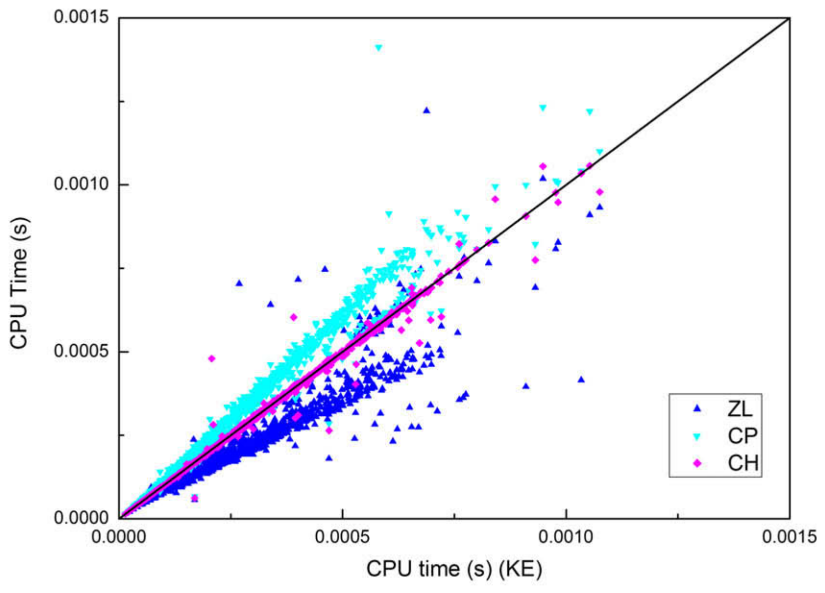
Fig. 12. Search path length comparisons among the four algorithms.

2.3.4. Current cell and position reset

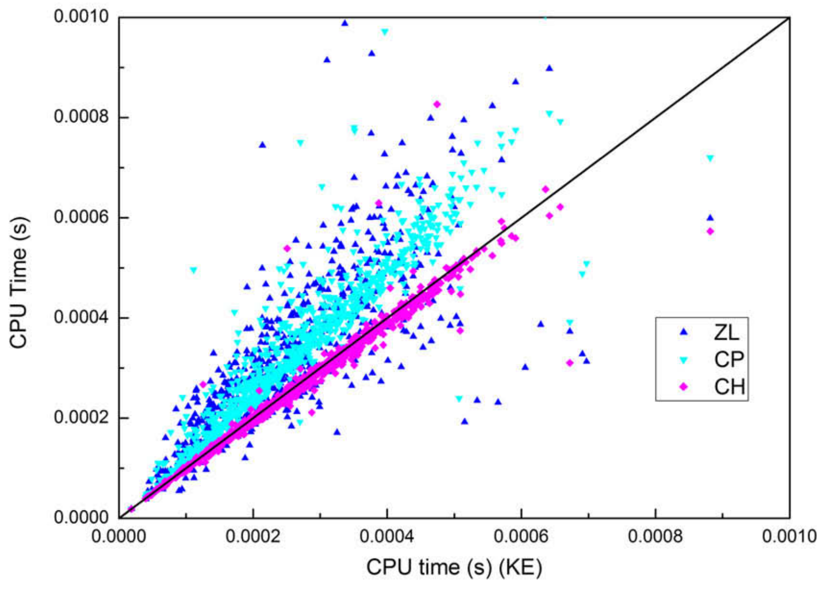
The particle localization process will continue after selecting the cell C' as the new current cell, which is the neighbor of the cell C and determined by the exit face. On the other hand, the new current position P could be selected using the cross point of the trajectory and the exit face, or just the center position of the cell C' for the simplicity.

2.3.5. Special boundary treatment

The previously published known vicinity algorithm might meet problems in case that the search path meets any internal boundary inside the grid or the external boundaries. The new neighbor selection algorithm was developed in the present study to overcome such difficulty. If the exit face selected is the boundary, then there will be two choices to continue searching. Choice (a): choose another face that fails the PIC test. If there is no alternative face, the process turns to Choice (b):



(a) 3D tetrahedral grid without boundary.



(b) 3D polyhedral grid with boundary.

Fig. 13. CPU running time comparisons among the four algorithms.

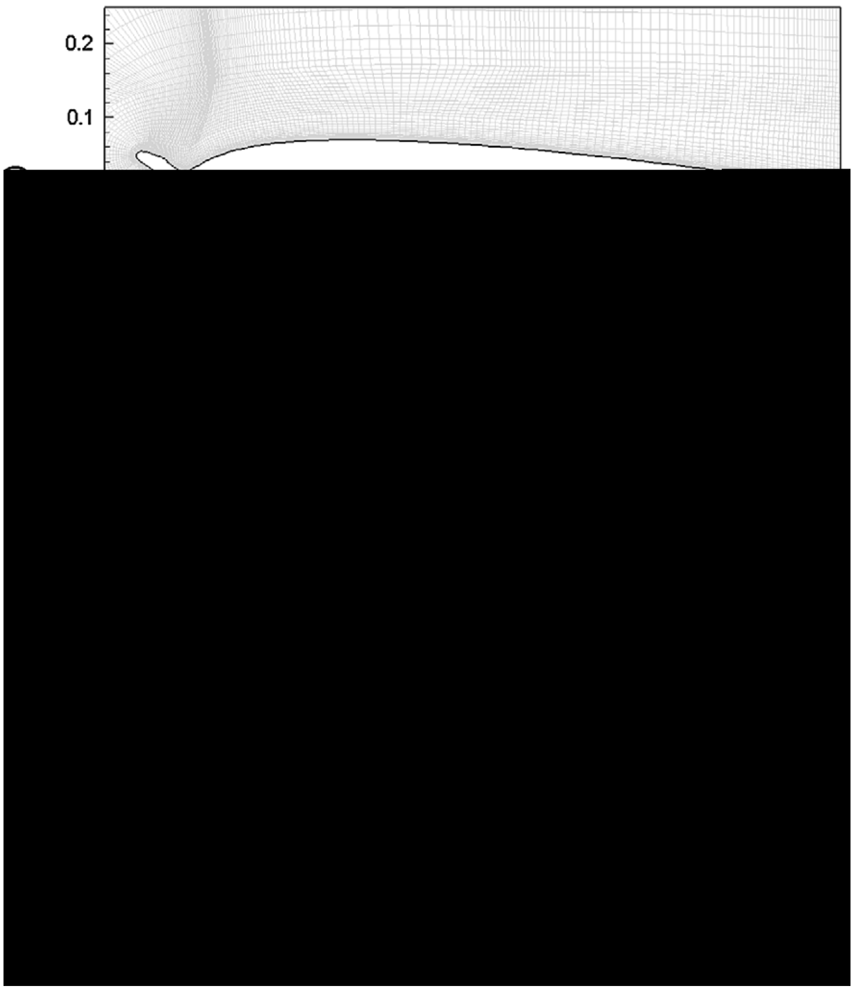
choose the face which next to the boundary and different from the previous cross faces. The face connected to the boundary would have the priority over others.

For example, the search path from point S_1 to E_1 shown in Fig. 9(a), will select the boundary face *BC* as the exit face after entering the red cell *ABCD* from the previous exit face *AB*. Thus, the face *BC* should be discarded and face *CD* will be reselected as the new exit face according to Choice (a). Similarly for the search path from point S_2 to E_2, face *AB* or *CD* both could be the new exit face by Choice (b) when the path crosses the boundary face *BC*.

The detailed judgment for Choice (b) is listed below. In Fig. 9(b), the current position *P* is inside the cell *C*, while the trajectory PP' crosses the face *F* (the gray shaded area), which is a boundary of the cell *C* and the only face where the TFI test failed. Here the Choice (b) promoted method to continue search and by-pass through this boundary face.

Step (1). Determine the faces next to the face *F*, such as face F' (the line shaded area), which was connected to the face *F* with the node N_1 and N_2. The boundary check and repetition check was continuously conducted to exclude those boundary or repetitive faces.
Step (2). Determine all cells adjacent to the cell *C* by the faces found in previous step, such as the cell C', which was connected with the cell *C* through face F'.
Step (3). Select the cell in the direction close to that of the trajectory, i.e., the maximum of value $(\overline{PP'} \cdot \overline{CC'})/(|\overline{PP'}| \cdot |\overline{CC'}|)$.

This algorithm will succeed when the initial and target positions both lie inside a connected grid. The repetitive check was incorporated nevertheless with respect to robustness, and to avoid the dead-lock, which might occurred exceptionally, such as bad grids not well connected, or bad starting/target position outside the grid, and so on. However, during the particle tracking, the algorithms' failure means that the predicted target position locates outside the computational domain. The



[redacted]

Fig. 14. The computational grid and streamlines of the flow field.

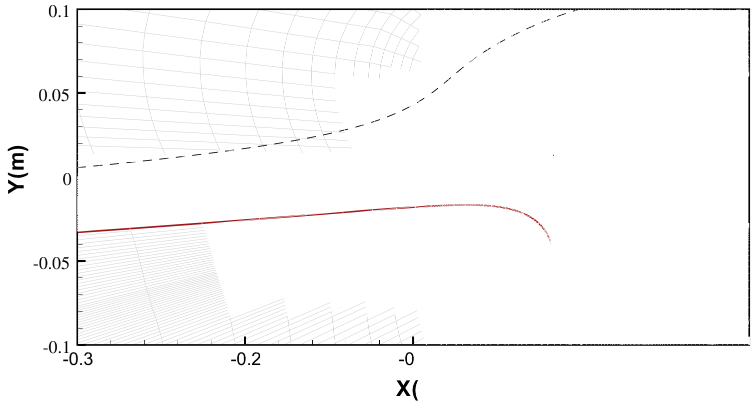
trajectory and boundary intersection should be determined by collision computation of the particle and boundary, as such stated in Ref. [1].

3. Implementation and comparison

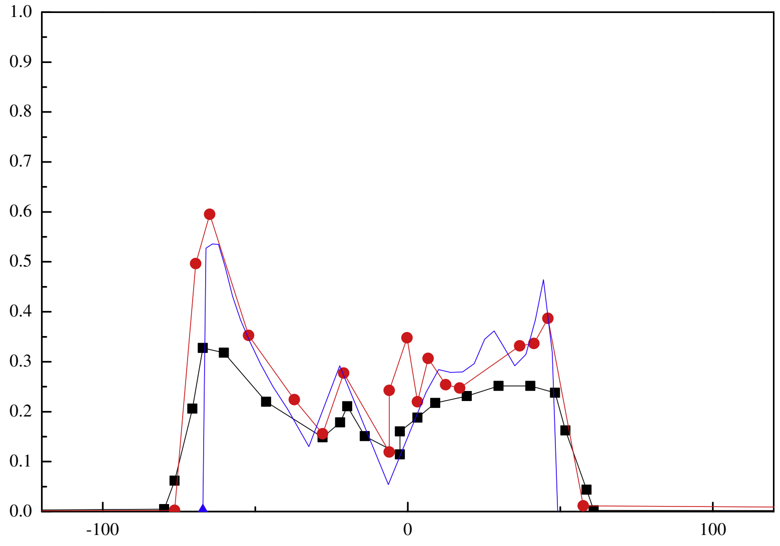
The present and previous algorithms were implemented and applied to practical problems, and their performances were compared in this section. Programming the present algorithm is relative simple according to the details given in previous section.

3.1. Initial cell guess

A cell was randomly chosen in the entire grid as the guess of the starting cell for the second kind of the PLP, and its centroid position was selected as the starting position. Four different search paths from the four randomly guessed cell to the same target position generated by our algorithm were demonstrated in Fig. 10(a) and (b), respectively using the unstructured and structured grids. As can be seen from the both figures, all the search paths reached the target successfully, although the intersection of the paths and the internal boundary occurred.



(a) Computed droplet trajectories (solid) and streamlines (dashed).



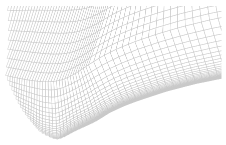
3.2. Comparison with the previous algorithms

Table 1 listed the comparison between the current algorithm (abbreviated as KE) and the previously published three methods based on the geometry relationship test, such as, ZL [11], CP [10] and CH [12], which considered the PIC test, the neighbor selection and the boundary treatment.

Firstly, different search paths obtained by these four algorithms for the same pair of the starting and target positions are illustrated in Fig. 11, where there are three cases of encounter with different grids and boundaries happened during the search: (a) a 3D tetrahedral grid without any boundary; (b) a 2D triangular grid with a circular internal boundary; (c) a 3D polyhedral grid with external boundary.

The paths were quite different for the different methods, except those generated by KE and CH was identical for their same neighbor selection method. It is worthwhile to noted that (Fig. 11(b) and (c)), only the path generated by the improved algorithm (KE) can reach the target successfully with special boundary consideration. Such behavior did not take place when using other approaches which terminate the process as encountering the boundary.

Furthermore, the performance comparisons were conducted between there previous algorithms and KE. The comparison involved the path length and the CPU running time. One thousand pairs of starting and target positions in two 3D grids were selected randomly from the grids used in Fig. 11(a) and (c).

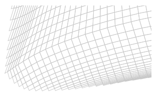


The path length comparison is displayed in Fig. 12. For a simple 3D tetrahedral grid without any internal or external boundaries met in the search path, all algorithms can reach the target positions. The path length, defined by the amount of the cell passed, of ZL is much longer than that of the other approaches in most conditions. The lengths of CH and KE are equal and the shortest amongst these four approaches. However, for more complex 3D polyhedral grid with the boundaries encountered in the search path, the previously published three algorithms may fail(as indicated by the biggest path length 50) in certain conditions. Moreover, search paths derived by KE were still the shortest under most conditions.

The CPU running time consumed by successfully seeking the paths was compared in Fig. 13. Similar comparison trend to the path length were found. The CPU time consumed by CP was the longest because of the direct intersection computation used in TFI test, which also demonstrated the efficiency of the geometry test. Although ZL could be most efficient in some simple case for its less operation, it fails and turns to the worst scenario in most complex cases. KE has good performance and acceptable computation time.

4. Sample case: water impingement computation

Papadakis et al. [16] presented experiments conducted to build the water impingement database for the study of aircraft icing. The impingement tests were carried out in the NASA Glenn Icing Research Tunnel, and the ice shapes selected for the



impingement tests were the simulated glaze ice shapes computed with the NASA Glenn LEWICE 2.2 for a NACA23012 airfoil using FAR25 Appendix C icing conditions [16]. In their test, the mean volume diameters (MVD) of the droplet were 20, 52 and 111 μm, which was composed of a distribution with 10 drop sizes measured during their previous impingement tests. Furthermore, the comparison of LEWICE and experimental impingement data for NACA23012 airfoil with 45-min glaze ice shape was given.

The impingement computation based on our particle localization algorithm and the Lagrangian particle dynamics model similar to Widhalm [7] was performed with the icing condition given in Ref. [16]. Fig. 14 demonstrated the computational grid generated to solve the flow field and streamlines near the ice horn. Figs. 15–17 illustrated 36 water trajectories and the streamlines released from the up and down limitations of the droplet impingement region. The local water impingement efficiency of this study (KE) was given and compared to the results taken from the experiment and LEWICE.

Excellent agreement between numerical and experimental impingement limitation were found in Figs. 15–17 for each droplet size, where the local impingement efficiency distribution of KE agrees well with LEWICE throughout the droplet impingement region. It should be noted that, the absolute impingement efficiency of the numerical simulation, both KE and LEWICE, was slightly higher than the experiment value at elevated MVDs of 52 μm and 111 μm. Such difference maybe because that the current particle dynamics models in hybrid Eulerian–Lagrangian framework did not fully consider the much larger droplets. Such droplets maybe deform or break-up during the trajectories and decrease the local impingement efficiency.

5. Conclusions

Extensions and improvements to the known vicinity algorithm for both kinds of particle localization problem in the hybrid Eulerian–Lagrangian model for arbitrary polygon and polyhedral grid were presented and implemented. Even without knowing the initial cell, the new algorithm can be used to determine the target cell hosting the target position by carefully searching for neighbor when the search path meets the boundary. Compared with the previous algorithms, the side function offers a more formal description than P2L or T2L test for the PIC and TFI tests. The computation efficiency was enhanced by the binary search methods due to its capability to accelerate the PIC and TFI test for more complex polygon cells.

The good agreement between the numerical experiments and the sample case indicated that, (1) the proposed method was applicable to both structured and unstructured 2D/3D grids; (2) it can locate the cell containing the target position successively even there were boundaries sitting on the search path, under which the previous algorithms may be difficult to proceed; (3) the search path generated and the time consumed by our algorithm were shorter in most of application cases than the ones obtained by previous algorithms.

Acknowledgments

The authors would like to thank the reviewers for their valuable suggestions and comments, and also thank Dr. Feng Yang for helpful advices and proof-reading.

Appendix A

Some definitions used in this algorithm were summarized in this appendix for a quick reference.

A.1. The particle location problem (PLP)

In general, the particle location problem was those to determine a cell in the Eulerian grid hosting a given position. There are two kinds of the particle localization problem: Those with prior knowledge of a nearby location, and those without. The first kind was formally defined by Haselbacher et al. [1]: "Given a grid, a particle position, and the cell which contains that particle position, determines the cell which contains a nearby particle position." The second one could be characterized as: Given a grid and a particle position, to locate the cell containing that particle position.

Some terms involved will be explained with Fig. 18.

The starting position, or the initial position P, as represented by the hollow triangle, is the position preset for the first kind of PLP, or guessed for the second kind of PLP.

The starting cell C is the cell hosting the starting position P. The difference between the two kinds of the PLP is whether the starting cell is known or not.

The target position P', as represented by the hollow square, is the position waiting to be determined the hosting cell.

The target cell C' is the cell hosting the target position P'.

The particle trajectory from P to P', as indicated by the bold line, is the particle movement trajectory, which is dominated by the equations of motion for the particle.

The search path, as indicated by the dash lines, is generated by the particle locating algorithm, and consisted of the center position of the cell passed during the search.

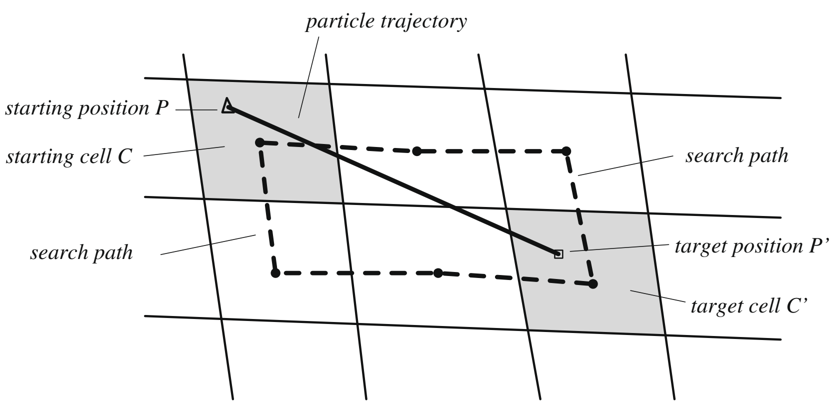


Fig. 18. Scheme of the particle location problem.

A.2. The known vicinity algorithm

The known vicinity algorithm is the locating method, which generally starts from a cell in the vicinity of the target position to be located, then jumps from neighbor to neighbor in the known grid, until finds the cell hosting the target position. It is typically best if the position can be located in a few attempts [3]. However, it was extended in this study to deal with the second kind PLP, without known the vicinal starting position, which needs the longer search path.

A.3. Side function

A function was utilized to check the relationship between the given points and faces, and widely applied in this improved known vicinity algorithm. Detail definition and mathematical description are given in Section 2.3.1.

A.4. PIC (particle in cell) test

PIC test was referred to the test to determine whether a particle lies inside a cell or not. A simple and effective procedure was suggested by Zhou and Leschziner [11] to do the PIC test: "moving along the cell faces (or the segments that join two consecutive cell vertices in 2D) anti-clockwise and check if the particle lies to the left of all the cell faces. If this is the case, within a given cell, the particle is within the cell".

A.5. P2L (particle-to-the-left) test

The P2L test in ZL algorithm was the process to check if the particle lies to the left of a given cell face. It was extend to 3D by Chorda et al. [12], and renamed as P2I (particle-towards-the-inside). Previously, the P2L test requires that the nodes of a given face must be sorted in anti-clockwise. However, the P2L test with the side function was extended in this study in both 2D and 3D grids to any cell with sorted faces. The systemic description is given in Section 2.3.2.

A.6. TFI (trajectory and face intersection) test

The TFI test was defined as the intersection test between the particle trajectory and the cell face, where the trajectory is always from the current particle position to the target position during the search process.

A.7. T2L (trajectory-to-the-left) test

The T2L test is used to check whether the particle trajectory lies to the left of given points (in 2D) or faces (in 3D), and firstly introduced in the approaches of ZL and CH. Similar as the P2L test, the T2L test with the side function in this study was extended to be applied to any cell with sorted faces, and not limited to the cell faces sorted in anti-clockwise.

A.8. Internal boundary

The internal boundary is consisted of the boundaries of the unreachable cells or holes in the grid, as demonstrated in Fig. 19. They are all the locations that the particle cannot enter. Although the edges/faces of the such cells or holes must have some boundary condition assigned, here they are defined as internal boundary to be separated from the other external boundaries.

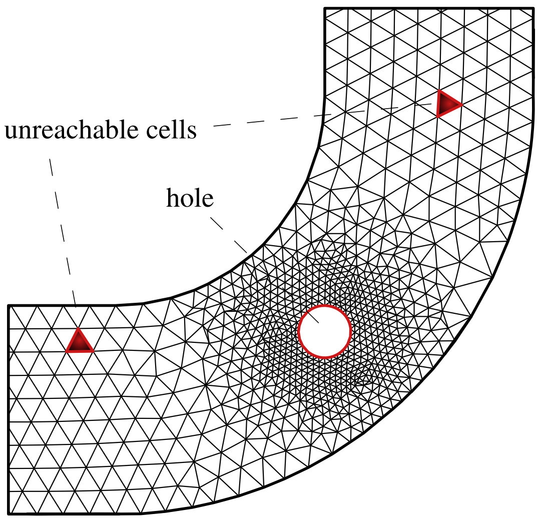


Fig. 19. Demonstration of the unreachable cells and holes.

A.9. Special boundary treatment

The special boundary treatment was the means utilized to deal with the boundary problems, including the internal or external boundary, during the search process. More details could be found in Section 2.3.5.

References

[1] A. Haselbacher, F. Najjar, J. Ferry, An efficient and robust particle-localization algorithm for unstructured grids, J. Comput. Phys. (225) (2007) 2198–2213.
[2] R. Lohner, J. Ambrosiano, A vectorized particle tracer for unstructured grids, J. Comput. Phys. 91 (1990) 22–31.
[3] R. Lohner, Robust, vectorized search algorithms for interpolation on unstructured grids, J. Comput. Phys. 118 (1995) 380–387.
[4] G. Li, M.F. Modest, An effective particle tracing scheme on structured/unstructured grids in hybrid finite volume/pdf Monte Carlo methods, J. Comput. Phys. 173 (2001) 187–207.
[5] S. Apte, K. Mahesh, P. Moin, J. Oefelein, Large-eddy simulation of swirling particle-laden flows in a coaxial-jet combustor, Int. J. Multiphase Flow 29 (2003) 1311–1331.
[6] J. Petera, L. Weatherley, A. Hume, T. Gawrysiak, A finite element algorithm for particle/droplet trajectory tracking, tested in a liquid–liquid system in the presence of an external electric field, Comput. Chem. Eng. 31 (2007) 1369–1388.
[7] M. Widhalm, A. Ronzheimer, J. Meyer, Lagrangian particle tracking on large unstructured three-dimensional meshes, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, 2008.
[8] T. Westermann, Localization schemes in 2d boundary-fitted grids, J. Comput. Phys. 101 (1992) 307–313.
[9] X.-Q. Chen, Efficient particle tracking algorithm for two-phase flows in geometries using curvilinear coordinates, Numer. Heat Transfer A – Appl. 32 (4) (1997) 387–405.
[10] X.-Q. Chen, J. Pereira, 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, Numer. Heat Transfer B – Fundam. 35 (1) (1999) 41–63.
[11] Q. Zhou, M. Leschziner, An improved particle-locating algorithm for Eulerian–Lagrangian computations of two-phase flows in general coordinates, Int. J. Multiphase Flow 25 (1999) 813–825.
[12] R. Chorda, J. Blasco, N. Fueyo, An efficient particle-locating algorithm for application in arbitrary 2d and 3d grids, Int. J. Multiphase Flow 28 (2002) 1565–1580.
[13] G. Martin, E. Loth, D. Lankford, Particle host-cell determination in unstructured grids, Comput. Fluids 38 (2009) 101–110. doi:10.1016/j.compfluid.2008.01.005.
[14] G. Coppola, S.J. Sherwin, J. Peir, Nonlinear particle tracking for high-order elements, J. Comput. Phys. 172 (2001) 356–386.
[15] G.E. Bredon, Topology and Geometry, Springer Science + Business Media, LLC, New York, 1993.
[16] M. Papadakis, A. Rachman, S.-C. Wong, K.E. Hung, C.S. Bidwell, Water impingement experiments on a NACA 23012 airfoil with simulated glaze ice shapes, in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, 2004.