RECEIVED
SEP 01 2000
OSTI

AIAA 2000-2361

**AN INVESTIGATION OF TWO-
DIMENSIONAL CAD GENERATED
MODELS WITH BODY DECOUPLED
CARTESIAN GRIDS FOR DSMC**

Thomas J. Otahal, Michael A. Gallis, and Timothy J. Bartel
Engineering Sciences Center
Sandia National Laboratories
Albuquerque, NM 87185-0827
USA

**34th AIAA Thermophysics
Conference**
19-22 June 2000 / Denver, CO

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

AN INVESTIGATION OF TWO-DIMENSIONAL CAD GENERATED MODELS WITH BODY DECOUPLED CARTESIAN GRIDS FOR DSMC

Thomas J. Otahal⁺, Michael A. Gallis⁺, and Timothy J. Bartel[†]

Engineering Sciences Center
Sandia National Laboratories¹
Albuquerque, NM 87185-0827
USA

Abstract

This paper presents an investigation of a technique for using two-dimensional bodies composed of simple polygons with a body-decoupled uniform Cartesian grid in the Direct Simulation Monte Carlo method (DSMC). The method employs an automated grid pre-processing scheme beginning from a CAD geometry definition file, and is based on polygon triangulation using a trapezoid algorithm⁴. A particle-body intersection time comparison is presented between the Icarus DSMC⁸ code using a body-fitted structured grid, and using a structured body-decoupled Cartesian grid with both linear and logarithmic search techniques. A comparison of neutral flow over a cylinder is presented using the structured body fitted grid, and the Cartesian body de-coupled grid.

Nomenclature

<i>uh</i>	cut cell triangulation half
<i>lh</i>	cut cell triangulation lower half
<i>n</i>	total number of grid cells; number of polygon vertices
<i>m</i>	total number of triangles in body interior
<i>P</i>	general polygon
<i>D</i>	discriminator function
<i>fom</i>	figure of merit
<i>ip</i>	set of all ray-body intersection points
<i>A(i, j)</i>	function that returns the intersected area between triangle <i>i</i> and triangle <i>j</i> in two-dimensions
<i>i, j</i>	cell indices; triangles

Introduction

The direct simulation Monte Carlo (DSMC) method, developed by Bird⁵, is used to model highly rarified non-continuum gas dynamics using statistical

calculations of macroscopic flow properties on a background grid. The grid in DSMC is used for converting statistical properties to macroscopic flow properties and for choosing collision partners within a grid cell, and as such DSMC grids are not restricted to regular grid cells where all cell boundaries originate from a vertex.

The standard DSMC algorithm for particle movement is performed on a cell by cell basis using the current velocity vector of the particle to give traversal length and direction. Given a particles location in a cell, it is necessary for the simulation to keep a list of all neighboring cells in order to move the particle to a new cell in the event it has sufficient velocity to leave its current cell location during the time step.

Grid strategies for DSMC can be divided into three broad categories:

Unstructured body fitted: The topology and geometry for unstructured grids are completely irregular in order to conform to the body. These irregularities in cell structures necessitate additional calculations to determine if a particle has left a cell during the movement phase. In

⁺ Member of Technical Staff, AIAA Member

[†] Principal Member of Technical Staff, AIAA Member

¹ Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-ACO4-94AL85000.

addition, the neighbor list for each cell in the grid must be constructed before the simulation begins and is expensive to re-calculate in the event of dynamic grid refinement. Traditionally, domain decomposition methods are used for unstructured body fitted grids on parallel computing platforms. A complication arises when a particle moves outside of the local domain of a processor, since the body geometry has been split up among each of the processors. Therefore, time-accurate simulation can be complex. Unstructured grids can be generated for complex bodies automatically by many commercially available grid generation software systems.

Structured body fitted: A structured grid has a regular topology and an irregular geometry. The irregular geometry of a structured grid can be used to conform it to curved shapes, but complex curvatures require the grid to be divided into distinct structured regions of varying curvature in order to accommodate the body. The process of building a structured body-fitted grid requires great amounts of user interaction, and is generally a lengthy process. The regular topology of the grid allows fast calculation of particle movement outside of the cell. An example of a body fitted grid for circular cylinder is given in Figure 1.

Figure 1. Structured body fitted grid Figure.

Structured body-decoupled: A structured body decoupled grid does not conform to the body, and can have a regular topology and geometry. This allows for fast particle movement through the grid cells, but demands more pre-processing calculation of cells cut by the boundaries of the body. The principle advantages of using decoupled Cartesian grids in DSMC are fast grid generation and fast particle movement calculations through gas phase cells. These two advantages come at

the cost of slower particle-body intersection determination for arbitrary bodies, and additional calculation to determine good cell shape functions for grid cells intersected by a portion of the body. An example of a body-decoupled grid is given in Figure 2.

In this work we investigate the applicability of structured body-decoupled grids to particle simulation codes and in particular in DSMC.

There are three primary motivations for this work:

1. Decrease the time involved in creating a body geometry description and corresponding grid to achieve results that are physically significant.
2. Simplify particle movement through the grid, and defer all grid complexity to interactions with the body surface.
3. Allow the body to be composed of any number of closed simple geometric primitives (polygons). The shape of the body is not altered to conform to a grid, and preserves the boundary information contained in a CAD geometry file describing it.

Figure 2. Structured body decoupled grid Figure.

The proposed structured body de-coupled uniform Cartesian grid best meets these three requirements, and is implemented for a two-dimensional CAD generated body geometry in this paper.

The de-coupling of the grid from the body raises an immediate question how to determine a particle-surface intersection efficiently, since the grid no longer contains exact information about the boundaries of the body. A shaded body with 15 numbered cut cells is shown in Figure 3.

Figure 3. Cartesian grid with numbered cut cells.

A novel implementation of particle moves is used, based on the observation that locating particle-object intersection points is exactly analogous to ray tracing in computer graphics.

Method Description

The Icarus DSMC⁸ code was modified to accommodate the new grid strategy and movement algorithm for two dimensional bodies and grids. The user first draws the desired body shape for modeling in a CAD system with appropriate dimensions and spatial location. The body shape must be composed of simple polygons. A uniform Cartesian grid is also specified using a text file that is read by the Icarus code.

Before the simulation begins, each of the grid cells must be pre-processed by calculating how much of the cells area lies outside of the closed body. This is accomplished by first finding a triangulation of each polygon constituting the body, and using the resulting collection of triangles to calculate cell intersection areas. Grid cells to be pre-processed are divided into an upper triangle half (*uh*), and lower triangle half (*lh*). Let $A(i, j)$ return the shared intersection area between triangle *i* and triangle *j*. If *n* is the total number of grid cells over which *i* iterates, and *m* is the total number of triangles in the triangulation of the body interior, then each cell area in the grid is calculated as:

$$area_i = \sum_{j=1}^m (A(uh_i, tri_j) + A(lh_i, tri_j))$$

The calculation of the cell area can be zero, in which case the cell is tagged as being completely inside of the body. If the cut cell area is equal to the cell area, then the cell is tagged as being completely outside of the body. Cells in which the calculated area outside of the body is greater than zero and less than the cells original area are tagged as cut cells.

The particle movement routine in Icarus calls an Intersection subroutine that returns a true/false intersection value, intersection coordinates, and a surface normal at the point of intersection for a given particle start coordinates and end coordinates.

Polygon Triangulation Algorithm

Shapes with high degrees of curvature require a large number of segment edges in the simple polygons used to describe them. This large number of edges may cause the pre-processes to become slow if care is not taken in selection of the algorithm to perform polygon triangulation. A brief sketch of the algorithm used is provided, but please see O'Rourke¹ and Seidel⁴, for full implementation details.

Polygon Triangulation Problem statement:

*Given the coordinates of the *n* vertices of a polygon *P*, find the *n-3* diagonals that partition *P* into *n-2* triangles.*

Figure 4 illustrates a convex polygon of 5 vertices triangulated with 2 diagonals into 3 triangles.

Figure 4. Triangulated convex polygon with 5 vertices.

The "ear removal" algorithm for polygon triangulation is well known, and runs in time $O(n^2)$ when *n* is the number of vertices in the polygon. Research in Computational Geometry has produced algorithms that run in time $O(n \log n)$ using a monotone partition of a polygon achieved by

trapezoidalization. A polygon P can be defined as monotone with respect to a line L , if every line orthogonal to L intersects the boundary of P in at most one point or line segment. The property of a monotone polygon chain allows the vertices of the polygon to be sorted with respect to the line of monotonicity in linear time. The polygon can then be triangulated in linear time by simply traversing the sorted list of vertices and removing a triangle formed by joining with two vertices immediately above the vertex under consideration.

This strategy is not completely general, however, since not all polygons are monotone. Thus, any algorithm wishing to exploit this property must first divide a given polygon into monotone sections, and then triangulate each monotone piece in linear time.

The basic algorithm from O'Rourke¹ is:

1. Sort polygon vertices by y coordinate.
2. Perform plane sweep to construct trapezoidalization.
3. Partition into monotone polygons by connecting from interior cusps.
4. Triangulate each monotone polygon in linear time.

The first step of the algorithm takes time $O(n \log n)$ for a general polygon. Step two forms a trapezoidalization of the polygon by maintaining a data structure of intersected edges along a horizontal sweepline through every vertex in the sorted list formed in step one. A polygon containing no interior cusps is monotone, so these vertices must be removed by joining diagonals from the interior cusps to vertices in an adjacent sweepline. The resulting monotone polygons are then triangulated in linear time.

Complex body shapes can contain thousands of vertices in the polygons used to describe them. A monotone polygon triangulation algorithm provides a $O(n \log n)$ running time to ensure a tractable calculation for polygons with many vertices.

Space Partitioning using KD-Trees

The particle-body intersection query used for the movement subroutine is analogous to the problem of ray tracing in computer graphics. The query is illustrated in Figure 4, and is formally stated as:

Given a ray r with starting coordinate p and termination coordinate q , what is the first point of intersection i with a body in scene S ?

Figure 5: Ray r intersects the first body at point i .

Ray tracing is used to generate highly detailed 2D images of 3D geometric scenes using virtual light rays emitted from every pixel in the 2D image. Each virtual light ray interacts with bodies in the scene it traverses by being absorbed, scattered, or reflected upon intersection with a body. The virtual light ray is projected a finite length into the scene, and may intersect many objects during its projection at which time the ray acquires new color information for its parent pixel. A typical image may require the transmission of roughly one million light rays to form the final image. In DSMC, the completion of single time step may involve the movement of an equal number of simulation particles.

The basic linear time algorithm for finding the set of all particle-body intersection points given n objects in the scene, a query ray r , and a routine $Int(r, obj)$ that returns the intersection point of the ray with an object if one occurred is:

$$ip = \bigcup_{i=1}^n Int(r, obj_i)$$

The members of the set ip must then be sorted by distance from the starting coordinate p along ray r in linear time to find the first point of intersection with a body in the scene.

A great amount of work has been performed in ray tracing intersection search algorithms to achieve a running time better than linear in the number of objects in the scene. The algorithm employed by Cleary et al³ utilizes a uniform grid laid over the scene to facilitate skipping over empty space before an intersection test with the body is necessary. Other algorithms employ a bounding box around all objects in the scene to delay intersection tests until a ray is very close to the object.

The algorithm introduced by Fussell and Subramanian² is implemented for our method to address the problem of intersection search. A brief overview of the algorithm is given here, and the full implementation can be found in the reference. The algorithm constructs a balanced search tree for the objects in the scene, and can be used to find the first point of intersection with the body in $O(\log n)$ running time.

The algorithm used is based on a multidimensional binary search tree called a KD-tree. The number of children at each node in the tree is given by the discriminator D , and an integer between 0 and $k-1$:

$$D = l \bmod k$$

For three dimensions $k = 3$, and $D \in (0,1,2)$.

The KD-tree is recursively constructed for a scene using the quantity fom to determine the best choice for scene partition at the current level of the tree. Each node in the tree partitions the domain into two spaces via a line segment in two-dimensions or a plane in three-dimensions. The choice of partition location at each node in the tree is based on the body geometry in the space under consideration. The equation used to decide the fitness of a particular partition is:

$$fom = absolute(l_cnt - r_cnt) + shr_cnt$$

r_cnt and l_cnt are the number of objects on each side of the partition, and shr_cnt is the number objects split by the partition. Objects split by the partition are duplicated and passed down all child nodes of the tree. The best choice for fom is found by searching iteratively along each of the coordinate axis directions in the scene, with a resolution parameter governing termination for search in a particular direction.

The KD-tree construction process will ideally yield a balanced tree structure that can be searched in logarithmic time in the number of objects in the scene. The following code is used to traverse the tree structure:

1. If at leaf node, use linear intersection to see if ray has reached the body. If true, return.
2. Else, classify ray with respect to the partition line as LR, RL, L, or R.
3. Recursively call the traverse routine with the children of the current node with respect to the ordering produced in step 2.

All objects are stored in the leaf nodes of the tree, so a linear intersection test must be performed to determine if the ray has hit the body. If the tree node is not a leaf,

then the ray needs to be classified with respect to the partition for the node. Rays classified as RL or LR must be passed to both child nodes of the tree. Rays classified as L or R must be passed to only one child of the current node. The traversal procedure is called recursively until an intersection point is found.

The construction of the KD-tree assures that the ray will find the first point of intersection along its travel path. Only objects immediately near the ray are searched for intersection, as governed by how many partitions for the scene are constructed. For a balanced KD-tree, each decision at a node effectively cuts the search space for object intersection in half.

The linear time particle-body intersection search algorithm is intractable for the purposes of this investigation. The KD-Tree provides a logarithmic search time for particle-body intersection, and is scalable to bodies with large numbers of edges.

Code used

The Sandia DSMC code Icarus was used for the simulations. Icarus was developed following Bird's approach⁵ by Bartel and co-workers⁸ for massively parallel applications, and can be run on a variety of computational platforms. The test cases presented here were executed on an Intel 400 MHz Pentium II PC. Each run of the cases used approximately 200,000 particles and did not exceed 7 hours of run time.

Test Case Results

The new gridding scheme was tested using a cross flow cylinder case. Despite the apparent simplicity of the simulated geometry, the cylinder poses a significant challenge for intersection search, since in this case the cylinder was discretized with 232 facets. A linear search algorithm for this case proved to be unacceptable in terms of running time, since the algorithm was searching for potential interactions with all of the facets. The KD-tree based searching algorithm can efficiently search for intersection in any arrangement of the 232 facets, allowing general shapes not described by simple equations to be modeled. The cylinder problem therefore presents a significant challenge for the performance of the method.

For the purposes of our comparison test two grids were required. The first one was a cartesian 75x75 background grid. A cylinder that has its origin at the beginning of the axis and a radius of 1 was overlaid (see figure 1). The second one, a body fitted grid of 6,500 cells giving approximately the same resolution was (see figure 2) also used for the simulation.

The ambient flow was assumed to be pure argon. The conditions of the simulation are given in table 1.

Gas	Ar
Velocity	1365 m/s
Number density	$10^{19} \text{ (m}^{-3}\text{)}$
Temperature	180 K

Table 1. Free stream properties.

The simulation used a timestep of $1\mu\text{s}$ and steady state time was observed after 20 ms, long enough for a compression flow to reach steady state.

Figures 3a and 3b present the density contour lines for the two simulations. The flow moves the positive x direction. We see that the compression of the flow on the side of the cylinder forms a boundary layer that in turn reflects the incoming particles creating a diffuse shock layer. The number density on the surface reaches a maximum of about 10 times the free stream density. Unlike the body fitted grid, no adaptation close to the surface was made to the cartesian grid.

Figures 4a and 4b present the temperature contour lines. The temperature increases from its free stream value of 300K to a temperature of 1800K in the area of the shock layer from $x = -1.64 \text{ m}$ to $x = -1.4 \text{ m}$. We note that both the cartesian and the body fitted grid produced almost identical results. This is to be expected since the physics of the simulation are identical and indicates that the implementation of the cartesian grids was correct.

Both the density and temperature contour lines are distorted close to the boundaries for both cases. This disturbance is an effect of the interaction with an incoming flow boundary.

The cartesian grid for this case turned out to be more expensive than the body fitted grid by a factor of 7%. This overhead is associated with the more complex surface collision searching algorithm of the cartesian grid. It should be noted that this test case is particularly unfavorable for the cartesian grid since most of the particles are clustered close to the cylinder creating a significant demand for surface interactions. As previously said the cartesian grid treats the cylinder as any shape with the same number of facets.

This example demonstrates some of the advantages and the disadvantages of the cartesian grid.

- 1) Although the search for particle location is faster in the cartesian grid, the surface interactions are slower, resulting in a modest increase in the particle movement time.
- 2) Setting up the cartesian grid once the CAD input is given is trivial and requires no skill from the side of the user.

Conclusions

In this paper the applicability of structured body-decoupled cartesian grids to particle simulations and in particular DSMC was examined. Cartesian background grids decoupled from the surface description offer an efficient and elegant way of modeling complicated surfaces that cannot be described analytically. The simplification of pre-processing comes with a modest overhead of more expensive surface collisions and additional pre-processing costs for cut-cell area determination. Careful implementation of the algorithms used for particle-body intersection and cut cell area determination is necessary to keep the computational time costs at a minimum. Body fitted structured grids for simple shapes are likely to be faster than any other scheme. There are cases, however, when constructing a body fitted grid for a complex shape requires prohibitive amounts of user time. Structured body-decoupled cartesian grids allow faster time to calculation for a reasonable increase in simulation running time. Cartesian grids also offer an attractive way to perform local adaptation and refinement, due to the less restrictive grid alignment requirements for DSMC. Work is currently in progress to extend the method to three dimensions.

Acknowledgements

The authors would like to thank Dr. Wahid L. Hermina, Dr. Justine E. Johannes and Dr. Jeff Payne of Sandia National Laboratories for their helpful comments. This work was performed at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by the Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References:

- ¹J. O'Rourke, "Computational Geometry in C, Second Edition", Cambridge University Press, 1998.
- ²D.F. Fussell, K.R. Subramanian, "Fast Ray Tracing Using K-D Trees", Masters Thesis, The University of Texas at Austin, 1987.
- ³J.G. Cleary, G. Wyvill, "Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision", The Visual Computer (1988) Volume 4, pages 65-83.
- ⁴R. Seidel, "A Simple and Fast Incremental Randomized Algorithm for Computing Trapezoidal Decompositions and for Triangulating Polygons", Computational

Geometry: Theory and Applications Volume 1 (1991),
pages 51-64.

⁵G. A. Bird, "Molecular Gas Dynamics and the Direct
Simulation of Gas Flows", Oxford University Press, 1994.

⁶R. N. Nance, R. G. Wilmoth, H. A. Hassan, "A
Comparison of Grid-Definition Schemes for DSMC",
AIAA 96-0604, 34th Aerospace Sciences Meeting and
Exhibit, January 15-18, Reno, NV.

⁷D. F. Rault, "Towards an Efficient Three-Dimensional
DSMC Code for Complex Geometry Problems", AIAA
1992.

⁸T. J. Bartel, S. Plimpton, J.E. Johannes, J. J. Payne,
"Sandia Report SAND 96-0591 Sandia Corporation,
Albuquerque, NM".

Figure 3a. Cartesian grid, Number Density

Figure 3b. Body fitted grid, Number density

Figure 4a. Cartesian grid, Translational temperature

Figure 4b. Body fitted grid, Translational temperature