Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 53 (2007) 329–344
An International Journal computers & mathematics with applications
www.elsevier.com/locate/camwa

Efficient random walks in the presence of complex two-dimensional geometries

Prabhu Ramachandran[*], M. Ramakrishna, S.C. Rajan

Department of Aerospace Engineering, IIT Madras, Chennai 600 036, India

Received 13 September 2005; accepted 27 February 2006

Abstract

This paper details an efficient algorithm for particles undergoing random walks in the presence of complex, two-dimensional, solid boundaries. The scheme is developed for the simulation of vortex diffusion using the random vortex method. Both vortex blobs and sheets are handled. The relevant modifications for using the algorithm with the vorticity redistribution technique are also discussed. The algorithm is designed to be used in the framework of an existing fast multipole implementation. A measure for the geometric complexity of a body is introduced and the algorithm's efficiency is studied as various parameters are changed for bodies of varying complexity.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Random walk; Vortex diffusion; Domain decomposition; Geometric complexity

1. Introduction

The primary aim of this paper is to present an efficient algorithm for particles undergoing a random walk in the presence of complex two-dimensional geometries. This has immediate application to the simulation of vortex diffusion in the vicinity of complex solid bodies.

Vortex methods [1,2] are grid free, particle based and are typically used to solve time dependent, incompressible, viscous fluid flows. These methods discretize the flow into interacting elements of vorticity called vortex blobs and/or vortex sheets. Vortex blobs are usually circular in shape. Vortex sheets are flat and usually [3] have zero thickness. These vortex elements are tracked in time as per the governing differential equations. The computation typically involves two steps: convection and diffusion. Convection involves the computation of the velocities induced by the constituent vortex elements on each other and the effect of the solid boundaries. Using the computed velocity field, the vorticity is convected in time. This can be performed very efficiently using various acceleration techniques [4–6]. Complicated geometries do not usually pose severe problems in this step.

The second computational step involves the diffusion of the vortex elements. There are several techniques for vortex diffusion. The random vortex method (RVM) introduced by Chorin [7] is based on the fact that the diffusion

[*] Corresponding address: Department of Aerospace Engineering, IIT Bombay, Mumbai 400076, India.
E-mail address: prabhu@aero.iitb.ac.in (P. Ramachandran).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2006.02.050

of vorticity in a fluid of viscosity ν can be approximated by giving the vortex particles a random displacement with a zero mean and total variance $2\nu t$, where t is time. Therefore, during each time step, each particle must be given an independent random displacement with zero mean and variance $2\nu\Delta t$, where Δt is the time step. To account for the boundary layer, Chorin [3] proposed a vortex sheet algorithm and introduced the concept of a "numerical layer". This is a thin layer around the body and the vortex sheets in this layer are only given a diffusion displacement normal to the local surface. If a blob enters this region, it is to be converted into a sheet of corresponding strength. Conversely, if a sheet leaves this layer, it is converted to a blob. The references [3,2,8,9] provide more details on the theory and the implementation of the RVM.

Computationally, the difficulty with the RVM in the presence of arbitrary geometries is that the vortex elements may penetrate the solid bodies or even go across them due to their random displacements. When a vortex particle strikes the surface of a body, the usual practice is to reflect the blobs specularly. Some researchers [10], delete any particles that enter the body. Irrespective of the scheme being used, it is difficult to efficiently determine whether the particle strikes or crosses a complicated solid body.

Deterministic diffusion schemes for vortex methods simulate diffusion without using a probabilistic approach. These methods do not suffer from the noise present in the RVM. The Particle Strength Exchange (PSE) method due to Degond and Mas-Gallic [11] and the Vorticity Redistribution Technique (VRT) due to Shankar and Van Dommelen [12,13] are commonly used methods. Both algorithms produce very accurate results [14,12]. However, these methods are computationally more expensive than the RVM. Complex geometries also cause complications for these schemes. Ploumhans et al. [15,16] have developed a method to handle the accurate redistribution of vorticity using the PSE scheme in the presence of arbitrary bodies. Their method is specifically tailored to the PSE scheme. The VRT is similar in principle to the PSE. In order to simulate diffusion, one distributes the circulation of a given vortex blob to other blobs that are located within a multiple of the diffusion length scale, $\sqrt{K\nu\Delta t}$, where K is a constant chosen [13] as 12. A linear programming problem is solved in order to determine the amounts of circulation that are to be distributed. If no solution is possible, new vortices of zero strength are added adaptively until the solution is feasible. In [17] an overview of the VRT is provided. The VRT does not require special treatment at the boundaries, except that one must ensure the obvious condition that there is no transfer of vorticity across a boundary. In the presence of complex geometries one has to find particles in a neighbourhood of the diffusing vortex element so that the line between the particles is not intersected by a boundary. In general this is not easy to determine.

In this paper an efficient methodology is developed for handling the above difficulties with diffusion in the presence of arbitrary two-dimensional shapes. Certain ideas from the fast multipole algorithms are used in order to make the computations more efficient.

Other work reported in literature that address diffusion in the vicinity of arbitrary geometry do not quantify geometric complexity. This paper provides a simple measure of this geometric complexity. The algorithm developed here is shown to scale efficiently as this complexity increases. The method is first described in detail using the random vortex method and is later extended to the vorticity redistribution technique.

The basic idea behind the algorithm is to discretize the geometry into linear segments. The domain containing the blobs and solid boundaries is then decomposed into a quad-tree of cells, much like the computational cells of the fast multipole methods [18,4,5,19]. The particles are then tracked along the cells as they move. If the particle strikes a body it is reflected appropriately. Sheet–Blob conversion issues are also considered. The algorithm is designed so that it is possible to extend an existing adaptive fast multipole method implementation.

It is to be noted here that the present work does not propose a new diffusion scheme. The aim of the paper is to provide an efficient means to use existing diffusion schemes in the presence of complex geometries. The main aims of the paper are as follows.

- Provide an efficient algorithm to handle randomly moving particles in the presence of complex geometries.
- Apply the algorithm to the random vortex method and show how to adapt it for use with the VRT.
- Use components of the fast multipole algorithm in order to perform efficient domain decomposition. In the process the paper also provides a generalized version of the domain decomposition for source and target particles similar to the work of Strickland and Baty [20].
- Provide a simple measure for geometric complexity.
- Demonstrate that the algorithm works efficiently for complex geometries.

The algorithms developed in this paper are somewhat involved and detailed descriptions are necessary in order to make it easy for the reader to understand and implement them. A description of the present algorithm is available with a brief introduction on vortex methods as an internal report [21] by the same authors.

2. Vortex diffusion in the presence of arbitrary boundaries

Irrespective of the diffusion technique used, there are important issues that need to be considered when diffusing vorticity in the presence of arbitrary boundaries. For the RVM, the sheets are constrained to a random displacement perpendicular to their plane. Further, the presence of the numerical layer requires that the blobs be converted to sheets and vice versa. To this end, one is required to track the particles and reflect them as and when they strike a solid surface. For simple geometries this can be trivial. For arbitrary bodies it can become highly complicated and inefficient to do this. The conversion of blobs to sheets and vice versa complicates the situation further. The VRT or any similar scheme requires particles within some radius of a diffusing particle to be identified. The line connecting the diffusing particle and its neighbors must not intersect any solid boundaries. The list of interacting particles must be carefully chosen. Hence, it is clear that in order to simulate the flow past arbitrarily shaped bodies, an efficient algorithm to deal with these complications must be obtained. The following are important issues to be considered when developing such an algorithm.

1. Discretization of the body geometry: In the present work this is done by discretizing the body into a set of linear panel elements.
2. Ensuring that the particle paths do not penetrate solid walls: In general it is important to be able to check if an arbitrary line segment intersects part of the boundary.
3. Particles may have extent: These particles have to be handled carefully because it is to be ensured that no part of the particle penetrates the surface. Vortex blobs are typically circular and vortex sheets are oriented line segments.
4. The entire procedure must be computationally efficient and not unduly hard to program.

The techniques to deal with the above are first discussed for the RVM. These can be modified to work for the VRT. The basic idea is as follows. The body geometry is discretized into linear panels. The domain of diffusing particles is split based on a modification of the adaptive fast multipole method (AFMM) [4]. The particles that strike a panel are reflected specularly. Particles that enter or leave the numerical layer are converted to sheets or blobs respectively. The methodology is now discussed in greater detail.

2.1. Domain decomposition

The most important step in the efficient formulation of the problem is to perform a domain decomposition. An adaptive quad-tree structure is used to organize the interacting particles and panels. It is noted that a quad-tree structure is also used in adaptive fast multipole methods (AFMM). Hence an effort is made to draw an analogy between the presently developed algorithm and the one used in the AFMM. It is shown that the AFMM domain decomposition can be generalized and used for both the present scheme and for the AFMM.

The domain is split up into cells that contain all the particles (blobs and sheets) and the panels. Depending on the number of vortex elements and panels in a particular cell, it may be split into four daughter cells. An unsplit cell is referred to as a *childless* or *leaf* cell. The split cell is called a *parent*. A *branch* cell is synonymous with a parent cell. The cell splitting can also be controlled based on the size of the cell.

The first cell, that contains all the particles and panels is defined to be at level 0. If a cell is at level l, cells of the next level $l+1$ are obtained by splitting it into four daughter cells. Consider a cell b at a level l. The colleagues of cell b are defined as all the cells at the same level l that share either a side or corner with it. This is illustrated in Fig. 1.

In order to simplify and generalize the subsequent discussions, the particles are viewed as *causes* or *effects*. As *causes*, they influence the *effects*. The *effects* are merely influenced by the *causes*. Treating the particles in this manner results in a very general algorithm for the domain decomposition that can be used efficiently in a wide variety of schemes. Consider the following example where the AFMM is used to compute velocities due to blobs. The computational domain contains blobs and passive particles. The passive particles by definition are *effects* as they do not induce a velocity on anything. The blobs influence all particles (blobs and passive particles). Therefore, the blobs are *causes* and both the blobs and passive particles are *effects*. This example brings out the fact that the interacting elements have two types of manifestations, *causes* and *effects*.

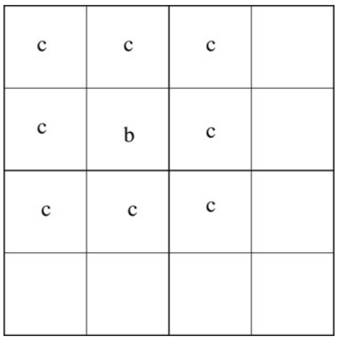

Fig. 1. The colleagues of cell *b*.

The general case is complicated by the addition of sheets and panels. In the present case of diffusion using the RVM, in the presence of arbitrary geometries, the following points are to be noted.

1. Blobs do not influence the random walk of other blobs.
2. Blobs and sheets are influenced by panels.
   (a) Blobs and sheets can intersect panels.
   (b) The terminal position of a blob may be in the numerical layer of a panel, resulting in its conversion to a sheet. Similarly, the terminal position of a sheet might be outside the numerical layer resulting in its conversion to a blob.

Therefore the panels are the *cause* elements and everything else is an *effect* element. Generally, the number of *causes* and *effects* are not the same. In the present case the number of *effects* (blobs and sheets) far exceed the number of *causes* (panels).

The idea behind the cell splitting is similar to that used in fast multipole techniques and is based on the proximity and number of the interacting *cause* and *effect* elements. For the case of a diffusing particle, given a particle path, one has to determine if the path will intersect the body. The following points are to be noted.

1. As the particle passes through a cell, it is required to check if its path intersects with any of the panels in that cell. If there are many panels in a cell then there are more intersection checks that have to be made. Hence, it is a good idea to reduce the size of the cell and thereby reduce the number of checks in that cell. It is to be noted that if the cell size is too small (i.e. much smaller than the length of the path), then creating more cells would mean that the particle path is to be traced through more number of cells (tracking a particle through cells also involves computational effort). In practice an optimal choice of cell size is to be made based on a few numerical experiments.
2. If there are very few panels and particles in a cell it is efficient to leave the cell as it is.
3. If a cell *b* contains a large number of *causes* (panels) and very few *effects* (particles) and if a neighbor *c* contains a large number of *effects* then it is likely that a particle present in *c* will enter cell *b*. By splitting *b* the intersection checks made on all such particles entering from a neighbor may be reduced.
4. Just as in the previous case, if a cell *b* contains a large number of *effects* and few *causes* then the decision to split it depends on the nature of its colleagues.

The domain decomposition used by the AFMM as applied to blobs and tracer particles can be expressed in terms of *causes* and *effects* as follows.

A. If there are a large number of *causes* and *effects* in a cell, then it would be inefficient to perform a direct computation between the *cause* and *effect* elements in the cell. Therefore the cell needs to be split into daughter cells.
B. If there are a very small number of *causes* and *effects* in the cell, then it is faster to perform the direct computation between the *cause* and *effect* elements in the cell rather than further subdividing the cell. Therefore the cell should be left alone and stored as a leaf/childless cell.
C. If there are a large number of *cause* elements but small number of *effects* in the cell *b*, then the decision to split depends on the nature of its colleagues. This is because, if there are a larger number of *effects* in any of the colleagues, *c*, then it would be expensive to do the computations between the *causes* in cell *b* and the *effects* in *c*. Hence, if any of the colleagues *c*, of cell *b* have a large number of *effects*, then cell *b* is split into daughter cells.

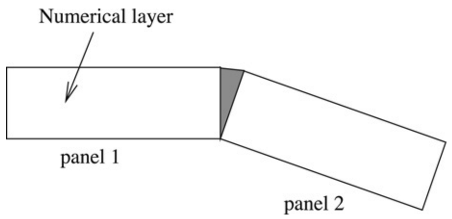

Fig. 2. The shaded region indicates the region between two panels which does not belong to either panel.

This will reduce the number of computations that have to be performed between the *causes* of cell b and the *effects* in c. If there are no such colleague cells then the cell b can be stored as a leaf/childless cell.

D. Similarly, if there are a large number of *effect* elements but small number of *causes* in the cell b, then it should be split if any of its colleagues c, have a large number of *causes*. If there are no such colleague cells then the cell b can be stored as a leaf/childless cell.

As opposed to the traditional domain decomposition employed in the AFMM, where only causes (blobs, charges, etc.) are considered, the above domain decomposition (A through D) can handle situations where the velocity on blobs and passive particles are computed.

Clearly, there is a direct correspondence between items 1 through 4 and A through D in the above discussion. Hence, the same domain decomposition algorithm can be used without loss of efficiency for both the present scheme and fast multipole methods. In [22,23] the authors apply this domain decomposition algorithm to a fast multipole scheme for panel methods. Strickland and Baty [20] also propose a similar scheme to handle independent "source" (cause) and "target" (effect) fields. Their scheme creates separate cells for the sources and targets whereas the present scheme creates a single set of cells with each cell containing both causes and effects.

Using such a domain decomposition algorithm one constructs a mesh of childless cells that divides the particles based on their proximity and number. In the present case, for the efficient treatment of diffusion, the existence of a panel in a cell is determined by some representative point of the panel. For example the center of the panel can be used. Since the panels have extent, a single point representation of the panel is insufficient because one has to check if the path of a particle intersects the panel. It is possible that a cell does not contain the representative point and yet has the panel passing through it. This problem is solved by associating a panel with more than one cell and is discussed in the next section.

2.2. Domain decomposition in the vicinity of a panel

As the panel has extent and in the case of the RVM, the panel has a numerical layer associated with it, it must be treated as a box. This box has a length equal to that of the panel and a height equal to that of the numerical layer. If the body is curved, there are further complications since the region between two boxes as indicated in Fig. 2 does not belong to either box. A blob entering this region will not be converted into a sheet. This problem can be alleviated by using a trapezium instead of a rectangle for the box. The trapezium can be found by extending the top side of each box till they intersect. Then the sides of the trapezium are bounded by the panel, the plane parallel to the panel at the height of the numerical layer and the other two sides are shared with neighboring boxes. The resulting trapezium is henceforth called a *viscous box*. Even if the panels move/deform, since the motion of the panel is known, the new viscous box can be easily obtained.

It is to be noted that Fig. 2 is for the case of a convex surface. For a concave surface, the viscous boxes will be formed by the intersection of the top sides of the viscous boxes.

Once the body geometry is specified, the geometry of the viscous boxes are known. Using an implementation of the domain decomposition algorithm discussed in Section 2.1, the domain is first decomposed in terms of the panel centers or some representative point, called the control point, inside the viscous box. Thus, each viscous box is initially identified with one childless cell. Due to the extent of the viscous box, parts of it will also be in other cells. In Fig. 3 the viscous box is initially assigned to cell 1 alone. All its neighboring cells also contain parts of the viscous box and this information must be updated in those cells. A cell is said to contain a viscous box if a side of the box passes through the cell. All the childless cells are modified based on this.

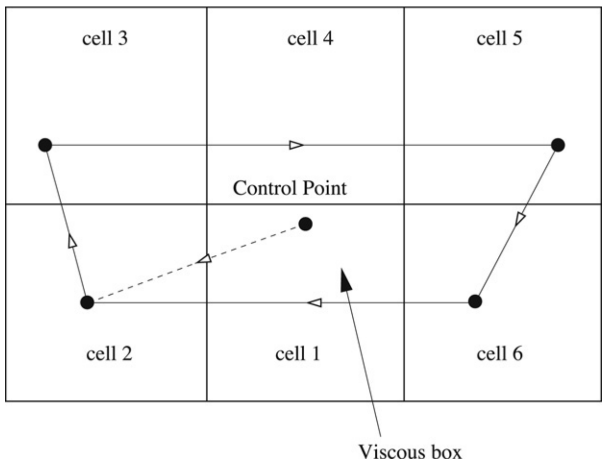

Fig. 3. A viscous box passing through various cells.

The following procedure is applied to modify the childless cells that are created by the domain decomposition. A viscous box is considered. For each of these boxes the control point is within some cell created by the domain decomposition. Starting at the control point, a path is traced to an edge of the box. The path continues along lines constituting the trapezium. In this process, the different childless cells that are traversed are modified to reflect the existence of a part of this viscous box in them. Care is to be taken to avoid repetitions when the cell already contains the particular box. The procedure is illustrated in Fig. 3, where the viscous box passes through the cells numbered 1 through 6. After performing the domain decomposition only cell 1 will contain the particular viscous box. After performing the above mentioned procedure, cells 2, 3, 4, 5 and 6 are also made to contain part of the viscous box. The process is repeated for all the viscous boxes. This procedure will fail if a cell is completely enclosed by the viscous box because in that case no side of the trapezium will pass through the cell. This can be easily prevented by enforcing the condition that every cell has a side larger than the smallest side of all the trapezia.

The pseudo-code for tracking a side of the viscous box is given in Algorithm 1. z_1 and z_2 are the end points of the line (side of the viscous box) that are being tracked. The most vital component in this algorithm is the one that tracks the side of the trapezium through the various cells using the **FindNextCell** function. It is evident that this tracking algorithm is also important for the case of the random walk of particles. The details of the algorithm for tracking a line segment are elucidated first.

Algorithm 1 TrackLine(box, z_1, z_2, C)

```
$NextCell$ = **FindNextCell** ($z_1, z_2, C$)
**if** $NextCell \neq C$ **then**
   **if** $NextCell$ does not contain $box$ **then**
      Set $box$ in $NextCell$.
   **end if**
   TrackLine($box, z_1, z_2, NextCell$)
**end if**
```

The current discussion is restricted to two dimensions and hence the complex plane is used to describe the algorithm. Let the start and end points of the straight line being tracked be labelled z_1 and z_2 respectively, where z is the complex coordinate. Let z_{12} be the line joining z_1 and z_2. Assume that the tracking is started at a childless cell C which contains the point z_1.

It is determined if the point z_2 lies within the current cell C. If it does, then the line segment z_{12} is wholly in the cell C. If it does not, then it crosses over into one of the colleagues of C, intersecting the side of the cell at z_{tmp}. This side identifies the colleague C_1 as shown in Fig. 4. Then, the childless cell C_{tmp} that contains z_{tmp} is found. This will be either C_1 or one of C_1's descendents or one of its ancestors. In Fig. 4, C_{tmp} is one of C_1's descendents. In order to continue tracking from the cell C_{tmp}, the process is repeated with $C = C_{tmp}$ and $z_1 = z_{tmp}$. In this fashion, the cells

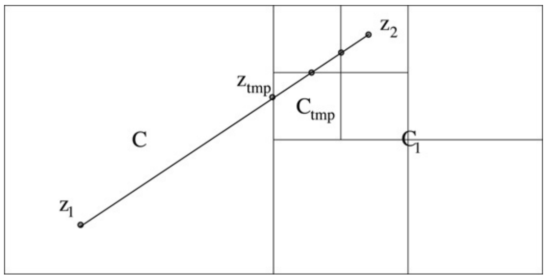

Fig. 4. Schematic of a line passing through various cells. The line joining z_1 and z_2 originates in the cell C and passes through the descendents of its colleague C_1.

through which z_{12} passes can be found. The pseudo-code to find the next cell (**FindNextCell**) is given in Algorithm 2.

Algorithm 2 FindNextCell(z_1, z_2, C)

```
if Line $z_{12}$ crosses any side of $C$ then
  Find the side, $S$, of $C$, that the line $z_{12}$ crosses.
  Find intersection of $S$ and $z_{12}$ and store as $z_{tmp}$.
  Find the colleague, $C_1$, of $C$, that shares $S$.
  Find the childless ancestor or descendent $C_{tmp}$, of $C_1$, that contains $z_{tmp}$.
  $z_1 = z_{tmp}$
  return $C_{tmp}$.
else
  return $C$
end if
```

Using implementations of the above algorithms it is easy to find all the cells that contain parts of a viscous box. After applying the algorithms to all the viscous boxes, it is possible to check for intersections of particle paths with the viscous boxes in each cell. The details of this methodology as applied to the RVM are discussed next.

2.3. *Diffusion using the random vortex method*

Diffusion of blobs and sheets using the RVM is accomplished by giving them random displacements. The sheets are given displacements perpendicular to the local panel surface and the blobs are given random displacements in each coordinate direction (x and y in two dimensions). The paths of the diffusing blobs and sheets are to be checked for intersections with any panel. Clearly, the displacement of the blob or sheet at a given time step is a straight line. If a blob enters a viscous box it is to be converted to a sheet and vice versa. A variant of the Algorithms 1 and 2 are used here and described below. The following observations need be noted before the algorithm is presented. Every blob is initially associated with a childless cell of the mesh. As the cells are critical to the tracking algorithm, the diffusion is done one cell at a time. It is also to be remembered that one side of each viscous box is associated with a linear panel that represents the solid surface. The other sides of the box are used to represent the numerical layer.

A blob in a childless cell, C, is considered. It's initial position is z_1 and the final position is z_2. The line joining the two points, z_{12}, is tracked through all the childless cells in a manner similar to the Algorithm 1. This is also illustrated in Fig. 4. First, it is determined if the point z_2 lies within the current cell C. If it does then the blob is checked for reflections with panels inside the current cell. If the point z_2 is not inside the cell C, then the line segment z_{12} crosses the cell. Therefore, the segment of z_{12} inside the current cell is found using z_{tmp} obtained from Algorithm 2. This segment is checked for intersections with all panels in the current cell. If there is no intersection then the algorithm proceeds from the next cell. If there is an intersection, the reflected path is computed. The same algorithm is then applied to the reflected path. The pseudo-code embodying the above discussion is given below.

The final value of z_2 is the final position of the particle. The methodology to check for intersections between the particle path and the panels is discussed next. For the case illustrated in Fig. 5, the panel associated with the viscous

Algorithm 3 CheckPath(z_1, z_2, C)

```
if Line $z_{12}$ crosses any side of $C$ then
    Find the side, $S$, of $C$, that the line $z_{12}$ crosses.
    Find intersection of $S$ and $z_{12}$ and store as $z_{tmp}$.
    $Last = false$
else
    $z_{tmp} = z_2$
    $Last = true$
end if
if Line joining $z_1$ and $z_{tmp}$ intersects any panel in $C$ then
    Find the panel that is intersected first.
    Find reflected ray and store the path in new $z_1, z_2$.
    CheckPath($z1, z2, C$)
else if $Last == false$ then
    Find the colleague, $C_1$, of $C$, that shares $S$.
    Find childless descendent or ancestor $C_{tmp}$, of $C_1$, that contains $z_{tmp}$.
    $z_1 = z_{tmp}$
    CheckPath($z1, z2, C_{tmp}$)
end if
```

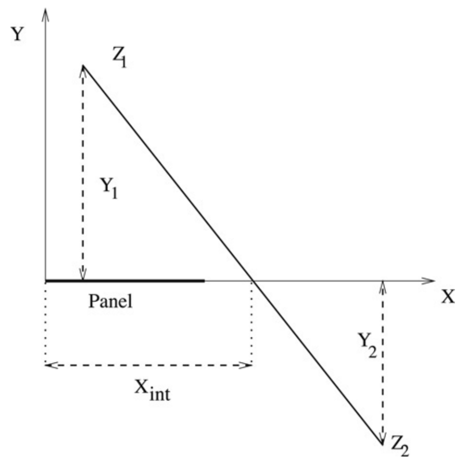

Fig. 5. Illustration of algorithm used to check intersections of blobs with a panel.

box and the relevant points are transformed to a local coordinate system. If the line joining z_1 and z_2 is the path of the particle, it is evident that it is not always necessary to compute the value of x_{int}, the point where the line intersects the x axis. One can eliminate quite a few intersection checks by looking at the relative signs of y_1 and y_2. Therefore, in the present implementation, the first check is to see if an intersection is possible. If it is, then the value of x_{int} is computed and compared to see if $0 \leq x_{\text{int}} \leq l$ where l is the length of the panel. In some cases there can be more than one panel in the same cell that the line crosses. In such a case all the intersecting panels are considered and the panel that has the closest intersection point is to be chosen for the reflection and the corresponding distance is given by, $\sqrt{(x_{\text{int}} - x_1)^2 + y_1^2}$. The intersection point is given as $(x_{\text{int}}, 0)$ and the final point is reflected such that $z_2 = (x_2, -y_2)$. It is possible to use a different reflection scheme or even delete the intersected blob, as done in [10].

The same algorithm can be used for the sheets. However, it can be simplified since the displacement of the sheets is only perpendicular to the local surface. The terminal position of the particle determines whether it is a sheet or a blob. If sheets are not used at all as in [10,24–26] then the complications due to the conversion are removed.

Usually, vortex blobs have a finite core and it may be necessary to ensure that the blob core does not penetrate a panel. In such a case the intersection check must be done by using the points closest to the panel and not the center

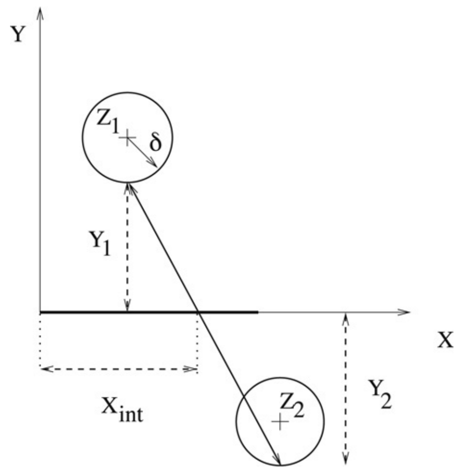

Fig. 6. Illustration of algorithm used to check intersections of blobs having a finite core radius, δ, with a panel.

of the blob. In local coordinates this reduces to the bottom of the blob as shown in the Fig. 6. This can be easily incorporated in the present algorithm by doing the following. In Fig. 6, instead of considering the actual heights in the local coordinate system, the radius of the blob, δ, is subtracted from the heights and the resulting line is used to perform the check. This merely requires two extra subtractions. However, a blob with a core does require little more care than this. Since the blob has extent, it is not sufficient to check for intersections inside the current cell alone. One must also check for intersections in all cells that the blob will pass through by virtue of its extent. This is done by treating the blob as a square, finding all the cells that it influences as it passes through a cell and checking for intersections with panels in each of them. This is not difficult to do but does require some care and effort in order to be performed efficiently. It also increases the computational time taken by the algorithm since the number of cells to consider is larger.

One final point to note is that there are cases where the angle between two adjacent panels is very small and forms a concave region. If a particle having a finite core radius performs a random walk into this region, it is possible for it to become stuck between the two panels as it approaches the intersection of the two panels. To avoid this case one must store the length of the path between two consecutive reflections. If the length is reducing and tending to zero the particle could be placed at the corner without undergoing any further reflections. Alternatively, the particle may be reflected along the bisector of the angle between the two panels.

In this fashion, the collisions of moving particles with the solid boundary are handled. Using the algorithms discussed above, it is clear that vortex diffusion in the presence of arbitrary boundaries can be carried out when using the RVM. The algorithm is also immediately applicable to particle advection. In this case, the displacements of the particles are not drawn from a random number generator but depend on the velocity field and time step. By using these displacements it is possible to perform advection in the presence of complex geometries. In the next section the modifications to the algorithm in order to handle the VRT are discussed.

2.4. *Diffusion using the vorticity redistribution technique*

In the VRT, neighboring particles of a given particle are defined as those which lie within a radius, $\sqrt{K\nu\Delta t}$. The vorticity of a particle is to be redistributed to its neighbors. To successfully implement the VRT in the presence of arbitrary boundaries it is imperative that one does not redistribute vorticity to particles that are separated by a solid boundary. Hence, given two interacting particles, it is required to check if the line z_{12} joining the particles intersects any panel. The check is simpler here, since unlike the RVM, the length of z_{12} is bounded by $\sqrt{K\nu\Delta t}$. Further, if the line joining any two particles intersects a panel, then the recipient particle can be immediately rejected. There are also no conversion and reflection issues. Similarly, when new particles are to be introduced by the diffusion procedure, checks can be applied so that the new particles are not introduced across a boundary. It is apparent that the algorithms developed for the RVM can be modified and used for the VRT.

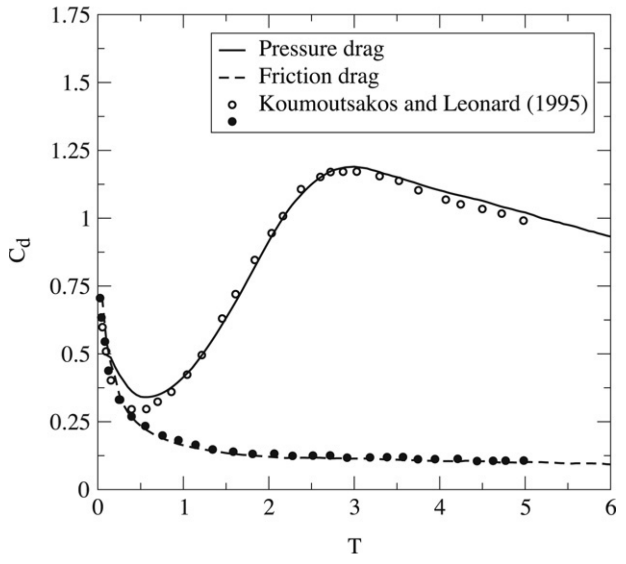

Fig. 7. C_d versus T curves for an impulsively started cylinder at $Re = 1000$. $T = Ut/R$ where U is the free stream velocity, R the radius of the cylinder and t time in seconds. The circles are the values obtained by Koumoutsakos and Leonard [14] and the lines represent the computations in [27].

The domain decomposition for the VRT will be slightly different from the RVM since for each diffusing particle, neighboring particles have to be found efficiently. Therefore, a cell must be split into children depending on the number of panels *and* blobs in it (i.e. both of them must be *cause* elements for the domain decomposition algorithm). If the smallest cell size is not smaller than $\sqrt{K\nu\Delta t}$, then only the colleagues of the cells need be checked for neighboring particles. The viscous box height can also be chosen as $\sqrt{K\nu\Delta t}$. After refining the mesh, all blobs that are within a viscous box can be identified and tagged. Consequently, only tagged blobs need be checked for intersections with panels. If a blob is not tagged, then it can safely diffuse to the particles in its vicinity. This also holds when new particles are introduced. The modified algorithms can now be applied to the VRT in the presence of arbitrary boundaries.

The next section demonstrates the utility and capabilities of the algorithms developed here using numerical computations.

3. Numerical results

The utility and accuracy of the developed algorithm in simulating well-known benchmark problems is first demonstrated. The flow past an impulsively started circular cylinder in an infinite mass of stationary fluid is considered as a benchmark problem. This is a well studied problem and extensive computational data is available. In [9], the authors simulate diffusion using the RVM and employ the developed algorithm to handle diffusion. Preliminary results of their computations are compared with results from the high accuracy simulations of Koumoutsakos and Leonard [14]. The agreement is found to be reasonable. Subsequently, in [27], high resolution simulations are performed using the RVM for diffusion along with the algorithm detailed in the present work. The algorithm is applied to both the advection and diffusion of the particles. The results from these computations are compared with those of Koumoutsakos and Leonard [14] who employ the PSE technique to simulate diffusion. Their computations are considered to be a DNS of the problem. In Fig. 7, the drag coefficient is plotted versus a non-dimensional time. The solid line plots the pressure drag and the dashed line plots the friction drag. The symbols represent results from Koumoutsakos and Leonard [14]. Clearly, the agreement is very good. It is important to note that the agreement is excellent for both the pressure and friction drag. This shows that the developed algorithm captures the diffusion of vorticity accurately, while being efficient and applicable to arbitrary geometries.

In order to demonstrate the efficiency of the algorithm developed, a complex body geometry is considered. Particles are distributed on the surface of the body and diffused using random walks. The computational time taken by the algorithm for a fixed number of time steps is plotted as various parameters are varied. In order to

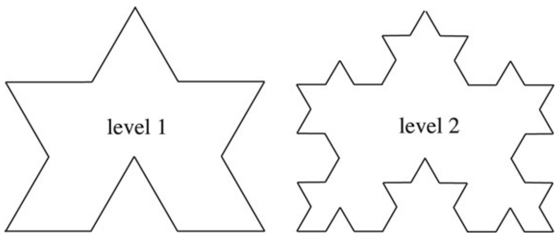

Fig. 8. Illustration of the geometry chosen for study at level 1 and 2.

Table 1
Geometric complexity of the body

Construction level	Number of sides	Geometric complexity, $\mathcal{C}$
0	3	1
1	12	3
2	48	11
3	192	43
4	768	171

characterize the geometric complexity of the body, a measure for the geometric complexity is necessary. Using a fractal dimension in this context is not useful because geometries considered in such computations are not truly fractal and can at best be finite iterations of a fractal construction. Using a complexity measure from information theory (e.g. Kolmogorov complexity) also does not seem appropriate for the present study. The following properties for the geometric complexity seem desirable:

1. The measure must be purely geometric and must not depend on the number of panels used to discretize the body.
2. The measure must be a scalar and insensitive to rotation, translation and scaling.
3. If the number of bodies is multiplied by an integer k, then the measure must also scale by k.
4. The measure must be easy to compute.

A simple measure for the geometric complexity, $\mathcal{C}$, of a body that satisfies all of the above can be defined as follows. Let $\mathcal{G}$ be the sum of the absolute change in the angle of inclination of the tangent as the contour of the body is traced. Consider an equilateral triangle, if one starts at a vertex and traces the contour of the body, it is clear that at each vertex, the inclination of the tangent changes by $2\pi/3$ rad. There are three such vertices and therefore for an equilateral triangle, $\mathcal{G} = 2\pi$. In similar fashion it can be seen that for simple geometries like triangles, rectangles, closed convex polygons and circles, $\mathcal{G} = 2\pi$. Hence, the geometric complexity is defined in terms of $\mathcal{G}$ as $\mathcal{C} = \mathcal{G}/2\pi$.

Evidently, this measure is incapable of differentiating between a circle and a convex polygon. However, for a closed body that has concave depressions and convex projections, it does show an increase in complexity. Any concavity or projection complicates the intersection algorithm because such regions would increase the number of intersections that a randomly diffusing particle would make with the body.

For the present study, in order to construct geometries with varying complexities easily, a fractal construction is chosen. At level 0 of the construction, an equilateral triangle is considered. Each side of the triangle is split in the manner of a Koch snowflake [28]. The first and second levels of construction are shown in Fig. 8. One can easily compute the complexity of such a body at any level of construction. Table 1 shows the variation of complexity as the number of levels of construction of the body shown in Fig. 8 increases. As is evident, the complexity of the body increases rapidly.

In order to test the present scheme, point vortices are distributed on the inner or outer surface of the body. The particles are diffused and the computational time taken is plotted as different parameters are varied. Since we are interested in testing the random walk algorithm alone, no convection is performed. Figs. 9 and 10 show the particles and the body used after 50 time steps. In Fig. 9 the particles are placed on the outer surface of the body and diffused. In Fig. 10 the particles are distributed on the inner surface of the body. 49 152 particles are used and the body has 576 equal sized panels. The body chosen is at level 3 and has a geometric complexity of 43. As is evident, none of the particles cross the body. The maximum number of *cause* and *effect* particles allowed per cell is fixed for all simulations

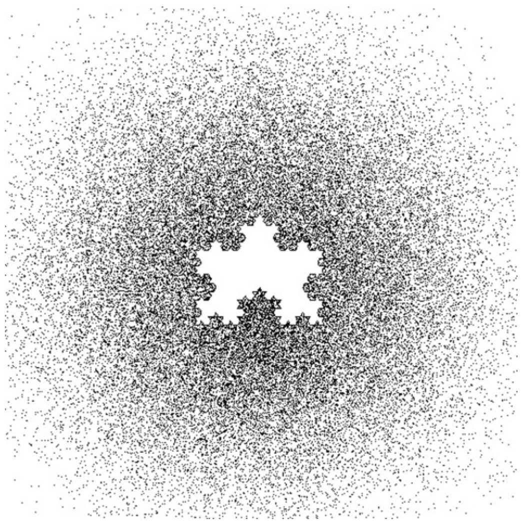
Fig. 9. Simulation of diffusion outside the body at level 3 of its construction. 49 152 vortex blobs are used in the simulation. The blobs are initially distributed on the outer surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps.

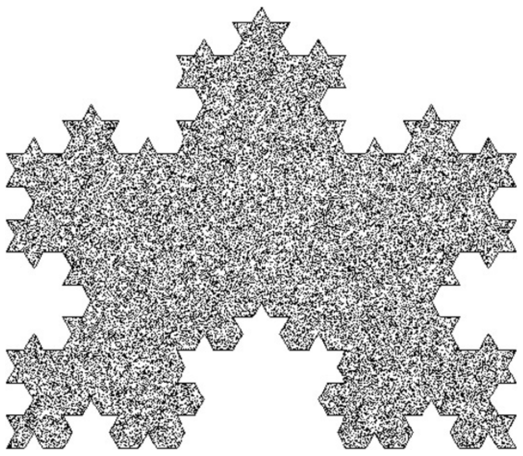
Fig. 10. Simulation of diffusion inside the body at level 3 of its construction. 49 152 vortex blobs are used in the simulation. The blobs are initially distributed on the inner surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps.

as 10. The non-dimensional length of a side of the equilateral triangle that is used to generate the body (i.e. the body at level 0) is unity. For simplicity, no sheet–blob conversion is performed. In order to stress-test the algorithm, the viscosity ν, and the time step Δt are chosen such that large random displacements are generated. In the present work ν and Δt are chosen such that the standard deviation of the random displacement is $\sigma/L = \sqrt{2\nu\Delta t}/L = 0.1$, where L is the length of the side of body. This is 10% of the size of the body and is quite a large displacement. The computations in [14] for the flow past a cylinder at a Reynolds number of 40 use a Δt of 0.02. Simulating this using the RVM would therefore require the generation of random displacements having a σ/L of about 0.032. The present displacement is about 3 times larger and therefore seems a reasonable choice to use when testing the diffusion algorithm.

To study the efficiency of the algorithm, the computational time is plotted against the complexity of the body in Fig. 11. For each body of given complexity the number of panels used is 768. The panels are equally sized. 49 152 vortex particles are used. The complexity of the body varies from 1 to 171. All other parameters are held fixed. The dashed line is for the internal diffusion case and the solid line is for the external diffusion case. It is clear that despite the great increase in complexity, there is only a 15% increase in the computational time. Due to more number of reflections, the internal diffusion case takes more computational time.

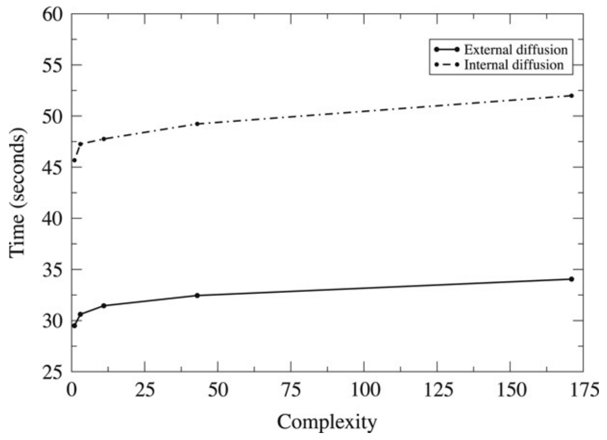

Fig. 11. Plot showing the variation of time taken for 50 time steps versus geometric complexity of the body. 49 152 vortex blobs are used in the simulation and 768 equal sized panels are used for the body.

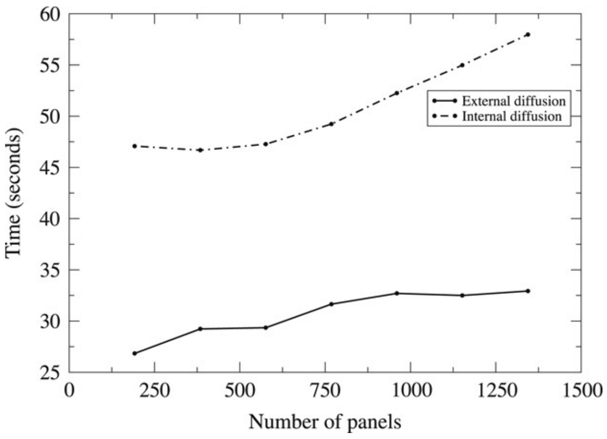

Fig. 12. Plot showing the variation of time taken for 50 time steps versus number of panels used. 49 152 vortex blobs are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction).

Fig. 12 plots the variation of the computational time as the number of panels is changed for a given number of particles (49 152) and complexity (43). Here again, despite a seven fold increase in number of panels there is only a 25% change in the time taken. As the number of panels increases, the time taken increases almost linearly. This indicates that the algorithm is efficient. Fig. 13 plots the variation of the computational time versus number of blobs used in the simulation. As expected, this is linear. 576 equal sized panels are used for the body and the complexity of the body chosen is 43.

For a simulation of diffusion inside the body at level 3 (complexity 43) with 768 panels and 49 152 particles the maximum number of cause and effect particles is chosen as 1000. This ensures that there is only one cell in the computation and hence all the particles are checked for intersections with all the panels. It is seen that this computation takes more than 45 times the time taken when the maximum number of causes and effects is set to 10. This clearly demonstrates the efficiency of the new procedure as compared to naive intersection checks without any domain decomposition.

In order to demonstrate the speed of the algorithm in realistic situations the uniform flow past two different complex body shapes is simulated. The reason for the simulations are two fold. The first is to demonstrate that the algorithm works in a realistic situation with advection and sheet blob conversions even when complex shapes are considered. The second is to provide an estimate of the efficiency of the algorithm as compared to the other computations (like the fast multipole velocity evaluation) involved in the simulation. For the simulation, an adaptive fast multipole method [4], using the modified domain decomposition has been implemented to compute the velocity due to the blobs on each

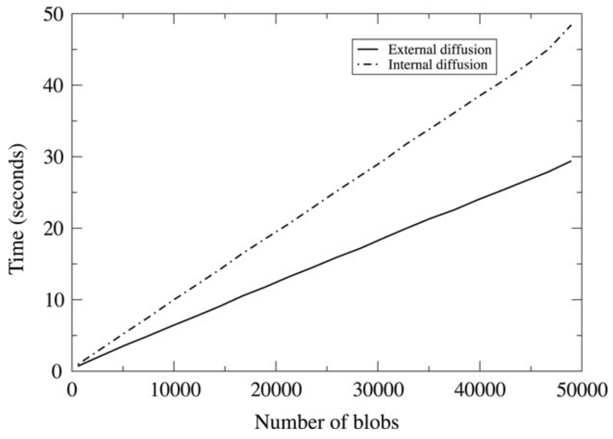

Fig. 13. Plot showing the variation of time taken for 50 time steps versus number of blobs used in the simulation. 576 equal sized panels are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction).

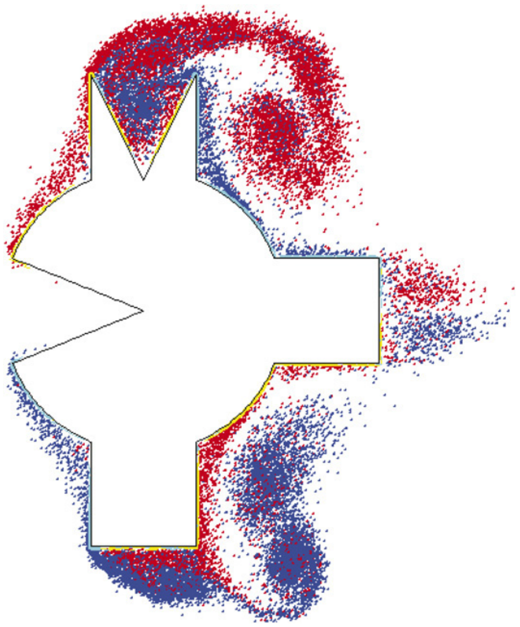

Fig. 14. Vorticity distribution for flow past the body at $T = 1.32$.

other. A linear vortex panel method is used to satisfy the no penetration condition on the boundary. A fast multipole technique [22] is also used to evaluate the velocity field due to the panels. Results for the random vortex diffusion scheme are shown. In the first case, the body has a geometry as shown in Fig. 14. 434 equal sized panels are used to discretize the shape. The viscous boxes use the same panels as the ones used to satisfy the no penetration on the body. For the simulation, the relevant parameters are chosen as $Re = 1000$ and $\Delta T = 0.0044$, where $T = Ut/R$, U is the free stream velocity and R is the radius of the cylinder. The numerical layer height is taken as $\epsilon = 0.011$ units. The resulting vorticity distribution at the end of 301 time steps is shown in Fig. 14. There are about 25 000 particles in the flow at this time. The red colored blobs and yellow colored sheets represent clockwise vorticity. The blue blobs and cyan sheets represent anti-clockwise vorticity. The strength of each blob is 7.868×10^{-4} units. As is evident, none of the particles have entered the body, indicating a successful implementation of the diffusion algorithm. At the end of 301 time steps, (i.e. a total time of $T = 1.32$), the computational time taken by the diffusion algorithm is about 3.44% of the total time taken by the fast multipole method. This corresponds to 2.5% of the time taken for the entire simulation.

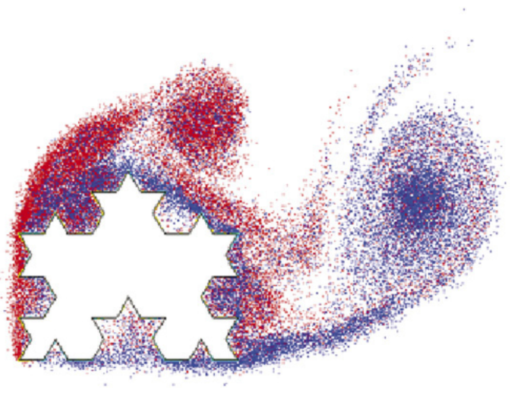

Fig. 15. Vorticity distribution for flow past a complex shape at $T = 2.5$.

In Fig. 15 the vorticity distribution for the flow past the shape given in Fig. 8 at level 2 and having complexity of 11 is shown. 576 equal sized panels were used to discretize the body. The flow is at an angle of 45° to the horizontal. The Reynolds number is based on the width of the body. The flow parameters are chosen such that the Reynolds number is 1000. The figure shows the vorticity distribution at the end of 1000 time steps using a $\Delta T = 0.0025$. At this time there are about 45 000 particles in the flow. It is evident that none of the particles have entered the body. For this simulation, the time taken for the diffusion algorithm is about 2.1% of the time taken for the fast multipole algorithm and only 1.6% of the total simulation time.

Considering the fact that the shapes are arbitrary, the above times for the two different simulations are certainly acceptable. From the earlier discussions, it is evident that an increase in the number of panels or geometric complexity will not change the computational time of the diffusion significantly. It is also evident that once a wake structure is created, the diffusion algorithm will perform more efficiently. This is because a greater fraction of the particles are in the wake and these particles require fewer intersection checks due to their distance from the body.

4. Conclusions

In this paper a new and efficient technique for random walks in the presence of arbitrary body geometries in two dimensions has been successfully developed. Since the technique uses existing components of the adaptive fast multipole algorithm, it is easier to implement. The method has been shown to work for flows using the RVM for diffusion. It is easy to modify the algorithm developed here to work for the vorticity redistribution technique. The modifications are also discussed. A measure for the geometric complexity of a body is introduced. The paper demonstrates that the entire scheme is very efficient. In principle, it should be possible to extend this work to three dimensions but this will necessarily be more involved and complex.

Acknowledgement

The authors would like to thank Prof. Vinita Vasudevan of the Department of Electrical Engineering, IIT-Madras, for her suggestions.

References

[1] G.-H. Cottet, P. Koumoutsakos, Vortex Methods: Theory and Practice, University Press, Cambridge, March 2000.
[2] E.G. Puckett, Vortex methods: An introduction and survey of selected research topics, in: R.A. Nicolaides, M.D. Gunzburger (Eds.), Incompressible Computational Fluid Dynamics — Trends and Advances, Cambridge University Press, 1991, p. 335.
[3] A.J. Chorin, Vortex sheet approximation of boundary layers, Journal of Computational Physics 27 (3) (1978) 428–442.
[4] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM Journal on Scientific and Statistical Computing 9 (4) (1988) 669–686.
[5] L. van Dommelen, E.A. Rundensteiner, Fast, adaptive summation of point forces in the two-dimensional Poisson equation, Journal of Computational Physics 83 (1989) 126–147.
[6] C.R. Anderson, An implementation of the fast multipole method without multipoles, SIAM Journal on Scientific and Statistical Computing 13 (4) (1992) 923–947.

[7] A.J. Chorin, Numerical study of slightly viscous flow, Journal of Fluid Mechanics 57 (4) (1973) 785–796.

[8] A.Y. Cheer, Unsteady separated wake behind an impulsively started cylinder in slightly viscous fluid, Journal of Fluid Mechanics 201 (1989) 485–505.

[9] P. Ramachandran, M. Ramakrishna, S.C. Rajan, Particle based flow solvers for incompressible flows in two dimensions: Impulsively started flow past a circular cylinder, Journal of the Aeronautical Society of India 53 (2) (2001) 102–110.

[10] N.R. Clarke, O.R. Tutty, Construction and validation of a discrete vortex method for two-dimensional incompressible Navier–Stokes equations, Computers and Fluids 23 (6) (1994) 751–783.

[11] P. Degond, S. Mas-Gallic, The weighted particle method of convection–diffusion equations, part 1: The case of an isotropic viscosity, part 2: The anisotropic case, Mathematics of Computation 53 (188) (1989) 485–526.

[12] S. Shankar, A new mesh-free vortex method. Ph.D. Thesis, The Florida State University, FAMU-FSU College of Engineering, 1996.

[13] S. Shankar, L. van Dommelen, A new diffusion procedure for vortex methods, Journal of Computational Physics 127 (1996) 88–109.

[14] P. Koumoutsakos, A. Leonard, High-resolution simulations of the flow around an impulsively started cylinder using vortex methods, Journal of Fluid Mechanics 296 (1995) 1–38.

[15] P. Ploumhans, G.S. Winckelmans, J.K. Salmon, Vortex particles and tree codes: I. flows with arbitrary crossing between solid boundaries and particle redistribution lattice; II. vortex ring encountering a plane at an angle, in: A. Givoannini, G.-H. Cottet, Y. Gagnon, A.F. Ghoniem, E. Meiburg (Eds.), Vortex Flows and Related Numerical Methods III (Proceedings of the Third International Workshop on Vortex Flows and Related Numerical Methods Toulouse, France, August 1999), in: European Series in Applied and Industrial Mathematics, ESAIM, vol. 7, 1999, pp. 335–348.

[16] P. Ploumhans, G.S. Winckelmans, Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry, Journal of Computational Physics 165 (2000) 354–406.

[17] K. Takeda, O.R. Tutty, D.A. Nicole, Parallel discrete vortex methods on commodity supercomputers; an investigation into bluff body far wake behaviour, in: Vortex Flows and Related Numerical Methods III, (Proceedings of the Third International Workshop on Vortex Flows and Related Numerical Methods Toulouse, France, August 1999), in: European Series in Applied and Industrial Mathematics, ESAIM, vol. 7, 1999, pp. 418–428.

[18] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 73 (1987) 325–348.

[19] J. Makino, Yet another fast multipole method without multipoles — pseudo-particle multipole method, Journal of Computational Physics 151 (2) (1999) 910–920.

[20] J.H. Strickland, R.S. Baty, Modification of the Carrier, Greengard and Rokhlin FMM for independent source and target field, Journal of Computational Physics 142 (1) (1998) 123–128.

[21] P. Ramachandran, M. Ramakrishna, S.C. Rajan, An efficient vortex diffusion algorithm for flow past arbitrary bodies in two dimensions. Technical Report AE:CFL:TR:2001:1, IIT-Madras, Computers and Fluids Lab, Dept. Aerospace Eng. IIT-Madras, Chennai, INDIA - 600 036, 2001.

[22] P. Ramachandran, S.C. Rajan, M. Ramakrishna, A fast, two-dimensional panel method, SIAM Journal on Scientific Computing 24 (6) (2003) 1864–1878.

[23] P. Ramachandran, S.C. Rajan, M. Ramakrishna, A fast multipole method for higher order vortex panels in two dimensions, SIAM Journal on Scientific Computing 26 (5) (2005) 1620–1642.

[24] H. Lin, M. Vezza, R.A. McD Galbraith, Discrete vortex method for simulating unsteady flow around pitching aerofoils, AIAA Journal 35 (3) (March 1997) 494–499.

[25] I. Taylor, M. Vezza, Prediction of unsteady flow around square and rectangular cylinders using a discrete vortex method, Journal of Wind Engineering and Industrial Aerodynamics 82 (1999) 247–269.

[26] I. Taylor, M. Vezza, Calculation of the flow around a square section cylinder undergoing forced transverse oscillations using a discrete vortex method, Journal of Wind Engineering and Industrial Aerodynamics 82 (1999) 271–291.

[27] P. Ramachandran, Development and study of a high-resolution two-dimensional random vortex method. Ph.D. Thesis, Department of Aerospace Engineering, IIT-Madras, 2004.

[28] H.-O. Peitgen, H. Jurgens, D. Saupe, Chaos and Fractals: New Frontiers of Science, Springer-Verlag, New York, 1992.