MULTIPLE STATIONARY AND MOVING BOUNDARY HANDLING IN CARTESIAN GRIDS

Kerem Pekkan

Georgia Institute of Technology, Atlanta, GA., U.S.A. kerem.pekkan@bme.gatech.edu

ABSTRACT

A Cartesian grid generation methodology is developed for unsteady control volume computational fluid dynamic (CFD) solvers. Arbitrary combinations and numbers of moving and stationary boundaries are allowed to exist in the two-dimensional Cartesian grid template. Specific definitions for the possible cases, CFD solver requirements and moving geometry handling algorithms of Cartesian grids are described. Applications are selected from bio-fluid dynamics and aerospace propulsion to demonstrate the capability of the method.

Keywords: Cartesian grids, moving boundaries, computational fluid dynamics

1. INTRODUCTION

Moving boundary problems of nature[1][2] and technology[3][4][5] is an attractive research area. Besides the trivial computational difficulties of interface movement, additional complexity is introduced due to surfaces that are irregular and not aligned with the orthogonal coordinate planes. Moreover in some problems multiple stationary and moving surface combinations can exist.

In moving boundary problems, in addition to the structured[6] and unstructured boundary confirming grids[7], embedded Chimera like moving zonal procedures[8] and level-set methods[9], Cartesian or cut-cell techniques can be used to define the solution domain. Cartesian grid approach is an efficient, versatile and rapid geometry definition[10][11][12]. Grid types consist of irregular cut-cells and uniform rectangular volumes that are located near the boundaries and in the far field, respectively. As will be demonstrated Cartesian grid intersections depend on the local topological character of the boundary curves, therefore making the Cartesian approach problem independent and suitable for broad research interests.

Unless a structured boundary layer grid (obtained by normal offsetting of the boundary) and optional tetrahedral transition elements are introduced, Cartesian grids may not be in the best possible quality for the given application but still can be preferred due to their high turnaround time and wide ranges of applicable geometry. For boundaries with many sharp corners it is another alternative to unstructured (tetrahedral) and multi block structured grids.

When Cartesian grids are used, even in a simple two-dimensional problem, triangular, quadrilateral and pentagonal elements may coexist. Since the geometric possibilities are close to unlimited a systematic approach is necessary. Structured programming, CAD algorithms and approaches from constructive solid geometry are employed for the realization of this highly geometric task.

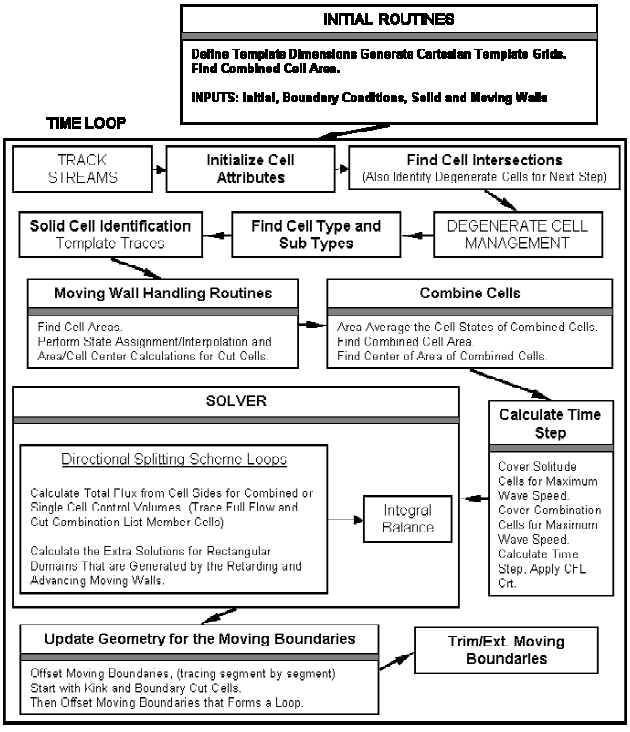

Figure 1. Solution algorithm and time loop.

Fig. 1 shows the main components of a moving boundary CFD solver that employs Cartesian grids. Details of the grid generation routines will be described next, followed by selected applications.

2. DEFINITIONS

The Cartesian grid terminology for stationary boundaries is quite complete in the literature[13][14]. In this section an extension will be made to cover domains that involve multiple stationary and moving boundaries.

In Fig. 2, a rectangular "template" containing stationary (solid) and moving boundaries, having five and seven "square cells" along x- and y- directions is plotted. Given the length and width of the template, integer number of square grids can be generated inside. Template boundary sides are termed as EAST, WEST, SOUTH and NORTH, where inflow/outflow, reflective, transmissive, injecting, moving wall boundary conditions can be specified. Also for the side of any cell inside the template, a boundary condition can be assigned when needed for specific applications.

In all problems, solid and moving walls are bordering the flow field of interest. "Curves" that form the boundaries of solid/moving walls are specified as line segments, in an order so that when the curve parameter increases, solid bodies are always on the left.

In a general problem there may be moving and solid boundaries connected in series, as shown in Fig. 2. These series of curves are termed as "streams." In streams intersection points of moving and solid curves are labeled as "kink points". A stream can be composed of a single moving or solid curve. Streams are allowed to loop, forming voids or closed bodies, with coinciding start and end points.

A boundary curve, passing through a template cell, intersects its sides in two points. Tracing the curve with the solid region being on the left-hand side, first intersection is denoted as "point 1a" and second as "point 2a." Cell intersection points are stored with respect to coordinates relative to the cell.

If number of intersection points in a cell exceeds two, cell types that are not covered by the solvers' cell-type-domain may appear. These cells need special treatment and named as "degenerate cells" in the Cartesian literature [13]. In that case, extra intersection points will be labeled and stored as "point 1b" and "point 2b," to be used in degenerate cell handling routines, §3.7 A simple degenerate cell example is given in Fig. 2.

3. CARTESIAN GRID GENERATION ROUTINES

Cartesian Grid Solvers does not require a separate grid

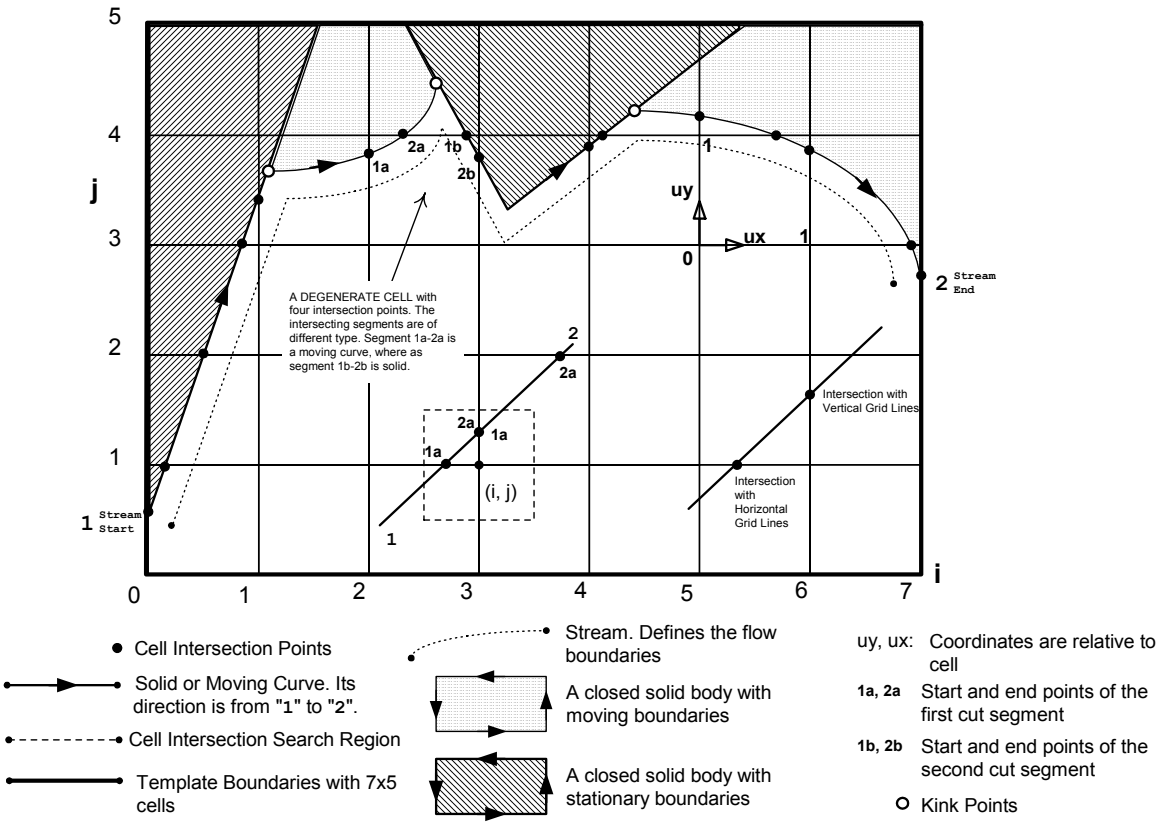

Figure 2. Terminology for cell intersections

generator program, while for arbitrary moving complex geometries; generation of the template grids and finding local intersections is a complicated geometric task.

Unlike boundary-conforming grids where global re-meshing, at each time step, alters the positions of grid points, Cartesian grid points are stationary. Thus for Cartesian control volumes that are not cut by a moving boundary, a geometric conservation law or Jacobian transformations is not needed.

3.1 Modeling Solid/Moving Wall Boundaries

Curves that form the boundaries of solid or moving walls are approximated with line segments. A line segment is the lowest object of the stream structure. It is defined by the parametric equation of straight line. Parametric representation of line segments is found useful especially in finding cell intersections. For this type of representation position vectors of the start and end points of the straight line are needed. A single curve parameter defines the position of any point on the line segment. Segment center velocity and its magnitude are also stored to allow for variable offsetting along a moving curve types.

The curve structure used in the code composed of the line segments, number of line segments that make up the curve and type of curve, which may be moving or solid. Line segment sequence of each solid/moving curve is arranged so that solid bodies are always on left. The geometric information of the input curves are kept in the following structure:

```
struct curve {
              struct line seg[N_SEGMT] ;
              int type ;   /* SOLID or MOVE */
              int n_segmt ;      /* end indis of seg[] */
              };
```

3.2 Cell Size and Number of Cartesian Grid Lines

Since the domain is rectangular and the cells are square, number of grid lines in x- and y- directions is dependent on each other. First the minimum number of grids in each direction is calculated, which is the coarsest possible grid for the given template dimensions. Then, if an extra refinement is required, number of grids is increased in both directions with the same ratio. If the template side dimensions are not whole numbers, then they should be expressed in rational form, i.e. numerator over denominator. In that case, to calculate cell size and template grid line locations accurately, integer arithmetic needs to be performed.

3.3 Finding Streams

A general geometry is composed of arbitrary number of solid and moving curves. At the start of the problem, i.e. at initial state, some part of the solid boundaries may be covered with moving curves and may not saw the flow region. For a regressing material this situation is sketched in Fig. 3. As the solution proceeds, these solid boundaries will expose out and start affecting the flow. To obtain the transient solution without any restart, at each time step the flow boundaries should be identified. Curves defining flow boundaries or wetted areas are called “streams” and the corresponding process is called “stream generation.”

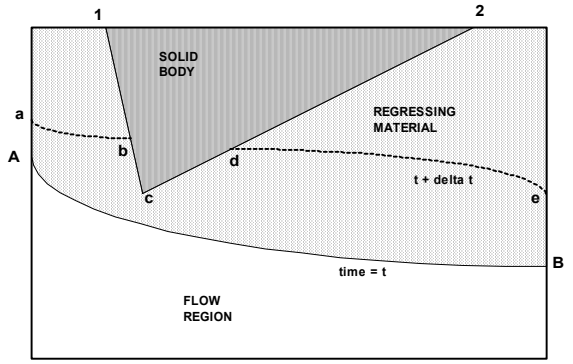

Figure 3. A Solid Boundary Exposing out Which is Initially Covered with a Regressing Material. (A-B is the Stream at Time t. a-b.b-c.c-d.d-e is the Stream at a Later Time.)

In Fig 3, initially there is one moving curve (A-B) and one solid curve (1-c.c-2). Solid curve is composed of two line segments and the initial stream is made up of the moving curve (A-B) only. As the boundary moves, at a later time step the number of moving curves increases to two. The stream is now made up of three curves (a-b, b-c.c-d, d-e). In a general problem the number of streams may be more than one. Algorithms developed in this study allow multiple streams, cover situations involving merging and break-up of moving boundaries and can also generate loop streams.

If there are no moving walls and all of the geometry that defines fluid boundaries is solid, then each solid wall is assigned as a new stream. If there are moving walls, new streams are generated by tracing moving and solid curves alternatively. In this trace, “A stream can form a closed loop or start and end at a template boundary” is the basic rule. Until this rule is satisfied, each trimmed/extended moving curve is traced first in its start direction and then towards its end. During these traces, intersections with other curves will be detected. Each new detected curve during this trace is kept in the order as a member of the generated stream. Boundaries that form loops can also be detected by this procedure.

3.4 Cartesian Cell Intersections

The basic geometry is illustrated in Fig 1. For each line segment, of each stream, intersections with vertical and horizontal grid lines are searched. If an intersection is found, intersection point and its type (moving or solid) are stored in the cell structure relative to cell coordinates. The coordinates should be specified relative to square cells because of the accuracy considerations. This algorithm is different than the one proposed in [1]. In that study boundary curves were traced and cell coordinates are

defined as integer variables taking discrete values to overcome the accuracy problem. The intersection routine that is developed here considers the sense of each line segment and differentiates different cases. Including specific orientations of segments, like cases parallel to Cartesian grid lines. Two intersections are allowed and typical for each Cartesian cell. If more than two intersections are found, their positions are stored for degenerate cell considerations.

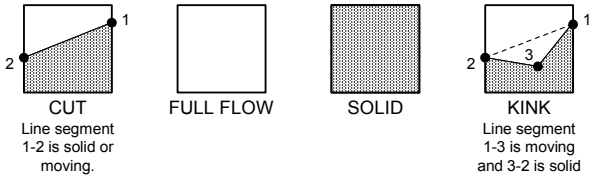

Figure 4. Basic cell types (SOLID, FULFLOW, CUT-M/S, KINK)

For geometries involving moving and solid boundaries, there are five basic cell types. These are: full flow, solid, cut-solid, cut-move and kink cells, Fig. 4. Cut cells contain a single curve segment, which may be moving or stationary. Kink cells are cut cells where a solid curve ends and a moving curve starts or vice versa. The same convention also holds for segments inside cells: the solid part is on left, in the direction of curve parameter increase.

3.5 Cell Structure

In the solution domain each square formed by the grid lines defines a control volume, named as cell. Number of cells are equal to the number of grids. For each cell, besides flow variables the following information is also stored:

- Basic cell type (Fig. 4)
- Cell sub-type (Fig. 5)
- Position vector of first and second intersection points.
- Intersection types of first and second intersections. (Which depends whether the cell is intersected at that point by a solid or a moving curve.)
- Position vector of first and second degenerate intersection points and their types.
- Position vector of the kink point.
- List number that the cell is combined.
- Cell area.
- Position of cell center.
- The boundary condition specified for any of the cell sides.

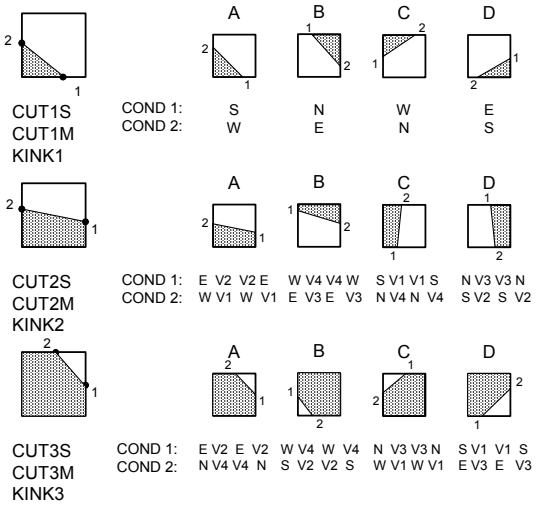

Figure 5. Cell sides (E, W, S, N) and four corner vertices (V1, V2, V3, V4) are labeled, defining the possible locations of the intersection points.

3.6 Cell Sub-Types

The intersection type (cut segment crossing whether in or out of the Cartesian cell), and the cell side each intersection point is located determine the cell sub-type. As a solver convention, these sub-types are grouped and presented in Fig. 5. Depending on the conditions given in Fig 5, the solver differentiates 48 different cell sub-types. Although only cut-moving/solid (CUT-M/S) cells are drawn, for each cut cell sub-type there is also a corresponding KINK cell with intersection points at the same locations. In that case the kink point, point-3 in Fig. 4, is located at an arbitrary position inside the Cartesian cell.

3.7 Degenerate Cells

In the solution domain, if arbitrary intersections with the input geometry are allowed, some cells that are not recognized by the Cartesian solver may appear. These cells are named as "degenerate cells" in the Cartesian literature. (A simple example is a cell with more than two intersections, Fig. 2)

PROBLEM TYPE	NUMBER of UNIT GEOMETRIES
P1	*32*
P2	*8*
P3	*64*
$P4_{diagonal}$	*16*
P4	*88*

Table 1. Number of possible unit geometries for each degenerate cell problem type.

By increasing grid size or slightly shifting the input geometry some of the problematic cases can be overcomed. However such remedies work only for bodies that are not moving. For applications involving continuously changing shapes and offsetting, these problematic geometries must be identified and suitably approximated.

Depending on the number of intersections in the degenerate Cartesian cell, geometrically possible problem types can be grouped in to four. These problem topologies will be labeled as P1, P2, P3 and P4 cells (With one, two, three and four intersections in a Cartesian cell respectively.) For each problem type, the rotations and symmetries of the basic geometry should be considered, together with the type (moving or solid), of the intersecting boundary. The total number of unit operations that is taken into account for each problem type is given in Table 1.

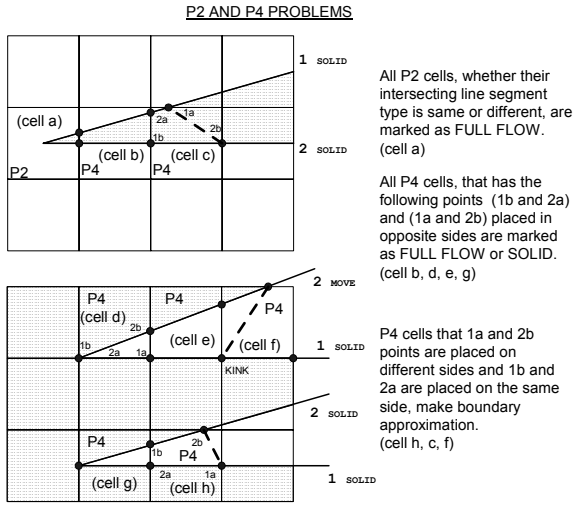

Figure 6. P2 and P4 type degenerate cells: In all figures dashed lines represent the final approximated boundary after degenerate cell corrections. Both line segments are of same type (Both MOVE and both SOLIDS). For the definitions of cell intersection points: 1a, 2a, 1b and 2b refer to Fig. 2.

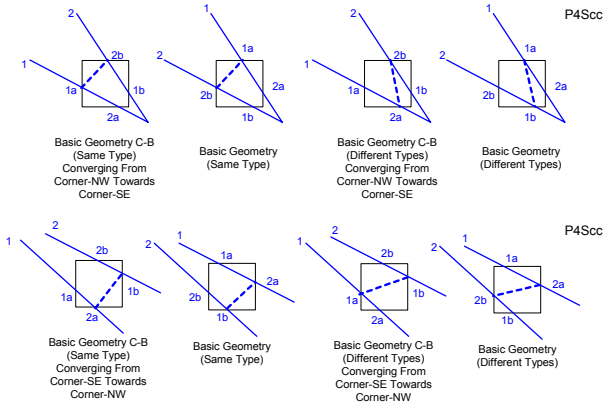

Figure 7. Diagonal P4 problem -one of the four sub-types. The four cases that are shown on the left are same type of curve intersections. Different type intersections are plotted on right.

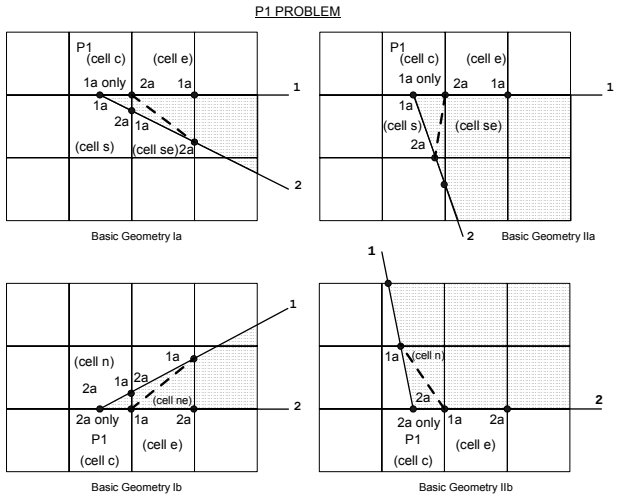

Figure 8. P1 problem: P1 problem is possible if the segments that make a sharp corner are of the same type and one of the segments coincide with the template grid lines. There are two basic geometry types. (And each type has two variations, a and b) For the basic geometry type 1a (top left figure), the ambiguity in cell intersections is corrected by; Mark (cell c) and (cell s) as FULL FLOW. Move Point 1a of (cell se) to Point 2a of (cell e). Each of the four plotted topologies has also four different orientations depending on the position of the cell that is marked 1a/2a-only.

To disclose the scope of the work some examples of the primary geometries and rules that are used in the code are given in the Figs. 6, 7 and 8. For degenerate cells with intersections of different curve types (solid/move), approximated geometry is sharp and defines a kink type of cell. Without considering these topologies a general moving body Cartesian solver is not possible.

3.8 Marking Solid Cells

Procedures describe in the previous sections specify the geometric properties for each cut cell in the solution domain. The remaining cells are either solid or full flow. For solid cells, no flow solution is needed. Therefore they must be distinguished from the fluid flow cells before starting the solution.

The procedure that is used is similar to the one used in [1]. All the Cartesian cells are traced first horizontally and then vertically. During each trace, cell type does not changed until, a cut cell is reached, and at that cell, the marking type is reversed and switched to either FULFLOW or SOLID. The tracing proceeds with this marking type afterwards. By taking into account the detected cut cell sub-type, initial cell type assumption and previous type marks are corrected. The algorithm is tested in various geometries and found to be working flawlessly in all cases considered so far.

The solid cell marking procedure of [1] does not take the cut cell sub-type into account. For this reason an extra trace, either in horizontal or vertical direction is needed. Even with this extra trace, ambiguous geometries are still possible. In this study since in marking type switches,

depending on the cut cell sub-type initial cell type assumption is corrected and the number of horizontal and vertical traces are decreased to two.

3.9 Cell Combinations

Intersections of arbitrary line segments may produce tiny cells, which minimize the time step size. If a fast Cartesian solver is the aim, these tiny cells must be combined and treated as a single control volume. The segment mid-point normal rule [1] is practiced here. Which will be referred as the best combination.

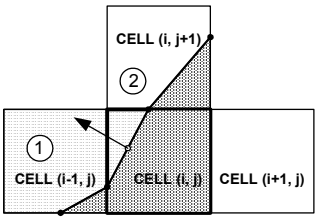

Figure 9. Cell (i, j) is combined with the first of the two alternatives.

Around confined regions and for cut cells with neighboring template boundaries, the best possible combination may not exist or the planned combination may produce an undesired size increase that decreases the local spatial accuracy. In order to acquire a consistent control volume size as much as possible, throughout the template, maximum three alternatives of the four neighboring cells are returned to the code in the order of the best combination possibility. In cases when the best combination cell does not exist other alternatives are considered in order, Fig. 9.

For all the combined cells in the template, cell states are area-averaged, area and cell center of the combined cells are found. A different list number is assigned to each combination.

In Fig. 10, Cartesian grid Information for various template locations of the letters ‘S’ and ‘A’ are plotted. Close-up regions demonstrate some critical locations and treatment of degenerate cells. The numbers shown on Cartesian cell centers, represent the combination list that the cell belongs. F, C and S stand for FULLFLOW, CUT and SOLID respectively. Due to the boundary movement, cell information is regenerated at each time step.

3.10 Boundary Offsetting, End Trimming/Extending

Prior to the next time step, segment center points are displaced in the wall velocity direction, for cells that contain a moving type cut segment,. New curve points for the moving wall are generated without loosing the curve parameter sense. Curves that form loops are separately detected and displaced. Moving curve ends are trimmed or extended to a nearest solid wall or a template boundary.

At the end of the trimming and extending process, each moving curve should start from or end at a solid boundary /template side, or form a loop, so that the stream generation algorithms that are discussed in the previous sections can trace the moving and solid curves without any breaks.

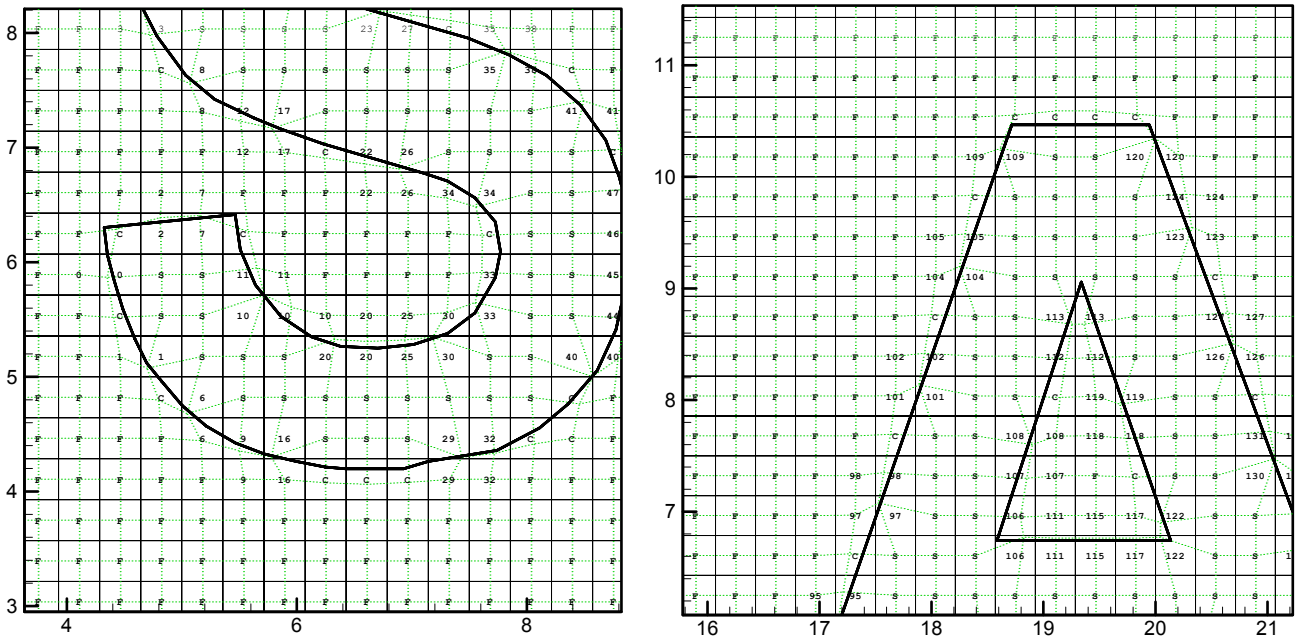

Figure 10. Cell type and combination information. Lower left corner of letter “S” and top right of letter “A”.

4. MOVING WALL EXAMPLES

Geometry handling and moving wall-offsetting functions will be demonstrated via three examples. Geometric problems encountered during normal offsetting involve edge separation and degenerative close loops. These are related to the local curvature and offsetting distance. For the following examples wall offset velocity is constant.

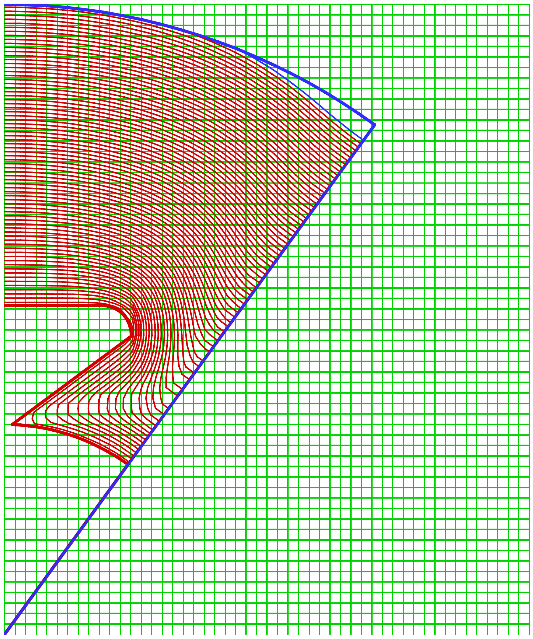

Figure 11. Offsetting in a Cartesian template. One of the four consecutive time steps are plotted

Fig. 11 is a regressing diameter pipe with an arbitrary shape. This geometry is selected because of its high convex curvatures, which may cause problems in geometry dependent offsetting codes.

As the result of detailed considerations of degenerate cells, some simple merging and break up situations can be detected, without any extra merge/break-up routines. Break-up region is realized by marking some degenerate cells as full flow. Fig. 12 demonstrates an offsetting sequence leading to break-up. Template is square of side 8 mm. Offset velocity is 2.5mm/s and grid size is 46x46.

Wall offsetting accuracy can be assessed by comparing exact flow area vs. perimeter plots. For an expanding and contracting circular body, Fig 13, such plots are generated for two grid sizes. A difference of ~10% is observed for the selected coarse grid, Fig 14. The circular body is initially represented by a 20-segment polygon and wall speed was constant. Increasing the grid size 1/3 resulted closer values.

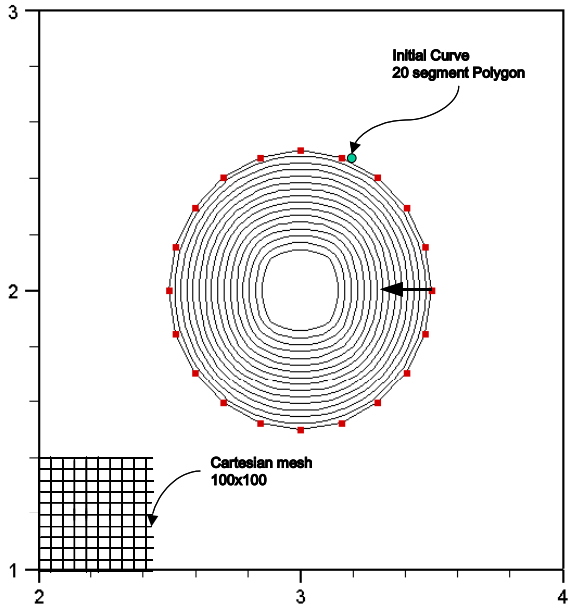

Figure 13. Contracting circle in 100x100 grid. Wall Speed 5 units/sec, Template size 4 units square, Time step 0.001 sec. (Part of the mesh is shown)

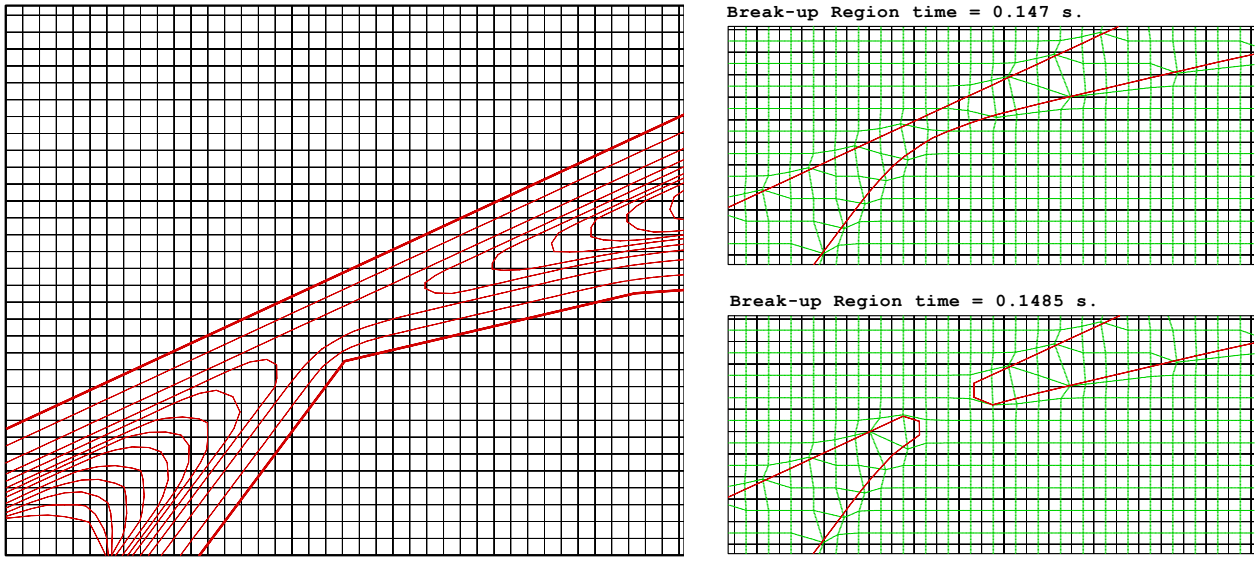

Figure 12. Break-up of an arbitrary solid body. Demonstrating the degenerate cell handling procedures and stream formation. The initial geometry is drawn using thick lines. Not all the zones are plotted. During break-up, time step is modified

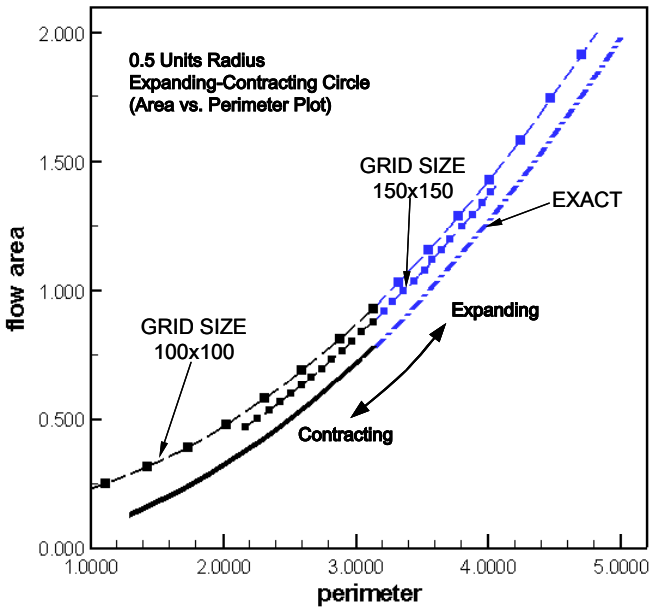

Figure 14. Area vs. Perimeter comparison for an expanding-contracting flow area.

REFERENCES

[1] M. S. Triantafyllou, G. S. Triantafyllou, "Hydrodynamics of Fishlike Swimming," *Annual Review of Fluid Mechanics*, Vol. 33 pp.33-53 (2000)

[2] Zhi B. Gao, et. al., "Bioprosthetic Heart Valve Leaflet Motion Monitored by Dual Camera Stereo Photogrammetry," *Journal of Biomechanics*, Vol.33 pp.199-207 (2000)

[3] K. Pekkan, A. Ucer, "A 2D Moving Boundary Cartesian Grid Solver for Internal Flow Fields of SPRM's," *Advances in Rocket Performance, Life and Disposal*, RTO Specialists Meeting NATO-AVT-089, Denmark, (2002)

[4] Ravi Ramamurti, "Simulation of Flow about Flapping Airfoils Using Finite Element Incompressible Solver," *AIAA Journal*, Vol 39(2) p.8 (2001)

[5] K. Stein, R. Benney, T. Tezduyar, J. Potvin, "Fluid-Structure Interactions of a Cross Parachute: Numerical Simulation," *Computer Methods in Applied Mechanics and Engineering*, Vol 191, pp. 673-687 (2001)

[6] W. Shyy, H. S. Udaykumar, M. M. Rao, R. W. Smith, *Computational Fluid Dynamics with Moving Boundaries*, Taylor-Francis, (1996)

[7] P. I. Crumpton, B. Giles, "Implicit Time-Accurate Solutions on Unstructured Dynamic Grids", *International Journal for Numerical Methods in Fluids*, Vol 25 pp.1285-1300 (1997)

[8] Kozo Fujii, "Unified Zonal Method Based on the Fortified Solution Algorithm",*Journal of Computational Physics* , Vol 118 pp.92-108 (1995)

[9] J. E. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science*, Cambridge University Press, (1999)

[10] Randall J. LeVeque, Donna Calhoun, "Cartesian Grid Methods for Fluid Flow in Complex Geometries," *Computational Modeling in Biological Fluid Dynamics*, IMA Vol. 124, Eds. Lisa J. Fauci and Shay Gueron, Springer-Verlag, (1999)

[11] H. Forrer, R. Jeltsch, "A higher-order boundary treatment for Cartesian-grid methods," *Journal of Computational Physics*, Vol.140 pp.259-277 (1998)

[12] H. Johansen, P. A. Colella, "A Cartesian grid embedded boundary method for Poisson's equation on irregular domains," *Journal of Computational Physics*, Vol.147 pp.60-85 (1998)

[13] J. J. Quirk, " An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies," *Computers and Fluids*, Vol.23(1) pp.125-142 (1994)

[14] M. J. Aftosmis, J. E. Melton, M. J. Berger, "Adaptation and surface modeling for Cartesian mesh methods;" AIAA Paper 95-1725-CP, pp.881-891 (1995)